An Efficient Algorithm for Solving Word Equations*

[Extended Abstract]

Wojciech Plandowski
Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warszawa, Poland
wojtekpl@mimuw.edu.pl

ABSTRACT

We present the first DEXPTIME algorithm which solves word equations i.e. finds a finite representation of all solutions of an equation in a free semigroup. We show how to use our approach to solve two new problems in PSPACE which deal with properties of the solution set of a word equation:

- deciding finiteness of the solution set,
- deciding boundness of the set of maximal exponents of periodicity of solutions.

The approach can be generalized to solve in PSPACE three problems for expressible relations, namely the emptiness of the relation, finiteness of the relation and boundness of the set of maximal exponents of periodicity of elements of the relation.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

General Terms

Algorithms

Keywords

word equations

1. INTRODUCTION

We start with the definition of a problem. Let Σ and Θ be two disjoint sets. These sets are called the alphabet of *constants* and the alphabet of *variables*, respectively. We use the convention that variables are denoted by block letters and the constants by lower-case letters. A word equation is a pair (u, v), usually denoted by $u = v$, such that u, $v \in (\Theta \cup \Sigma)^+$. The word u is called the *left-hand side* and the word v is called the *right-hand side* of the equation $u = v$. A *solution* of a word equation $u = v$ is a substitution of variables by words in Σ^* such that after the substitution the word which is obtained from u is the same as the word which is obtained from v. Denote by w^i, for a natural number i, the word $w \dots w$ where w is copied i times.

Example 1. Consider the equation $abX = Xba$ with one variable X. It is not difficult to check that all words of the form $(ab)^i a$ for $i \geq 0$ are solutions of the word eqaution. It can be proved that there are no other solutions.

Example 2. Consider the equation $XaXbY = aXYbX$ with two variables X and Y. Since the lengths of aX and Xa are the same, the word equation is equivalent to a system of word equations $Xa = aX$ and $XbY = YbX$. It can be proved that the solutions of the first equation are of the form $X = a^i$ for $i \geq 0$. Using this fact the equation $XbY = YbX$ can be rewritten as $a^i bY = Yba^i$. It can be proved that the last equation have the solutions $Y = (a^i b)^j a^i$ for $j \geq 0$. Putting everything together the solutions of the initial equaitons are $X = a^i$ and $Y = (a^i b)^j a^i$, for $i \geq 0$ and $j \geq 0$.

In the rest of this short introduction we will need the notion of a minimal and complete set of unifiers.

Example 3. Consider the equation $X = aYbZa$. The solution set of this equation can be described by formulae $X = aybza$ and $Y = y$ and $Z = z$ where y and z may be replaced by any words.

In our last example a solution set is described by a formula with unfixed parts. Such a formula is called a *unifier*. It represents an infinite family of solutions. In our examples we found descriptions of all solutions of word equations. Such a description is called *complete*. Of course a good description is the input equation. It is short and describes all solutions. However, it does not satisfy an important condition which such a description should satisfy - it does not allow to solve problems dealing with solution sets of word equations.

The formula in our example could be replaced by the following one $X = auvbza$ and $Y = uv$ and $Z = a$ which also represents an infinite family of solutions of the equation $X = aYbZa$. However we can obtain this unifier from the previous one by replacing y by uv and z by a. This

*The material for paper was partially created during visits of the author in August 1999 and November 2001 in Turku University, Turku, Finland.
Supported by KBN grant 4T11C04425.

STOC'06 May 21–23, Seattle, Washington, USA.

means that the second unifier is a nontrivial instance of the first one. A solution of a word equation which is not a nontrivial instance of a unifier which is also a solution of the word equation is called *minimal solution*. In our previous examples all formulae describe minimal solutions.

Example 4. Consider the equation $XY = YX$. It can be proved that the solution set of this equation can be described by formulae $X = u^i$ and $Y = u^j$, for $i \geq 0$, $j \geq 0$. However, the formulae do not describe only minimal solutions since, for instance, the solution $X = u^2$ and $Y = u^4$ is an instance of $X = u$ and $Y = u^2$. The formulae which describe minimal and complete set of unifiers which are solutions of $XY = YX$ are $X = u^i$ and $Y = u^j$ and $gcd(i, j) = 1$ where gcd stands for greatest common divisor.

Our examples suggest that the solution set of a word equation can be described by some formula with integer parameters. However our next example shows that it is not the case even in case of two variables.

Example 5. Consider the equation $XabY = YbaX$. The set of pairs which constitute a solution set of this equation is closely related to standard pairs which in turn are closely related to Sturmian words [8]. Consequently, the set of pairs cannot be expressed by formulae with integer parameters.

If we bound our considerations to constant-free word equations the solution sets of word equations with one, two and three variables are expressible by formulae with integer parameters but the solution set of the word equation $XYZ = ZVX$ with four variables is not expressible by such formulae [7].

In the satisfiability problem we ask whether an input word equation has a solution. The problem was open for a long time until its decidability has been proved by Makanin [14]. The algorithm by Makanin is very famous in theoretical computer science community since its proof of correctness is one of the most complicated proofs of correctness of an algorithm existing in literature. The problem of decidability has been solved but it remained to classify the exact complexity of the problem. The complexity of the original Makanin's algorithm was a composition of several exponential functions. Since that time various scientists proposed improvements to the algorithm which improved the complexity of the algorithm [9, 21, 12, 3]. Current version of the algorithm works in $EXPSPACE$ [6]. The whole algorithm with the proof of correctness occupies 50 pages, [4].

In past decade three different algorithms have been proposed. Two of them solve the problem nondeterministically in time polynomial in n - the size of the input equation and $\log N(n)$ - the logarithm of the size of the minimal length solution of a word equation of length n [19]. The function $N(n)$ can be bounded using the analysis of Makanin's algorithm. Then the bound is triple exponential function. In [16] the author proposed completely different approach which results with a bound which is double exponential. With this bound the algorithms in [19] work in NEXPTIME. The lower bound for $N(n)$ is single exponential and it is conjectured that this is the proper bound for $N(n)$. If this conjecture is true, then the algorithms in [19] work in NP. The satisfiability problem is NP-hard [1] so if the conjecture is true, then the problem is NP-complete.

The third algorithm for the satisfiability problem works in $PSPACE$ [17]. The approach in this paper is a consequence of a deeper analysis of the work in [16].

We propose the fifth algorithm which is similar to the algorithm in [17] but it is conceptually simpler. However, we pay for this simplicity - the proof of correctness of our algorithm is approximately three times longer than the proof of correctness of the algorithm in [17]. The algorithm is a side effect of our considerations and it works in $PSPACE$.

The satisfiability problem answers the question whether or not the solution set of a word equation is empty. It does not answer the question what is the solution set. We present the first DEXPTIME algorithm which solves word equations, that is that answers this more general question. Our approach is a nontrivial modification of a construction in [17].

There are two results which deal with related problems. The first one is by Razborov [20] in which an algorithm which finds a finite representation of all solutions of equations in free groups is presented. Similarly as in our algorithm the representation of all solutions is a finite graph. Razborov's algorithm extends Makanin's algorithm for satisfiability of equations in free groups [15]. Nobody computed its complexity but since it is based on Makanin's algorithm for free groups, it is reasonable to assume that its complexity is at least the complexity of Makanin's algorithm for free groups which is nonprimitive recursive [13]. Our algorithm finds the representation of all solutions of a word equation in DEXPTIME. Since, using the techniques of [5], our construction can be extended to free groups [18], our approach is much more efficient than Razborov's. The techniques of [5] are modifications of the techniques in [17].

The second paper in the area is a paper by Jaffar [9] which presents an algorithm which generates minimal and complete set of unifiers which are solutions of a word equation. If this set is finite, then the algorithm stops. Otherwise it works infinitely long. The algorithm extends Makanin's algorithm for satisfiability of word equations [14]. Again nobody computed its complexity when it stops but since it is based on Makanin's algorithm for word equations, it is reasonable to assume that its complexity is at least the complexity of Makanin's algorithm which is $EXPSPACE$. Our algorithm is more efficient. It finds in DEXPTIME a finite representation of a complete set of unifiers and always stops. However, the set of unifiers which is found by our algorithm does not have to be minimal.

The result of our algorithm is a finite multigraph. A multigraph is an oriented graph which can have several different edges which connect the same pair of vertices. The vertices of our multigraph are labelled by polynomial size word equations. The edges are labelled by polynomial size objects. Each path between two distinguished vertices corresponds to one family of solutions which can be easily derived from the information included in labels of the vertices and of the edges of the path.

We show how to use our approach to solve two new problems in PSPACE:

- deciding finiteness of the solution set,
- deciding boundness of the set of maximal exponents of periodicity of solutions.

The satisfiability problem for word equations is related to the decidability of existential theory of concatenation or, in other words, to the decidability of the computation of a boolean value of a closed existential boolean formula built on word equations. An example of such a boolean formula

is:

$$\exists X \exists Y \exists Z [XabY = YbaZ \wedge Z = XY \vee \neg(aX = Xa)].$$

The problem of computing of a boolean value of such formula is in PSPACE [5]. The theory of concatenation is interesting for logicians since it is one of the simplest theories.

Existential boolean formulae which contain free variables describe relations. An example of such a formula with two free variables X and Y is:

$$\exists Z \exists U [XabU = ZbaX \vee \neg(X = YZ \wedge abY = Uba)].$$

Such formula with k free variables describes a relation consisting of k-tuples of words which after replacing free variables by these words turn the value of the formula to true. Such relations are called *expressible*. In [2, 10, 11] tools for checking expressibility of relations are considered. Using the constructions of [10, 4], it is possible to replace an existential boolean formula with free variables by an equivalent existential formula which is a disjunction of exponential number of polynomial size word equations. This allows to use our approach to find in DEXPTIME a finite description of an expressible relation which corresponds to an input existential formula. The approach can be used to decide in PSPACE whether such a relation is empty or finite or if the exponents of periodicity of the words which are components of tuples belonging to the relation are bounded.

Closed existential formulae built on word equations and on regular constraints are considered [21]. An example of such a formula is:

$$\exists X \exists Y [Xa = bY \vee X = Y \wedge \neg X \in a^* \wedge Y \in (bb \cup a)^*].$$

As previously we can consider formula with free variables. Using the techniques of [5], our algorithm may be generalized to find in DEXPTIME a representation of a set of evaluations of free variables which satisfy nonclosed existential boolean formula built on word equations and on regular constraints [18]. The generalization of our approach can be used to decide in PSPACE whether such a relation is finite or if the exponents of periodicity of the words which are components of tuples bolonging to the relation has bounded exponent of periodicity [18].

Again using the techniques of [5] all our constructions may be generalized to work in free semigroups with inversion and in free groups [18].

Due to space limitations we omit proofs of most lemmata.

2. PRELIMINARIES

Let w be a word. The length of w is denoted by $|w|$. A word p is a *prefix* of a word w if there is a word s such that $w = ps$. A word s is a *suffix* of a word w if there is a word p such that $w = ps$. A word s is a *subword* of a word w if there are words u and v such that $w = usv$. Two words w and w' *conjugate* if $w = uv$ and $w' = vu$ for some words u and v. Denote by $w[i..j]$ the subword of w starting at position i in w and ending at position j in w. The empty word is denoted by 1.

Let Σ and Σ' be two sets called *alphabets*. Elements of alphabets are called *letters*. A *morphism* $h : \Sigma^* \to \Sigma'^*$ is a function satisfying $h(uv) = h(u)h(v)$, for each $u, v \in \Sigma^*$. Note that a morphism is uniquely determined by its values on the letters of the alphabet Σ. Indeed, if $u = a_1, \ldots, a_k$ where $a_i \in \Sigma$, then $h(u) = h(a_1) \ldots h(a_k)$. A *permutation morphism* is a morphism h such that, for each a in Σ, $|h(a)| = 1$ and h is one to one. A *length preserving morphism* is a morphism h such that, for each a in Σ, $|h(a)| = 1$. A *nonerasing morphism* is a morphism h such that, for each a in Σ, $h(a) \neq 1$.

The *length* of a word equation $u = v$ is a number $|u| + |v|$. Let Σ' be a set disjoint with $\Sigma \cup \Theta$. Let Θ' be a set of variables which occur in the word equation $u = v$. A *solution* of a word equation $u = v$ is a morphism $h : (\Sigma \cup \Theta')^* \to (\Sigma \cup \Sigma')^*$ such that it is invariant on Σ, i.e. $h(a) = a$ for $a \in \Sigma$ and such that $h(u) = h(v)$. Observe here that h is invariant on all constants of the equation $u = v$. Observe also that the trivial equation $a = a$ which does not contain variables has exactly one solution - the empty solution.

We say that a solution h of a word equation e is *minimal* if there is no solution h' of e and a nonerasing and nonpermutation morphism $\sigma : (\Sigma \cup \Sigma')^* \to (\Sigma \cup \Sigma')^*$ such that $\sigma \circ h' = h$ where $\circ$ is a composition of functions. Observe here, that since h' and h are invariant on constants occuring in $u = v$, σ is also invariant on constants occuring in $u = v$. Intuitively, h is not minimal if we can find more general solution h' from which we can generate $h(X)$, for $X \in \Theta'$, by replacing in $h'(X)$, for $X \in \Theta'$, letters of Σ' by words over $\Sigma \cup \Sigma'$ in a consistent way (the same letters are replaced by the same words). A solution h of e, such that for some variable X, $h(X)$ contains a constant in Σ', is called a *unifier solution*. By replacing in $h(X)$, for $X \in \Theta$ the letters in Σ', by arbitrary words in a consistent way we obtain from h an infinite family of solutions of e. As it will be clear in the next section when we search minimal solutions we may assume that $|\Sigma'| \leq |\Theta|$. In next part of the paper we consider only solutions h of a word equation $u = v$, that satisfy $h(u) \neq 1$. Observe here that if $h(u) = 1$, then the equation $u = v$ is constant-free and $h(X) = 1$, for each variable X occuring in the equation $u = v$.

Let h be a solution of a word equation $u = v$. An *exponent of periodicity* of the solution h is a maximal fractional number α such that a word w^α is a subword of $h(u) = h(v)$ for some w. We use here the convention that if k is a fractional number, then x^k has sense only if $k|x|$ is an nonegative integer. For instance, the expression $(aba)^{2\frac{2}{3}}$ is an abbreviation of the word $abaabaab$ and $(aba)^{2\frac{1}{4}}$ has no sense.

Throughout the paper n denotes the length of the input equation e.

We will use standard notions of combinatorics of words. We say that a number p is a *period* of a word w if $w[i] = w[i+p]$ for all i that both sides of the equation are defined. We call also a period of a word w a subword of w of length which is a period. The smallest period of a word is called *the period*. If the period of a word w is at most $|w|/2$, then we say that w is *periodic*. A nonempty word is *primitive* if it is not a power of a different word. Clearly, the period of a word is primitive. We will use the following lemma.

PROPOSITION 1 (PERIODICITY LEMMA). *If p and q are periods of a word w and $p + q \leq |w|$, then $gcd(p, q)$ is also a period of w where gcd stands for greatest common divisor.*

In our considerations we use *expressions with integer exponents*. Such an expression is a compact representation of a word. The set of expressions with integer exponents is the smallest family of expressions which satisfies the following rules:

- each $w \in \Sigma^*$ is an expression with integer exponents,

- if x and y are expressions with integer exponents, then xy is an expression with integer exponents,
- if $x \in \Sigma^*$ and k is an positive integer, then x^k is an expression with integer exponents.

In the last case the word x is called a *period* and the integer k is called an *exponent*. Clearly, each expression with integer exponents is an abbreviation of a word which can be obtained from the expression by replacing each of its fragments of the form x^k by k repetitions of x. Observe here that the expression $((ab)^{100}a)^{200}$, which is of height 2, is not an expression with integer exponents in our sense, since all expressions with integer exponents are of height at most one. Observe also that small expression with integer exponents can be a compact description of a very long word. The *size* of the expression is its denotational length.

In our considerations we use *expressions with integer parameters* which after replacing parameters by numbers represent words. Let Δ be a set of integer paramenters and Σ an alphabet of constants. Then the set of expressions with integer parameters is defined as the smallest family of expressions which satisfies the following rules:

- each $w \in \Sigma^*$ is an expression with integer parameter,
- if x and y are expressions with integer parameters, then xy is also an expression with integer parameters,
- if $x \in \Sigma^*$ and $\mu \in \Delta$ then $(x)^\mu$ is an expression with integer parameters.

In the last case the word x is called a *period* and the integer parameter μ is called an *exponent*. Exponents which are integer parameters represent nonnegative integer numbers. As in the previous definition the expression $((ab)^\mu a)^\nu$ where $\mu, \nu \in \Delta$ is not an expression with integer parameters since it is of height 2 and expressions with integer parameters in our sense are of height at most one.

In our considerations if x is a period then each letter of x occurs in it exactly once. Moreover, if two periods occur in the same expression then either they are conjugate or they do not contain a common letter. Such an expression *expr* can be uniquely represented in a *compressed* form

$$compress(expr) = w_1(p_1)^{\alpha_1} w_2(p_2)^{\alpha_2} \dots (p_k)^{\alpha_k} w_{k+1}$$

where w_i, $p_i \in \Sigma^*$, and p_i is a period, and last letters of w_i and p_i are different so that the period p_i cannot be extended to the left, and the period p_i is not a period of the word $(p_i)^{\alpha_i} b$ where b is the first letter of the word w_{i+1} so that the period p_i cannot be extended to the right, and α_i is a **nonconstant** linear function on integer parameters with a constant which can be a fractional number.

Observe here that the word $w = ghcdabcdabab$ can be abbreviated by $gh(cdab)^2ab$ or by $ghcdabcd(ab)^2$ but the abbreviations are not exponential expressions with integer parameters since their exponents are not integer parameters. Those two abbreviations cannot be also candidates for $compress(w)$ since the expression w has no periods. We have $compress(w) = w$.

By *parametrized word equation* we mean an equation e of the form $expr_1 = expr_2$ where $expr_1$ and $expr_2$ are expressions with integer parameters which are built of letters in $\Theta \cup \Sigma$. In our considerations no variable is contained in a period of $expr_1$ or $expr_2$. For a parametrized equation e of the form $u = v$, by $compress(e)$ we mean the equation $compress(u) = compress(v)$.

We use a partial order between linear functions on integer parameters containing fractional constants. We write $\alpha_1 \leq \alpha_2$ iff for any integer parameter μ, the coefficient at integer paramater μ in α_1 does not exceed the coefficient at μ in α_2 and the constant coefficient in α_1 does not exceed the constant coefficient in α_2. For instance, $2\mu_1 + 1\frac{2}{3} \leq 3\mu_1 + \mu_2 + 2$ but it is not true that $\mu + 2 \leq 1000$.

A *factorization of a word* w is a sequence of nonempty words $w_1, \dots, w_l$ such that $w = w_1 \dots w_l$. A *factorization* is a function $\mathcal{F}$ which takes a word and returns its factorization.

We consider systems of linear multiside equations on integer parameters. A multiside equation is an extension of a common equation. A multiside equation is of the form $s_1 = s_2 = \dots = s_k$, for some $k \geq 1$, where s_i is a function of variables. The solution of the multiside equation is such a substitution of variables that makes all the sides s_i equal. For instance a one-side linear equation on integer parameters is $2\mu + 3\nu + 4$ where $\mu, \nu \in \Delta$. A solution of it is any substitution of integer parameters. A three-side linear equation on integer parameters is, for instance, $2\mu + 3\nu + 4 = 3\mu + 2\nu + 5 = 4\omega$. Its has only one solution $\mu = 1$, $\nu = 2$, $\omega = 3$.

3. HOW TO COPE WITH MINIMAL UNIFIER SOLUTIONS

Fix a word equation $e : u = v$ and its minimal (unifier or nonunifier) 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.

LEMMA 1. *Let h be a minimal (unifier or nonunifier) solution of a word equation $e : u = v$. Then each subword of $h(u)$ of length at least 2 has an occurrence which contains a cut.*

Let h be a minimal unifier solution of e. Let $c \in \Sigma'$ be a symbol which occurs in one of $h(X)$, for some variable X. It does not occur in e. The symbol c is either the last one in $h(u)$ and then is a suffix of some word $h(X)$ for some $X \in \Theta$ or is the first symbol of an interval of size two. This interval has an occurrence over a cut so c is a suffix of at least one word $h(X)$ for some variable $X \in \Theta$. This is true for each such symbol c. This means that there are at most $|\Theta|$ of such symbols. This means that h is a nonunifier solution of an equation which is obtainable from e in the following way. Replace each variable X in e by X or by a word Xc_X where $c_X \in \Sigma'$. At least one variable should be replaced by Xc_X. Denote the obtained equation by e'. Note, that the opposite is also true i.e. if e' has a solution then e has a unifier solution. Denote by $Equations(e)$ the set of all equations which can be obtained from e in this way. We proved that

LEMMA 2. *1. Each minimal unifier solution of e can be translated into a minimal nonunifier solution of one of the equations in $Equations(e)$.*

2. Each solution of an equation of $Equations(e)$ can be translated into a unifier solution of e.

We may thus restrict the search to all minimal nonunifier solutions of equations in the set $Equations(e)$. From this moment assume that the alphabet of constants is Σ and that it contains the alphabet Σ'. The size of the alphabet Σ is $O(n^2)$.

4. THE REPRESENTATION OF ALL SOLUTIONS

The representation of a solution set of a word equation e of size n is a finite multigraph G in which each vertex v is labelled by a word equation $e(v)$ of size $O(n^2)$. In G labels of different vertices are different and labels of different edges connecting the same pair of vertices are different. Edge $v_1 \rightarrow v_2$ of G is labelled by a tuple $(\sigma, R, \tau, \mathcal{S}, p)$ where

- σ is a substitution of constants in Σ by words in Σ^+ such that $|\sigma(a)| = O(n^2)$. The mapping σ can be extended to a nonerasing morphism $\sigma : \Sigma^* \rightarrow \Sigma^*$ in a natural way:

  $$\sigma(a_1 \dots a_k) = \sigma(a_1) \dots \sigma(a_k),\ \sigma(1) = 1$$

  where $a_i \in \Sigma$.

- R is a function which takes a variable in Θ and returns a word in Σ^* of length $O(n^4)$. We put no restrictions on R. In particular, $R(X)$ may be the empty word.

- τ is a substitution which takes a constant in Σ and returns an expression with integer parameters of the form q or $q^{\mu+t}$ or $q^{\mu+t}d$ where q is a word in Σ^+ of length $O(n)$, $\mu \in \Delta$, t is a fractional number such that $2 \leq t < 3$ and d is a letter which breaks the period q in $q^{1+t}d$ or, equivalently, $q^{1+t+\frac{1}{|q|}} \neq q^{1+t}d$. We can easily extend τ into a morphism which leads from words over Σ to expressions with integer parameters in Δ.

- τ is such that, if $\tau(a) = q^{\mu+t}$ or $\tau(a) = q^{\mu+t}d$ and $\tau(a') = q'^{\mu'+t'}$ or $\tau(a') = q'^{\mu'+t'}d'$ and q and q' share a constant, then q and q' conjugate.

- $\mathcal{S}$ is a system of linear multiside equations over variables in the set of integer parameters Δ. It has the following properties

  - Each multiside equation of the system $\mathcal{S}$ corresponds to exactly one variable in $e(v_2)$. The number of sides of such multiside equation is the same as the number of occurrences of its variable in $e(v_2)$.
  - The coefficients at integer parameters are positive integers. The sum of them for one side of multiside equation does not exceed $3n$.
  - The constants in multiside equations do not exceed $9n$.
  - The system has a solution.

- p is a function which takes a variable in $e(v_2)$ and returns a word in Σ^* of length $O(n)$. We put no restrictions on p. In particular, $p(X)$, for some variable X, may be the empty word.

Clearly, the labels of the vertices of G and of the edges of G can be stored in space polynomial in n. Consequently the number of vertices of G and the number of edges of G is singly exponential in n.

There is an edge $v_1 \rightarrow v_2$ in G labelled by a fivetuple $(\sigma, R, \tau, \mathcal{S}, p)$ if and only if the equation $e(v_2)$ can be obtained from $e(v_1)$ by the following procedure.

- **Step 1. Substitution of constants.** Create an equation e' from the equation $e(v_1)$ by replacing each constant c of $e(v_1)$ by a word $\sigma(c)$.

- **Step 2. Extension of variables.** Create an equation e'' from the equation e' in the following way. An expression $XR(X)$ of e', where $X \in \Theta$, is replaced by X for each occurrence of X in $e(v_1)$. In this step it is required that $R(X)$ is a prefix of all words which start just to the right of each occurrence of X in e'. The word $R(X)$ does not have to be the longest possible.

- **Step 3. Creation of variables.** Create an equation e''' from the equation e'' in the following way. We put a variable X, which does not occur in $e(v_1)$ and occurs in $e(v_2)$, in m places inside e'' where m is the number of occurrences of X in $e(v_2)$.

- **Step 4. Creation and deletion of integer parameters.** Create the equation $e(v_2)$ on the basis of e'''. This step consists of two substeps.

  - **Step 4A. Creation of integer parameters.** Create equation with integer parameters e^{IV} from e''' in the following way. Each constant c of e is replaced by a word $\tau(c)$.
  - **Step 4B. Deleting integer parameters by extension of variables.** Consider a compressed representation $compress(e^{IV})$ of e^{IV}. Create the equation $e(v_2)$ without integer parameters from $compress(e^{IV})$ in the following way. An expression of the form $Xp(X)^{\alpha_i}$ is replaced by X in i-th occurrence of X in $compress(e^{IV})$. α_i is a linear function on integer parameters at it may be different for different i. If the occurrence of X is i-th occurrence of X in e^{IV}, then α_i is the i-th side of the multiside equation of $\mathcal{S}$ corresponding to X. We assume that, if the subexpression of $compress(e^{IV})$ to the right of this occurrence of X starts from a constant, then α_i is a constant function and the word abbreviated as $p(X)^{\alpha_i}$ is a prefix of this subexpression. If the subexpression of $compress(e^{IV})$ starts from a period, then it is in form $p(X)^{\beta}$ for some linear possibly constant function on integer parameters β and $\alpha \leq \beta$.

Let $v_1 \rightarrow v_2$ be an edge in graph G labelled by a fivetuple $(\sigma, R, \tau, \mathcal{S}, p)$. Let h be a solution of $e(v_1)$ and $\vec{\mu}$ be a vector of integers being a solution of $\mathcal{S}$. Then we can obtain a solution $h_{\sigma,R,\tau,\vec{\mu},p}$ of $e(v_2)$ in the following way. Let

$$h_{\sigma,R,\tau,\vec{\mu},p}(X) = \tau(\sigma(h(X))R(X))^{\vec{\mu}}p(X)^{\alpha(\vec{\mu})},$$

if X occurs in $e(v_1)$ and α is any side of a multiside equation in $\mathcal{S}$ which corresponds to the variable X, and

$$h_{\sigma,R,\tau,\vec{\mu},p}(X) = p(X)^{\alpha(\vec{\mu})},$$

if X occurs in $e(v_2)$ but does not occur in $e(v_1)$ and α is any side of a multiside equation in $\mathcal{S}$ which corresponds to the variable X. In the formulae $\alpha(\vec{\mu})$ denotes the value of α under the substitution $\vec{\mu}$ and $\tau(x)^{\vec{\mu}}$ is a word which can be obtained from $\tau(x)$ after substituting $\vec{\mu}$ for integer parameters occurring in $\tau(x)$. Clearly, $h_{\sigma,R,\tau,\vec{\mu},p}$ is a solution of the word equation $e(v_2)$. In this way we can construct for different solutions $\vec{\mu}$ of $\mathcal{S}$ different solutions of $e(v_2)$ on the basis of the solution g of $e(v_1)$. Let $v_1, v_2, \ldots, v_k$ be a path in G. Then, in the above way, on the basis of any solution of $e(v_1)$ we can construct a family of solutions of $e(v_k)$. Consider the equation $a = a$ with no variables. Then each path in G which starts in $a = a$ and ends in e generates from the empty solution of $a = a$ a family of solutions of e. Denote this family of solutions by $S(e)$. The solutions in $S(e)$ does not have to minimal or nonunifier but the following is true.

LEMMA 3. *Each minimal nonunifier solution of e is in $S(e)$.*

The proof of Lemma 3 is long and complicated. Due to space limitations we omit it here. As a consequence of Lemma 3 and Lemma 2 we have that each path of G starting with $a = a$ and ending with $Equations(e)$ or e corresponds to a family of solutions of e and each minimal solution is in one of the families. Observe here that the using Steps 1-4 of our algorithm we can change any equation in $Equations(e)$ to e. This means that all minimal solutions of e correspond to paths leading from the equation $a = a$ to the equation e. This means that G is a representation of all solutions of e. Since the graph G can be generated in DEXPTIME we have.

THEOREM 1. *It is possible to find a finite representation G of a set of all solutions of e in DEXPTIME.*

5. DECIDING EMPTINESS OF THE SET OF SOLUTIONS

We show now a simplified version of our algorithm which can be used to solve satisfibility problem for word equations:

Input: a word equation e
Output: 'true' if and only if e has a solution

Since there is one to one correspondence between equations of length $O(n^2)$ and vertices of the graph G in our algorithms we identify vertices of G with the equations of length $O(n^2)$. Our algorithm is similar to the one in [17] but simpler. The main part is the same although the graphs the algorithms are working on are different.

$eq := (a = a)$;
while $eq \neq e$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G;
  $eq := eq'$;
return true;

The main difference between our algorithm and the algorithm in [17] is as follows. The algorithm in [17] works on compressed equations of size $O(n^3)$. Our algorithm works on uncompressed equations of smaller size namely of size $O(n^2)$. The procedures of checking whether there is an edge (v_1, v_2) in considered graphs are also similar. The procedure in [17] consists of three steps. Our procedure in the previous section consists of four steps. However we can simplify our procedure. Then the simplified procedure consists of three steps and is very similar to the one in [17].

The simplified procedure is as follows. There is an edge (v_1, v_2) in G if $e(v_2)$ can be obtained from $e(v_1)$ in the following way.

- **Step 1. Substitution of constants.** In this step we obtain an equation e' with integer exponents from the equation $e(v_1)$ in the following way. Replace constants of $e(v_1)$ by expressions of size $O(n^3)$ with integer $O(n)$ bit exponents in consistent way that is the same constants should be replaced by the same expressions. This subsitution defines a morphism $\sigma : \Sigma^* \to \Sigma^*$ such that $\sigma(a)$ is the word abbreviated by the expression which replaces a in this step.

- **Step 2. Creation of new variables.** In this step we obtain an equation e'' with integer exponents from the equation e' in the following way. Put a variable which does not occur in $e(v_1)$ but occurs in $e(v_2)$ m times in m places in e'. We cannot put a variable inside a period of e'.

- **Step 3. Extension of variables** In this step we obtain the equation $e(v_2)$ from the equation e'' in the following way. Replace expressions of the form XR_X by X in each place in e'' the variable X occurs in e''. In this step R_X is a word which occurs to the right of each occurrence of X in uncompressed version of e''. Observe here that since uncompressed version of e'' may be of exponential size, the size of R_X may be exponential although it can be represented by polynomial size expression with integer exponents.

The simplified procedure can be obtained from the procedure in Section 4 by combining Steps 1 and 4A of the four step procedure into Step 1 of the simplified procedure and Steps 2 and 4B of the four step procedure into Step 3 of the simplified procedure.

The exponents which appear in expressions of the simplified procedure are singly exponential functions in n. The correctness of such a choice is a consequence of the fact that they correspond to integer parameters μ in four step procedure and the coefficients μ are connected via systems of linear multiside equations with coefficient being at most linear in n. The systems have solutions. Such systems, when they have solutions, have solutions the components of which are singly exponential in n [4].

This means that if an equation $e(v_2)$ can be obtained from $e(v_1)$ by four step procedure, then $e(v_2)$ can be obtained from $e(v_1)$ by the simplified procedure. However, the simplified procedure is not equivalent to the four step procedure. It can happen that $e(v_2)$ can be obtained from $e(v_1)$ by the simplified procedure but not by four step procedure.

Denote by G' the graph the vertices of which are labelled by different word equations of length $O(n^2)$ and there is an edge (v_1, v_2) in G' if and only if $e(v_2)$ can be obtained from $e(v_1)$ by the simplified procedure. The considerations of this section gives.

LEMMA 4. *G' is a subgraph of G.*

By Lemma 4 and by considerations of the previous section, if e has a solution then there is a path from $a = a$ to the equation e in G'. To complete the proof of correctness of our algorithm we need the following lemma.

LEMMA 5. *If there is a path from the equation $a = a$ to the equation e in G', then e has a solution.*

PROOF. To prove the lemma it is enough to prove that, if (v_1, v_2) is an edge in G' and $e(v_1)$ has a solution, then $e(v_2)$ has a solution. Let g be a solution of $e(v_1)$ and let σ and R_X be as in the simplified procedure. Then h defined by

$$h(X) = \sigma(g(X))R_X,$$

if X occurs in both $e(v_1)$ and $e(v_2)$, and

$$h(X) = R_X,$$

otherwise, is a solution of $e(v_2)$. □

6. DECIDING FINITENESS OF THE SET OF SOLUTIONS

We show now how to apply the graph G to solve two problems in PSPACE. The first one is:

Input: a word equation e
Output: 'true' if and only if e has infinite number of solutions

Observe that checking whether there is an edge (e_1, e_2) in G between two equations can be done in NP. It is enough to guess its label $(\sigma, R, \tau, \mathcal{S}, p)$ and check whether e_2 can be obtained from e_1 using Steps 1-4.

The number of solutions is infinite if and only if the set of numbers $\{|h(u)| : h \text{ is a solution of } e\}$ do not have upper bound. This may happen only in one of three cases:

- e has a unifier solution,
- one of the systems $\mathcal{S}$ on some path in G leading from $a = a$ to e has infinite number of solutions,
- there is a path in G leading from $a = a$ to e which contains a loop which generate infinite family of solutions.

All three conditnions can be checked in NPSPACE and hence in PSPACE. Indeed to check the first one we have to check whether there is a path leading from $a = a$ to one of the equations in $Equations(e)$. This can be done by the following nondeterministic algorithm:

eq:= $(a = a)$;
while $(eq \notin Equations(e))$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G;
  eq:=eq';
return true;

To check the second condition we can use the following nondeterministic algorithm:

eq:= $(a = a)$.
$infinite$:= false;
while $(eq \neq e)$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G;
  if $\mathcal{S}$ being a part of the label of (eq, eq')
    has infinite number of solutions
  then $infinite$:=true;
  eq:=eq';
return $infinite$;

To check the third condition we can use the following nondeterministic algorithm.

eq:= $(a = a)$; $loop\_is\_found$:=false;
while $(eq \neq e)$ **do**
  decide nondeterministically whether execute
    the assignment e':= eq;
    and if yes then execute $loop\_is\_ok$:=false;
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G;
  // Let the label of (eq, eq') be $(\sigma, R, \tau, \mathcal{S}, p)$
  if σ or,
    for some solution $\vec{\mu}$ of $\mathcal{S}$,
    $\tau^{\vec{\mu}}$ is not a length preserving morphism
  then $loop\_is\_ok$:= true;
  $loop\_is\_found$:=$(eq' = e')$
  eq:=eq';
return $loop\_is\_found$ and $loop\_is\_ok$;

We proved the theorem.

THEOREM 2. *Deciding whether a word equation has infinite number of solutions can be done in PSPACE.*

7. DECIDING BOUNDNESS OF THE SET OF MAXIMAL EXPONENTS OF SOLUTIONS

We give now a solution in PSPACE of the following problem:

Input: a word equation e
Output: 'true' if and only if the expenents of periodicity of solutions of e are bounded

The solution is based on the following lemma. The implication from right to left is obvious but the other implication requires a proof. The proof of it uses the constructions from the proof of Lemma 3.

LEMMA 6. *The exponents of periodicity of the solutions of e are unbounded if and only if one of the following conditions is satisfied*

- *e has a unifier solution or,*
- *one of the systems $\mathcal{S}$ on some path in G leading from $a = a$ to e has infinite number of solutions.*

The conditions of Lemma 6 can be checked using the algorithms in the previous section. Hence, we proved

THEOREM 3. *Deciding boundness of the set of maximal exponents of periodicity of solutions of a word equation can be done in PSPACE.*

8. A FINITE REPRESENTATION OF EXPRESSIBLE RELATIONS

Let ϕ be a boolean existential formula which is built on word equations with free variables $X_1, \ldots, X_k$. First using

de Morgans' law we change the formula into an existential formula ϕ' which does not contain not operator and which is built on word equations and word inequalities. The inequalities are changed into equivalent bolean existential formulae of polynomial size which are built on word equations [10]. Next we move existential quantifiers into the begining of the formula obtaining a formula ϕ''. In this way we obtain a polynomial size existential formula ϕ''' which is built on word equations and which contains only 'or' and 'and' operators. Now we change the formula into formula ϕ^{IV} which is disjunction of conjunctions of word equations. The number of disjunctions is at most exponential and the number of word equations in a conjuction block is polynomial. We now use the transformation in [4] to change each block of conjunctions into one equivalent equation. Now we have an existential formula with free variables $X_1,\ldots, X_k$ which is a disjuntion of word equations. Denote the set of these equations by $E(\phi)$.

The representation of an expressible relation corresponding to ϕ is a multigraph G' which can be obtained from G in the following way. The vertices of G' and their labels are the same as in G. The set of edges are the same. The difference is in the labels of the edges in G'. Let $v_1 \rightarrow v_2$ be an edge in G labelled by $(\sigma, R, \tau, \mathcal{S}, p)$. Then in G' the label of the edge is $(\sigma, R', \tau, \mathcal{S}, p')$ where R' is a restriction of R to variables $X_1, \ldots, X_k$ and p' is a restriction of p to variables $X_1, \ldots, X_k$. Having a restricion g to variables X_i of a solution of $e(v_1)$ we can use the same formulas as the ones in Section 4 to build a restriction h to variables X_i of a solution of the equation $e(v_2)$. Indeed, in the formulas it is enough to replace R by R' and p by p'. The paths which corresponds to families of elements of the relation expressible by the formula ϕ lead now from the trivial equation $a = a$ to equations in $E(\phi)$. Clearly, having the graph G we can construct in DEXPTIME the graph G'. Hence, we have

THEOREM 4. *A finite representation G' of the elements of the relation expressioble by a formula ϕ can be computed in DEXPTIME.*

9. THE EMPTINESS OF AN EXPRESSIBLE RELATION

The problem we solve in this section is the following.

Input: an existential boolean formula ϕ built on word equations
Output: 'true' if and only if the relation expressible by ϕ is not empty

The algorithm for checking the emptiness of a relation expressible by a formula ϕ is a consequence of our considerations of the previous sections.

eq:= $(a = a)$;
guess an equation e' in $E(\phi)$;
while $eq \neq e'$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G';
  eq:= eq';
return true;

Generating the equation e' in $E(\phi)$ can be done in NP by creating first the formula ϕ'''. Next by guessing a block of conjunctions in the formula ϕ^{IV} by choosing exactly one formula of each subexpression of ϕ''' which is a disjunction of subformulas of ϕ'''. In this way we obtain a conjunction of word equations which we change into equivalent word equation e'.

Using our algorithm for checking whether there is an edge in G we can check in PSPACE whether there is an edge in G'. Consequently we have.

THEOREM 5. *Checking the emptiness of an expressible relation is in PSPACE.*

10. THE FINITENESS OF EXPRESSIBLE RELATIONS

The problem we solve in this section is the following.

Input: an existential boolean formula ϕ built on word equations
Output: 'true' if and only if the relation expressible by ϕ is finite

The approach is similar to that in Section 6 but we have to be more careful. We have to check whether the set

$$\{(|h(X_1)|, |h(X_2)|, \ldots, |h(X_k)|) :$$

$$h \text{ is a solution of an equation in } E(\phi)\}$$

is finite. Before we formulate conditions which are equivalent to finiteness of an expressible relation we introduce some definitions. Let $\mathcal{S}$ be a system of linear multiside equations over integer parameters in Δ and let α be a linear function over integer paramaters in Δ. We say that α is *unbounded* with respect to $\mathcal{S}$ if and only if

$$\max\{\alpha(\vec{\mu}) : \vec{\mu} \text{ is a solution of } \mathcal{S}\} = \infty$$

Denote by $IntPar(expr)$, for an expression $expr$ with integer parameters, the set of integer parameters occurring in $expr$. Denote by $Alph(w)$, for a word w, the set of letters which occur in w.

A relation expressible by ϕ is infinite if and only if one of the following conditions is satisfied.

- There is an equation in $E(\phi)$ which does not contain one of variables X_i, for $1 \leq i \leq k$.
- There is a unifier solution of an equation in $E(\phi)$ such that $h(X_i)$, for some $1 \leq i \leq k$, contains a letter in Σ'.
- There is a system $\mathcal{S}$ of multiside equations being a part of a label of an edge in G' contained in a path leading from the equation $a = a$ to an equation in $E(\phi)$ such that the following condition is satisfied. Let $LinFun$ be the set of sides of multiside equations in $\mathcal{S}$ which correspond to variables X_i, for $1 \leq i \leq k$. Then one of the functions in $LinFun$ is unbounded with respect to $\mathcal{S}$.
- There is a path $path$ leading from the equation $a = a$ to one of the equations in $E(\phi)$ such that the following condition is satisfied. There is a solution h corresponding to a subpath $path'$ leading from $a = a$ to an equation eq such that the following condition is satisfied. Let $(\sigma, R', \tau, \mathcal{S}, p')$ be a label of the edge (eq, eq') which is the next in $path$ after $path'$. Then one of the functions in

  $$IntPar(\cup_{1\leq i\leq k}\tau(\sigma(h(X_i))R'(X_i)))$$

  is unbounded with respect to $\mathcal{S}$.

- There is a path leading from $a = a$ to an equation e' in $E(\phi)$ which contains a loop which generate infinite family of elements of the relation expressible by ϕ.

Observe here that checking the conditions three and four which should be satisfied by the system $\mathcal{S}$ can be done by linear programming. Hence it can be done in polynomial time.

The first condition can be checked by guessing an appropriate equation in $E(\phi)$ which can be done in NP.

The second condition is satisfied in one of two cases. Either $h(X_i)$, for some $1 \leq i \leq k$, ends by a letter in Σ' or all of them end by a letter in Σ. In the first case h is a solution of an equation e'' in $Equations(e')$ where $e' \in E(\phi)$ which can be obtained from e by replacing one of variables X_i by $X_i c_{X_i}$ where $c_{X_i} \in \Sigma'$. Hence, this case can be checked by guessing the equation e'' and then by checking its satisfiability. The second case can be checked by the following procedure. In the procedure $Equations'(e')$ is the set of those equations in $Equations(e')$ which can be obtained from e' by replacing some variables different from X_i, for $1 \leq i \leq k$, by a words Xc_X where $c_X \in \Sigma'$.

eq:= $(a = a)$;
$Alph$:=$\emptyset$;
guess an equation e' in $E(\phi)$;
while $(eq \notin Equations'(e'))$ **or**
  $Alph \cap \Sigma' = \emptyset$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G';
  //Let $(\sigma, R', \tau, \mathcal{S}, p')$ be the label of the edge
  $Alph$:=$Alph(\sigma(Alph)) \cup \cup_{1 \leq i \leq k} Alph(R'(X_i))$;
  $Alph$:=$Alph(\tau(Alph)) \cup \cup_{1 \leq i \leq k} Alph(p'(X_i))$;
  eq:=eq';
return true;

In the above algorithm we compute the set $Alph$. This set contains letters that occur in the words $h(X_i)$, for $1 \leq i \leq k$, where h is any solution of the family of solutions of the equation eq which correspond to the path we walked on. In the algorithm we use the fact that h is a morphism, then $Alph(h(u)) = Alph(h(Alph(u))$.

To check the third condition we can use the following nondeterministic algorithm:

eq:= $(a = a)$.
$infinite$:= false;
guess an equation $e' \in E(\phi)$;
while $(eq \neq e')$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G';
  //Let $(\sigma, R', \tau, \mathcal{S}, p')$ be the label of (eq, eq')
  //Let $LinFun$ be the sides of $\mathcal{S}$ corresponding
  //to varibles X_i, for $1 \leq i \leq k$
  if there is $f \in LinFun$ which is
    unbounded with respect to $\mathcal{S}$
  then $infinite$:=true;
  eq:=eq';
return $infinite$;

To check the fourth condition we can use the following nondeterministic algorithm:

eq:= $(a = a)$.
$infinite$:= false;
guess an equation $e' \in E(\phi)$;
while $(eq \neq e')$ **do**
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G';
  //Let $(\sigma, R', \tau, \mathcal{S}, p')$ be the label of (eq, eq')
  $Alph$:=$Alph(\sigma(Alph)) \cup \cup_{1 \leq i \leq k} Alph(R'(X_i))$;
  if there is $f \in IntPar(\tau(Alph))$ which is
    unbounded with respect to $\mathcal{S}$
  then $infinite$:=true;
  $Alph$:=$Alph(\tau(Alph)) \cup \cup_{1 \leq i \leq k} Alph(p'(X_i))$;
  eq:=eq';
return $infinite$;

In the above procedure we use the fact that, if f is a morphism which takes a word in Σ^* and returns an expression with integer parameters, then

$$IntPar(f(w)) = IntPar(f(Alph(w)).$$

To check the fifth condition we can use the following nondeterministic algorithm.

eq:= $(a = a)$; $loop\_is\_found$:=false; $Alph$:=$\emptyset$;
guess an equation $e' \in E(\phi)$;
while $(eq \neq e')$ **do**
  decide nondeterministically whether execute
    the assignment e'':= eq;
    and if yes then execute $loop\_is\_ok$:=false;
  nondeterministically generate next equation eq'
    such that (eq, eq') is an edge in G';
  // Let the label of (eq, eq') be $(\sigma, R', \tau, \mathcal{S}, p')$
  if σ restricted to $Alph$ or,
    for some solution $\vec{\mu}$ of $\mathcal{S}$,
    $\tau^{\vec{\mu}}$ restricted to
    $Alph(\sigma(Alph)) \cup \cup_{1 \leq i \leq k} Alph(R'(X_i))$
    is not a length preserving morphism
  then $loop\_is\_ok$:= true;
  $loop\_is\_found$:=$(eq' = e'')$
  $Alph$:=$Alph(\sigma(Alph)) \cup \cup_{1 \leq i \leq k} Alph(R'(X_i))$;
  $Alph$:=$Alph(\tau(Alph)) \cup \cup_{1 \leq i \leq k} Alph(p'(X_i))$;
  eq:=eq';
return $loop\_is\_found$ and $loop\_is\_ok$;

We proved the theorem.

THEOREM 6. *Checking finiteness of an expressible relation is in PSPACE.*

11. BOUNDNESS OF EXPONENTS OF PERIODICITY OF TUPLES IN EXPRESSIBLE RELATION

In this section we solve the following problem.

Input: an existential boolean formula built on word equations
Output: 'true' if and only if there is no finite upper bound for the exponents of periodicity of words being components of tuples in the relation expressible by ϕ.

Our algorithm is based on the following lemma. The proof of this lemma is similar to the proof of Lemma 6. Due to space limitations we omit it here.

LEMMA 7. *Maximal exponents of periodicity of tuples of a relation expresible by a formula ϕ are unbounded if and only if one of the following conditions is satisfied.*

- *There is an equation in $E(\phi)$ which does not contain one of variables X_i, for $1 \leq i \leq k$.*
- *There is a unifier solution of an equation in $E(\phi)$ such that $h(X_i)$, for some $1 \leq i \leq k$, contains a letter in Σ'.*
- *There is a system $\mathcal{S}$ of multiside equations being a part of a label of an edge in G' contained in a path leading from the equation $a = a$ to an equation in $E(\phi)$ such that the following condition is satisfied. Let $LinFun$ be the set of sides of multiside equations in $\mathcal{S}$ which correspond to variables X_i, for $1 \leq i \leq k$. Then one of the functions in $LinFun$ is unbounded with respect to $\mathcal{S}$.*
- *There is a path path leading from the equation $a = a$ to one of the equations in $E(\phi)$ such that the following condition is satisfied. There is a solution h corresponding to a subpath $path'$ leading from $a = a$ to an equation eq such that the following condition is satisfied. Let $(\sigma, R', \tau, \mathcal{S}, p')$ be a label of the edge (eq, eq') which is the next in path after $path'$. Then one of the functions in*

$$IntPar(\cup_{1 \leq i \leq k} \tau(\sigma(h(X_i))R'(X_i)))$$

is unbounded with respect to $\mathcal{S}$.

The conditions of Lemma 7 can be checked using the algorithms in the previous section.

We proved the theorem.

THEOREM 7. *Checking boundness of exponents of periodicity of tuples of an expressible relation is in PSPACE.*

12. ACKNOWLEDGMENTS

The author wishes to thank Prof. Wojciech Rytter (Warsaw University, Poland) and two anonymous referees for many detailed comments which allowed on significient improvement of the final version of this paper.

13. REFERENCES

[1] D. Angluin. Finding pattern common to a set of string. In *Proceedings of Symposium on the Theory of Computing STOC'79*, pages 130–141. ACM Press, 1979.

[2] J. R. Buchi and S. Senger. Definability in the existential theory of concatenation and undecidable extensions of this theory. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 22:337–342, 1987.

[3] V. Diekert. personal communication, 1998.

[4] V. Diekert. Makanin's algorithm. In M. Lothaire, editor, *Algebraic aspects of combinatorics on words*. Cambridge University Press, 2002.

[5] V. Diekert, C. Gutierrez, and C. Hagenach. The existential theory of equations with rational constraints in free groups is pspace-complete. In *Proceedings of Symposium on Theoretical Aspects of Computer Science STACS'01, Lecture Notes in Computer Science 2010*, pages 170–182, 2001.

[6] C. Gutierrez. Satisfiability of word equations with constants is in exponential space. In *Proceedings of the Annual Symposium on Foundations of Computer Science FOCS'98*, pages 112–119. IEEE Computer Society Press, 1998.

[7] Y. I. Hmelevski. Equations in free semigroup. *Trudy Matematiczeskovo Instituta Steklova*, 107, 1971. English translation: *Proceedings of Steklov Institute of Mathematics 107(1971)*, American Mathematical Society, Providenca R.I., 1976.

[8] L. Ilie and W. Plandowski. Two-variable word equations. *Theoretical Informatics and Applications*, 34:467–501, 2000.

[9] J. Jaffar. Minimal and complete word unification. *Journal of the ACM*, 37(1):47–85, 1990.

[10] J. Karhumaeki, F. Mignosi, and W. Plandowski. The expressibility of languages and relations by word equations. *Journal of the ACM*, 47(5):483–505, 2000.

[11] J. Karhumaeki, F. Mignosi, and W. Plandowski. On the expressibility of languages by word equations with a bounded number of variables. *Bulletin of the Belgian Mathematical Society*, 8(2), 2001.

[12] A. Kościelski and L. Pacholski. Complexity of makanin's algorithm. *Journal of the ACM*, 43(4):670–684, 1996.

[13] A. Kościelski and L. Pacholski. Makanin's algorithm is not primitive recursive. *Theoretical Computer Science*, 191:145–156, 1998.

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

[15] G. S. Makanin. Equations in a free group. *Izv. Akad. Nauk SSR Ser. Mat.*, 46:1199–1273, 1983. In Russian; English transl. in *Math. USSR Izv.* **21**, 1983.

[16] W. Plandowski. Satisfiability of word equations is in nexptime. In *Proceedings of the Symposium on the Theory of Computing STOC'99*, pages 721–725. ACM Press, 1999.

[17] W. Plandowski. Satisfiability of word equations is in pspace. In *Proceedings of the Annual Symposium on Foundations of Computer Science FOCS'99*, pages 495–500. IEEE Computer Society Press, 1999.

[18] W. Plandowski. paper in preparation, 2006.

[19] W. Plandowski and W. Rytter. 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 1443*, pages 731–742. Springer, 1998.

[20] A. A. Razborov. On systems of equations in a free group. *Izv. Akad. Nauk SSR Ser. Mat.*, 48:779–832, 1984. In Russian; English transl. in *Math. USSR Izv.*, **25**, 115-162, 1985.

[21] K. Schulz. 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 572*, pages 85–150. Springer, 1992.