Solving Two-Variable Word Equations*
(Extended Abstract)

Robert Dąbrowski and Wojtek Plandowski

Institute of Informatics
University of Warsaw
Banacha 2, 02-097 Warszawa, Poland
{r.dabrowski,w.plandowski}@mimuw.edu.pl

Abstract. We present an algorithm that solves word equations in two variables. It computes a polynomial size description of the equation's solutions in time $O(n^5)$. This additionally improves the result by Ilie and Plandowski [8] by giving the currently fastest algorithm to decide solvability of two-variable word equations.

1 Introduction

One of the most famous and most complicated algorithms existing in literature is *Makanin's algorithm* [13]. The algorithm takes as an input a word equation and decides whether or not the equation has a solution. It has been improved several times. The algorithm's currently best version works in EXPSPACE [6] and occupies (including the proof of correctness) over forty pages [5].

Recently new algorithms to decide solvability of general word equations have been found [15,17]. The first one works nondeterministically in polynomial time with respect to the length of the input equation and the logarithm of the length of its minimal solution. Since the best upper bound for the length of the minimal solution is double exponential [16], then with this bound the algorithm in [15] works in NEXPTIME. The algorithm in [17] works in PSPACE.

Obviously the algorithms solving the problem of satisfiability of general word equations cannot be called efficient. We cannot even expect efficiency since the problem is NP-hard [1,11]. However, if we concentrate on selected classes of word equations, then there do exist polynomial time algorithms either to decide solvability, or to describe solutions of word equations.

For instance, an efficient algorithm that solves word equations in one variable is known [3]. It works in $O(\#_x \log n)$ time, where n is the length of the input equation and $\#_x$ is the number of variable occurrences. For two-variable word equations, there exist two polynomial time algorithms [2,8] that determine solvability. The best one works in $O(n^6)$ time. There is also an efficient $O(n \log^2 n)$ time algorithm for restricted equations with two variables [14].

There are two algorithms that solve general word equations [9,18]. The first one generates representation of solutions which is a set of unifiers. If this set

* Supported by KBN grant 4T11C04425.

J. Diaz et al. (Eds.): ICALP 2004, LNCS 3142, pp. 408–419, 2004.

is finite the algorithm terminates. The second algorithm generates a finite representation of all solutions in form of a finite graph. It works on equations in free groups and is based on Makanin's algorithm for free groups which is not primitive recursive [12]. Both algorithms cannot be called efficient. Existence of a polynomial time algorithm to solve a word equation in two variables has remained up to now an open problem. In this papers we present the first polynomial time algorithm that solves equations in two variables. By solving a word equation we mean an algorithm that finds a polynomial description of all of its solutions.

2 Notation

A *factorization of a word* w is a sequence of words $w_1, \ldots, w_l$ such that $w = w_1 \ldots w_l$. A *factorization* is a function $\mathcal{F}$ such that it takes a word and returns some factorization of this word.

Definition 1. *For a primitive word $P \in \Sigma^\star$ we define P-factorization as follows. For any word $w \in \Sigma^\star$ there exists a unique representation $w = w_0 \cdot P^{k_1} \cdot w_1 \cdot \ldots \cdot P^{k_n} \cdot w_n$ where $n \geq 0$, $k_i \geq 0$ for any $1 \leq i \leq n$ and (1) w_i does not contain P^2 as a factor for any $0 \leq i \leq n$; (2) P is both a proper prefix and suffix of w_i for any $0 < i < n$; (3) P is a proper suffix of w_0 or $w_0 = 1$; (4) P is a proper prefix of w_n or $w_n = 1$. Then the* P-factorization *of w is the ordered sequence $w_0, P^{k_1}, w_1, \ldots, P^{k_n}, w_n$. The* size *of the P-factorization is*

$$\sum_{i=0}^{n} |w_i| + |P| + \sum_{i=1}^{n} \log k_i.$$

Our next definition comes from [10] and is quite technical. The idea of it is to consider factorizations which have quite strong property. Let a word y occurs inside a word x. If we place y over its occurrence inside x and compare both factorization of y and the part of the factorization of x which is under y, then the factorizations are almost the same except the beginning and end. A formal definition follows.

Definition 2. *Let $\mathcal{F}$ be a factorization. Let $\mathcal{F}(x) = x_1, \ldots, x_j$ and $\mathcal{F}(y) = y_1, \ldots, y_k$ for some words x and y. The factorization $\mathcal{F}$ is* synchronizing *iff for some non-negative integer parameters l and r if $k > l + r$ then there exist $l' \leq l$ and $r' \leq r$ such that the following condition holds. Denote $u = y_1 \ldots y_{l'}$ and $v = y_{k-r'+1} \ldots y_k$ (*border *factors: l' starting and r' ending ones). If y occurs in x starting at position i then (1) positions $i + |u|$ and $i + |y| - |v|$ in x are starting positions of factors, say x_p and x_q, respectively; (2) the sequences of factors $x_p, \ldots, x_q$ and $y_{l'+1}, \ldots, y_{k-r'}$ are identical; (3) the occurrence of u at position i in x covers at most $l-1$ factors of x; (4) the occurrence of v at position $i + |y| - |v|$ in x covers at most $r - 1$ factors of x.*

Proposition 1 (Karhumäki, Mignosi, Plandowski [10]). *Given a primitive word P, the P-factorization is synchronizing with $l = r = 2$.*

Definition 3. *By a* k-ary word generator *we denote a function* $w : N^k \to \Sigma^\star$ *that allows for a compact representation of a family of words* $\{w(n_1, \ldots) | n_1, \ldots \in N\}$. *In this paper* unary *or* binary generators *are used and they are typically represented by expressions, i.e.* unary generator $u^i v$ *representing set of words* $\{u^i v | i \geq 0\}$ *for certain* $u, v \in \Sigma^\star$*; or* binary generator $(u^i v)^j u^i v$ *representing set of words* $\{(u^i v)^j u^i v | i, j \geq 0\}$ *We shall distinct a* constant *as a word generator with zero arity.*

Definition 4. *By a* rotation *we mean a mapping* $rot : \Sigma^\star \to \Sigma^\star$ *defined for any* $a \in \Sigma$, $w \in \Sigma^\star$ *by* $rot(aw) = wa$. *A composition of* t *rotations is denoted by* rot^t, $t \geq 0$.

If $w = rot^t(u)$, for some t then we say that w and u *conjugate* or *are conjugates.*

Definition 5. *By a* primitive root *we mean a mapping* $root : \Sigma^\star \to \Sigma^\star$ *defined for any* $u \in \Sigma^\star$ *by* $root(u) = v$ *iff* $v \in \Sigma^\star$ *is of minimal length and such, that* $u = v^k$ *for some* $k \geq 1$.

Definition 6. *Given an equation (or a system of equations)* E, *by* $Sol(E)$ *we denote the set of its solutions. In case of a* multiple-variable *word equations, by* $Sol_x(E)$, *for a variable* x *of* E, *we denote a language which is the set of the* x *components of the solutions of* E.

3 Systems of Equations

We introduce tools that let us solve some specific systems of word equations.

3.1 System S_1

Let S_1 be the following system of word equations in two variables $|u| < |x|$, where $A, B, C, D \in \Sigma^\star$, CD is primitive.

$$S_1 : \begin{cases} u \cdot A \cdot x = x \cdot B \cdot u \\ CD \cdot u = u \cdot DC \end{cases}$$

We can prove the following lemma.

Lemma 1. *Given a system of equations* S_1 *of length* n, *it is posible to find in time* $O(n)$ *the following representation of* $Sol_x(S_1)$.

(α) *At most one binary generator of the form* $x_{j,k} = (P^j Q)^k P^{j+c} P'$ *for certain* $P, Q \in \Sigma^*$ *of length* $O(n)$, $|c| \leq n$, P *primitive,* P' *a prefix of* P, P *not a prefix of* Q *and any* $j, k \geq 0$, *or of the form* $x_{j,k} = (P^j Q)^k P'$ *for* P *primitive and* P *not a prefix of* Q *and any* $j, k \geq 0$.

(β) *A set of* $O(n)$ *unary generators* $x_j = P^j Q$ *for certain* $P, Q \in \Sigma^*$ *of length* $O(n)$, P *primitive,* P *not a prefix of* Q, $j \geq 0$.

3.2 System S_2

Let S_2 be the following system of distinct word equations in two variables $|u| < |x|$, where $A, B, C, D \in \Sigma^\star$. We assume $|A| = |B| \leq |C| = |D|$ and $A \neq C$ or $B \neq D$.

$$S_2 : \begin{cases} u \cdot A \cdot x = x \cdot B \cdot u \\ u \cdot C \cdot x = x \cdot D \cdot u \end{cases}$$

We can prove the following lemma.

Lemma 2. *Given a system of equations S_2 of length n, it is possible to find in time $O(n^2)$ the following representation for $Sol_x(S_1)$.*

(α) At most one binary generator $x_{j,k} = (P^jQ)^kP^{j+b}P'$ for some P, Q of length $O(n)$ and for some constant $|b| \leq n$.
(β) A set of $O(n)$ unary generators $x_j = P^jQ$ for some P, Q of length $O(n)$, P primitive and $j \geq 0$.
(γ) A set of $O(n)$ constants.

3.3 System S_3

We distinguish a system S_3 which consists of one equation in two variables u, x, where $A, B \in \Sigma^\star$.

$$S_3 : u \cdot A \cdot x = x \cdot B \cdot u$$

There is a a close connection between such equations and Sturmian words [8]. Moreover, as already noticed by Hmelevskii [7], S_3 has a non-empty set of solutions iff there exist $P, Q, R \in \Sigma^\star$ such, that $A = PQR$ and $B = QRP$. The following lemma holds.

Proposition 2 (Ilie, Plandowski [8]). *Given an equation S_3 of length n, it is possible to find in time $O(n)$ a finite set of substitutions (computed on the basis of the graph induced by the equation) that represents $Sol(S_3)$.*

Therefore, if the problem reduces to a single equation of type S_3, then we can terminate the algorithm.

4 Interlaced Sequences

Denote $[C_i]_{i=1}^k = C_1 \ldots C_k$. Let $A(x) = [xA_i]_{i=1}^k$ and $B(x) = [B_jx]_{j=1}^k$ be two sequences of equal length over a variable x and coefficients $A_i, B_j \in \Sigma^\star$. The equation

$$E : A(x) \cdot y = y \cdot B(x)$$

is called a *singleton equation* in variables x, y. The size of it denoted by n; we additionally assume that x is both a prefix and a suffix of y. We introduce the

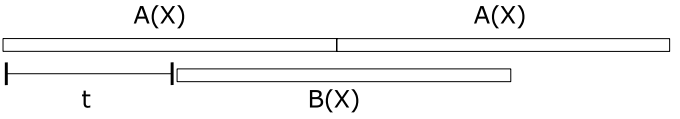

Fig. 1. Conjugation

technique of *interlaced sequences* that allows us to solve E. Fix $(x, y) \in Sol(E)$. Then $A(x)$ and $B(x)$ are conjugated (by y), that is

$$A(x) = rot^t(B(x))$$

for some $t \geq 0$. In other words $B(x)$ is a subword of $A(x)A(x)$, see Figure 1.

To find $Sol_x(E)$ we consider separately *simple* and *complex* solutions. Simple solutions correspond to the case when one of the ends of x in $B(x)$ or in $A(x)$ drops inside a constant A_i or B_j.

4.1 Simple Solutions

In the first case we consider *simple* x only. First, we take all $O(n^2)$ factors of A_i, B_j as possible *constants* x of length $O(n)$. Second, we consider all $O(n)$ prefixes and suffixes of A_i, B_j as periods. Each of them creates $O(n)$ *unary generators* $x = P^jQ$ where P is primitive, $j \geq 0$, and Q is a proper prefix of P. Totally we get $O(n^2)$ unary generators.

4.2 Complex Solution

In the remaining case we may assume that no occurrence of x either starts or ends within any of the coefficients. Therefore to solve the conjugation of $A(x)$ and $B(x)$ it suffices to consider k possible *interlaced sequences* of coefficients A_i and B_j. Fix the interlace to be $I : A_1, B_1, A_2, B_2, \ldots, A_k, B_k$; renumerate the coefficients if necessary.

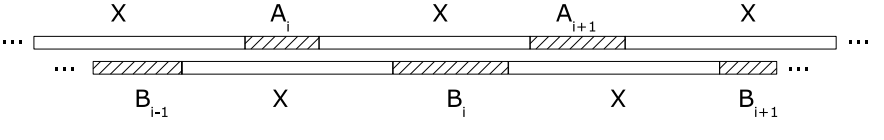

Fig. 2. Interlaced sequences

Case 1 If all coefficients A_i are equal and all coefficients B_i are equal and A_i and B_i are of equal length, then E degenerates to an equation of type S_3.

Case 2 If all coefficients are of equal length, then interlace results in a system S_2 of equations of length $O(n)$. Since Case 1 does not hold, we can find among the equations two which form a system S_2 where $|A| = |B| = |C| = |D|$ and $C \neq A$ or $B \neq D$. Then only cases (β) or (γ) may hold in Lemma 2. Hence, this case results in $O(n)$ constants and at most one unary generator.

Case 3 In the remaining case every interlace contains two consecutive coefficients $|A| \neq |B|$. It results in a system of equations as depicted in Figure 3. Let A be shorter and to the left to B (the other cases are symmetric).

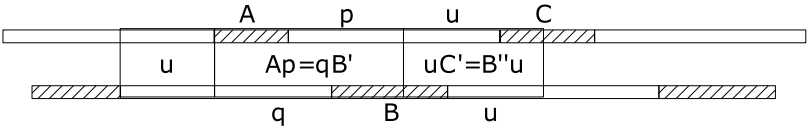

Fig. 3. System of equations

Since $pu = uq = x$ then the system is equivalent to the following one.

$$\begin{cases} u \cdot A \cdot x = x \cdot B' \cdot u \\ B'' \cdot u = u \cdot C' \end{cases}$$

It is possible to find the primitive roots of all coefficients in total time $O(n)$, hence all k systems can be reduced to systems of type S_1 in total time $O(n)$.

One remark should be done in the reasoning in this case when $|C| < |B'|$. In that case it is not possible to calculate directly the coefficient C' in the second equation. Note, however, that then, since $|u| \geq |B''| = C'$, C' is of the form CC'' where C'' is a prefix of u which is just to the right of C. We know that B'' is a prefix of u and we know the length of C'' which is $|B''| - |C|$ so we can compute C'' and therefore also C'. Hence, this case results in at most one binary generator and $O(n)$ unary generators.

Lemma 3. *Given a* singleton equation E *of length* n*, it either degenerates to a single equation of type* S_3*, or it is possible to find in time* $O(n^2)$ *the following representation of* $Sol_x(E)$*.*

- (α) *A set of* $O(n)$ *binary generators* $x_{j,k} = (P^jQ)^kP^{j+b}P'$ *for certain* P*,* Q*,* R *of lengths* $O(n)$*,* $|b| \leq n$*,* P *primitive,* $j, k \geq 0$ *or* $x_{j,k} = (P^jQ)^kP'$ *for* P*,* Q*,* P' *of lengths* $O(n)$*,* P *primitive* $j, k \geq 0$*.*
- (β) *A set of* $O(n^2)$ *unary generators* $x_j = P^jQ$ *for certain* P, Q *of lengths* $O(n)$*,* P *primitive,* $j \geq 0$*.*
- (γ) *A set of* $O(n^2)$ *constants* x *of lengths* $O(n)$*.*

5 Singleton Equations

Again we consider two sequences $A(x)$ and $B(x)$ as defined in the previous section and the singleton equation they induce, but this time we relax the condition $|A(x)| = |B(x)|$ and assume only that the number of x in $A(x)$ and $B(x)$ are the same. This leads to two *skew* types of singleton equations.

5.1 Singleton+ Equations

We assume $\sum_{i=1}^{k} |A_i| > \sum_{i=1}^{k} |B_i|$. By *singleton+* equations we denote equations of the form

$$E : A(x) \cdot y = y \cdot B(x) \cdot B'x'$$

where $B' \in \Sigma^\star$, where x' is both a nontrivial prefix and suffix of x and $|B'x'| = |A(x)| - |B(x)|$ and x is a prefix and suffix of y. We can prove the following lemma.

Lemma 4. *Given a* singleton+ *equation E of length n, it is possible to find in time $O(n^3)$ the following representation of $Sol_x(E)$.*

(α) *A set of $O(n)$ binary generators $x_{j,k} = (P^jQ)^kP^{j+b}P'$ for certain P, Q, R of lengths $O(n)$, $|b| \leq n$, P primitive, j, $k \geq 0$, or $x_{j,k} = (P^jQ)^kP'$ for certain P, Q, P' of lengths $O(n)$ and j, $k \geq 0$.*
(β) *A set of $O(n^3)$ unary generators $x_j = P^jQ$, for certain $P, Q \in \Sigma^n$, P primitive, $j \geq 0$.*
(γ) *A set of $O(n^2)$ constants x of lengths $O(n)$.*

5.2 Singleton- Equations

We assume $\sum_{i=1}^{k} |A_i| < \sum_{i=1}^{k} |B_i|$. By *singleton-* equations we denote equations of the form

$$E : A(x) \cdot y \cdot A'x'' = y \cdot B(x)$$

where $A' \in \Sigma^\star$, x'' is both a nontrivial suffix and prefix of x and $|A'x''| = |B(x)| - |A(x)|$. We can prove the following lemma.

Lemma 5. *Given a* singleton- *equation E of length n, it is possible to find the following representation of $Sol_x(E)$.*

(α) *A set of $O(n)$ binary generators $x_{j,k} = (P^jQ)^kP^{j+b}P'$ for certain P, Q, R of lengths $O(n)$, $|b| \leq n$, P primitive, $j, k \geq 0$.*
(β) *A set of $O(n^3)$ unary generators $x_j = P^jQ$, for certain $P, Q \in \Sigma^n$, P primitive, $j \geq 0$.*
(γ) *A set of $O(n^2)$ constants x of lengths $O(n)$.*

6 Single-Periodic Solutions

Our goal is to solve equation E in two variables x, y for which x is known to be of the form $x = P^iQ$, for some P, Q of lengths $O(n)$, P primitive, P not a prefix of Q and any $i \geq 0$. First, we use the algorithm in [3] to solve E when the value of x is a fixed word, namely in cases $i = 0$ and $i = 1$, i.e. $x = Q$ and $x = PQ$. Then the algorithm works in $O(n^2 + \#_y \log n)$ time where $\#_y$ is the number of occurrences of the variable y in the equation. Since $\#_y = O(n)$ it totally works in $O(n^2)$ time. In the remaining part we may concentrate on the case $i \geq 2$.

Our considerations work on P-factorizations. We use two data structures which can be computed in linear time on the basis of P-factorizations of words. The first data structure is an *overlap array* for a word y. This is an array which says for each position i whether $y[i..|y|] = y[1..|y|-i+1]$. The second data structure is a *prefix array* which for each position i of y says the length of the longest prefix of y which starts at position i in y. Both data structures are standard ones in text algorithms [4]. However, they are computed for explicite representation of y. We compute them for words which are given by a P-factorization of y. In our case it can happen that the size of P-factorization is of smaller order than the size of y (see the definition of the size of a P-factorization of a word). Both arrays can be, however, computed in linear time with respect to the size of the P-factorization of y. We can prove the following theorem.

Theorem 1. *Let P and Q be two words such that they are of length $O(n)$ and P is primitive and P is not a prefix of Q. Let i be an integer parameter. All solutions in which the variable x is of the form P^iQ can be found in $O(n^2)$ time.*

7 Double-Periodic Solutions

Our goal now is to solve equation E in two variables x, y for which x is known to be of the form $x = (P^iQ)^kP'$ or $x = (P^iQ)^kP^{i+c}P'$, for some $P, P', Q \in \Sigma^n$, P primitive, P not a prefix of Q, constant c and any $i \geq 0$. We split our considerations into n cases for $i = 0, 1, \ldots n-1$ and $i \geq n$. Starting from that point the proof follows the lines the proof of Theorem 1 where P^iQ plays the role of P, k plays the role of i and instead of P-factorization we work on P^iQ-factorizations. We can prove the following theorem.

Theorem 2. *Let P, P' and Q be three words such that they are of length $O(n)$, P is primitive and P is not a prefix of Q. Let i and k be be two integer parameters and c be an integer constant such that $|c| \leq n$. All solutions in which the variable x is of the form*

$$(P^iQ)^kP'$$

or of the form

$$(P^iQ)^kP^{i+c}P'$$

can be found in time $O(n^3)$.

8 Canonization

We revise a data structure that allows for efficient comparison of concatenated coefficients. Let Π be a set of words over an alphabet Σ of finite size and of total length $\sum_{u\in\Pi}|u| = n$. We consider two words $u = u_1 \cdot \ldots \cdot u_k$ and $v = v_1 \cdot \ldots \cdot v_l$ where $u_i, v_j \in \Pi$ and k, l are fixed. Our aim is to verify quickly whether u is a prefix of, or equal to, v. We follow the reasoning introduced originally in Section 4 of [3].

Proposition 3 (Dąbrowski, Plandowski [3]). *Given a finite set Π of words over an alphabet Σ and of total length $n = \sum_{u\in\Pi}|u|$, after an $O(n)$-time preprocessing it is possible to answer in time $O(1)$ if for given a, b being some prefixes of words in Π it is true, that a is a prefix of b.*

Definition 7 (Dąbrowski, Plandowski [3]). *For given set Π of words by a* prefix array *$Pref[u, j]$ for a word $u \in \Pi$ and $1 \le j \le |u|$ we mean the longest prefix of a word in Π which occurs at position j in u.*

Proposition 4 (Dąbrowski, Plandowski [3]). *Given a finite set Π of words of total length $n = \sum_{u\in\Pi}|u|$ and over an alphabet Σ, it is possible to construct the prefix array for Π in time $O(n)$.*

Remark 1. It clearly follows from the propositions, that after $O(n)$-time preprocessing it is possible to answer in constant time whether $u \in \Pi$ starts at position i in $v \in \Pi$.

We say that a word equation is in *canonical* form if its sides start with different variables and end with different variables. Now, we show how we transform an input word equation to its canonical form. If both parts of the equation start or end with the same symbol (constant or variable) then the symbol is reduced. Another case is when one side starts (ends) with a variable and the other starts (ends) with the same variable preceeded by a coefficient.

$$E : A \cdot x \cdot \ldots = x \ldots$$

In such case A is clearly a period of x and the case results in a set of $O(n)$ *unary generators* representing $Sol_x(E)$. The only difficult part is one when one side starts (ends) with a variable and the other starts (ends) with the other variable preceeded by a coefficient.

$$E : A \cdot y \cdot \ldots = x \ldots$$

In such case a set of $O(n)$ *constants* representing $Sol_x(E)$ which are prefices of A is considered first and then a substitution $x := Ax$ is executed. Now, both sides of the equation start with different variables. Similarly, as above we proceed with ends of sides of the equation. Now, the equation is in canonical

form. However its size can be quadratic with respect to the size of the original equation if the constant A in the substitution is large. This is why we do not apply the substitutions directly. Instead, we put before or after each occurrence of an appropriate variable an abbreviation which tells that it is the place for a constant A. Now, for such an equation, using the data structures we said about, we can, for instance, verify in linear time whether a pair of words x, y is a solution although the equation can represent an equation of quadratic size. Similarly, we can find a representation of a P-factorizations of all constants of the equation in linear time although the total size of the factorizations given explicitely can be larger than linear.

9 Main Result

Let E be a two-variable word equation in canonical form, namely it starts and ends with distinct variables. Fix $A(x) = [xA_i]_i$ and $B(x) = [B_j x]_j$ to be the longest respective sequences in one variable.

$$E : A(x) \cdot y \cdot \phi(x, y) = y \cdot B(x) \cdot B'y \cdot \psi(x, y)$$

The case of $|x| = |y|$ leads immediately to a one-variable word equation, which we can handle efficiently as described in [3]. Hence, due to the problem's symmetry, fix $(x, y) \in Sol(E)$ such that $|x| < |y|$. Since the equation is in canonical form then x is both a prefix and suffix of y.

The algorithm that solves E is iterated. A single iteration splits E into E' and E'' and either returns a representation of $Sol_x(E')$ by means of generators or reduces E' to an equation of system S_3. In the former case the algorithm we use the results of Section 6 and Section 7. In the latter case it follows to iterate with E''. Finally, the iterations either result in a system S_2 or a single equation S_3.

Denote by $|A(x)|_x$ the number of occurrences of the variable x in $A(x)$. Similarly, denote by $|B(x)|_x$ the number of occurrences of the variable x in $B(x)$. To perform a single iteration three cases need to be considered.

9.1 $|A(x)|_x \leq |B(x)|_x$

We consider the shortest $B'(x) \leq B(x) = B'(x)B''(x)$ such that $|B'(x)|_x = |A(x)|_x$ and $B'(x)$ ends with x.

If $|A(x)| = |B'(x)|$ then we reduce the problem to solving a singleton equation $E' : A(x)y = yB'(x)$. We either terminate with a representation of $Sol_x(E')$ by means of generators or reduce E' to a system S_3 and iterate with an equation $E'' : \phi(x, y) = B''(x)B'y\psi(x, y)$. In the latter case it either can be shortened to a canonical form or $O(n)$ constant candidates for x can be found or a period of x can be found and $Sol_x(E'')$ can be represented by $O(n)$ unary generators $x_j = P^j Q$, P, Q of lengths $O(n)$, P primitive, $j \geq 0$.

If $|B(x)| + |x| \geq |A(x)| > |B(x)|$, then we reduce the problem to solving a singleton+ equation $E' : A(x)y = yB(x)B'x'$ where $|A(x)| = |B(x)| + |B'x'|$.

We assume x' is nontrivial; otherwise a prefix of B' of length $|A(x)| - |B(x)|$ is a period of x.

If $|A(x)| > |B(x)| + |x|$, then $A(x)y = yC(x)x'$ for some prefix x' of x and the number of occurrences of x in $C(x)$ is bigger than this number in $A(x)$. In such case since the word $C(x)x'$ occurs in $A(x)A(x)$ it occurs only in a simple way, i.e. one of x in $C(x)x'$ touches a constant of $A(x)$. This means that in this case we have $O(n^2)$ unary generators for x.

If $|B(x)| > |A(x)| + |x|$, then $A(x)y = yC(x)x'$ for some prefix x' of x and the number of occurences of x in $C(x)$ is smaller than then this number in $A(x)$. In this case we have the same situation as in the previous one.

If $|A(x)| < |B(x)| \leq |A(x)| + |x|$ then we reduce the problem to solving a singleton- equation $E' : A(x)yA'x' = yB(x)$ where $|A(x)| + |A'x'| = |B(x)|$. We assume x' is nontrivial; otherwise a prefix of A' of length $|B(x)| - |A(x)|$ is a period of x.

9.2 $|A(x)|_x > |B(x)|_x$ and $|y| \geq |A(x)| - |B(x)|$

Since y is long enough, then we consider the shortest prefix $B'(x) < B(x)$ such that $|A(x)|_x = |B(x)|_x + |B'(x)|_x$. We follow the reasoning in the previous case to solve $E' : A(x) \cdot y = B(x) \cdot B' \cdot B'(x)$ or reduce to $E'' : B'(x)\phi(x, y) = y\psi(x, y)$ which is strictly shorter in terms of number of occurrences of y.

9.3 $|A(x)|_x > |B(x)|_x$ and $|y| < |A(x)| - |B(x)|$

Since $|y| < |A(x)|$ then either $y = A'(x)A'$ or $y = A'(x)Ax'$ for certain prefix $A'(x) < A(x)$. In the former case A' is a period of x; there is a total number of $O(n)$ possible periods and they yield $O(n^2)$ unary generators that represent $Sol_x(E)$. Therefore, we assume y ends with x'. There are $O(n)$ possible ways to choose $A'(x)$. Fix $A'(x)$.

We consider now the end of the equation. By symmetry $y = x''C'(x)$ for some $C'(x) = [C_i x]_{i=1}^k$. Starting from that point we follow the reasoning in [8]. It is proved there that either we end up with a unary or constants or system S_1 or system S_2 or one special case. The last case can be reduced to $xAy = yBx$ and $A = B$. Then the equation reduces to $(xA)^i y \ldots = y(Ax)^j Ay \ldots$ with $j < i$ and further to $(xA)^{i-j+1} x \ldots = y \ldots$, which is shorter and we consider it in the same manner as the input equation E.

Theorem 3. *Let E be an equations in two variables x, y in canonical form. Then it either reduces to an equation $xAy = yBx$ for some $A, B \in \Sigma^\star$ or it is possible to establish, that $(x, y) \in Sol(E)$ only if x is of the following form.*

- (α) *A set of $O(n)$ candidates of the form $x = (P^iQ)^j P^{i+b} P'$ or for some P, $P'Q$ of length $O(n)$, P primitive, P not a prefix of Q, $b \leq n$ and $i, j \geq 0$ or of the form $x = (P^iQ)^j P'$ for some P, Q, P' of length $O(n)$ and i, $j \geq 0$.*
- (β) *A set of $O(n^3)$ candidates of the form $x = P^iQ$ for some P, Q of lengths $O(n)$, P primitive, P not a prefix of Q and any $i \geq 0$.*

(γ) *A set of* $O(n^2 \log n)$ *candidates* x *of length* $O(n)$.

Therefore combining the theorems presented in the paper, we can find the representation for any equation in two variables. Namely, we reduce the equation to canonized form, establish candidates for one of the solution component, and then solve the original equation by substitution the periodical candidates. The total time to solve the equation is $O(n^5)$.

Theorem 4. *Given an equation* E *in two variables* x, y*, it is possible to find in time* $O(n^5)$ *a polynomial representation of its solutions.*

References

1. Angluin D., Finding pattern common to a set of string, *in Proc. STOC'79*, 130-141, 1979.
2. Charatonik W., Pacholski L., Word equations in two variables, *Proc. IWWERT'91*, LNCS 677, 43-57, 1991.
3. Dąbrowski R., Plandowski W., On word equations in one variable, *Proc. MFCS'02*, LNCS 2420, 212-221, 2002.
4. Crochemore M., Rytter W., *Text Algorithms*, Oxford University Press, 1994.
5. Diekert V., Makanin's algorithm, Chapter 13 in M. Lothaire, *Algebraic Combinatorics on Words*, Cambridge University Press, 2002.
6. Gutierrez C., Satisfiability of word equations with constants is in exponential space, in: *Proc. FOCS'98*, IEEE Computer Society Press, Palo Alto, California.
7. Hmielevskii Yu.I., Equations in free semigroups, *Proc. Steklov Institute of Mathematics*, Amer. Math. So., 107, 1976.
8. Ilie L., Plandowski W., Two-variable word equations, *RAIRO Theoretical Informatics and Applications* **34**, 467-501, 2000.
9. Jaffar J., Minimal and complete word unification, *Journal of the ACM* **37**(1), 47-85, 1990.
10. Karhumäki J., Mignosi G., Plandowski W., The expressibility of languages and relations by word equations, *Journal of the ACM*, Vol. **47**, No 5, May 2000, pp. 483-505.
11. Koscielski A., Pacholski L., Complexity of Makanin's Algorithm, *Journal of the ACM* **43**(4), 670-684, 1996.
12. Koscielski A., Pacholski L., Makanin's algorithm is not primitive recursive, *Theoretical Computer Science* **191**(1-2):145-156, 1998.
13. Makanin G. S., The problem of solvability of equations in a free semigroup, *Mat. Sb.*, **103**(2), 147-236. In Russian; English translation in: *Math. USSR Sbornik* **32**, 129-198, 1977.
14. Neraud J., Equations in words: an algorithmic contribution, *Bull. Belg. Math. Soc.* **1**, 253-283, 1994.
15. Plandowski W., Rytter W., Application of Lempel-Ziv encodings to the solution of word equations, in: *Proc. ICALP'98*, LNCS 1443, 731-742, 1998.
16. Plandowski W., Satisfiability of word equations with constants is in NEXPTIME, *Proc. STOC'99*, ACM Press, 721-725, 1999.
17. Plandowski W., Satisfiability of word equations with constants is in PSPACE, *Proc. FOCS'99*, IEEE Computer Society Press, 495-500, 1999.
18. Razborov A. A., On systems of equations in a free group, *Izv. Akad. Nauk SSSR*, Ser. Mat. 48:779-832, 1984. In Russian; English translation in: Math. USSR Izvestija, 25, 115-162, 1985.