6 Conclusions

We proved that the complexity of deciding if a word equation with constants is solvable is EXPSPACE. On the other side, the known lower bound is NP-hard. The conjecture in [16] is that the problem is NP-complete.

It is interesting to note that the improvement in the complexity of the decision problem presented in this paper gives also an improvement for the upper bound on the length of minimal solutions of word equations, triple exponential as shown above. Rytter and Plandowski [16] showed recently, using compression of solutions, that if the length of a minimal solution to a word equation $\mathcal{E}$ is $L >> |\mathcal{E}|$, then there is a non-deterministic algorithm running in time polynomial in lg L. Unfortunately, due to the current bounds on lengths of minimal solutions (triple exponential in $|\mathcal{E}|$), this is still not enough to improve the above Theorem 5.

It is also interesting to note the result mentioned Remark 1 above, and in general the use of different parameters in the measuring of the complexity of solving equations (the classical is the length of the equation $|\mathcal{E}|$). In this direction, we isolated a reasonable subclass of word equations (those whose number of ocurrences of variables is bounded by a constant) whose decision problem, we conjecture, can be proved tractable.

Finally, it is worth noticing that the improvement of the upper bound presented in this paper works also for the case of word equations with regular constraints (see [18], and the proof in forthcoming [6]).

References

[1] H. Abdulrab, 1987. Résolution d'équations sur les mots: étude et implémentation LISP de l'algoritme de Makanin, Ph.D. dissertation, Univ. Rouen, Rouen.

[2] A.V. Aho, 1990. Algorithms for finding patterns in strings, in Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), Elsevier Sc. Pub., pp. 256-300.

[3] F. Baader, J.H. Siekmann, 1994. Unification Theory, in Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 2, (D. Gabbay et al, ed.) Clarendon Press, Oxford.

[4] D. Benanav, D. Kapur and P. Narendran, 1985. Complexity of matching problems, in Proceedings of the 1st. Int. Conference on Rewriting Techniques and Applications, J.-P Jouannaud, ed., LNCS 202, pp. 417-429.

[5] V.K. Bulitko, 1970. Equations and Inequalities in a Free Group and a Free Semigroup, Tul. Gos. Ped. Inst. Učen. Zap. Mat. Kafedr. Vyp. 2 Geometr. i Algebra (1970), 242-252.

[6] V. Diekert, 1998. Makanin's Algorithm for Solving Word Equations with Regular Constraints, (Preliminary version of the chapter in M. Lothaire, *Algebraic Combinatorics on Words*.) Report Nr. 1998/02, Fakultät Informatik, Universität Stuttgart.

[7] C. Gutiérrez, 1998. Solving Equations in Strings: On Makanin's Algorithm, in Proceedings of LATIN'98, Third Latin American Symposium on Theoretical Informatics, Campinas, Brazil, April 1998, C. Lucchesi, A. Moura, ed., LNCS 1380, pp 358-373.

[8] J.I. Hmelevskiĭ, 1971. Equations in free semigroups, Trudy Mat. Inst. Steklov. 107 (1971). English translation: Proc. Steklov Inst. Math. 107 (1971).

[9] J. Jaffar, 1990. Minimal and Complete Word Unification. Journal ACM, Vol. 37, No.1, January 1990, pp.47-85.

[10] A. Kościelski, L. Pacholski, 1996. Complexity of Makanin's Algorithm. Journal of the ACM, Vol. 43, July 1996, pp. 670-684.

[11] A. Lentin, 1972. Equations in Free Monoids, in Automata Laguages and Programming (M. Nivat ed.), North Holland, 67-85.

[12] G.S. Makanin, 1977. The problem of solvability of equations in a free semigroup. Math. USSR Sbornik **32**(1977) (2), 129-198.

[13] Yu. Matiyasevich, 1968. A connection between systems of word and length equations and Hilbert's Tenth Problem (in russian), English translation in: Seminars in Mathematics, V. A. Steklov Mathematical Institute, 8, 61-67, 1970.

[14] J.P. Pécuchet, 1981. Equations avec constantes et algoritme de Makanin, Thèse de doctorat, Laboratoire d'informatique, Rouen.

[15] G.D. Plotkin, 1972. Building-in equational theories, Mach. Int. 7, 1972, pp. 73-90.

[16] W. Rytter and W. Plandowski, 1998. Application of Lempel-Ziv encodings to the solution of word equations, in Larsen, Proceedings of the 25th ICALP, Aarhus, 1998.

[17] J. Siekmann, 1972. A modification of Robinson's Unification Procedure, M.Sc. Thesis.

[18] K.U. Schulz, 1993. Word Unification and Transformation of Generalized Equations. Journal of Automated Reasoning **11**: 149-184, 1993.

[19] K.U. Schulz, 1990. Makanin's Algorithm for Word Equations: two improvements and a generalization, in Schulz, K.U. ed., Word Equations and related topics, LNCS 572, pp. 85-150.

For the case $\overline{S_i} \subseteq S_{i+1}$ the only difference is that now $\mathsf{left}(y) \preceq \mathsf{left}(x)$ and hence we get a negative sign:

$$|U(S_i)| - |U(S_{i+1})| = -|U(\mathsf{left}(y), \mathsf{left}(x)|.$$

Fact 7. *There are no more than Vn^{V^2} different subbases S which have a convex chain $S, S_2, \ldots, S_n$ landing in x_0 of length n.*

Proof. By enumerating the elements in $\mathcal{D}$, let us say $D_1, \ldots, D_{|\mathcal{D}|}$, and grouping identical summands in the equation in Fact 5, we get:

$$|U(S)| = \alpha_1 D_1 + \cdots + \alpha_{|\mathcal{D}|} D_{|\mathcal{D}|} + |U(\text{col}\,(x_0))|,$$

with $\alpha_1 + \cdots + \alpha_{|\mathcal{D}|} = n$ and $\alpha_j \geq 0$ integers. So the number $|U(S)|$ can take no more than $n^{|\mathcal{D}|}$ values. From Fact 4 we know that S_x is completely determined by x and $|U(S)|$. Hence there are no more than $Vn^{|\mathcal{D}|}$ subbases S with the desired property. Finally just note that $|\mathcal{D}| \leq V^2$.

Fact 8. *The number of different subbases S which have a convex chain $S, S_2, \ldots, S_l$ landing in x_0 of length $l \leq n$ is less than Vn^{V^2+1}.*

Proof. For each length l use Fact 7 and calculate $\sum_{i=1}^{l} Vi^{V^2}$.

Fact 9. Finally, denote by L the length of the longest chain which lands in x_0. From Facts 3 and 8, we have $\frac{M-2N}{V} \leq VL^{V^2+1}$, so we get $L \geq (\frac{M-2N}{V^2})^{\frac{1}{V^2+1}}$. □

Proof of Theorem 2. By Theorem 4, for $n = (\frac{M-2N}{V^2})^{\frac{1}{V^2+1}}$ there is a clean convex chain $S_1, \ldots, S_t, \ldots, S_n$ where S_t is the turning point of the chain. Hence $S_1, \ldots, S_t$ or $S_n, \ldots, \overline{S_t}$ is a *monotone* chain of length $k \geq n/2$.

Let U be a strict unifier of GE. By Lemma 4 we have a domino tower $B_1C_1, \ldots, B_kC_k$ of height k associated to the chain, all of whose words are $B_jC_j = U(\text{col}\,(x_{i_j})) \in \{U(\text{col}\,(x)) : x \in \mathcal{X}\}$. There are V variable bases, so for every i, in $S_i, \ldots, S_{i+V}$ necessarily appear two subbases of the same variable. Because all subbases are different (the chain is clean), $|B_{i+V}| = |U(S_{i+V})| > |U(S_i)| = |B_i|$. So we can apply Proposition 1 to this domino tower to conclude that there is a word B_jC_j $(= U(\text{col}\,(x)$ for some $x \in \mathcal{X})$ of the form P^sQ with P non-empty and $s + 1 > \frac{k}{V|\mathcal{X}|^2} > \frac{n/2}{V(V/2)^2} = \frac{2}{V^3}[\frac{M-2N}{V^2}]^{\frac{1}{V^2+1}}$.

5 Complexity

Theorem 5 *Let $|\mathcal{E}|$ denote the length of the word equation $\mathcal{E}$. The satisfiability problem for word equations is in the following complexity classes:*

DTIME$(2^{2^{O(|\mathcal{E}|^3)}})$, i.e., double exponential deterministic time.

DSPACE$(2^{O(|\mathcal{E}|^3)})$, i.e., exponential deterministic space.

Proof. Let us analyze $\mathsf{Makanin}$. The worst case time-complexity for Gen is exponential in $|\mathcal{E}|$: the number of all possible generalized equations which can be built from $\mathcal{E}$. As for the time-complexity of the Search procedure, the key point is the bound $K(\mathcal{E})$ which is $4|\mathcal{E}|^2(\frac{|\mathcal{E}|^3(p(\mathcal{E})+1)}{2})^{4|\mathcal{E}|^2+1} + 4|\mathcal{E}|$ (see proof of Theorem 3.) On the other side the procedure $\mathsf{Transform}$ takes time bounded by an exponential in the number of boundaries of its input GE (essentially the number of all possible generalized equations which can be built from GE. See [7].) The rest is to calculate how much time does the search take. A rough estimation is the number of nodes in the subtree formed by all the generalized equations with $|BD| < K(\mathcal{E})$ which is exponential in $K(\mathcal{E})$.

As for the space-complexity, using an argument from [6], the space needed to generate non-deterministically with $\mathsf{Transform}$ a new node from a given GE is roughly the number of boundaries of GE. But we need to generate only nodes with less than $K(\mathcal{E})$ number of boundaries. Hence the algorithm uses space in NSPACE$(K(\mathcal{E}))$, and consequently, by Savitch's Theorem, DSPACE$(2^{O(|\mathcal{E}|^3)})$. □

Corollary 1 *If a word equation $\mathcal{E}$ is solvable, then the length of a minimal solution of $\mathcal{E}$ is triple exponential in $|\mathcal{E}|$.*

Proof.(Sketch). The path to a solved node in $\mathcal{T}(\mathcal{E})$ is no longer than the number of nodes in the tree, that is, double exponential. Also, it is not difficult to see that if a node has a solution of length l, its parent has a solution of length no more than $2l$ (see for example the code for $\mathsf{Transform}$ in [7]). Now, because a solution in a solved node has length exponential in $|\mathcal{E}|$, it follows that the length of a solution of $\mathcal{E}$, the root of $\mathcal{T}(\mathcal{E})$, is no more than triple exponential in $|\mathcal{E}|$. □

Remark 1 It is interesting to note (see Theorem 2) that our bounds depend heavily on V (the number of *ocurrences* of variables), as opposed to $|\mathcal{E}|$ (the length of the equation). Moreover, noting that for word equations with V bounded by a fixed constant v_0, their exponent of periodicity is a function polynomial on the length of the equation (for example, check the proof of Lemma 1.3 in Makanin's paper [12]), we can conclude the following: Let V be bounded by a fixed constant v_0 which is not part of the input. Then, the satisfiability problem for word equations where $V \leq v_0$ is in PSPACE, result that is rather interesting when dealing with equations with fixed small number of ocurrences of variables. Moreover, we conjecture that this last problem is tractable, i.e., there is a deterministic polynomial time algorithm which solves it.

2. *Each subbase S_x has its* dual *(the corresponding column in the dual variable), denoted $S_{\bar{x}}$ or $\underline{S}_x$. This pair is called a* boundary equation *and denoted $S_x \sim \underline{S}_x$. Note that if U is a unifier of GE, then $U(S_x) = U(\underline{S}_x)$.*

Definition 5 *Let $S_1, S_2, \ldots, S_n$ be subbases of GE.*

1. *$S_1 \subseteq S_2$ (S_1 is a* suffix *of S_2) iff $S_1 = (b_1, i)$ and $S_2 = (b_2, i)$ and $b_2 \preceq b_1$. Note that the second boundary in both subbases is the same.*

2. *A* monotone suffix chain *in GE is a sequence $S_1, S_2, \ldots S_n$ of subbases with $S_1 \subseteq S_2 \sim \underline{S}_2 \subseteq S_3 \sim \underline{S}_3 \subseteq \ldots \subseteq S_{n-1} \sim \underline{S}_{n-1} \subseteq S_n$.*

3. *A* convex *suffix chain is a sequence $S_1, \ldots, S_t, \ldots, S_n$ such that $S_1, S_2, \ldots, S_t$ and $S_n, S_{n-1}, \ldots, S_t$ are monotone suffix chains.*

   Note that when $t = 1$ or $t = n$ we have chains as in (2), i.e. convex chains generalize monotone chains.

The next lemma—whose proof is an easy check—shows the precise relationship between suffix chains and domino towers.

Lemma 4 *Let $S_1, \ldots, S_k$ be a suffix monotone chain of GE and U a strict unifier of GE. Suppose S_j is a subbase of x_{i_j}. Then*

1. *$U(S_j)$ is a suffix of $U(S_{j+1})$ for all $j = 1, \ldots, n$.*

2. *Define $B_j = U(S_j)$ and C_j such that $U(\mathrm{col}(x_{i_j})) = B_j C_j$. Then the sequence of words $B_1C_1, \ldots, B_kC_k$ is a domino tower of height k.*

We need two more concepts in order to state the next results. A convex chain $S_1, \ldots, S_n$ is said to *land* in a variable base x, if $S_n = (b, i)_x$ with $i = \mathsf{right}(x)$. A convex chain will be called *clean* if each subbase appears just once.

The next proposition has a rather technical proof. It is sketched in [7]. (Also compare [9], Lemmas 3.1, 3.2, and [18], Lemma 5.9 and Appendix.)

Proposition 2 *For each non-empty subbase S of $GE \in \mathcal{T}(\mathcal{E})$, there is a clean convex chain of non-emtpy subbases $S, S_2, \ldots, S_n$ landing in a variable base x.*

We arrived to the key point in our proof. In [9] there is a similar result with a bound logarithmic in the number of boundary equations. The main result of this paper is its improvement to a polynomial bound in the number M of boundaries. The essence of the idea is to use the extra information provided by the fact that the domino towers come from generalized equations, hence the number of possible different B_j's (see Definition 3 and Lemma 4) can be bounded by the square of the number of variable bases.

Theorem 4 *Let $GE(M, N, V)$ be a node of $\mathcal{T}(\mathcal{E})$ which has a strict unifier. Then there is a clean convex chain of length bigger than $(\frac{M-2N}{V^2})^{1/(V^2+1)}$.*

Proof. Let U be a strict unifier of $GE(M, N, V)$.

Fact 1. *Each subbase is completely determined by a base x and a boundary i.*
Proof. Just note that a subbase is of the form $(b, i)_x$ with $b = \mathsf{left}(x)$.

Fact 2. *The number of different non-emtpy subbases of $GE(M, N, V)$ is no less than $M - 2N$.*
Proof. From Fact 1 and Lemma 2(3), in the worst case (each $i \in BD$ occurs only in one base) there are as many different non-emtpy subbases as different boundaries, discounting the left and right boundaries of bases.

Fact 3. *There is a variable base x_0 such that at least $(M - 2N)/V$ non-empty different subbases have a convex chain landing in x_0.*
Proof. By Proposition 2, each subbase has a convex chain landing in a variable base x. But there are V variables bases in GE, hence there is a variable base x_0 with the required property.

Fact 4. *Each subbase S is completely determined by its underlying base and the natural number $|U(S)|$ (the length of the word $U(S)$).*
Proof. From Fact 1, $S = (b, i)_x$ is completely determined by its base x and the boundary i. Now because U is strict, $i \neq j$ iff $|U(b, i)| \neq |U(b, j)|$.

Fact 5. *Let $S_1, \ldots, S_n$ be a convex chain landing in x_0. Then the length of $U(S_1)$ is:*

$$\begin{aligned} |U(S_1)| &= (\sum_{i=1}^{n-1} |U(S_i)| - |U(S_{i+1})|) + \\ &\quad + (|U(S_n)| - |U(\mathrm{col}(x_0))|) + |U(\mathrm{col}(x_0))|. \end{aligned}$$

Proof. Use the telescopic rule.

Fact 6. *Each of the summands in the equation in Fact 5 above is in the set*

$$\begin{aligned} \mathcal{D} &= \{\pm|U(\mathsf{left}(x), \mathsf{left}(y))| : \mathsf{left}(x) \preceq \mathsf{left}(y) \\ &\quad \text{and } x, y \text{ are variable bases}\} \end{aligned}$$

Proof. By definition of convex chain, for each i, $S_i \sim \underline{S}_i \subseteq S_{i+1}$ or $S_i \sim \underline{S}_i \supseteq S_{i\pm1}$. First suppose that the latter case happens, and denote $\underline{S}_i = (\mathsf{left}(x), j)_x$ and $S_{i+1} = (\mathsf{left}(y), j)_y$. Then $\mathsf{left}(x) \preceq \mathsf{left}(y)$ and note that the second boundary j must be the same in both subbases. Now, from the definition of unifier, $U(S_i) = U(\underline{S}_i)$, hence

$$\begin{aligned} |U(S_i)| - |U(S_{i+1})| &= |U(\underline{S}_i)| - |U(S_{i+1})| \\ &= |U(\mathsf{left}(x), j)| - |U(\mathsf{left}(y), j)| \\ &= |U(\mathsf{left}(x), \mathsf{left}(y))|. \end{aligned}$$

The next result can be found in [12] and in [9]. The new bound, exponentially better than Jaffar's [9] is our contribution.

Theorem 2 *Let $GE(M, N, V)$ be a node of $\mathcal{T}(\mathcal{E})$ which has a unifier with exponent of periodicity p. Then $p + 1 \geq \frac{2}{V^3}(\frac{M-2N}{V^2})^{\frac{1}{V^2+1}}$.*

Before proving it, we use it to prove the following:

Theorem 3 Makanin *is correct and terminates.*

Proof. Let $\mathcal{E}$ be a word equation, and define $K(\mathcal{E}) = 4|\mathcal{E}|^2(\frac{|\mathcal{E}|^3(p(\mathcal{E})+1)}{2})^{4|\mathcal{E}|^2+1} + 4|\mathcal{E}|$ where $p(\mathcal{E})$ is as in Theorem 1.

The termination of $\mathsf{Makanin}(\mathcal{E})$ reduces to showing that $\mathsf{Search}(\mathsf{Gen}(\mathcal{E}), K(\mathcal{E}))$ terminates. Observe first that for fixed parameters M, N there are only finitely many different generalized equations $GE(M, N)$. Second, every non-visited node $GE(M, N) \in S$ has a parent with $M \leq K(\mathcal{E})$ (line 7 of Search) and $N \leq 2|\mathcal{E}|$ (Lemmas 1(3) and 2(3)). Finally note that every node has only finitely many children (definition of Gen and Transform). Hence there is a fixed finite number (depending solely on $|\mathcal{E}|$) which bounds the size of S at any stage. Now, because in each loop one more element of S is visited, Search will eventually stop. (Notice that for the union of sets in line 10 of Search we need to check if two generalized equations are the same: this is staightforward.)

Makanin is correct. If $\mathcal{E}$ has no unifier, then by Lemma 3 there is no solved node in $\mathcal{T}(\mathcal{E})$. Hence Search will never reach line 6. Therefore eventually all nodes will be visited and Search will output FAILURE.

Now suppose that $\mathcal{E}$ has a unifier. Then by Theorem 1, it has a unifier with exponent of periodicity $p(\mathcal{E})$. From Lemma 3 it follows that there is a branch in $\mathcal{T}(\mathcal{E})$ ending in a node labelled with a solved generalized equation SGE. By Lemmas 1(1) and 2(1), it follows that each node $GE(M, N, V)$ in the branch has a strict unifier with exponent of periodicity $p' \leq p(\mathcal{E})$. Also from Theorem 2 we have $\frac{2}{V^3}(\frac{M-2N}{V^2})^{1/(V^2+1)} \leq p' + 1$. So we can conclude, using $V \leq N \leq 2|\mathcal{E}|$, that $M \leq V^2[\frac{V^3(p(\mathcal{E})+1)}{2}]^{V^2+1} + 2N \leq K(\mathcal{E})$. Hence all the nodes in the branch eventually will be in S, so Search will visit SGE and check that it is solved (line 5) and return SUCCESS. □

4 Proof of Theorem 2

The proof is essentially the construction of some combinatorial objects called *domino towers* (Definition 3). The general lines of the proof are as follows:

(1) Long domino towers imply big exponent of periodicity of some of its constituent words. This is Proposition 1.

(2) Domino towers can be constructed from a solution to a generalized equation $GE \in \mathcal{T}(\mathcal{E})$. This fact, which uses the relations generated by the boundary sequences of the bases, is stated in Lemma 4 and Proposition 2.

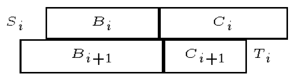
(3) A big number of boundaries in GE implies the existence of long domino towers of those in (2). This is Theorem 4, a counting argument.

Let us begin defining formally the chains of words.

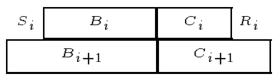
Definition 3 *A* domino tower *is a sequence of words $B_1C_1, \ldots, B_kC_k$ (B_i and C_i non-empty) such that for all $1 \leq i < k$*

1. *There are (possible empty) words S_i such that $B_{i+1} = S_iB_i$.*
2. *There are (possibly empty) words R_i, T_i such that $C_iR_i = C_{i+1}T_i$.*

By condition 2, there are two possible cases for each i:



or



The length of the sequence is called the height *of the domino tower.*

The following result—whose proof can be found in [19]—establishes a relationship between the length of domino towers and the exponent of periodicity of some of its building blocks (the words B_iC_i's).

Proposition 1 *Let $\mathcal{X} = \{X_1, \ldots, X_N\}$ be a set of non-empty words. Suppose the sequence of words $B_1C_1, \ldots B_kC_k$ is a domino tower of height k and each $B_iC_i \in \mathcal{X}$. If for all i, $|B_{i+m}| > |B_i|$, then some word B_tC_t has the form $B_tC_t = P^sQ$, where P is non-empty and $s + 1 \geq \frac{k}{mN^2}$.*

Our next goal will be to generate—from the data of GE—long domino towers whose building blocks (the B_iC_i's) are the variables in $\mathcal{X}$. Then, using the Proposition above, we shall conclude that one variable will have big exponent of periodicity. The next definitions give the elements needed to construct the domino tower (as is shown in Lemma 4 below).

Definition 4 *Let GE be a generalized equation, and let x be a variable base of GE.*

1. *A* subbase of *x, S_x, is a column of the form $(\mathsf{left}(x), i)$ with $i \in E_x$ and $i \neq \mathsf{right}(x)$. If $i = \mathsf{left}(x)$ the subbase is called* empty.

1. *For each pair of duals $x = (x, (e_1, \dots, e_n))$ and $\overline{x} = (x, (\overline{e_1}, \dots, \overline{e_n}))$, and for every sub-index s it holds $U(e_s, e_{s+1}) = U(\overline{e_s}, \overline{e}_{s+1})$. In particular $U(col(x)) = U(col(\overline{x}))$.*

2. *For each constant base bs of label c, $U(col(bs)) = c$.*

U is strict *if $U(i, i+1) \neq \epsilon$ for every $i \in BD$. The* index *of U is the number $|U(b_1, b_M)|$, where b_1 is the first and b_M the last element of BD. The* exponent of periodicity *of U is the maximal exponent of periodicity of the words $U(col(x))$, where x is variable base.*

2 The Algorithm

Makanin's Algorithm consists of the generation of a tree in which all the solutions of $\mathcal{E}$ (if any) can be found. This is acomplished by two main procedures, Gen, whose input is a word equation, and output a finite set of generalized equations, and Transform, whose input is a non-solved generalized equation, and output a finite set of generalized equations. Lemmas 1 and 2 below review their properties. Unfortunately we do not have enough space here to give a thorough description of them (The detailed code and the proofs of Lemmas 1, 2 and 3 which state their properties can be found in [7]).

For a word equation $\mathcal{E}$, define its associated *Makanin's tree*, $\mathcal{T}(\mathcal{E})$, recursively as follows:

- The root of $\mathcal{T}(\mathcal{E})$ is $\mathcal{E}$.
- The children of $\mathcal{E}$ are $\mathsf{gen}(\mathcal{E})$.
- For each node $GE \neq$ root, the set of its children is $\mathsf{Transform}(GE)$.

Lemma 1 (Properties of Gen) *(Compare Lemma 2.3 in [18], Generalization Theorem in [9]). Let $\mathcal{E}$ be a word equation. The following assertions hold:*

1. *If $\mathcal{E}$ has a unifier with exponent of periodicity p then some $GE \in \mathsf{gen}(\mathcal{E})$ has a strict unifier with exponent of periodicity p. Conversely, if some $GE \in \mathsf{Gen}(\mathcal{E})$ has a unifier, then $\mathcal{E}$ has a unifier.*

2. *For each $GE \in \mathsf{gen}(\mathcal{E})$, every boundary is the right or left boundary of a base. Also, every boundary sequence consists precisely of these two boundaries.*

3. *For $GE \in \mathsf{gen}(\mathcal{E})$, the number of bases of GE does not exceed $2|\mathcal{E}|$.*

Lemma 2 (Properties of Transform) *(Compare Theorems 4.7, 4.8 in [18], procedure REDUCE in [9]). Let GE be a non-solved generalized equation (node) of $\mathcal{T}(\mathcal{E})$. The following assertions hold:*

1. *If GE has a strict unifier U with index I and exponent of periodicity p, then $\mathsf{Transform}(GE)$ has an element GE' that has a strict unifier U' with index $I' < I$ and exponent of periodicity $p' \leq p$. Conversely, if an element of $\mathsf{Transform}(GE)$ has a unifier, then GE has a unifier.*

2. *For each $GE' \in \mathsf{Transform}(GE)$, each boundary of GE' occurs in the boundary sequence of some base.*

3. *For $GE' \in \mathsf{Transform}(GE)$, the number of bases of GE' does not exceed the number of those in GE.*

Lemma 3 (Properties of $\mathcal{T}(\mathcal{E})$) *(Compare Corollary 4.9 in [18]). Let $\mathcal{E}$ be a word equation. Then $\mathcal{E}$ has a unifier if and only if $\mathcal{T}(\mathcal{E})$ has a node labelled with a solved generalized equation (i.e. with all variables bases empty.)*

Lemma 3 immediately gives a semi-decision procedure because it is easy to decide if a base is empty or not. But in general, the tree could be infinite. Makanin showed that there exists a constant $K(\mathcal{E})$ which depends only on $\mathcal{E}$ which allows to restrict the search to a finite subtree of $\mathcal{T}(\mathcal{E})$. For a current bound on $K(\mathcal{E})$ see the proof of Theorem 3 below.

```
Makanin($\mathcal{E}$)

1.  $K \leftarrow K(\mathcal{E})$  ▷ bound of the search
2.  $S \leftarrow$ Gen($\mathcal{E}$)
3.  Search($S, K$)
```

```
Search($S, K$)

1.  if all elements of $S$ are visited then
2.      return FAILURE
3.  else
4.      pick a non-visited $GE(M, N) \in S$
5.      if $GE$ is solved then
6.          return SUCCESS
7.      else if $M > K$ then
8.          mark $GE$ as visited; Search($S, K$)
9.      else
10.         $S \leftarrow S \cup$ Transform($GE$)
11.         mark $GE$ as visited; Search($S, K$)
```

3 Correctness and Termination

The cornerstones of Makanin's algorithm are the next two theorems. The first is based on a deep result in word combinatorics, stated by Bulitko in 1970, whose bound was improved by Kościelski and Pacholski [10].

Theorem 1 *Let $\mathcal{E}$ be a word equation which has a unifier. Denote by $p(\mathcal{E})$ the minimum among the exponent of periodicity of all unifiers of $\mathcal{E}$. Then $p(\mathcal{E}) \leq 3|\mathcal{E}|2^{1.07|\mathcal{E}|}$.*

$p(\mathcal{E})^c$ for some constant $c \approx |\mathcal{E}|^2$, But we know that there are only a finite number of nodes with M bounded by that constant. Hence the search is finite.

Summarizing, the complexity of Makanin's Algorithm has three main points: (1) the bound of the exponent of periodicity of word equations, (Theorem 1, which is optimal), (2) the relationship between the number of boundaries of a generalized equation and its exponent of periodicity (Theorem 2, whose bound we improved exponentially in this paper), and (3) possible ways of doing the search. Currently there is no known better way of doing it than by doing a complete search of the subtree of those nodes with number of boundaries $M \leq p(\mathcal{E})^c$, which amounts to a complexity exponential in $p(\mathcal{E})$. This gives a time-complexity for Makanin's Algorithm double exponential in $|\mathcal{E}|$. Also by noticing, as pointed out by Diekert in [6], that the tree can be non-deterministically generated in the space used by the biggest node (hence in our case exponential in $|\mathcal{E}|$), it follows that the satisfiability problem for word equations is in NEXPSPACE, and using Savitch's Theorem, conclude that it is in EXPSPACE.

Many of the arguments in this paper are implicit in Makanin's work and that of his followers. We give an overview of the whole algorithm (along the lines of [7]) in order to make the paper as self-contained as possible. The essential arguments are in Section 4, especially Theorem 4.

1 Basic Concepts and Definitions

Let $\mathcal{C} = \{a_1, \ldots, a_r\}$ be a finite set of *constants*, and $\mathcal{V} = \{v_1, v_2, \ldots\}$ be an infinite set of *variables*. A *word* w over $\mathcal{C} \cup \mathcal{V}$ is a (possibly empty) finite sequence of elements of $\mathcal{C} \cup \mathcal{V}$. The *length* of w, denoted $|w|$, is the length of the sequence. The *exponent of periodicity* of a word w is the maximal number p such that w can be written as uv^pz for some words u, v, z with v non-empty.

A *word equation* $\mathcal{E}$ is a pair (w_1, w_2) of words over $\mathcal{C} \cup \mathcal{V}$, usually written as $w_1 = w_2$. The number $|\mathcal{E}| = |w_1| + |w_2|$ is the *length* of the equation $\mathcal{E}$. Note that in an equation $\mathcal{E}$ only a finite number of variables occur, let us say $\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathcal{V}$. A *unifier* of $\mathcal{E}$ is a sequence $U = (U_1, \ldots, U_n)$ of words over $\mathcal{C} \cup \mathcal{V}$ such that both sides of the equation become graphically identical when we replace all ocurrences of x_i by U_i, for each $i = 1, \ldots, n$. The *exponent of periodicity* of the unifier U is the maximal exponent of periodicity of the words U_i. Usually a unifier U with every $U_i \in \mathcal{C}^*$ is called a *solution*.

The key concept in Makanin's algorithm is that of *generalized equation*. The version presented here follows [7] (compare also [12], [9], [18].)

Definition 1 (Generalized Equation) *A* generalized equation *GE consists of*

1. *Two finite disjoint sets $\mathcal{C}$ and $\mathcal{X}$, the* labels.
2. *A finite linear orderd set $(BD, \preceq)$, the* boundaries.
3. *A finite set BS of* bases. *A* base *bs is a label together with an ordered sequence of boundaries, i.e. $bs = (t, (e_1, \ldots, e_n))$, where $n \geq 2$, $t \in \mathcal{C} \cup \mathcal{V}$, $e_i \in BD$ and $e_i \preceq e_{i+1}$. The bases are subject to the following conditions:*[1]
   (a) *For each variable $x \in \mathcal{X}$, there are exactly two bases with label x, called* duals *(abusing notation denoted by x and $\overline{x}$ respectively). Also, their respective boundary sequences must have the same length.*
   (b) *For each base bs with $t \in \mathcal{C}$, its boundary sequence has exactly two elements and they are consecutives in the order $\preceq$.*

Example 1 *The generalized equation (1) of the introduction is formally: $\mathcal{C} = \{a, b\}$, $\mathcal{X} = \{x, y\}$, $BD = \{1, \ldots, 6\}$ and $BS = \{(a, (2, 3))$, $(a, (24, 5))$, $(b, (3, 4))$, $(x, (1, 2))$, $(x, (5, 6))$, $(y, (1, 3))$, $(y, (4, 6))$.*

Some definitions and conventions to easy the notation: The boundary sequence of a base bs is denoted by E_{bs}. A base $bs = (t, E_{bs})$ is called *constant* if $t \in \mathcal{C}$, and *variable* if $t \in \mathcal{X}$. The first element in E_{bs} is called the *left* boundary of the base, denoted $\mathsf{left}(bs)$, and the last, the *right* boundary, $\mathsf{right}(bs)$.

Letters x, y, z will be used as metavariables for variable bases. Also letters $i, j, \ldots$ will denote boundaries. A pair (i, j) of boundaries with $i \leq j$ is called a *column* of GE. The column of a base x is defined as $\mathrm{col}(x) = (\mathsf{left}(x), \mathsf{right}(x))$. A base is *empty* if $\mathsf{left}(x) = \mathsf{right}(x)$. A generalized equation is *solved* if all its variable bases are empty.

By $GE(M, N, V)$ we will denote the generalized equation with parameters $M = |BD|$, $N = |BS|$ and $V = 2|\mathcal{X}|$ (the number of boundaries, of bases, and of variables bases respectively.)

Definition 2 (Unifier) *Let U be a function that assigns to each column $(i, i+1)$ of GE a word over $\mathcal{C} \cup \mathcal{V}$. Note that U can be extended by concatenation to all columns by $U(i, j) = U(i, k)U(k, j)$ if $i \preceq k \preceq j$ and $U(i, i) = \epsilon$, the empty word. U is called a* unifier *of GE if the following properties are satisfied:*

[1]These are the normalizing conditions: (a) says that we have exactly two ocurrences of each variable. It is necessary also to record all known information about identical subwords in these two ocurrences. This is the role of the boundary sequence, which intuitively is coding: "the word between boundaries e_i and e_j is equal to that between $\overline{e_i}$ and $\overline{e_j}$". (b) says that constants have length 1.

node (the previous bound was logarithmic).

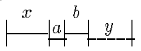
Using (1) and (2) it can be shown that we have to search for solutions only in those nodes of the tree which are "small", and because there is a finite number of them, the procedure succeds.

Finally, let us mention that the only known lower bound for this problem is NP-hard, see e.g. [4]. Alternatively, it follows from the fact that integer programming problems (with non-negative solutions) can be coded as systems of word equations in an alphabet of one letter.

Outline of the Algorithm

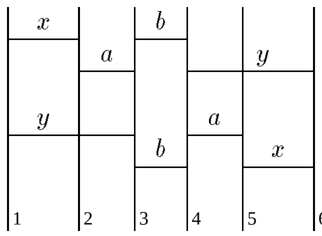
Given a word equation $\mathcal{E}$, Makanin's algorithm can be thought of as the generation of a possibly infinite tree $\mathcal{T}(\mathcal{E})$ whose root is $\mathcal{E}$, and whose nodes are generalized equations which approximate the solutions of $\mathcal{E}$. The tree $\mathcal{T}(\mathcal{E})$ has the property that the word equation $\mathcal{E}$ has a solution if and only if there is a node in the tree which is solved (i.e. as an equation it has no variables, hence it is easily detectable). The idea of a tree in which the nodes are aproximations to the solutions was essentially contained in [11], [15].

Makanin's achievement is having devised a way to restrict the search to a finite part of that tree. To this end, he used the two key results mentioned above. In order to implement this aparently simple idea, a better datatype for word equations was needed.

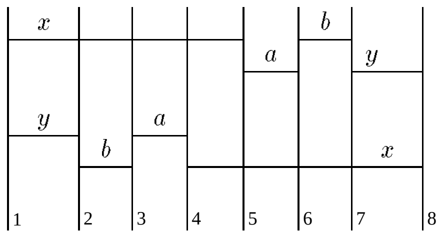
Makanin introduced the concept of *generalized equation*: a graphic representation of approximations to a solution. Consider the equation $xaby = ybax$. The variables x, y represent unknown words. Graphically $xaby$ is represented as 

where the length of the horizontal line in the case of the variables is unknown. The vertical lines are called *boundaries*. An approximation to a solution is essentially to make some decision about how both sides of the equation overlap. See for example two possible overlappings for the equation $xaby = ybax$ in the diagrams below. In general, there may be many such overlappings.

After this initial setup, the algorithm proceeds by replacing equals by equals (elimination of variables) from left to right. For example, in diagram (1), you can replace $y = xa$ (the first two columns) into the y in between boundaries 4 and 6 (note that a new overlapping must be guessed), and so on. Each step will be one new node in the tree. This process in principle increases the number of boundaries. (The number of variables is kept bounded by the trick of replacing the equation by an equivalent system which has exactly two ocurrences of each variable.)



(1)



(2)

Hence a *generalized equation* consists of: (1) A finite set of boundaries, (2) A finite set of generalized variables and constants (called *bases*). A base is defined by a label and a list of relevant boundaries, starting from its left end to its right end, subject to some technical normalization conditions. Standard concepts like solution, exponent of periodicity, etc. for word equations have their counterparts in generalized equations. Note that a generalized equation is a finite object with two parameters: the number M of boundaries and the number N of bases.

It can be proved that all nodes (generalized equations) in $\mathcal{T}(\mathcal{E})$ have a number of bases bounded by $2|\mathcal{E}|$ (due to the fact of having only two ocurrences of each variable). Hence, the "size" of a node (generalized equation) of $\mathcal{T}(\mathcal{E})$ is essentially determined by its number of boundaries M. Also, note that for a given fixed number of bases and boundaries, there are only finitely many different generalized equations.

As we pointed out, the two key results in proving the termination of Makanin's algorithm involve what is called the *exponent of periodicity* of a solution, i.e. the maximal n for which the solution contains a subword of the form w^n. Rephrased in this new language, the results are: (1) There is a constant $p(\mathcal{E})$ with the property that if $\mathcal{E}$ has a solution, it has also a solution with exponent of periodicity $p(\mathcal{E})$ (Kościelski and Pacholski [10] proved that $p(\mathcal{E}) \in 2^{\Theta(|\mathcal{E}|)}$), and (2) Any solution to a node with large number of boundaries must have big exponent of periodicity. Our main contribution is the exponential improvement of this last dependence, i.e. the proof that if a node of $\mathcal{T}(\mathcal{E})$ has M boundaries, then the exponent of periodicity of any solution to that node must be larger than $\Omega(M^{1/c})$, for $c \approx |\mathcal{E}|^2$ for M big enough (The old bound was of the order of lg M, see e.g. [7], [9].)

So the algorithm works as follows: If $\mathcal{E}$ has a solution, it has a solution with exponent of periodicity $n \leq p(\mathcal{E})$. But that solution must show up in a node of $\mathcal{T}(\mathcal{E})$ for whose number of boundaries, M, holds $M^{1/c} \leq n$, that is $M \leq$

Satisfiability of Word Equations with Constants is in Exponential Space

Claudio Gutiérrez
Department of Mathematics
Wesleyan University
Middletown CT 06459, U.S.A.
cgutierrez@wesleyan.edu

Abstract

In this paper we study solvability of equations over free semigroups, known as word equations, particularly Makanin's algorithm, a general procedure to decide if a word equation has a solution. The upper bound time-complexity of Makanin's original decision procedure (1977) was quadruple exponential in the length of the equation, as shown by Jaffar. In 1990 Kościelski and Pacholski reduced it to triple exponential, and conjectured that it could be brought down to double exponential. The present paper proves this conjecture. In fact we prove the stronger fact that its space-complexity is single exponential.

Introduction

Solving equations in equationally defined free algebras (Unification) is a widely used technique in Computer Science, see e.g. [3]. In particular, solving equations in free semigroups, i.e. word equations, is of great interest in e.g. associative rewriting and completion, string unification in PROLOG-3, extensions of string rewrite systems, unification in some theories with associative non-commutative operators, and in symbolic mathematical packages.

The problem of solving word equations was considered at least since the late fifties by A. Markov (see [8]). Partial solutions were known long ago: in the late sixties Hmelevskiĭ [8] solved the problem for equations in three variables, Matiyasevich [13] solve it for the case in which each variable occurs at most twice, and in the seventies Lentin [11], Plotkin [15], and Siekmann [17] gave semi-decision procedures.

In 1977 Makanin [12] solved the problem in its complete generality giving us an algorithm to decide if arbitrary systems of word equations have solutions (the case of systems of equations reduces easily to the case of only one equation). This decision procedure was later extended by Jaffar [9] to give all possible solutions to an equation as well. In the meantime, there has been some work simplifying various aspects of the algorithm and even some implementations [14], [1], [19], [18]. Also, Schulz [19] generalized the result for the case of variables with regular constraints.

Jaffar in [9] calculated an upper bound for the running time of Makanin's algorithm which was four times exponential in the length of the equation. Later Kościelski and Pacholski [10] improved it to non-deterministic triple exponential time. But a more detailed analysis, see [7], [6], shows an upper bound of double exponential space-complexity (i.e. no more than than triple exponential time-complexity).

In the present paper we prove that Makanin's Algorithm has a space-complexity upper bound which is single exponential in the length of the word equation. This result improves the known upper bound complexity from 2-EXPSPACE to EXPSPACE, and proves as a corollary Kościelski-Pacholski's conjecture in [10] about improving the upper bound, using the ideas present in Makanin [12], to double exponential.

Makanin's algorithm is essentially a search for solutions in a tree whose nodes are *generalized equations* (a datatype that generalizes word equations). The generation of the tree is not difficult. The problem is to restrict the search to a finite subtree. This is obtained by using two non trivial results:
(1) For every word equation $\mathcal{E}$ there is a constant $p(\mathcal{E})$, depending solely on $\mathcal{E}$, such that if $\mathcal{E}$ has a solution, then it also has a solution in which each subword of the form w^n (w repeated n times) satisfies $n \leq p(\mathcal{E})$. This is the place where Kościelski and Pacholski reduced the time-complexity by one exponential.
(2) If a node has a big "size", then any solution of it must contain a subword w^n with large n. This is the point on which we concentrate in this paper. In fact, we prove that if a node has a solution, then that solution must contain a subword w^n with n at least polynomial in the "size" of that