# Solving Equations in Strings: On Makanin's Algorithm

Claudio Gutiérrez

Wesleyan University, Middletown, CT 06459, U.S.A.

**Abstract.** We present a further simplification of Makanin's algorithm, still the only known general procedure for solving string equations. We also give pseudo-code, a thorough analysis of its complexity, and complete proofs of correctness and termination.

## 1 Introduction

Checking if two strings are identical is a rather trivial problem. Theoretically it corresponds to solving an equation with both sides constant. For example, are these strings equal?

$$ababababbbbabaaabbbba \stackrel{?}{=} abababab bbababaaabbbba$$

Finding patterns in strings is slightly more complicated. This corresponds to solving equations in strings, one of whose sides is a constant, the text, and the other contains patterns (variables). For example, are there strings $s_1$ and $s_2$ in the alphabet $\{a, b\}$ such that when replacing $x$ by $s_1$ and $y$ by $s_2$ in

$$xxabxby \stackrel{?}{=} abaabababaaabbababababa$$

you get the same string on both sides? Equations of this kind are not difficult to solve. Indeed, many cases of this problem have very efficient algorithms and are the subject of the field of pattern matching (see [2]).

Finding solutions to equations in strings in general (i.e. where both sides contain variables) is a surprisingly difficult problem.[1] Try to find a solution to this simple equation (or show it has none):

$$xaxby \stackrel{?}{=} bybyx$$

Partial solutions to this problem were known long ago: in the seventies Lentin [7], Plotkin [11] and Siekmann [12] gave semi-decision procedures (which give a solution if the equation has one, but if not, could run forever). In 1971, Hmelevskiĭ [6] solved the problem for equations in three variables.

[1]The current bound on its time computational complexity is $O(2^{2^{2^{|\mathcal{E}|}}})$ where $|\mathcal{E}|$ is the length of the equation $\mathcal{E}$. Other anecdotical numbers: The paper in which Makanin presented the algorithm for the first time has 70 pages; later simplified versions (Jaffar, Schulz) have more than 30 pages each. Also there have been at least two Ph.D. theses [1], [10], studying this algorithm, possible simplifications and implementations.

In 1977 Makanin [8] solved the problem in its complete generality giving us the first (and still the only known) algorithm to find solutions for arbitrary string equations. It was later extended by Jaffar [5] 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 [10], [1], [14], [13].

The problem of solving equations in (equationally defined free) algebras is a well-established area in computer science called Unification, with a wide range of applications (see [3]). Solving equations in strings has potential applications in many areas e.g. string unification in PROLOG-3, extensions of string rewrite systems, unification in some theories with associative non-commutative operators, which, due to the current state of the art of the problem, are still of no practical use. This highlights the importance of studying the only currently known general algorithm for solving string equations, its complexity and possible improvements.

Makanin's original paper focused on proving that the question "Does the word equation $\mathcal{E}$ has a solution?" is decidable. He was not interested in either complexity or implementation. Afterwards Pécuchet, Abdulrab, Jaffar and Schulz, among others, simplified some of the technicalities of the algorithm and its proof of correctness and termination, and started to approach the problem from a computational point of view. On the other hand, Jaffar, Kościelski and Pacholski started a systematic study of its complexity. In this paper, we present one more step towards its simplification which also gives better complexity bounds.

First, we introduce a substantially simpler datatype for the concept of *generalized equation* which considerably simplifies the algorithm, making it more understandable and allowing shorter and simpler proofs of the correctness and termination of the algorithm (compare [5], [13]).

Secondly, we introduce the *associated diophantine equations* for an equation, which prune the search tree significantly, and by itself could possibly give another approach to solve string equations.

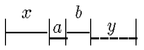
Third, we give a thorough analysis of the complexity of the algorithm, obtaining smaller bounds (although still in the same complexity class) than Jaffar's [5] (on which [13] and [9] are based).

Last, but not least, we include complete proofs of correctness and termination, and present for the first time pseudo-code ready to be implemented in any language. Finally let us say that our presentation owes much to Schulz [13], particularly in Section 4. We use the terms *word* and *string* interchangeably.

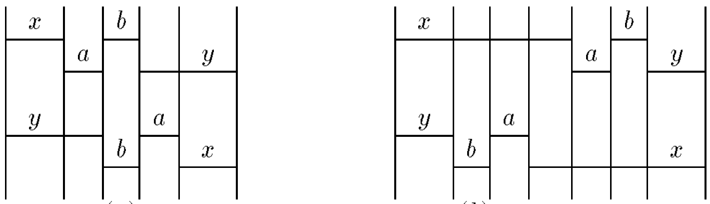
## 2 Word Equations: basic concepts and examples

**Definition 1.** 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 $\mathcal{E}$ 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 occurrences 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 $U_i$.

It is very convenient to have a graphical idea of word equations. Consider the equation $xaby = ybax$. The variables $x, y$ represent unknown words. Graphically, $xaby$ will be represented as 

where the length of the horizontal line in each case is unknown, except those of the constants which are always of unit length. The vertical lines will be called *boundaries*. In this representation, the equation has a solution if there is a way of consistently overlapping both sides such that the words (represented by segments of horizontal lines) between boundaries are the same. In general, there may be many such overlappings. Below we show two possibilities among many others (we draw the variables in different levels in order to highlight the limits of each variable):



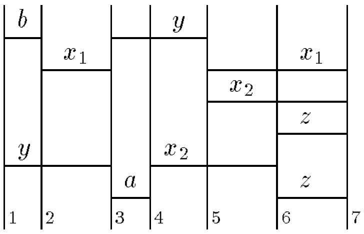
$(a)$ $(b)$

The next step is to replace equals by equals (elimination of variables) from left to right, e.g. in case (a) we can replace $y = xa$ in the other occurrence of $y$. After this, we have to guess the order of some boundaries again, and so on.

This example contains the basic idea at the heart of the algorithm: (i) guess an ordering of the boundaries, that is, which comes first, which second, and so on, for all the initial boundaries on both sides of the equation, and (ii) proceed from left to right replacing equals by equals.

But this naive recipe has some problems: (1) the number of occurrences of some variables starts growing after replacement, (2) what to do in cases where there is no evident replacement (cf. example (b) above), and (3) you can go on forever (cf. the equation $xa = ax$). That is why a more elaborate idea is needed.

The starting point is to build, for each word equation, an arrangement like the above. It is convenient (to avoid problem 1) to work with an equivalent system of equations in which each variable occurs no more than twice. Note that this is always possible. Consider for example the equation $bxyx = yaxz$. It is equivalent to $bx_1yx_1 = yax_2z$ and $x_1 = x_2$. One possible arrangement of boundaries will look like




(1)

Note that $x_1 = x_2$ can be easily expressed in the same arrangement (see the last two columns). It also will be convenient to have exactly two copies of each variable in the arrangement (that is why we put one more copy of $z$ on top of it in the last column). This presentation of a word equation is the starting point of Makanin's algorithm.

## 3 Generalized Equations

The main concept in Makanin's algorithm is that of *Generalized Equation*, essentially a datatype that codifies arrangements as those shown above. The version presented here differs somewhat from the classical ones [8], [5], [13], allowing a considerably simpler algorithm.

**Definition 2.** A *generalized equation* $GE$ consists of

1. Two finite sets $\mathcal{C}$ and $\mathcal{X}$, the *labels*.
2. A finite linear ordered set $(BD, \preceq)$, the *boundaries*.
3. A finite set $BS$ of *bases*. A *base* $bs$ has the form $(t, (e_1, \ldots, e_n))$, where $n \geq 2$, $t \in \mathcal{C} \cup \mathcal{X}$, and $E_{bs} = (e_1, \ldots, e_n)$ is a sequence of boundaries ordered by $\preceq$.

subject to the following conditions:[2]

(C1) For each $x \in \mathcal{X}$, there are exactly two bases with label $x$, called *duals*, and (abusing notation) denoted by $x$ and $\bar{x}$ respectively. Also, their respective boundary sequences $E_x, E_{\bar{x}}$ must have the same length.

(C2) For each base $bs$ with $t \in \mathcal{C}$, the boundary sequence $E_{bs}$ has exactly two elements and they are consecutives in the order $\preceq$.

Some definitions and conventions to ease the notation: 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 $\textsc{left}(bs)$, and the last, the *right* boundary, $\textsc{right}(bs)$.

[2] The data above is intended to represent arrangements like 1. So we must impose some additional conditions. (C1) says that we have exactly two occurrences of each variable. The boundary sequences are to record known information about identical columns in these pairs of variables. Intuitively they are coding 'the word between column $e_i$ and $e_j$ is equal to that between $\bar{e}_i$ and $\bar{e}_j$'. (C2) says that constants have length 1.

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 $GE$. Columns $(i, i)$ are called *empty*; columns $(i, i+1)$ are called *indecomposable*. The column of a base $bs$ is defined as $\mathrm{col}(bs) = (\mathrm{LEFT}(bs), \mathrm{RIGHT}(bs))$. A base is *empty* if its column is empty. A generalized equation is *solved* if all its variable bases are empty.

**Definition 3.** A *unifier* of $GE$ is a function $S$ that assigns to each indecomposable column of $GE$ a word over $\mathcal{C} \cup \mathcal{V}$ (extend it by concatenation to all non-empty columns of $GE$) with the following properties:

1. For each constant base $bs$ of label $c$, $S(\mathrm{col}(bs)) = c$.
2. For every pair of dual variables $x, \bar{x}$, and for every $e_j \in E_x$, $S(e_1, e_j) = S(\bar{e}_1, \bar{e}_j)$ (recall $\bar{e}_1, \bar{e}_j \in E_{\bar{x}}$). In particular $S(\mathrm{col}(x)) = S(\mathrm{col}(\bar{x}))$.

$S$ is *strict* if $S(i, i+1)$ is non-empty for every $i \in BD$. The *index* of $S$ is the number $|S(b_1, b_M)|$, where $b_1$ is the first and $b_M$ the last element of $BD$. The *exponent of periodicity* of $S$ is the maximal exponent of periodicity of the words $S(\mathrm{col}(x))$, where $x$ is a variable base.

**Definition 4.** For a generalized equation $GE$, and $c \in \mathcal{C}$, the *associated system of linear diophantine equations*, $L(GE, c)$, is defined by:

1. A variable $Z_i$ for each indecomposable column $(i, i+1)$ of $GE$.
2. For each pair of dual variables bases $(x, (e_1, \ldots, e_n))$ and $(x, (\bar{e}_1, \ldots, \bar{e_n}))$ define $(n-1)$ equations, for $j = 1, \ldots, n-1$:

$$\textstyle\sum_{e_j \preceq i \prec e_{j+1}} Z_i \ = \sum_{\bar{e}_j \preceq i \prec \bar{e}_{j+1}} Z_i$$

3. For each constant base $(t, (i, i+1))$, define the equation $Z_i = 1$ if $t = c$ and $Z_i = 0$ if $t \neq c$.

**Lemma 5.** *If $GE$ has a unifier, then $L(GE, c)$ is solvable for each $c \in \mathcal{C}$.*

*Proof.* Let $S$ be a unifier of $GE$ and $c \in \mathcal{C}$. Define $Z_i = |S(i, i+1)| - D_c$ where $D_c$ is the number of occurrences of constants different from $c$ in the word $S(i, i+1)$. Using the fact that $S$ is a unifier, it is easy to check that this is a solution to $L(GE, c)$.

Checking solvability of systems of linear diophantine systems is decidable, although expensive (NP-complete). A generalized equation $GE$ whose system $L(GE, c)$ is solvable for all $c \in \mathcal{C}$ is called *admissible*.

### 3.1 The translation from word equations to generalized equations.

Given a word equation $\mathcal{E}$, we can obtain (possibly) many generalized equations by a procedure like that of Sect.2: for each possible overlapping of both sides of $\mathcal{E}$, proceed as in examples in Sect.2, and then check admissibility. The detailed description of the algorithm and the checking of the properties below is straightforward, so we will omit it.

**Lemma 6.** *There exists an algorithm* GEN *which for every word equation* $\mathcal{E}$ *outputs a finite set* $\text{GEN}(\mathcal{E})$ *of generalized equations with the following properties:*

1. *$\mathcal{E}$ has a unifier with exponent of periodicity $p$ if and only if some $GE \in \text{GEN}(\mathcal{E})$ has a strict unifier with exponent of periodicity $p$.*
2. *For each $GE \in \text{GEN}(\mathcal{E})$, every boundary is the right or left boundary of a base. Also, every boundary sequence contains exactly these two boundaries.*
3. *For $GE \in \text{GEN}(\mathcal{E})$, the number of bases of $GE$ does not exceed $2|\mathcal{E}|$.*
4. *Every $GE \in \text{GEN}(\mathcal{E})$ is admissible.*

As an illustration let us show an element in $\text{GEN}(bxyx = yaxz)$, the one corresponding to the arrangement (2) in Sect.2. The corresponding generalized equation is: $\mathcal{C} = \{a, b\}$, $\mathcal{X} = \{x_1, x_2, y, z\}$, $BD = \{1, \ldots, 7\}$ and $BS = \{(b, (1, 2)),$ $(a, (3, 4)),$ $(x_1, (2, 3)),$ $(x_1, (5, 7)),$ $(y, (3, 5)),$ $(y, (1, 3)),$ $(x_2, (4, 6)),$ $(x_2, (5, 7)),$ $(z, (6, 7)), (z, (6, 7))\}$.

## 4 The transformation Algorithm

Now we know that every word equation $\mathcal{E}$ has a set of generalized equations $\text{GEN}(\mathcal{E})$ equivalent to it in the sense of Lemma 6. Hence the problem is reduced to work on generalized equations.

Given a generalized equation, the basic idea of the algorithm—as was shown in Sect.2—is the succesive replacement of equal variables from left to right. The naive idea is to pick the leftmost and biggest variable (called the carrier) and transport all its columns to the position of its dual. Unfortunately sometimes not all its columns can be moved without losing essential information (see example (b) in Sect.2). What is to be done? Answering this question is the motivation of the following two definitions. Let us fix throughout this section a non-solved generalized equation $GE = (\mathcal{C}, \mathcal{X}, BD, BS)$.

**Definition 7.** The *carrier* of $GE$, denoted $x_c$, is the non-empty variable base with smallest left boundary. If there is more than one, $x_c$ is the one with largest right boundary. If there is still more than one, choose one among them randomly. We will denote $l_c = \text{LEFT}(x_c)$ and $r_c = \text{RIGHT}(x_c)$.

The *critical boundary* of $GE$ is defined as $cr = \min\{\text{LEFT}(y) : r_c \in \text{col}(y)\}$ if the set is non-empty, and $cr = r_c$ if not.

**Definition 8.** Let $bs$ be base of $GE$, $bs$ is not the carrier. Then

1. $bs$ is *superfluous* if $\text{col}(bs) = (i, i) \prec l_c$.
2. $bs$ is *transport* if $l_c \preceq \text{LEFT}(bs) \prec cr$ or $\text{col}(bs) = (cr, cr)$.
3. $bs$ is *fixed* if it is not superfluous and not transport.

Note that all variable bases with $\text{LEFT}(x) \prec l_c$ are empty by definition of the carrier. Also, each base—except the carrier—is exactly one of these: superfluous, transport or fixed, depending on what region of the diagram below its left boundary is:

| $b_1$ | $l_c$ | $cr$ | $r_c$ | $b_M$ |
|---|---|---|---|---|
| superfluous | transport | fixed | fixed | |

To ilustrate these definitions with the examples in Sect.2. In (a) the carrier is $y$, $l_c = 1$, $r_c = cr = 3$. In (b) the carrier is $x$, $l_c = 1$, $cr = 4$ and $r_c = 5$.

Now we know *what* bases should be moved: the transport bases. It is time to define *where* to move them. The next definition points to this problem.

**Notation.** For each boundary $l_c \preceq i \preceq r_c$ in $BD$, let us introduce a new symbol $i^{tr}$ (which will indicate the place where the boundary $i$ should go) and denote $\mathrm{tr}(E_x) = \mathrm{tr}(e_1, \dots, e_n) = (e_1^{tr}, \dots, e_n^{tr})$.

**Definition 9.** A *print* of $GE$ is a linear order $\preceq$ on the set $BD \cup \{i^{tr} : i \in [l_c, r_c]\}$ satisfying the following conditions:

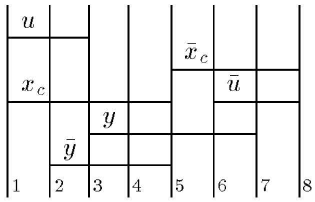
1. $\preceq$ extends the order of $BD$ and $j^{tr} \prec k^{tr}$ for $l_c \preceq j \prec k \preceq r_c$.
2. $\mathrm{tr}(E_c) = \bar{E}_c$. (The structure of the carrier overlaps its dual.)
3. If $x$ is transport, $\bar{x}$ fixed, then if for some $e_i \in E_x$, $e_i^{tr} = \bar{e}_i$, then $\mathrm{tr}(E_x) = E_{\bar{x}}$. (The order $\preceq$ is consistent with the boundary sequence information.)
4. If $(c, (i, j))$ is a constant base, then $i, j$ (and also $i^{tr}, j^{tr}$ if $i, j \in [l_c, r_c]$) are consecutive in the order $\preceq$. (Constants are preserved.)

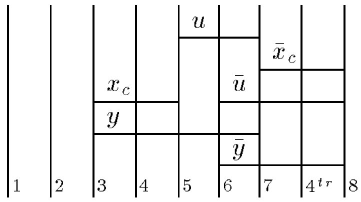
Finally we are ready to present the heart of the Makanin's algorithm, the procedure TRANSPORT, which corresponds to the replacement of equals by equals from left to right. Once we have the classification of bases and a print (a guess about where each boundary to be transported will go), things are relatively straightforward: leave the fixed bases untouched and move all transport bases. All the intrincacies of the algorithm will then rely on the carrier and its dual (lines 1-5 and 7-8). We need some notation to describe it: for a set of boundaries $A$, $E_x \cap A$ will denote the subsequence of $E_x$ of the elements in $A$. Similarly, $E_x \cup A$ is the supersequence of $E_x$ obtained by adding the elements of $A$ in the corresponding order. $E_c$ is a shorthand for $E_{x_c}$. Let $\preceq$ be a print of $GE$.

TRANSPORT($GE, \preceq$)
1. **if** $cr \prec r_c$ **then**
2. $\quad E_c \leftarrow E_c \cap \{i \in BD : cr \preceq i\}$
3. $\quad \bar{E}_c \leftarrow \bar{E}_c \cap \{i \in BD : cr^{tr} \preceq i\}$
4. **else** $/*\ cr = r_c$
5. $\quad E_c \leftarrow \mathrm{tr}(E_c) \quad /*$ recall $\mathrm{tr}(E_c) = \bar{E}_c$
6. **for** each transport base $bs \in BS$ **do**
7. $\quad E_c \leftarrow E_c \cup \{i\ : i \in E_{bs}$ and $cr \prec i\}$
8. $\quad \bar{E}_c \leftarrow \bar{E}_c \cup \{i^{tr} : i \in E_{bs}$ and $cr \prec i\}$
9. $\quad E_{bs} \leftarrow \mathrm{tr}(E_{bs})$
10. **for** each variable base $x \in BS$ with $\mathrm{col}(x) = \mathrm{col}(\bar{x})$ **do**
11. $\quad E_x \leftarrow (\mathrm{RIGHT}(x), \mathrm{RIGHT}(x))$
12 $\quad E_{\bar{x}} \leftarrow E_x$
13. $BS \leftarrow \{bs \in BS : cr \preceq \mathrm{LEFT}(bs)\}$
14. $BD \leftarrow \{i \in BD : i \in E_{bs}$ and $bs \in BS\}$
15. $\mathcal{X} \leftarrow \{x \in \mathcal{X} : (x, E) \in BS\}$
16. $\mathcal{C} \leftarrow \{c \in \mathcal{C} : (c, E) \in BS\}$
17. **return** $(\mathcal{C}, \mathcal{X}, BD, BS)$.

*Remarks.* First note that fixed bases are left untouched. Lines 1-5 process the carrier and its dual: If $cr \prec r_c$, then $x_c, \bar{x}_c$ are shrunk. If $cr = r_c$, $x_c$ is transported completely onto $\bar{x}_c$. Lines 6-9 process transport bases: add new boundary equations to the carrier (lines 7-8) and give the new position (line 9). Lines 10-12 optimize: once two duals overlap, they are not necessary anymore. Lines 13-16 eliminate superfluous bases, boundaries, and labels respectively.

An example will help. In the diagrams below, on the left there is a generalized equation $GE$ (suppose $E_{\bar{y}} = (2, 4, 5)$, $E_y = (3, 5, 7)$ and $E_z = (\text{LEFT}(z), \text{RIGHT}(z))$ for all the other bases) and on the right $\text{TRANSPORT}(GE, \preceq)$ for a print $\preceq$ with $1^{tr} = 5$, $2^{tr} = 6$, $3^{tr} = 7$ and $5^{tr} = 8$. Note that $4^{tr}$ introduces a new boundary.




The critical boundary of $GE$ is 3. Because $3 \prec 5 = r_c$, $x_c$ and $\bar{x}_c$ are shrunk. Next, the transport bases $u, \bar{y}$ are moved to their new positions. Note that when moving $\bar{y}$ we lose information, e.g. that $\bar{y}$ and $y$ have a common segment, column $(3, 4)$. The algorithm keeps track of it by adding the boundary 4 to $E_c$ and $4^{tr}$ to $\bar{E}_c$, i,e. the new $E_c = (3, 4, 5)$ and $\bar{E}_c = (6, 4^{tr}, 8)$ (lines 7-8 in the code), and hence the segments continue to be equal through the 'boundary equation' which says that columns $(3, 4)$ and $(6, 4^{tr})$ are the same. Observe that $u$ produces no new boundary equation and the fixed bases $\bar{u}, y$ remained untouched. Finally, we can delete the boundaries to the left of $cr = 3$ (line 14).

The next lemma follows easily from the definitions and the code.

**Lemma 10.** $\text{TRANSPORT}(GE, \preceq)$ *is a generalized equation.*

Note that a generalized equation has only finitely many different prints. So the following procedure returns a finite set of generalized equations.

```
TRANSF(GE)
    S ← ∅
    Print ← the set of all prints of GE
    for each print ≼ ∈ Print do
        GE' ← TRANSPORT(GE, ≼)
        if GE' is admissible then
            S ← S ∪ {GE'}
    return S
```

**Lemma 11.** *1. If $GE$ has a strict unifier $S$ with index $I$ and exponent of periodicity $p$, then* $\text{TRANSF}(GE)$ *has an element $GE'$ which has a strict unifier $S'$ with index $I' < I$ and exponent of periodicity $p' \leq p$.*

*2. If an element of* $\text{TRANSF}(GE)$ *has a unifier, then $GE$ has a unifier.*

*Proof.* (Sketch). (1) Because $S$ is a unifier, in particular $S(\mathrm{col}(x_c)) = u_1 \cdots u_s = S(\mathrm{col}(\bar{x}_c))$, with $u_i \in \mathcal{V} \cup \mathcal{C}$. Hence, we have a function $f$ from the boundaries in $[l_c, r_c] \cup [\bar{l}_c, \bar{r}_c]$ to $\{1, 2, \ldots, s\}$ with $S(i,j) = u_{f(i)} \cdots u_{f(j)-1}$. Extend it by defining $f(i^{tr}) = f(i)$ for $i \in [l_c, r_c]$. Then the following order $\preceq$ in $BD' = BD \cup \{i^{tr} : i \in [l_c, r_c]\}$ is a print of $GE$:

- For $i, j \in BD$, define $i \preceq j$ iff $i \preceq_{BD} j$.
- For $l_c \preceq_{BD} j \preceq_{BD} r_c$, define $\bar{l}_c \preceq j^{tr} \preceq \bar{r}_c$.
- For $i, j \in BD'$ and $\bar{l}_c \preceq i, j \preceq \bar{r}_c$, define $i \preceq j$ iff $f(i) \leq f(j)$.

Define $GE' = \textsc{Transport}(GE, \preceq)$. In order to construct the unifier $S'$ of $GE'$, for $i \in BD'$: define $S'(i, i{+}1) = u_{f(i)} \cdots u_{f(i+1)-1}$ if $\bar{l}_c \preceq i \prec \bar{r}_c$, and $S'(i, i{+}1) = S(i, i+1)$ otherwise. Then $S'$ is a unifier of $GE'$ and strict if $S$ is strict. Also, from $l_c \prec_{BD} cr$ it follows that the index of $S'$ < index of $S$. Also, the exponent of periodicity of $S'$ does not exceed that of $S$ since $S'(\mathrm{col}(x))$ is a suffix of $S(\mathrm{col}(x))$ for any base $x$ of $GE'$.

For (2), suppose $S'$ is a unifier of some $GE' \in \textsc{Transf}(GE)$. Let $i, j$ be consecutive in $BD$. Define $S(i,j)$ as $S'(i,j)$ if $i, j \in BD'$; as $S'(i^{tr}, j^{tr})$ if $i^{tr}, j^{tr} \in BD'$; as $c$ if there is a constant base $(c, (i,j))$ in $GE$, and finally as the empty word if there is no base $(c, (i,j))$ in $GE$ and $i$ or $j$ is not in $BD'$. It can be checked that $S$ is a unifier of $GE$.

## 5 The Final Algorithm

Given 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 $\textsc{Gen}(\mathcal{E})$. (see lemma 6)
- For each node $GE$ (not the root), the set of its children is $\textsc{Transf}(GE)$.

**Theorem 12.** *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.*

*Proof.* Suppose $\mathcal{E}$ has a unifier. By Lemma 6 there is an element of $\textsc{Gen}(E)$ which has a strict unifier with some index $I_E$. By induction on the depth of the node, using lemma 11, it can be proven that $\mathcal{T}(E)$ has a branch, with each node labelled by a strict-unifiable generalized equation and the index decreases for every child. Since the index is non-negative, the branch is finite; hence there must be a node $GE$ for which $\textsc{Transf}$ does not apply. The only possibility is that $GE$ is solved.

On the other direction, induct again using lemmas 6 and 11.

Theorem 12 immediately gives a semi-decision procedure: examine all nodes of $\mathcal{T}(\mathcal{E})$ to find out if $\mathcal{E}$ has a solution. But in general, the tree could be infinite. Here comes the kernel of Makanin's algorithm: there exists a finite number $K_{\mathcal{E}}$ that bounds the number of nodes we have to visit.

```
MAKANIN(ℰ)
    K ← K_ℰ   /* bound of the search
    S ← GEN(ℰ)
    SEARCH(S, K)
```

```
SEARCH(S, K)
    if all elements of S are marked then
        return FAILURE
    else
        pick a non-marked GE = (𝒞, 𝒳, BD, BS) ∈ S
        if GE is solved then
            return SUCCESS
        else if |BD| > K then
            mark GE ; SEARCH(S, K)
        else
            S ← S ∪ TRANSF(GE)
            mark GE ; SEARCH(S, K)
```

## 6 Correctness and Termination

From now on, let us fix a word equation $\mathcal{E}$, and let $\mathcal{T}(\mathcal{E})$ be its associated Makanin tree. All generalized equations will be nodes of $\mathcal{T}(\mathcal{E})$. For $GE = (\mathcal{C}, \mathcal{X}, BD, BS)$ with parameters $M = |BD|$, $N = |BS|$, $V = 2|\mathcal{X}|$ (the number of variables bases) write $GE(M, N, V)$.

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 recently by Kościelski and Pacholski [9].

**Theorem 13.** *If a word equation $\mathcal{E}$ is unifiable, then there it has a unifier with exponent of periodicity $p_{\mathcal{E}} \leq 3|\mathcal{E}|2^{1.07|\mathcal{E}|}$.*

A simple proof (for a weaker bound) which gives a good intuition of why, if $\mathcal{E}$ is unifiable, there must be unifiers of this kind can be found in [8]. The next theorem is due to Makanin.

**Theorem 14.** *If $GE(M, N, V)$ is a node of $\mathcal{T}(\mathcal{E})$, then the exponent of periodicity of all its strict unifiers exceeds $\frac{2\log_V(M/N-2)}{V^3}$.*

The proof of this result is tricky, and the rest of the paper is devoted to it. Allow me, before proving it, to show why Makanin's algorithm works.

**Theorem 15.** Makanin *is correct and terminates.*

*Proof.* Let $\mathcal{E}$ be a word equation. Termination of Makanin($\mathcal{E}$) reduces to show that Search(Gen($\mathcal{E}$), $K_{\mathcal{E}}$) terminates.

Define $p = 3|\mathcal{E}|2^{1.07|\mathcal{E}|}$ and $K_{\mathcal{E}} = 2^{4p|\mathcal{E}|^3 \lg 2|\mathcal{E}| + \lg 2|\mathcal{E}|} + 4|\mathcal{E}|$. Now observe that there are only finitely many generalized equations $GE(M, N)$ with fixed parameters $M, N$, and that at every stage in SEARCH, every element $GE(M, N) \in S$ has $M \leq K_{\mathcal{E}}$ (line 7 of SEARCH) and $N \leq 2|\mathcal{E}|$ (Lem.6(3) and line 13 in TRANSPORT). Hence, because in each loop one more element of $S$ is marked, SEARCH will eventually stop.

MAKANIN is correct. If $\mathcal{E}$ has no unifier, then by Thm.12 there is no solved node in $\mathcal{T}(\mathcal{E})$. Hence SEARCH will never reach line 6. Therefore eventually all nodes will be marked and SEARCH will output FAILURE.

Now suppose that $\mathcal{E}$ has a unifier. Then by Thm.13, it has a unifier with exponent of periodicity less than $p$. From the proof of Thm.12 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 6(1) and 11(1), it follows that each node $GE(M, N, V)$ in the branch has a strict unifier with exponent of periodicity $p' \leq p$. Also from Thm.14 we have $\frac{2\log_V(M/N-2)}{V^3} \leq p'$. So we can conclude, using $V \leq N \leq 2|\mathcal{E}|$, that $M < 2^{0.5pV^3 \lg V + \lg N} + 2N \leq K_E$. 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.

Now let us prove Thm.14. The general lines are as follows: (i) From each $GE \in \mathcal{T}(\mathcal{E})$ we can obtain (using the relations generated by the boundary sequences of the bases) certain chains of words. This is Prop.25, whose proof is long and very technical; (ii) By a counting argument it follows that a large number of boundaries in $GE$ produces long chains of words. This is Prop.27; (iii) Combine (ii), using Lem.20, with a combinatorics result (Prop.17) establishing a relation between long chains of words and high exponent of periodicity.

Let us begin with the formal definitions of those chains of words.

**Definition 16.** 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 $C_iR_i = C_{i+1}T_i$.


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

The following result (whose proof can be found in [14]) establishes a relation between the length of a domino tower and the exponent of periodicity of some of its words.

**Proposition 17.** *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 $s + 1 \geq \frac{k}{mN^2}$.*

So we need to generate long domino towers whose building blocks are elements of $\mathcal{X}$. In this way, one variable will have a large exponent of periodicity.

**Definition 18.** Let $GE$ be a generalized equation, $x$ a variable base of $GE$.

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

$GE'$ will denote $\text{TRANSPORT}(GE, \preceq)$. Also $\text{LEFT}'$ is the corresponding function in $GE'$, etc. So, if $S = (\text{LEFT}(x), i)_x$ is a subbase of $GE$, then $S'$ will denote its 'image' in $GE'$, i.e. $(\text{LEFT}'(x), i^{tr})_x$ if $x$ is transport and $(\text{LEFT}'(x), i)_x$ otherwise. In case $S'$ is empty or it is not a subbase of $GE'$ (i.e. $x$ becomes empty in $GE'$ or $x = x_c$ and $i \prec cr$ in $GE$) then $S$ is called a *terminal* subbase.

**Definition 19.** Let $S_1, S_2, \ldots, S_n$ be subbases of $GE$.

1. Let $S_1 = (b_1, i)$ and $S_2 = (b_2, i)$ be subbases with the same second boundary. $S_1$ is a *suffix* of $S_2$ if $b_2 \preceq b_1$. We write $S_1 \subseteq S_2$.
2. A *(monotone) suffix chain* in $GE$ is a sequence $S_1, S_2, \ldots S_n$ of subbases with
$$S_1 \subseteq S_2 \sim \bar{S}_2 \subseteq S_3 \sim \bar{S}_3 \subseteq \ldots \subseteq S_{n-1} \sim \bar{S}_{n-1} \subseteq S_n$$
We will denote this chain by $S_1 \subseteq^* S_n$.
3. A *convex* suffix chain is a sequence $S_1, \ldots, S_t, \ldots, S_n$ such that $S_1 \subseteq^* S_t$ and $S_t \sim \bar{S}_t$ and $\bar{S}_t \supseteq^* S_n$. We write $S_1 \subseteq^* \supseteq S_n$. Note that when $t = 1$ or $t = n$ we have chains as in 1. (i.e. convex chains generalize monotone chains.)

The next lemma (whose proof is an easy check) shows the relation between suffix chains and domino towers.

**Lemma 20.** *Let $S_1, \ldots, S_k$ be a suffix chain of $GE$ and $U$ a 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 $U(col(x_{i_j})) = B_j C_j$. Then the sequence of words $B_1 C_1, \ldots, B_k C_k$ forms a domino tower of height $k$.*

The next lemmas have long (but straightforward) proofs by exhaustion of all possibles cases of the bases involved (transport, fixed, carrier, its dual). We will do one case to give the flavor of the technique.

**Lemma 21.** *Let $S_x \subseteq S_y$ in $GE$ and $S_x, S_y$ be non-terminal. Then*
1. *If $y$ is the carrier or its dual, then $S'_x \subseteq^* S'_y$ or $S'_x \supseteq^* S'_y$ in $GE'$.*
2. *If $y$ is neither the carrier nor its dual, then $S'_x \subseteq^* S'_y$ in $GE'$.*

*Proof.* Case (1) Let $S_x = (b_1, i)_x \subseteq (b_2, i)_y = S_y$.

(a) $y$ is the carrier. So $l_c = b_2 \preceq b_1$. Suppose first that $x$ is transport, i.e. $b_1 \prec cr$. It holds that $S'_x = (b_1^{tr}, i^{tr}) \supseteq (cr^{tr}, i^{tr}) \sim (cr, i) \supseteq (cr, i) = S'_y$ in $GE'$. Now, suppose $x$ is fixed, i.e. $cr \preceq b_1$. We have $S'_x = (b_1, i) \subseteq (cr, i) = S'_y$ in $GE'$. Note that this also works if $x$ is the dual of the carrier.

(b) Now, assume $y$ is the dual of the carrier, that is $\bar{l}_c = b_2 \preceq b_1$. Note that $x$ cannot be the carrier now. So let us suppose $x$ is neither the carrier nor its dual. Because the dual of the carrier is fixed (always), $x$ must be fixed too ($cr \prec \bar{l}_c = b_2 \preceq b_1$). Hence we have $S' = (b_1, i) \subseteq (cr^{tr}, i) = S'_y$ or $S'_x \supseteq S'_y$, depending on whether $b_1 \preceq c^{tr}$ or $c^{tr} \preceq b_1$.

**Lemma 22.** *Let $S_x \subseteq S_y \sim S_{\bar{y}} \subseteq S_z$ in $GE$ and $S_x$ be non-terminal.*

1. *If $z$ is the carrier or its dual and $y$ is the carrier or its dual, then $S_z$ is non-terminal and $S'_x \subseteq^* S'_z$ in $GE'$.*
2. *If $z$ is the carrier or its dual, $S_z$ is non-terminal, and $y$ is neither the carrier nor its dual, then $S'_x \subseteq^* \supseteq S'_z$.*
3. *If $z$ is neither the carrier nor its dual, $S_z$ is non-terminal, then $S'_x \subseteq^* S'_z$.*

**Lemma 23.** *Suppose $S_x \subseteq^* S_z$ in $GE$, and $S_x$, $S_z$ are non-terminal. Then*

1. *If $z$ is the carrier or its dual, $S'_x \subseteq^* \supseteq S'_z$ in $GE'$.*
2. *If $z$ is neither the carrier nor its dual, $S'_x \subseteq^* S'_z$ in $GE'$.*

*Proof.* A simultaneous induction for (1) and (2) on the length of the chain. The base cases are lemmas 21 and 22.

**Lemma 24.** *(convex chains) Let $S_x$ and $S_z$ be subbases of $GE$ which are non-terminal. Suppose there is a convex chain from $S_x$ to $S_z$ in $GE$. Then there is a convex chain from $S'_x$ to $S'_z$ in $GE'$.*

*Proof.* Induction on the length of $S_1 \subseteq^* S_t \sim \bar{S}_t \supseteq^* S_n$. Consider the turning point $t$ and the possibles cases for the chains $S_1 \subseteq^* S_t$ and $S_n \subseteq^* \bar{S}_t$.

All the preceding work was done in order to prove the next lemma. Extend the notation $S \subseteq B$ to allow $B$ to be a constant base, i.e. $S = (b, i) \subseteq (l, r) = B$ iff $i = r$ and $b \preceq l$. Similarly for $S \supseteq B$.

**Proposition 25.** *For each non-empty subbase $S$ of $GE \in \mathcal{T}(\mathcal{E})$, there is a convex chain $S = S_1, \ldots, S_n, B$ with $B = col(bs)$ for some base $bs$ of $GE$.*

*Proof.* Induction on the depth of structure of $\mathcal{T}(\mathcal{E})$. For elements $GE \in \textsc{Gen}(E)$, the only subbases are of the form $(\textsc{left}(x), \textsc{right}(x))$ (Lem.6(2)), so the statement is trivially true. Now suppose the statement is valid for $GE$. We will prove it for $GE' = \textsc{Transport}(GE, \preceq)$.

Let $S'$ be a non-empty subbase of $GE'$. It is an image of a subbase $S$ in $GE$, so by hypothesis there is a convex chain $S = S_1, \ldots, S_n, B = \mathrm{col}(bs)$ in $GE$ and $S$ is non-terminal because $S'$ is its image. If $S_n$ is non-terminal, applying Lem.24 it follows that $S'_1, \ldots, S'_n, B'$ is convex and $B' = \mathrm{col}'(bs)$.

So suppose $S_n$ is terminal. Let $t$ $(1 < t \leq n)$ be the smallest index such that $S_t, \ldots, S_n$ are all terminal subbases. So $S_{t-1}$ is non-terminal, and by lemma 24 there is a convex chain from $S_1'$ to $S_{t-1}'$ in $GE'$. Let us show that it can be completed to end with col$(bs)$ for some base $bs$. Denote $\bar{S}_{t-1} = (l_y, j)_y$ and $S_t = (l_z, j)_z$.

If $z$ is neither the carrier nor its dual, then $\bar{S}_{t-1}' = (l_y, j^{tr})_y \supseteq (j^{tr}, j^{tr})_z =$ col$'(z)$ in $GE'$ if $y$ is transport. If $y$ is fixed then $cr \preceq l_y$, hence $\bar{S}_{t-1} \subseteq S_z$ and so $S_1 \subseteq^* S_n$ because the chain is convex. Then $\bar{S}_{t-1}' = (l_y, j)_y \subseteq (cr, j) \sim (cr, j^{tr}) \supseteq (j^{tr}, j^{tr})_z$ in $GE'$.

If $z$ is the carrier then $j \preceq cr$ ($S_t$ is terminal), so $y$ is transport. Now, if $S_{t+1} = B$ constant, it must be transport, so $\bar{S}_{t-1}' = (l_y, j^{tr})_y \supseteq B'$. If $S_{t+1} = S_u$ with $u$ a variable, $u$ can neither be the carrier nor its dual (because $S_{t+1}$ is also terminal), hence $\bar{S}_{t-1}' = (l_y, j^{tr})_y \supseteq (j^{tr}, j^{tr})_z$ in $GE'$. If $z$ is the dual of the carrier the analysis is similar.

A *strict* convex chain is one in which each subbase appears just once. (Note that a subbase is characterized by its column *and* its base.)

**Lemma 26.** *Let $S_0 = (b_0, i)$ be a fixed subbase of $GE(M, N, V)$. The number of different subbases $S$ of $GE$ such that there is a strict convex chain $S = S_1, \ldots, S_j = S_0$ of length $j \leq k$ is less than $V^k$.*

*Proof.* Consider the set of chains $S_1, \ldots, S_j$ with $S_i \subseteq S_{i+1}$ or $S_i \supseteq S_{i+1}$ for each $i$. Clearly it contains the set of strict convex chains. For $j = 1$ note that if $(b, i) \subseteq (b_0, i)$ or $(b, i) \supseteq (b_0, i)$, $b$ must be a left boundary of a variable base, and there are less than $V$ such boundaries different from $b_0$. The general case follows by simple combinatorics, i.e. there are no more than $V^k$ chains of that type.

**Proposition 27.** *Let $GE(M, N, V) \in \mathcal{T}(\mathcal{E})$. Then there is a strict convex chain of length bigger than $\log_V(M/N - 2)$ in $GE$.*

*Proof.* A subbase is of the form $(b, i)_x$ with $i \in E_x$. There are at least $(M - 2N)$ different non-empty subbases (the number of boundaries—line 14 of Transport—minus the left and right boundaries of each base). By Prop.25, for each such subbase $S$ there is a convex chain $S = S_1, \ldots, S_n, B =$ col$(bs)$ for some base $bs$. But there are $N$ bases in $GE$, hence there is a base $bs_0$, such that at least $(M-2N)/N$ subbases have a convex chain to $bs_0$ (which is strict because all subbases were different). Now, by lemma 26, $V^k > (M - 2N)/N$, hence $k > \log_V(M/N - 2)$.

**Proof of Theorem 14.** By Prop.27, for $n = \log_V(M/N - 2)$ there is a strict convex chain $S_1, \ldots, S_t, \ldots, S_n$. Hence $S_1, \ldots, S_t$ or $S_n, \ldots, \bar{S}_t$ is a (monotone) chain of length $k > n/2$.

Let $U$ be a strict unifier of $GE$ and consider the domino tower $B_1C_1, \ldots, B_kC_k$ of height $k$ associated to the chain as in Lem.20, all of whose words $B_jC_j = U(\text{col}(x_{i_j}))$ are in $\{U(\text{col}(x)) : x \in \mathcal{X}\}$. There are $V$ variable bases, so, for every $i$ two subbases of the same variable must appear in $S_i, \ldots, S_{i+V}$. Now because all subbases are different (strict chain), $|B_{i+V}| = |U(S_{i+V})| > |U(S_i)| = |B_i|$. We conclude from Prop.17 that there is a word $B_jC_j$ of the form $P^sQ$ with $P$ non-emtpy and $s + 1 > \frac{k}{V|\mathcal{X}|^2} > \frac{2\log_V(M/N-2)}{V^3}$.

## 7 Final Remarks

There are three key points in estimating the time complexity of MAKANIN: first, bounds on $p_{\mathcal{E}}$, the exponent of periodicity of word equations. Thm.13 is almost optimal: it is known that $p_{\mathcal{E}} \geq 2^{0.29|\mathcal{E}|}$ (see [9]); The second point is bounds on $K_{\mathcal{E}}$, the depth of the search. Jaffar's estimate [5] was of the order $16N^{15}p(6|\mathcal{E}|^2)^2(2N)^{32p(6|\mathcal{E}|^2)N^5}$. We improved it to $2^{0.5p(|\mathcal{E}|)V^3 \lg V + \lg N}$ where $p(x) = 3x2^{1.07x}$ and $V \leq N \leq 2|\mathcal{E}|$; The third point, bounds on SEARCH. A rough bound is the number of all different $GE(M, N)$ with $N \leq 2|\mathcal{E}|$ and $M \leq K_{\mathcal{E}}$. There seems to be much room for improvement on these last two points. Also a finer analysis of TRANSPORT would imply a clearer picture of the interplay among prints, associated diophantine equations, and the kind of search needed. Rounding, the current time complexity bound on MAKANIN is triple exponential in $|\mathcal{E}|$.

Finally, let us say that it is easy to add two lines to TRANSPORT in order to get explicit solutions: we need an extra variable $U$ (a list of pair of boundaries) for each pair of duals to keep track of the value of the original variable. The proof of Lem.11 tells how they have to be updated.

## 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 Artif. Int. and Logic Prog., Vol. 2, (D. Gabbay et al, ed.), Clarendon Press, Oxford.
4. 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.
5. J. Jaffar, 1990. *Minimal and Complete Word Unification*, Journal ACM, Vol. 37, No.1, January 1990, pp.47-85.
6. J.I. Hmelevskiĭ, 1971. *Equations in free semigroups*, Trudy Mat. Inst. Steklov. 107 (1971). English translation: Proc. Steklov Inst. Math. 107 (1971).
7. A. Lentin, 1972. *Equations in Free Monoids*, in Automata Laguages and Programming (M. Nivat ed.), North Holland, 67-85.
8. G.S. Makanin, 1977. *The problem of solvability of equations in a free semigroup*, Math. USSR Sbornik **32**(1977) (2), 129-198.
9. A. Kościelski, L. Pacholski, 1996. *Complexity of Makanin's Algorithm*, Journal of the ACM, Vol. 43, July 1996, pp. 670-684.
10. J.P. Pécuchet, 1981. *Equations avec constantes et algoritme de Makanin*, Thèse de doctorat, Laboratoire d'informatique, Rouen.
11. G.D. Plotkin, 1972. *Building-in equational theories*, Mach. Int. 7, 1972, pp. 73-90.
12. J. Siekmann, 1972. *A modification of Robinson's Unification Procedure*, M.Sc. Thesis.
13. K.U. Schulz, 1993. *Word Unification and Transformation of Generalized Equations*, Journal of Automated Reasoning **11**: 149-184, 1993.
14. K.U. Schulz, 1990. *Makanin's Algorithm for Word Equations: two improvements and a generalization*, in LNCS 572, pp. 85-150.