**Basic Research in Computer Science**

**BRICS RS-00-13 Klíma & Srba: Matching Modulo Associativity and Idempotency is NP-Complete**

# Matching Modulo Associativity and Idempotency is NP-Complete

**Ondřej Klíma**
**Jiří Srba**

**BRICS Report Series** **RS-00-13**

**ISSN 0909-0878** **June 2000**

**Copyright © 2000, Ondřej Klíma & Jiří Srba.**
**BRICS, Department of Computer Science**
**University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work**
**is permitted for educational or research use**
**on condition that this copyright notice is**
**included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.**
**Copies may be obtained by contacting:**

**BRICS**
**Department of Computer Science**
**University of Aarhus**
**Ny Munkegade, building 540**
**DK–8000 Aarhus C**
**Denmark**
**Telephone: +45 8942 3360**
**Telefax: +45 8942 3255**
**Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:**

`http://www.brics.dk`
`ftp://ftp.brics.dk`
**This document in subdirectory** `RS/00/13/`

# Matching Modulo Associativity and Idempotency is NP–complete*

Ondřej Klíma[1] and Jiří Srba[2]

[1] Faculty of Science MU, Dept. of Mathematics, Janáčkovo nám. 2a, 662 95 Brno, Czech Republic, klima@math.muni.cz
[2] BRICS***, Department of Computer Science, University of Aarhus, Ny Munkegade bld. 540, DK-8000 Aarhus C, Denmark, srba@brics.dk

**Abstract.** We show that AI–matching (AI denotes the theory of an associative and idempotent function symbol), which is solving matching word equations in free idempotent semigroups, is NP-complete.
*Note: this is a full version of the paper [9] and a revision of [8].*

*Keywords:* unification, word rewriting, idempotent semigroups, complexity

## 1 Introduction

Solving equations appears as an interesting topic in several fields of computer science. Many areas such as logic programming and automated theorem proving exploit solving equations, and syntactic (Robinson) unification is a typical example of it. An important role is played also by semantic unification, which allows to use several function symbols with additional algebraic properties (e.g. associativity, commutativity and idempotency). Makanin (see [16]) shows that the question whether an equation in a free monoid has a solution is decidable. It can be generalized in the way that existential first-order theory of equations in a free monoid with additional regular constraints on the variables is decidable [19]. For an overview of unification theory consult e.g. [4].

AI–matching is one example of semantic unification where the considered equational theory is of one associative and idempotent

* The paper is supported by the Grant Agency of the Czech Republic, grant No. 201/97/0456 and by grant FRVŠ 409/1999.
*** Basic Research in Computer Science, Centre of the Danish National Research Foundation.

function symbol. In this paper we focus on a subclass of word equations which we call pattern equations. Pattern equations are word equations where we have on the left-hand side just variables and on the right-hand side only constants. In the usual interpretation, AI–matching is a AI–unification of systems of equations where all right-hand sides are variable–free. However, we can eliminate constants on the left-hand sides by adding new equations and so pattern equations are as general as AI–matching.

Many practical problems such as speech recognition/synthesis lead to this kind of equations. This work has been inspired by the papers [12] and [13] where the basic approach – syllable-based speech synthesis – is in assigning prosody attributes to a given text and segmentation into syllable segments. We examine the solvability of word equations in the variety of all idempotent semigroups, which we call stuttering equations. Their name comes from practical motivation. For example in speech recognition the speaker sometimes stutters some words and we would like to eliminate this effect and enable the correct variables assignment even in the case of stuttering. Therefore we allow to eliminate multiple occurrences of the same subword into only one occurrence, which can be modelled by the identity $x^2 = x$. The decidability of the satisfiability problem (even in the general case) is a consequence of the local finiteness of free idempotent semigroups and an exponential upper bound on the length of a minimal solution can be given ([6]). A polynomial time decision procedure for the word problem in a free idempotent semigroup can be also easily established. Recently it has been proved in [3] that AI–unification remains decidable even if additional uninterpreted function symbols in the equations are allowed.

Unification problems for the AI–theory have been investigated e.g. in [1, 2, 20], however, the complexity questions were not answered. In this paper we prove that there is a polynomial bound on the length of a minimal solution in the case of stuttering pattern equations and thus we show that the satisfiability problem is in NP. The proof exploits the confluent and terminating word rewriting system for idempotent semigroups by Siekmann and Szabo (see [21]). This means that the identity $p = q$ holds in a free idempotent semigroup if and only if the words $p$ and $q$ have the same normal form

w.r.t. the rewriting system $\{xx \to x \mid \mathsf{C}(x) \neq \emptyset\} \cup \{uvw \to uw \mid \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$, where $\mathsf{C}(y)$ denotes the set of letters of $y$.

Showing a reduction from 3–SAT to our problem, we prove its NP–completeness. This is a more general result than Theorem 7 in the paper by Kapur and Narendran [11], where they prove NP–hardness for AI–matching, where additional uninterpreted function symbols are allowed. In our proof we use only one associative and idempotent function symbol. NP–hardness means that the problem is probably difficult. One of the ways how to solve the problem is to use heuristic algorithms. They are the current field of interest in speech recognition.

## 2 Basic definitions

An idempotent semigroup (also called a band) is a semigroup where the identity $x^2 = x$ is satisfied. Let $\mathcal{C}$ be a finite set. We define a binary relation $\to \subseteq \mathcal{C}^* \times \mathcal{C}^*$ such that $uvvw \to uvw$ for $u, v, w \in \mathcal{C}^*$ and let $\sim$ be its symmetric, reflexive and transitive closure, i.e. $\sim := (\to \cup \to^{-1})^*$. Then the identity $p = q$ holds in a free band over $\mathcal{C}$ if and only if $p \sim q$ (completeness of the equational logic).

Let $\mathcal{C}$ be a finite set of *constants* and $\mathcal{V}$ be a finite set of *variables* such that $\mathcal{C} \cap \mathcal{V} = \emptyset$. A *word equation* $L = R$ is a pair $(L, R) \in (\mathcal{C} \cup \mathcal{V})^* \times (\mathcal{C} \cup \mathcal{V})^*$. A *system of word equations* is a finite set of equations of the form $\{L_1 = R_1, \ldots, L_n = R_n\}$ for $n > 0$. A *solution (in a free idempotent semigroup)* of such a system is a homomorphism $\alpha : (\mathcal{C} \cup \mathcal{V})^* \to \mathcal{C}^*$ which behaves as an identity on the letters from $\mathcal{C}$ and equates all the equations of the system, i.e. $\alpha(L_i) \sim \alpha(R_i)$ for all $1 \leq i \leq n$. Such a homomorphism is fully established by a mapping $\alpha : \mathcal{V} \to \mathcal{C}^*$. A solution is called *non-singular*, if $\alpha(x) \neq \epsilon$ for all $x \in \mathcal{V}$, where $\epsilon$ denotes the empty word. Otherwise we will call it *singular*. We say that a system of word equations (in a free idempotent semigroup) is *satisfiable* whenever it has a solution. For the introduction into word equations and combinatorics on words you can see [14], [15] and [18]. We refer to word equations in a free idempotent semigroup as *stuttering equations*.

In what follows we will use a uniform notation. The set $\mathcal{C} = \{a, b, c, \ldots\}$ denotes the alphabet of constants and $\mathcal{V} = \{x, y, z, \ldots\}$ stands for variables (unknowns) with the assumption that $\mathcal{C} \cap \mathcal{V} = \emptyset$.

We will use the same symbol $\alpha$ for the mapping $\alpha : \mathcal{V} \to \mathcal{C}^*$ and its unique extension to a homomorphism $\alpha : (\mathcal{C} \cup \mathcal{V})^* \to \mathcal{C}^*$. The empty word will be denoted by $\epsilon$ and the length of a word $w$ by $|w|$.

We exploit the fact that the word problem in a free band is decidable (see [7] and its generalization [10]), which is a consequence of the next lemma. Let $w \in \mathcal{C}^+$. We define

$\mathsf{C}(w)$ – the set of all letters that occur in $w$,
$0(w)$ – the longest prefix of $w$ in $\mathrm{card}(\mathsf{C}(w)) - 1$ letters,
$1(w)$ – the longest suffix of $w$ in $\mathrm{card}(\mathsf{C}(w)) - 1$ letters.

Let also $\overline{0}(w)$ resp. $\overline{1}(w)$ be the letter that immediately succeeds $0(w)$ resp. precedes $1(w)$.

**Lemma 1 ([7]).** *Let $p, q \in \mathcal{C}^+$. Then $p \sim q$ if and only if $\mathsf{C}(p) = \mathsf{C}(q)$, $0(p) \sim 0(q)$ and $1(p) \sim 1(q)$.*

It is obvious that if a stuttering equation system has a solution then it has always infinitely many solutions, which we show in the following lemma.

**Lemma 2.** *Let $\{L_1 = R_1, \ldots, L_n = R_n\}$ be a stuttering equation system with a solution $\alpha$. Then also any $\beta$ that satisfies $\alpha(x) \sim \beta(x)$ for all $x \in \mathcal{V}$ (we simply write $\alpha \sim \beta$) is a solution.*

*Proof.* Immediate. □

This gives an idea that we should look just for the solutions where $\alpha(x)$ is the shortest word in the $\sim$ class for each variable $x$. We introduce a size of a solution $\alpha$ as $\mathrm{size}(\alpha) := \max_{x \in \mathcal{V}} |\alpha(x)|$ and say that $\alpha$ is *minimal* iff for any solution $\beta$ of the system we have $\mathrm{size}(\alpha) \leq \mathrm{size}(\beta)$. Given a stuttering equation system it is decidable whether the system is satisfiable because of the local finiteness of free idempotent semigroups. The following lemma just gives a precise exponential upper bound on the size of a minimal solution.

**Lemma 3 ([6]).** *Let $k = \mathrm{card}(\mathcal{C}) \geq 2$ and let $\{L_1 = R_1, \ldots, L_n = R_n\}$ be a stuttering equation system. If the system is satisfiable then there exists a solution $\alpha$ such that $\mathrm{size}(\alpha) \leq 2^k + 2^{k-2} - 2$.*

In general it can be shown that there are stuttering equation systems such that all their solutions are at least exponentially large w.r.t.

the cardinality of the set $\mathcal{C}$ . Consider the following sequence of equations: $x_1 = a_1$ and $x_{i+1} = x_i a_{i+1} x_i$ for a sequence of pairwise different constants $a_1, a_2, \ldots$. For any solution $\alpha$ of the system we have that $|\alpha(x_i)| \geq 2^i - 1$.

In this paper we focus on a special kind of word equations which we call *pattern equations*.

**Definition 1.** *A* pattern equation system *is a set* $\{X_1 = A_1, \ldots, X_n = A_n\}$ *where* $X_i \in \mathcal{V}^*$ *and* $A_i \in \mathcal{C}^*$ *for all* $1 \leq i \leq n$. *A* solution *of a pattern equation system is defined as in the general case.*

*Remark 1.* In the usual interpretation AI–matching allows constants to appear also on the left-hand sides, i.e. the equations are of the type $X = A$ where $X \in (\mathcal{V} \cup \mathcal{C})^*$ and $A \in \mathcal{C}^*$. However, we can w.l.o.g. consider only pattern equations, since an equation of the type $X_1 a X_2 = A$ where $a \in \mathcal{C}$ can be transformed into $X_1 x X_2 = A$ and $x = a$, where $x$ is a new variable.

Two natural decidability problems (Pattern-Equation and Non-Singular-Pattern-Equation problem) appear in this context and are defined below.

**Definition 2.** *Given a pattern equation system* $\{X_1 = A_1, \ldots, X_n = A_n\}$ *as an instance of the* Pattern-Equation *problem, the task is to decide whether this system has a solution. If we require the solution to be non-singular we call the problem* Non-Singular-Pattern-Equation.

The Pattern-Equation problem for a single *stuttering* pattern equation $X = A$ is trivial since it is always solvable: $\alpha(x) = A$ for all $x \in \mathcal{V}$. On the other hand a system is not always solvable: e.g. $\{x = a,\ x = b\}$ has no solution.

We give an example of a pattern equation system and demonstrate its solutions.

*Example 1.* Let us have the following system where $\mathcal{C} = \{a, b\}$, $\mathcal{V} = \{x, y, z\}$ and the pattern equations are $\{yxy = aba,\ yz = a\}$. A singular solution exists $\alpha(x) = aba,\ \alpha(y) = \epsilon,\ \alpha(z) = a$, however, there is also a non-singular solution $\beta(x) = bab,\ \beta(y) = a,\ \beta(z) = a$ since $ababa \sim aba$ and $aa \sim a$. We have also another non-singular solution $\gamma(x) = b,\ \gamma(y) = a,\ \gamma(z) = a$.

Our goal is to show that a minimal solution of a stuttering *pattern* equation system is of a polynomial length. This implies that the problem of deciding whether a stuttering pattern equation system is satisfiable is in NP.

## 3 Rewriting system for idempotent semigroups

In this section we summarize several properties of the rewriting system by Siekmann and Szabo in [21] and prove some technical lemmas. First of all we have to give some definitions and results concerning rewriting systems as it can be found e.g. in [5].

A *rewriting system* $R$ over $\mathcal{C}$ is a subset of $\mathcal{C}^* \times \mathcal{C}^*$. The elements of $R$ will be called *rules*. Having such a system $R$ we can define a *rewrite relation* $\rightarrow \subseteq \mathcal{C}^* \times \mathcal{C}^*$ in the following way:

$$\forall p, q \in \mathcal{C}^* : p \rightarrow q \text{ iff } \exists (u, v) \in R, s, t \in \mathcal{C}^* : p = sut, q = svt.$$

The elements $(u, v)$ of $R$ will be often written as $u \rightarrow v$. For a word $q \in \mathcal{C}^*$ we write $q \not\rightarrow$ iff there is no $q'$ such that $q \rightarrow q'$ and we say that $q$ is in a *normal form*. We define the set of normal forms of $p \in \mathcal{C}^*$ as $\langle p \rangle = \{q \mid p \rightarrow^* q \not\rightarrow\}$. We say that $R$ (resp. the relation $\rightarrow$) is *terminating* iff there is no infinite sequence $p_1, p_2, p_3, \ldots \in \mathcal{C}^*$ such that $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow \ldots$. The system $R$ (resp. the relation $\rightarrow$) is *confluent* iff $\forall p, p_1, p_2 \in \mathcal{C}^* \exists q \in \mathcal{C}^*$ :

$$\text{if } (p \rightarrow^* p_1 \text{ and } p \rightarrow^* p_2) \text{ then } (p_1 \rightarrow^* q \text{ and } p_2 \rightarrow^* q).$$

The system $R$ (resp. the relation $\rightarrow$) is *locally confluent* iff $\forall p, p_1, p_2 \in \mathcal{C}^* \exists q \in \mathcal{C}^*$ :

$$\text{if } (p \rightarrow p_1 \text{ and } p \rightarrow p_2) \text{ then } (p_1 \rightarrow^* q \text{ and } p_2 \rightarrow^* q).$$

The following lemma shows the relationship between confluence and local confluence.

**Lemma 4 ([5]).** *Let $R$ be a terminating rewriting system. Then $R$ is confluent if and only if $R$ is locally confluent.*

It is easy to see that if $R$ is a confluent and terminating rewriting system, then a word $p \in \mathcal{C}^*$ has exactly one normal form, i.e. $\langle p \rangle = \{q\}$ for some $q$, and in such a case we simply write $\langle p \rangle = q$.

*Example 2.* Let $\{xx \to x \mid x \in \mathcal{C}^*, \mathsf{C}(x) \neq \emptyset\}$ be a rewriting system over $\mathcal{C}$. Then this system is terminating but it is not confluent. For $p = ababcbabc$ we have $p = (ab)(ab)cbabc \to abcbabc$ and $p = a(babc)(babc) \to (ab)(ab)c \to abc$ where $abcbabc$ and $abc$ are in normal form. It is easy to see that $\langle p \rangle = \{abc, abcbabc\}$.

In this paper we will exploit the rewriting system by Siekmann and Szabo in [21].

**Lemma 5 ([21]).** *The rewriting system* $\{xx \to x \mid x \in \mathcal{C}^*, \mathsf{C}(x) \neq \emptyset\} \cup \{uvw \to uw \mid u, v, w \in \mathcal{C}^*, \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$ *is confluent and terminating. Moreover for* $p, q \in \mathcal{C}^*$ *we have* $p \sim q$ *if and only if* $p$ *and* $q$ *have the same normal form w.r.t. the system.*

We will refer the rewriting system $\{xx \to x \mid \mathsf{C}(x) \neq \emptyset\} \cup \{uvw \to uw \mid \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$ as $RS$. Since $RS$ contains two different types of rewriting rules we denote $RS_1$ the rewriting system $\{xx \to x \mid \mathsf{C}(x) \neq \emptyset\}$ and $RS_2$ the rewriting system $\{uvw \to uw \mid \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$. The corresponding rewrite relations are denoted $\to$, $\to_1$ resp. $\to_2$ and for a word $p \in \mathcal{C}^*$ the set of its normal forms is denoted by $\langle p \rangle$, $\langle p \rangle_1$ resp. $\langle p \rangle_2$.

If we want to investigate the complexity issues for stuttering equations, the first question we have to answer is the complexity of checking whether some identity holds in a free band. We will show that the word problem (i.e. the problem whether $p \sim q$) can be decided in polynomial time by using the rewriting system $RS$. If we note that a string of length $k$ contains $\mathcal{O}(k^2)$ substrings (each substring is identified by its beginning and its length) we get that each reduction of $RS$ can be done in polynomial time. Since every reduction decreases the length of the word, we have a polynomial time decision algorithm for the word problem in a free band.

We know that $RS$ is confluent and terminating. Our goal in this section is to show that $RS_2$ is also a confluent and terminating rewriting system and that $\langle p \rangle = \langle \langle p \rangle_2 \rangle_1$.

We define a rewrite relation $\to_{2l} \subset \to_2$ such that $suvwt \to_{2l} suwt$ if and only if $|v| = 1$ and $\mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$. It is easy to see that $\to_2 \subseteq \to_{2l}^*$ and hence $\to_{2l}^* = \to_2^*$. The last relation we will use is $\to_{2m} \subset \to_2$, consisting of all rules that leave out the maximal number of letters in the following sense. Let $\mathsf{H}(w)$ resp. $\mathsf{T}(w)$ mean the first

resp. the last letter of the word $w$. We write $suvwt \to_{2m} suwt$ iff $\emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$ and the following conditions hold:

**(i)** $\mathsf{C}(s_0u) \neq \mathsf{C}(wt_0)$, for any suffix $s_0$ of $s$ and any prefix $t_0$ of $t$ (including empty $s_0$ or $t_0$, but not both)
**(ii)** $u = 0(u)\mathsf{T}(u)$
**(iii)** $w = \mathsf{H}(w)1(w)$.

Note that if $suvwt \to_{2m} suwt$ then the last letter of $s$ and the first letter of $t$ (if they exist) are new and different letters[1]. Also note that $\mathsf{T}(u)$ is the only occurrence of this letter in $u$ and we can write it as $u = 0(u)\overline{0}(u)$. Similarly $w = \overline{1}(w)1(w)$.

We remind that whenever $\to_2$ rewriting applies then so does $\to_{2m}$ and $\to_{2l}$. Moreover a word is in normal form w.r.t. $\to_2$ iff it is in normal form w.r.t. $\to_{2m}$ and iff it is in normal form w.r.t. $\to_{2l}$. In what follows, we will use these trivial observations without any explicit reference.

We show that $\langle p\rangle_{2m} = \langle p\rangle_2$. The inclusion $\langle p\rangle_{2m} \subseteq \langle p\rangle_2$ is obvious and the rest is the content of the following lemmas. For more transparent proofs we use the notation $s\underline{u}v\underline{w}t \to_2 suwt$ in the sense that $suvwt \to_2 suwt$ where $\emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$ (and the same for $\to_{2l}$, $\to_{2m}$). In the following, whenever we say that $u$ is a subword of $sut$, we always refer to the concrete (and obvious) occurrence of the subword $u$ in $sut$.

**Lemma 6.** *The relation $\to_{2m}$ is confluent and terminating.*

*Proof.* The termination is obvious. Let $p$ be a word and suppose that we can apply two different rules of $\to_{2m}$ on $p$, say $p = s_1\underline{u_1}v_1\underline{w_1}t_1 \to_{2m} s_1u_1w_1t_1$ and $p = s_2\underline{u_2}v_2\underline{w_2}t_2 \to_{2m} s_2u_2w_2t_2$.

Let us suppose that $u_1$ is a subword of $u_2v_2w_2$. Then the whole $u_1v_1w_1$ is a subword of $u_2v_2w_2$, because $u_2v_2w_2$ is followed by a new letter (if $t_2$ is non-empty), which is not contained in $u_1$. If $u_1v_1w_1$ is a subword of $u_2$ resp. $w_2$ then our two rules commute (i.e. they are independent of the order of their applications). If it is not the case, we will show that $v_1$ is a subword of $v_2$. Suppose that the occurrence of $\overline{0}(u_2)$ (the last letter of $u_2$) is in $u_1v_1w_1$, then it is surely in $u_1$.

[1] Observe that it doesn't hold in general that if $p \to_{2m} q$ then $spt \to_{2m} sqt$ for $s, t \in \mathcal{C}^*$. This means that $\to_{2m}$ is not a rewriting relation in the previously introduced sense.

Similarly if the occurrence of $\overline{1}(w_2)$ is in $u_1v_1w_1$, then it is in $w_1$. This implies that $v_1$ is a subword of $v_2$. Let us define $s_0$ as a prefix of $u_2$ and $t_0$ as a suffix of $w_2$ s.t. $s_0u_1v_1w_1t_0 = u_2v_2w_2$. Observe that $\mathsf{C}(s_0u_1) = \mathsf{C}(u_2) = \mathsf{C}(w_2) = \mathsf{C}(w_1t_0)$ and using the condition $(i)$ in the definition of $\rightarrow_{2m}$ we get that $s_0 = t_0 = \epsilon$. So we have that $u_1v_1w_1 = u_2v_2w_2$ and $v_1$ is a subword of $v_2$, which is a contradiction with the conditions $(ii)$ and $(iii)$.

If $u_1$ resp. $w_1$ is not a subword of $u_2v_2w_2$, and $u_2$ resp. $w_2$ is not a subword of $u_1v_1w_1$, then our two rules commute. □

*Remark 2.* From the previous proof we can see that two arbitrary applications of $\rightarrow_{2m}$, say $p = s_1\underline{u_1}v_1\underline{w_1}t_1 \rightarrow_{2m} s_1u_1w_1t_1$ and $p = s_2\underline{u_2}v_2\underline{w_2}t_2 \rightarrow_{2m} s_2u_2w_2t_2$, commute and they can be nested exactly in one of the following ways (up to symmetry):

1. $w_1 = w_1'q, u_2 = qu_2'$ and $p = s_1u_1v_1w_1'qu_2'v_2w_2t_2$
2. $u_1v_1w_1$ is a subword of $u_2$

**Lemma 7.** *$RS_2$ is a confluent and terminating rewriting system and $\langle p\rangle_{2m} = \langle p\rangle_2$ for any $p \in \mathcal{C}^*$.*

*Proof.* The termination is clear. If we have $u \rightarrow_2 v$, $u \rightarrow_2 w$ then there is $v_1$ and $w_1$ such that $v \rightarrow_2^* v_1$, $w \rightarrow_2^* w_1$, $u \rightarrow_{2m} v_1$ and $u \rightarrow_{2m} w_1$. Since $\rightarrow_{2m}$ is confluent, we have the confluence of $RS_2$. The equality $\langle p\rangle_{2m} = \langle p\rangle_2$ is a trivial consequence. □

**Lemma 8.** *For any $p, q \in \mathcal{C}^*$ such that $p = \langle p\rangle_2$ and $p \rightarrow_1 q$ it holds that $\langle q\rangle_2 = q$. In particular for a word $p \in \mathcal{C}^*$ we have $\langle\langle p\rangle_2\rangle_1 = \langle p\rangle$.*

*Proof.* Assume for the moment that $\langle q\rangle_2 \neq q$, which means that $q = suawt$ where $s, u, w, t \in \mathcal{C}^*$, $a \in \mathcal{C}$, $a \in \mathsf{C}(u) = \mathsf{C}(w)$, i.e. $q = s\underline{u}a\underline{w}t \rightarrow_{2l} suwt$. Then (up to symmetry) $p = su_1xu_2awt$ where $u_1, x, u_2 \in \mathcal{C}^*$, $u_1u_2 = u$ and $su_1$ has a suffix $x$ or $u_2awt$ has a prefix $x$. We discuss four different cases:

**1)** $x$ is a suffix of $u_1$
**2)** $x$ is a suffix of $su_1$ and $|x| > |u_1|$
**3)** $x$ is a prefix of $u_2aw$
**4)** $x$ is a prefix of $u_2awt$ and $|x| > |u_2aw|$.

In the case 1) we get $\mathsf{C}(x) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$ and we could also use the reduction $p = s\underline{u_1 x u_2} a \underline{w} t \to_{2l} s u_1 x u_2 w t$ since $a \in \mathsf{C}(u_1 x u_2) = \mathsf{C}(u) = \mathsf{C}(w)$. In the case 2) we may write $x = x_1 u_1$ and then $p = s u_1 x_1 u_1 u_2 a w t = s u_1 x_1 \underline{u} a \underline{w} t \to_{2l} s u_1 x_1 u w t$. Cases 3) and 4) are similar and all the four cases lead to a contradiction. □

## 4 Upper bound for the size of the solution

This section aims to prove that the PATTERN-EQUATION problem is in NP by giving a polynomial upper bound on the size of a minimal solution. In the following we assume implicitly that $A, B \in \mathcal{C}^*$. Realise that each reduction of $RS$ just leaves out some subword, the case $uvw \to uw$ is clear, and in the case $xx \to x$ we leave out the right occurrence of $x$ in the square. If we have a word $uAv$, we can speak about the *residual* of $A$ in the sense that the residual consists of all letter occurrences of $A$ that were not left out during the sequence of reductions. Moreover if we use two different sequences of reductions by $\to_{2m}$, which give normal form w.r.t. $\to_2$, then the residuals are the same after the both reduction sequences, since any two applications of $\to_{2m}$ commute by Remark 2.

**Lemma 9.** *Let $A$ and $B$ be in normal form and $AB \to_{2m} AB' \not\to_{2m}$ where $B'$ is the residual of $B$. Then the word $B'$ contains at most one square $x^2$. If $B'$ contains such a square, then $x^2$ arises from $xvx$ where $v$ is the word left out by the reduction rule $\underline{u}v\underline{w} \to_{2m} uw$, and $x$ is both a suffix of $u$ and a prefix of $w$. Moreover in the case when $B'$ contains a square we have $B' \to_1 \langle B' \rangle$.*

*Proof.* Assume that we have used $AB = s\underline{u}v\underline{w}t \to_{2m} suwt = AB'$ and $B'$ contains a square $x^2$. Since $B$ is in normal form, $x^2$ contains "space" of the cancelled $v$, i.e. $xx = u_1 w_1$ where $u_1$ is a suffix of $u$ ($u$ starts in $A$) and $w_1$ is a prefix of $wt$.

We show that $w_1$ is a prefix of $w$. In the case when $|w_1| > |w|$ we can deduce that occurrences of $\mathsf{T}(u_1)$ and $\mathsf{H}(t)$ must lie in the left $x$ since they are the first occurrences of the constants $\mathsf{T}(u_1)$ and $\mathsf{H}(t)$ in $B$ (from the maximality of $\to_{2m}$). It means that $x = u_1 w z$ where $z$ is a prefix of $t$. Since $\mathsf{C}(u_1) \subseteq \mathsf{C}(u) = \mathsf{C}(w) \subseteq \mathsf{C}(wz)$, we can reduce $xx = u_1 \underline{wz} u_1 \underline{wz} \to_2 u_1 wzwz$ and this is a contradiction with $B' \not\to_2$.

So, $w_1$ is a prefix of $w$. The last letter of $u_1$ is in the left $x$ and the first letter of $w_1$ is in the right $x$ and we see that $x^2$ arises from $xvx$ and $x$ is a suffix of $u$ and a prefix of $w$ (i.e. $u = u_0x$, $w = xw_0$).

We have $uwt = u_0xxw_0t \to_1 u_0xw_0t$. Let us denote $B''$ as the residual of $B'$; in fact $B'' = u_0''xw_0t$ where $u_0''$ is a suffix of $u_0$.

It is enough to show that the word $B''$ does not contain a square and in such a case we get $B' \to_1 \langle B' \rangle_1 = B''$. Assume that $u_0''xw_0t$ contains a square $y^2$. Recall $xw_0t = wt$ is a suffix of $B$. Thus $y^2$ is a subword of $u_0''xw_0$ since $\mathsf{H}(t) \notin \mathsf{C}(u_0''xw_0)$ and because of the similar arguments as in the second paragraph. Since $u_0''x$ and $xw_0 = w$ are subwords of $B$ then $y^2$ contains both $\mathsf{T}(u_0'')$ and $\mathsf{H}(w_0)$. However, $\mathsf{T}(x) = \mathsf{T}(u)$ is the first occurrence of this letter in $u_0''xw_0$ and it must be in the left $y$ and from the same reason $\mathsf{H}(x) = \mathsf{H}(w)$ is in the right $y$. This is impossible. So $u_0''xw_0t$ contains no square.

If $B'$ contains two (or more) squares then one is a subword of the other and we get two different residuals of $B'$. This is a contradiction with $B' \to_1 \langle B' \rangle_1 = \langle B' \rangle$. So $B'$ contains at most one square. □

*Remark 3.* The same arguments as in the proof of Lemma 9 give the following analogue. If $A_1$, $B$, $A_2$ are in normal form and $A_1BA_2 \to_{2m} A_1B'A_2 \not\to_{2m}$ and if the residual $B'$ of $B$ contains a square $x^2$, then $B' \to_1 \langle B' \rangle_1$ and $x$ has the same properties as in Lemma 9.

**Proposition 1.** *Let $A$ and $B$ be in normal form such that $\langle AB \rangle_2 = AB'$ where $B'$ is the residual of $B$, then $|B'| \leq |\langle B' \rangle|^2$.*

*Proof.* By Lemma 7 we have $\langle AB \rangle_2 = \langle AB \rangle_{2m}$ and we can use the maximal reductions. W.l.o.g. assume that the reductions $\to_{2m}$ did not leave out some prefix $B_1$ of the word $B$, otherwise we can start with the words $A$ and $B_2$ where $B = B_1B_2$. Remark 2 shows how two applications of $\to_{2m}$ can be nested. Since $A$ and $B$ are in normal form, we can see that any reduction $\to_{2m}$ uses some letters from both $A$ and $B$. Since $A$ is left untouched, we can write $A = s_{n+1}s_n \dots s_1$, $B = u_1v_1w_1 \dots u_nv_nw_nu_{n+1}$ where $s_i, u_i, v_i, w_i \in \mathcal{C}^*$ for all possible $i$ and we have $n$ reductions of the form

$$s_{n+1} \dots \underline{s_i \dots s_1u_1w_1 \dots u_i}v_i\underline{w_i} \dots u_{n+1} \to_{2m}$$

$$s_{n+1} \dots s_1u_1w_1 \dots u_iw_i \dots u_{n+1}$$

where $\mathsf{C}(v_i) \subseteq \mathsf{C}(s_i \dots s_1 u_1 w_1 \dots u_i) = \mathsf{C}(w_i)$ and $B'$ is of the form $B' = u_1 w_1 \dots u_n w_n u_{n+1}$.

Since each step of the maximal reduction needs a new letter (the letter that immediately succeeds $w_i$), we get an upper bound for $n$ (the number of steps in $\rightarrow_{2m}^*$), $n+1 \leq \mathrm{card}(\mathsf{C}(B))$. Let us denote $B'' = \langle B' \rangle_1 = \langle B' \rangle$ and $w_0 = \epsilon$. By induction (where $i = 1, \dots, n$) and by Lemma 9 applied on $A$ and $\langle w_0 u_1 \dots w_{i-1} u_i \rangle v_i w_i u_{i+1}$ we can see that $|B''| \geq \max_{i=1}^{n+1} \{|w_{i-1} u_i|\}$ since after every application $xx \rightarrow x$ we can find each $w_{i-1} u_i$ as a subword in the residual of $B$. Hence we get $|B''| \geq \max_{i=1}^{n+1} \{|w_{i-1} u_i|\} \geq \frac{1}{n+1} \sum_{i=1}^{n+1} |w_{i-1} u_i| = \frac{|B'|}{n+1}$ and from the fact $n+1 \leq \mathrm{card}(\mathsf{C}(B)) = \mathrm{card}(\mathsf{C}(B''))$ we can deduce that $|\langle B' \rangle|^2 = |B''|^2 \geq \mathrm{card}(\mathsf{C}(B'')) \cdot |B''| \geq (n+1) \frac{|B'|}{n+1} = |B'|$. □

The previous proposition can be generalized in the following way.

**Proposition 2.** *Let $A_1$, $B$ and $A_2$ be in normal form such that $\langle A_1 B A_2 \rangle_2 = A_1' B' A_2'$ where $A_1'$, $B'$, $A_2'$ are the residuals of $A_1$, $B$, $A_2$. Then $|B'| \leq 2 \cdot |\langle B' \rangle|^2$.*

*Proof.* We prove the assertion for $A_1' = A_1$ and $A_2' = A_2$, because in the case when we leave out some occurrences of letters in $A_1$ and $A_2$, we can reduce $A_i' \rightarrow_1^* \langle A_i' \rangle$ and start with these new surroundings $\langle A_1' \rangle$ and $\langle A_2' \rangle$ of the word $B$, since $\langle \langle A_1' \rangle B \langle A_2' \rangle \rangle_2 = \langle A_1' \rangle B' \langle A_2' \rangle$. So, $\langle A_1 B A_2 \rangle_2 = A_1 B' A_2$ where $B'$ is the residual of $B$.

We will use the maximal reduction again and for an arbitrary word $B_i \in \mathcal{C}^*$ we denote $B_i'$ its residual (after the applications of $\rightarrow_{2m}$). Three different cases must be discussed.

**1)** There is a reduction $\rightarrow_{2m}$ using letters from both $A_1$ and $A_2$.
**2)** There is a letter in $B$ which is not involved in any reduction $\rightarrow_{2m}$.
**3)** Otherwise.

In the case 1) we can write $B = B_1 v B_2$, $A_1 = s u_1$ and $A_2 = w_2 t$ where $s \underline{u_1 B_1} v \underline{B_2 w_2} t \rightarrow_{2m} A_1 B_1 B_2 A_2$. We apply twice Proposition 1 on the words $A_1 B_1'$ and $B_2' A_2$. We can deduce $|B'| = |B_1' B_2'| = |B_1'| + |B_2'| \leq |\langle B_1' \rangle|^2 + |\langle B_2' \rangle|^2 \leq 2 \cdot |\langle B_1' B_2' \rangle|^2 = 2 \cdot |\langle B' \rangle|^2$ where the last inequality holds, because by Remark 3 we have $\langle B_1' \rangle \langle B_2' \rangle \rightarrow_1 \langle B' \rangle$ (in the case when $\langle B_1' \rangle \langle B_2' \rangle$ contain a square) and so $|\langle B_1' B_2' \rangle| \geq \max\{|\langle B_1' \rangle|, |\langle B_2' \rangle|\}$, which implies that $2 \cdot |\langle B_1' B_2' \rangle|^2 \geq |\langle B_1' \rangle|^2 + |\langle B_2' \rangle|^2$.

In the case 2) we can write $B = B_1B_2B_3$ where $B_2$ is not involved in any reduction by $\rightarrow_{2m}$. Then we have $\langle B' \rangle = \langle B'_1 \rangle B_2 \langle B'_3 \rangle$ and so by Proposition 1 we get $|B'| = |B'_1B_2B'_3| = |B'_1| + |B_2| + |B'_3| \leq |\langle B'_1 \rangle|^2 + |B_2| + |\langle B'_3 \rangle|^2 \leq |\langle B'_1 \rangle B_2 \langle B'_3 \rangle|^2 = |\langle B' \rangle|^2$.

In the case 3) we can write $B = B_1v_1B_2B_3B_4v_2B_5$, $A_1 = su_1$, $A_2 = w_2t$ where

$$A_1BA_2 = s\underline{u_1B_1}v_1\underline{B_2B_3}B_4v_2B_5w_2t \rightarrow_{2m} A_1B_1B_2B_3B_4v_2B_5A_2,$$

$$A_1BA_2 = su_1B_1v_1B_2\underline{B_3B_4}v_2\underline{B_5w_2}t \rightarrow_{2m} A_1B_1v_1B_2B_3B_4B_5A_2$$

are the unique overlapping reductions (Remark 2). We have $B' = B'_1B_2B_3B_4B'_5$ and $B_0 = \langle B'_1B_2B_3 \rangle$ is a prefix of $\langle B' \rangle = \langle B_0B_4B'_5 \rangle$, and so $|B_0| \leq |\langle B' \rangle|$. Let us now observe that $|B'| = |B'_1B_2B_3B_4B'_5| = |B'_1B_2B_3| + |B_4B'_5| \leq |\langle B'_1B_2B_3 \rangle|^2 + |B_0B_4B'_5| \leq |B_0|^2 + |\langle B_0B_4B'_5 \rangle|^2 \leq 2 \cdot |\langle B' \rangle|^2$. □

**Lemma 10.** *Let $sxt$ be a word such that $\langle sxt \rangle_2 = sxt$ and $sxt$ contains another square $y^2$ ($|y| \leq |x|$) such that one of these occurrences of $y$ lies inside the $x^2$. Then one of the following conditions holds:*

1. *$y$ is a suffix of $s$ and a prefix of $x$*
2. *$y$ is a prefix of $t$ and a suffix of $x$*
3. *$y^2$ is a subword of $x$*

*Proof.* Since $y$ is a subword of $x^2$, we have $\mathsf{C}(y) \subseteq \mathsf{C}(x)$ and let us suppose that conditions 1. and 2. do not hold. If $|x| = |y|$ then we get $\mathsf{C}(x) = \mathsf{C}(y)$ and we can apply $\rightarrow_2$ (w.l.o.g. $\underline{x}v\underline{y} \rightarrow_2 xy$ where $v$ is both a prefix of $x$ and a suffix of $y$), which is a contradiction. Assume that $|y| < |x|$. Notice that the first and the last letter of $x$ are unique occurrences of these constants in the word $x$ because in another case we can apply $\rightarrow_{2l}$ on $x^2$. This implies that $y^2$ does not contain the first letter of the right $x$ and $y^2$ also does not contain the last letter of the left $x$, requiring that $y^2$ is a subword of $x$. □

*Remark 4.* The previous lemma shows that for two applications of the rules $xx \rightarrow_1 x$ and $yy \rightarrow_1 y$ on a word $p$ in normal form w.r.t. $\rightarrow_2$, one of the following conditions holds (up to symmetry):

1. $xx$ and $yy$ do not overlap

2. $yy$ is a subword of $x$
3. $x = x'z$, $y = zy'$ and $xx'zy'y$ is a subword of $p$

**Lemma 11.** *If $\langle sxxt\rangle_2 = sxxt$ and $sxt$ contains a square $y^2$ which is not in $sxxt$ then $y = s_1xt_1$ where $|s_1|, |t_1| \geq 1$.*

*Proof.* Since $y^2$ is not in $sxxt$, we have $y^2 = s_0xt_0$ where $s_0$ is a suffix of $s$ and $t_0$ is a prefix of $t$. If $x$ is not a subword of $y$ then $y = s_0x_1$, $y = x_2t_0$ where $x = x_1x_2$. Hence $s_0xxt_0 = \underline{s_0x_1}x_2x_1\underline{x_2t_0} \to_2 s_0x_1x_2t_0$ since $\mathsf{C}(x_2x_1) \subseteq \mathsf{C}(y) = \mathsf{C}(s_0x_1) = \mathsf{C}(x_2t_0)$. The case when $x_1$ or $x_2$ is an empty word (i.e. $x$ is a prefix or a suffix of $y$) is also included and we can conclude that $y = s_1xt_1$ and $|s_1|, |t_1| \geq 1$. □

**Proposition 3.** *Let $A$ and $B$ be in normal form,* $\mathrm{card}(\mathsf{C}(AB)) \geq 2$ *and $\langle AB\rangle_2 = AB$. Then $|AB| \leq |\langle AB\rangle|^2$.*

*Proof.* We denote $k$ the length of $\langle AB\rangle_1 = \langle AB\rangle$. The case $k = 2$ is trivial: there is no word (in normal form w.r.t. $\to_2$) over a 2–letter alphabet of length greater than 4. Now we assume that $k \geq 3$.

At first we have a look at the squares in $AB$. Since $A$ and $B$ are in normal form, each square has got some letters from $A$ and some from $B$. By Remark 4 we have that $AB$ contains at most two squares because the cases 1. and 2. of Remark 4 are impossible.

By Lemma 8, anytime during any reduction sequence by $\to_1$ the residuals of $A$ and $B$ remain in normal form and their concatenations are in normal form w.r.t. $\to_2$. So by the previous arguments there are at most two squares and by Lemma 11, if the reduction $\to_1$ introduces a new square then it is larger at least by two letters than the previous one. We use the sequence of reductions which in each step reduces the smallest square. Then the last reduction in this sequence reduces a square $x^2$ with the property $|x| \leq k$. The previous reduction reduces a square $y^2$ with the property $|y| \leq k-1$ (the case $|y| = |x| = k = |\langle AB\rangle|$ is a contradiction with $AB$ is in normal form w.r.t. $\to_2$). Thus we can see that for the length of $AB$ we have an upper bound $k + k + (k-1) + (k-2) + \ldots \leq k^2$. □

**Proposition 4.** *There is a polynomial $p : \mathbb{N} \to \mathbb{N}$, such that for an arbitrary $A_1$, $B$, $A_2 \in \mathcal{C}^*$ in normal form and $\langle A_1BA_2\rangle_2 = A_1'B'A_2'$ where $A_i'$, $1 \leq i \leq 2$, is the residual of $A_i$ and $B'$ is the residual of $B$, we have $|B'| \leq p(|\langle A_1BA_2\rangle|)$.*

*Proof.* We may assume that $\mathrm{card}(\mathsf{C}(B)) \geq 2$ and by Proposition 2 we know that $|B'| \leq 2 \cdot |\langle B' \rangle|^2$, which is of course less or equal to $2 \cdot |\langle A'_1 \rangle \langle B' \rangle|^2$. Since $\langle \langle A'_1 \rangle \langle B' \rangle \rangle_2 = \langle A'_1 \rangle \langle B' \rangle$ by Lemma 8, we can use Proposition 3 and we get that $2 \cdot |\langle A'_1 \rangle \langle B' \rangle|^2 \leq 2 \cdot |\langle A'_1 B' \rangle|^4$, which is again less or equal to $2 \cdot |\langle A'_1 B' \rangle \langle A'_2 \rangle|^4$. Analogously we have that $2 \cdot |\langle A'_1 B' \rangle \langle A'_2 \rangle|^4 \leq 2 \cdot |\langle A'_1 B' A'_2 \rangle|^8$. Thus we have $|B'| \leq p(|\langle A_1 B A_2 \rangle|)$ for the polynomial $p(n) = 2 \cdot n^8$. □

**Proposition 5.** *Let $p$ be a polynomial that satisfies the condition from Proposition 4. If a stuttering pattern equation system $\{X_1 = A_1, \ldots, X_n = A_n\}$ is satisfiable then there exists a solution $\alpha$ with $\mathrm{size}(\alpha) \leq \sum_{i=1}^{n} |X_i| \cdot p(|A_i|)$.*

*Proof.* Of course, we can assume that all $A_i$'s are in normal form. Let $\alpha$ be a solution of the stuttering pattern equation system $\{X_1 = A_1, \ldots, X_n = A_n\}$ which minimizes both $\mathrm{size}(\alpha)$ and the number of variables $x$ such that $|\alpha(x)| = \mathrm{size}(\alpha)$. Assume for the moment that there is some $x$ such that $\mathrm{size}(\alpha) = |\alpha(x)| > \sum_{i=1}^{n} |X_i| p(|A_i|)$. We may assume that $\alpha(x)$ is in normal form, otherwise we have a smaller solution.

We now reduce $\alpha(X_i) \rightarrow_{2m}^{*} \langle \alpha(X_i) \rangle_2$. If we look at an arbitrary residual $B'$ of an occurrence of $\alpha(x)$ in $\langle \alpha(X_i) \rangle_2$, we see that $|B'| \leq p(|A_i|)$ by Proposition 4. This means that there are at most $\sum_{i=1}^{n} |X_i| p(|A_i|)$ letter's occurrences in the residuals of all occurrences of $\alpha(x)$ in all $\langle \alpha(X_i) \rangle_2$.

By the assumption $|\alpha(x)| > \sum_{i=1}^{n} |X_i| p(|A_i|)$ we get that there is an occurrence of a letter $a$ in $\alpha(x)$, i.e. $\alpha(x) = u_1 a u_2$, that has been left out from all the occurrences of $\alpha(x)$ by the rule $\rightarrow_{2m}$. We can erase this occurrence of the letter $a$ from $\alpha(x)$ and we get a smaller solution $\beta$ s.t. $\beta(y) = \alpha(y)$ for $y \neq x$ and $\beta(x) = u_1 u_2$. The homomorphism $\beta$ is indeed a solution since $\alpha(X_i) \rightarrow_{2l}^{*} \beta(X_i)$. This is a contradiction because we have found a smaller solution. □

The previous considerations lead to the following corollary.

**Corollary 1.** *The* Pattern-Equation *problem is in NP.*

*Proof.* We can guess a solution $\alpha$ and by Proposition 5, if the system is satisfiable then there is a solution of a polynomial length. Checking whether $\alpha$ solves all the equations takes also polynomial time and so the problem is in NP. □

## 5 NP–hardness of the Pattern-Equation problem

In this section we show that the Pattern-Equation problem in a free idempotent semigroup is NP–hard. We use a reduction from the NP–complete problem 3–SAT (see [17]).

**Proposition 6.** *The* Pattern-Equation *problem is NP–hard.*

*Proof.* Suppose we have an instance of 3–SAT, i.e.

$$C \equiv C_1 \wedge C_2 \wedge \ldots \wedge C_n$$

is a conjunction of clauses and each clause $C_i$, $1 \leq i \leq n$, is of the form

$$l_1 \vee l_2 \vee l_3$$

where $l_j$, $1 \leq j \leq 3$, is a literal ($l_j$ is a variable from the set *Var*, possibly negated – we call it positive resp. negative literal). A valuation is a mapping $v : Var \to \{T, F\}$. This valuation extends naturally to $C$ and we say that $C$ is satisfiable if and only if there exists a valuation $v$ such that $v(C) = T$.

We construct a stuttering pattern equation system such that the system is satisfiable if and only if $C$ is satisfiable. The system will consist of the following sets of equations (1) – (6) and $\mathcal{C} = \{a, b, c\}$, $\mathcal{V} = \{x, s_1^x, t_1^x, s_2^x, t_2^x \mid x \in Var \cup \overline{Var}\} \cup \{y_a, y_b, y_c\}$ where $\overline{Var} = \{\overline{x} \mid x \in Var\}$ is a disjoint copy of *Var*. For the constants $a$, $b$ and $c$ there are three equations

$$y_a = a, \quad y_b = b, \quad y_c = c. \tag{1}$$

We define $\widetilde{x} = x$ if $x$ is a positive literal, $\widetilde{\neg x} = \overline{x}$ if $\neg x$ is a negative literal and for all clauses $C_i \equiv l_1 \vee l_2 \vee l_3$ we have the equation

$$y_a \widetilde{l_1} \widetilde{l_2} \widetilde{l_3} y_a = aba \tag{2}$$

for each $x \in Var$ we add the equations

$$y_b x \overline{x} y_b = bab \tag{3}$$

$$y_a x \overline{x} y_a = aba \tag{4}$$

and finally for each $x \in Var \cup \overline{Var}$ we have the following equations:

$$s_1^x x t_1^x = acb, \quad s_1^x y_c = ac \tag{5}$$

$$s_2^x x t_2^x = bca, \quad s_2^x y_c = bc. \tag{6}$$

The intuition behind the construction is following. If a variable $x$ is true then $x = b$ and if $x$ is false then $x = a$. The second equation ensures that at least one literal in each clause is true and the other equations imply consistency, i.e. a literal and its negation cannot be both true (false). In particular, the equation (3) means that at least one of $x$ and $\overline{x}$ contains $a$. Similarly for $b$ and (4). The last two equations make sure that a variable $x \in Var \cup \overline{Var}$ cannot contain both $a$ and $b$.

Suppose that $C$ is satisfiable, i.e. there is a valuation $v$ such that $v(C) = T$. Then we show that $\alpha$ defined below is a solution of our system. Let us state

$$\alpha(y_a) = a, \quad \alpha(y_b) = b, \quad \alpha(y_c) = c$$

and for all $x \in Var$ such that $v(x) = T$ let

$$\alpha(x) = b, \quad \alpha(\overline{x}) = a$$

$$\alpha(s_1^x) = ac, \quad \alpha(t_1^x) = b, \quad \alpha(s_1^{\overline{x}}) = a, \quad \alpha(t_1^{\overline{x}}) = cb$$
$$\alpha(s_2^x) = b, \quad \alpha(t_2^x) = ca, \quad \alpha(s_2^{\overline{x}}) = bc, \quad \alpha(t_2^{\overline{x}}) = a$$

and if $v(x) = F$ then

$$\alpha(x) = a, \quad \alpha(\overline{x}) = b$$

$$\alpha(s_1^x) = a, \quad \alpha(t_1^x) = cb, \quad \alpha(s_1^{\overline{x}}) = ac, \quad \alpha(t_1^{\overline{x}}) = b$$
$$\alpha(s_2^x) = bc, \quad \alpha(t_2^x) = a, \quad \alpha(s_2^{\overline{x}}) = b, \quad \alpha(t_2^{\overline{x}}) = ca.$$

Checking that $\alpha$ is a solution (even non-singular) is a routine. The only interesting equation is (2). This equation is also satisfied by $\alpha$ since we have the assumption that under the valuation $v$ there is at least one true literal in each clause.

Let us suppose that $\alpha$ is an arbitrary solution of our system and we find a valuation that satisfies $C$. The equation (3) implies that

$\mathsf{C}(\alpha(x)) \subseteq \{a, b\}$ for all $x \in \mathit{Var} \cup \overline{\mathit{Var}}$. We will conclude that it is not possible that $\mathsf{C}(\alpha(x)) = \{a, b\}$.

Suppose that it is the case and using the equations (5) we get that $\alpha(x)$ does not begin with the constant $a$. For the moment assume that $\alpha(x)$ begins with $a$. We have $ac = 0(acb) \sim 0(\alpha(s_1^x x t_1^x))$ and from (1) and (5) we get that $a \in \mathsf{C}(\alpha(s_1^x)) \subseteq \{a, c\}$. If $\mathsf{C}(\alpha(s_1^x)) = \{a\}$ then $\mathsf{C}(0(\alpha(s_1^x x t_1^x))) = \{a, b\}$ whereas $\mathsf{C}(0(acb)) = \{a, c\}$, which is a contradiction. Otherwise we have $\mathsf{C}(\alpha(s_1^x)) = \{a, c\}$ and we get $0(\alpha(s_1^x x t_1^x)) \sim aca \not\sim ac = 0(acb)$.

By the similar arguments and using the equations (6) we get that $\alpha(x)$ does not begin with the constant $b$. This yields that there are just three possibilities for $\alpha(x)$, namely $\alpha(x) \sim a$, $\alpha(x) \sim b$ or $\alpha(x) = \epsilon$.

By the equations (3) and (1) we know that for all $x \in \mathit{Var}$ at least $\alpha(x) \sim a$ or $\alpha(\overline{x}) \sim a$. The equation (4) implies that either $\alpha(x) \sim b$ or $\alpha(\overline{x}) \sim b$. Similarly for each clause, the equation (2) with (1) gives that there is $j$, $1 \leq j \leq 3$, such that $\alpha(\widetilde{l_j}) \sim b$. Let us finally define the valuation $v$ as $v(x) = T$ if $\alpha(x) \sim b$ and $v(x) = F$ if $\alpha(x) \sim a$ for each $x \in \mathit{Var}$. The valuation is consistent and it holds that $v(C) = T$.

This is enough to demonstrate that the Pattern-Equation problem is NP–hard since the reduction can be done effectively in polynomial time. □

It is not difficult to see that the same reduction as above would also work for the Non-Singular-Pattern-Equation problem, which is consequently also NP–hard. We can now formulate the main result of this paper.

**Theorem 1.** Pattern-Equation *and* Non-Singular-Pattern-Equation *problems are NP–complete.*

As an immediate corollary of this theorem (using Remark 1), we get the following result.

**Corollary 2.** *AI-matching with only one associative and idempotent function symbol is NP–complete.*

*Acknowledgements* We would like to thank Ivana Černá and Michal Kunc for their comments and suggestions.

# References


[1] Baader F.: The Theory of Idempotent Semigroups is of Unification Type Zero, J. of Automated Reasoning **2** (1986) 283–286.

[2] Baader F.: Unification in Varieties of Idempotent Semigroups, Semigroup Forum **36** (1987) 127–145.

[3] Baader F., Schulz K.U.: Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures, J. Symbolic Computation **21** (1996) 211–243.

[4] Baader F., Siekmann J.H.: Unification Theory, Handbook of Logic in Artificial Intelligence and Logic Programming (1993) Oxford University Press.

[5] Book R., Otto F.: String-Rewriting Systems (1993) Springer–Verlag.

[6] Černá I., Klíma O., Srba J.: Pattern Equations and Equations with Stuttering, In Proceedings of SOFSEM'99, the 26th Seminar on Current Trends in Theory and Practice of Informatics (1999) 369-378, Springer–Verlag.

[7] Green J.A., Rees D.: On semigroups in which $x^r = x$, Proc. Camb. Phil. Soc. **48** (1952) 35–40.

[8] Klíma O., Srba J.: Complexity Issues of the Pattern Equations in Idempotent Semigroups, Technical report FIMU-RS-99-02, Faculty of Informatics MU (1999).

[9] Klíma O., Srba J.: Matching Modulo Associativity and Idempotency is NP–complete, In Proceedings of MFCS'00, the 25rd International Symposium on Mathematical Foundations of Computer Science (2000) To appear.

[10] Kaďourek J., Polák L.: On free semigroups satisfying $x^r = x$ , Simon Stevin **64**, No.1 (1990) 3–19.

[11] Kapur D., Narendran P.: NP–completeness of the Set Unification and Matching Problems, In Proceedings of CADE'86, Springer LNCS volume **230** (1986) 489–495, Springer–Verlag.

[12] Kopeček I.: Automatic Segmentation into Syllable Segments, Proceedings of First International Conference on Language Resources and Evaluation (1998) 1275–1279.

[13] Kopeček I., Pala K.: Prosody Modelling for Syllable-Based Speech Synthesis, Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, Cancun (1998) 134–137.

[14] Lothaire M.: Algebraic Combinatorics on Words, Preliminary version available at `http://www-igm.univ-mlv.fr/~berstel/Lothaire/index.html`

[15] Lothaire, M.: Combinatorics on Words, Volume **17** of Encyclopedia of Mathematics and its Applications (1983) Addison-Wesley.

[16] Makanin, G. S.: The Problem of Solvability of Equations in a Free Semigroup, Mat. Sbornik. **103(2)** (1977) 147–236. (In Russian) English translation in: Math. USSR Sbornik **32** (1977) 129–198.

[17] Papadimitriou, C.H.: Computational Complexity, Addison-Wesley Publishing Company (1994), Reading, Mass.

[18] Perrin D.: Equations in Words, In H. Ait-Kaci and M. Nivat, editors, Resolution of Equations in Algebraic Structures, Vol. **2** (1989) 275–298, Academic Press.

[19] Schulz, K. U.: Makanin's Algorithm for Word Equations: Two Improvements and a Generalization, In Schulz, K.–U. (Ed.), Proceedings of Word Equations and Related Topics, 1st International Workshop, IWW-ERT'90, Tübingen, Germany, Vol. **572** of LNCS (1992) 85–150, Berlin-Heidelberg-New York, Springer–Verlag.

[20] Schmidt-Schauss M.: Unification under Associativity and Idempotence is of Type Nullary, J. of Automated Reasoning **2** (1986) 277–281.

[21] Siekmann J., Szabó P.: A Noetherian and Confluent Rewrite System for Idempotent Semigroups, Semigroup Forum **25** (1982).

# Recent BRICS Report Series Publications

**RS-00-13** **Ondřej Klíma and Jiří Srba.** ***Matching Modulo Associativity and Idempotency is NP-Complete*****. June 2000. 19 pp. To appear in** ***Mathematical Foundations of Computer Science: 25 th International Symposium*****, MFCS '00 Proceedings, LNCS, 2000.**

**RS-00-12** **Ulrich Kohlenbach.** ***Intuitionistic Choice and Restricted Classical Logic*****. May 2000. 9 pp.**

**RS-00-11** **Jakob Pagter.** ***On Ajtai's Lower Bound Technique for R-way Branching Programs and the Hamming Distance Problem*****. May 2000. 18 pp.**

**RS-00-10** **Stefan Dantchev and Søren Riis.** ***A Tough Nut for Tree Resolution*****. May 2000. 13 pp.**

**RS-00-9** **Ulrich Kohlenbach.** ***Effective Uniform Bounds on the Krasnoselski-Mann Iteration*****. May 2000. 34 pp.**

**RS-00-8** **Nabil H. Mustafa and Aleksandar Pekeč.** ***Democratic Consensus and the Local Majority Rule*****. May 2000. 38 pp.**

**RS-00-7** **Lars Arge and Jakob Pagter.** ***I/O-Space Trade-Offs*****. April 2000. To appear in** ***7th Scandinavian Workshop on Algorithm Theory*****, SWAT '98 Proceedings, LNCS, 2000.**

**RS-00-6** **Ivan B. Damgård and Jesper Buus Nielsen.** ***Improved Non-Committing Encryption Schemes based on a General Complexity Assumption*****. March 2000. 24 pp.**

**RS-00-5** **Ivan B. Damgård and Mads J. Jurik.** ***Efficient Protocols based on Probabilistic Encryption using Composite Degree Residue Classes*****. March 2000. 19 pp.**

**RS-00-4** **Rasmus Pagh.** ***A New Trade-off for Deterministic Dictionaries*****. February 2000.**

**RS-00-3** **Fredrik Larsson, Paul Pettersson, and Wang Yi.** ***On Memory-Block Traversal Problems in Model Checking Timed Systems*****. January 2000. 15 pp. Appears in Graf and Schwartzbach, editors,** ***Tools and Algorithms for The Construction and Analysis of Systems: 6th International Conference*****, TACAS '00 Proceedings, LNCS 1785, 2000, pages 127–141.**

**RS-00-2** **Igor Walukiewicz.** ***Local Logics for Traces*****. January 2000. 30 pp.**