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: full version of the paper appears as [8].

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 [15]) 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 [18]. 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 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 [11] and [12] 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. 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, 19], 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 [20]). 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 [10], 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.

Many proofs in this paper are not included due to space limitation and the full version can be obtained as [8].

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 ϵ 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 [13], [14] and [17]. We refer to word equations in a free idempotent semigroup as *stuttering equations*.

In what follows we will use a uniform notation. The set $\mathcal{C} = \{a, b, c, \dots\}$ denotes the alphabet of constants and $\mathcal{V} = \{x, y, z, \dots\}$ stands for variables (unknowns) with the assumption that $\mathcal{C} \cap \mathcal{V} = \emptyset$. We will use the same symbol α 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 ϵ 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 [9]), 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 $\operatorname{card}(\mathsf{C}(w)) - 1$ letters; $1(w)$ – the longest suffix of w in $\operatorname{card}(\mathsf{C}(w)) - 1$ 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 $1(p) \sim 1(q)$.*

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

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 α as $\operatorname{size}(\alpha) := \max_{x \in \mathcal{V}} |\alpha(x)|$ and say that α is *minimal* iff for any solution β of the system we have $\operatorname{size}(\alpha) \leq \operatorname{size}(\beta)$. 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 $k = \operatorname{card}(\mathcal{C}) \geq 2$ and let $\{L_1 = R_1, \dots, L_n = R_n\}$ be a stuttering equation system. If the system is satisfiable then there exists a solution α such that $\operatorname{size}(\alpha) \leq 2^k + 2^{k-2} - 2$.*

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 $a_1, a_2, \dots$. For any solution α 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 $\{X_1 = A_1, \dots, X_n = A_n\}$ 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.

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. Our goal is to show that a minimal solution is of a polynomial length.

3 Rewriting system for idempotent semigroups

In this section we summarize several properties of the rewriting system by Siekmann and Szabo in [20] and give some technical lemmas. First of all we have to give some definitions and results concerning rewriting systems (see e.g. [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 $\rightarrow \subseteq \mathcal{C}^* \times \mathcal{C}^*$*.

$\forall p, q \in \mathcal{C}^* : p \rightarrow q$ 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

$$\forall p, p_1, p_2 \in \mathcal{C}^* \exists q \in \mathcal{C}^* : \text{ if } (p \rightarrow p_1 \text{ and } p \rightarrow p_2) \text{ then } (p_1 \rightarrow^* q \text{ and } p_2 \rightarrow^* q).$$

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 $\langle p \rangle = q$. In this paper we will exploit the rewriting system by Siekmann and Szabo in [20].

Lemma 5 ([20]). *The rewriting system $\{xx \rightarrow x \mid x \in \mathcal{C}^*, \mathsf{C}(x) \neq \emptyset\} \cup \{uvw \rightarrow uw \mid u, v, w \in \mathcal{C}^*, \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$ 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 $\{xx \rightarrow x \mid \mathsf{C}(x) \neq \emptyset\} \cup \{uvw \rightarrow uw \mid \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$ as RS. Since RS contains two different types of rewriting rules we denote RS_1 the rewriting system $\{xx \rightarrow x \mid \mathsf{C}(x) \neq \emptyset\}$ and RS_2 the rewriting system $\{uvw \rightarrow uw \mid \emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)\}$. The corresponding

rewrite relations are denoted $\rightarrow$, $\rightarrow_1$ resp. $\rightarrow_2$ 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. It can be easily shown that the word problem (i.e. the problem whether $p \sim q$) can be decided in polynomial time by using the rewriting system RS.

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 $\rightarrow_{2l} \subset \rightarrow_2$ such that $suvwt \rightarrow_{2l} suwt$ if and only if $|v| = 1$ and $\mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$. It is easy to see that $\rightarrow_2 \subseteq \rightarrow_{2l}^*$ and hence $\rightarrow_{2l}^* = \rightarrow_2^*$. The last relation we will use is $\rightarrow_{2m} \subset \rightarrow_2$, 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 $suvwt \rightarrow_{2m} suwt$ if and only if $\emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$ and the following conditions hold:

(i) $\mathsf{C}(s_0 u) \neq \mathsf{C}(w t_0)$, 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 $suvwt \rightarrow_{2m} suwt$ 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 $\rightarrow_2$ rewriting applies then so does $\rightarrow_{2m}$ and $\rightarrow_{2l}$. Moreover a word is in normal form w.r.t. $\rightarrow_2$ iff it is in normal form w.r.t. $\rightarrow_{2m}$ and iff it is in normal form w.r.t. $\rightarrow_{2l}$. 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. We use the notation $su\underline{v}wt \rightarrow_2 suwt$ in the sense that $suvwt \rightarrow_2 suwt$ where $\emptyset \neq \mathsf{C}(v) \subseteq \mathsf{C}(u) = \mathsf{C}(w)$ (and the same for $\rightarrow_{2l}$, $\rightarrow_{2m}$). 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 $\rightarrow_{2m}$ is confluent and terminating.*

Remark 2. 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 (i.e. they are independent of the order of their applications) 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

---

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

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

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 $\langle \langle p \rangle_2 \rangle_1 = \langle p \rangle$.*

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 $uvw \rightarrow uw$ is clear, and in the case $xx \rightarrow x$ 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 $\rightarrow_{2m}$, which give normal form w.r.t. $\rightarrow_2$, then the residuals are the same after the both reduction sequences, since any two applications of $\rightarrow_{2m}$ commute by Remark 2.

Lemma 9. *Let A and B be in normal form and $AB \rightarrow_{2m} AB' \not\rightarrow_{2m}$ 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 $\underline{u}v\underline{w} \rightarrow_{2m} uw$, and x is both a suffix of u and a prefix of w. Moreover in the case when B' contains a square we have $B' \rightarrow_1 \langle B' \rangle$.*

Remark 3. The previous lemma has the following analogue. If A_1, B, A_2 are in normal form and $A_1 B A_2 \rightarrow_{2m} A_1 B' A_2 \not\rightarrow_{2m}$ and if the residual B' of B contains a square x^2, then $B' \rightarrow_1 \langle B' \rangle_1$ 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 $\rightarrow_{2m}$ did not leave out some prefix B_1 of the word B, otherwise we can start with the words A and B_2 where $B = B_1 B_2$. Remark 2 shows how two applications of $\rightarrow_{2m}$ can be nested. Since A and B are in normal form, we can see that any reduction $\rightarrow_{2m}$ uses some letters from both A and B. Since A is left untouched, we can write $A = s_{n+1} s_n \ldots s_1$, $B = u_1 v_1 w_1 \ldots u_n v_n w_n u_{n+1}$ where $s_i, u_i, v_i, w_i \in \mathcal{C}^*$ for all possible i and we have n reductions of the form

$$s_{n+1} \ldots \underline{s_i \ldots s_1 u_1 w_1 \ldots u_i} v_i \underline{w_i} \ldots u_{n+1} \rightarrow_{2m} s_{n+1} \ldots s_1 u_1 w_1 \ldots u_i w_i \ldots u_{n+1}$$

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

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 $i = 1, \ldots, n$) and by Lemma 9 applied on A and $\langle w_0 u_1 \ldots w_{i-1} u_i \rangle v_i w_i u_{i+1}$ we can see that $|B''| \geq \max_{i=1}^{n+1} \{|w_{i-1} u_i|\}$ since after every application $xx \to x$ 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 $n + 1 \leq \operatorname{card}(\mathsf{C}(B)) = \operatorname{card}(\mathsf{C}(B''))$ we can deduce that $|\langle B' \rangle|^2 = |B''|^2 \geq \operatorname{card}(\mathsf{C}(B'')) \cdot |B''| \geq (n+1) \frac{|B'|}{n+1} = |B'|$. $\qquad\square$

The previous proposition can be generalized in the following way.

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

Lemma 10. *Let $sxxt$ be a word such that $\langle sxxt \rangle_2 = sxxt$ and $sxxt$ 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*

Remark 4. The previous lemma shows that for two applications of the rules $xx \to_1 x$ and $yy \to_1 y$ on a word p in normal form w.r.t. $\to_2$, 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 $y = s_1 x t_1$ where $|s_1|, |t_1| \geq 1$.*

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

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 $\langle A_1 B A_2 \rangle_2 = A_1' B' A_2'$ where A_i', $1 \leq i \leq 2$, is the residual of A_i and B' is the residual of B, we have $|B'| \leq p(|\langle A_1 B A_2 \rangle|)$.*

Proof. We may assume that $\operatorname{card}(\mathsf{C}(B)) \geq 2$ 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 $p(n) = 2 \cdot n^8$. $\qquad\square$

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 α with $\operatorname{size}(\alpha) \leq \sum_{i=1}^n |X_i| \cdot p(|A_i|)$.*

Proof. Of course, we can assume that all A_i's are in normal form. Let α be a solution of the stuttering pattern equation system $\{X_1 = A_1, \ldots, X_n = A_n\}$ which minimizes both size(α) and the number of variables x such that $|\alpha(x)| = $ size(α). Assume for the moment that there is some x such that 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 β s.t. $\beta(y) = \alpha(y)$ for $y \neq x$ and $\beta(x) = u_1 u_2$. The homomorphism β is indeed a solution since $\alpha(X_i) \rightarrow_{2l}^{*} \beta(X_i)$. This is a contradiction because we have found a smaller solution. $\qquad\square$

Corollary 1. *The* PATTERN-EQUATION *problem is in NP.*

5   NP–hardness of the PATTERN-EQUATION problem

In this section we show that the PATTERN-EQUATION problem is NP–hard. We use a reduction from the NP–complete problem 3–SAT (see [16]).

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 $v : Var \rightarrow \{T, F\}$. 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

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

$$y_a x \overline{x} y_a = aba \qquad (4)$$

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

$$s_1^x x t_1^x = acb, \quad s_1^x y_c = ac \qquad (5)$$

$$s_2^x x t_2^x = bca, \quad s_2^x y_c = bc. \qquad (6)$$

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 $x \in \mathit{Var} \cup \overline{\mathit{Var}}$ 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 can easily define a solution of our system.

Let us suppose that α 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, $1 \le j \le 3$, 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 $v(C) = T$. $\qquad \square$

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 (using Remark 1), we get the following result.

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

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.: Matching Modulo Associativity and Idempotency is NP-complete, Technical report RS-00-13, BRICS, Aarhus University (2000).

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

[10] 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.

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

[12] 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.

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

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

[15] 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.

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

[17] 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.

[18] 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.

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

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