On the Solvability Problem for Restricted Classes of Word Equations

Florin Manea[1], Dirk Nowotka[1], and Markus L. Schmid[2(✉)]

[1] Department of Computer Science, Kiel University, 24098 Kiel, Germany
flm@informatik.uni-kiel.de, nowotka@zs.uni-kiel.de
[2] Fachbereich IV – Abteilung Informatikwissenschaften,
Trier University, 54286 Trier, Germany
MSchmid@uni-trier.de

Abstract. We investigate the complexity of the solvability problem for restricted classes of word equations with and without regular constraints. For general word equations, the solvability problem remains NP-hard, even if the variables on both sides are ordered, and for word equations with regular constraints, the solvability problems remains NP-hard for variable disjoint (i.e., the two sides share no variables) equations with two variables, only one of which is repeated. On the other hand, word equations with only one repeated variable (but an arbitrary number of variables) and at least one non-repeated variable on each side, can be solved in polynomial-time.

Keywords: Word equations · Regular constraints · NP-hardness

1 Introduction

A *word equation* is an equation $\alpha = \beta$, such that α and β are words over an alphabet $\Sigma \cup X$, where Σ is a finite alphabet of *constants* and $X = \{x_1, x_2, x_3, \ldots\}$ is an enumerable set of *variables*. A *solution* to a word equation $\alpha = \beta$ is a morphism $h : (\Sigma \cup X)^* \to \Sigma^*$ that satisfies $h(\alpha) = h(\beta)$ and $h(b) = b$ for every $b \in \Sigma$. For example, $x\mathtt{ab}y = \mathtt{b}yx\mathtt{a}$ is a word equation with variables x, y, constants $\mathtt{a}, \mathtt{b}$ and h with $h(x) = \mathtt{bab}$, $h(y) = \mathtt{aba}$ is a solution, since $h(x\mathtt{ab}y) = \mathtt{bababab a} = h(\mathtt{b}yx\mathtt{a})$.

The *solvability problem* for word equations, i.e., to decide whether or not a given word equation has a solution, has a long history with the most prominent landmark being Makanin's algorithm [11] from 1977, which showed the solvability problem to be decidable (see Chapter 12 of [10] for a survey). While the complexity of Makanin's original algorithm was very high, it is nowadays known that the solvability problem is in PSPACE (see [8,12]) and NP-hard (in fact, it is even believed to be in NP). Word equations with only a single variable can be solved in linear time [7] and equations with two variables can be solved in time $\mathcal{O}(n^5)$ [2]; it is not known whether there exist polynomial-time algorithms for solving word equations with at most k variables, for some $k \geq 3$.

If we require $\beta \in \Sigma^*$, i. e., only one side of the equation is allowed to contain variables, then we obtain the *pattern matching problem with variables* (or simply *matching problem*, for short), where the term *pattern* refers to the part α that can contain variables. The matching problem is NP-complete and, compared to the solvability problem for word equations, many more tractability and intractability results are known (see [4, 5, 13]). More precisely, while restrictions of numerical parameters (e. g., number of variables, number of occurrences per variable, length of substitution words, alphabet size, etc.) make the problem either polynomial-time solvable in a trivial way (e. g., if the number of variables is bounded by a constant) or result in strongly restricted, but still NP-complete variants (see [4]), structural restrictions of the pattern (e. g., of the order of the variables) are more promising and can yield rich classes of patterns for which the matching problem can be solved in polynomial-time (see [13]). For example, the matching problem remains NP-complete if $|\Sigma| = 2$, every variable has at most two occurrences in α and every variable can only be replaced by the empty word or a single symbol (or instead by non-empty words of size at most 3). On the other hand, non-trivial and efficient polynomial-time algorithms exist (see [3]), if the patterns are *regular* (i. e., every variable has at most one occurrence), the patterns are *non-cross* (i. e., between any two occurrences of the same variable x no other variable different from x occurs) or the patterns have a bounded *scope coincidence degree* (i. e., the maximum number of scopes of variables that overlap is bounded, where the scope of a variable is the interval in the pattern where it occurs).

Technically, all these results can be seen as tractability and intractability results for restricted variants of the solvability problem (in fact, as it seems, all NP-hardness lower bounds for restricted variants of the solvability problem in the literature are actually NP-hardness lower bounds for the matching problem). However, these results are disappointing in terms of how much they provide us with a better understanding of the complexity of word equations, since in the matching problem the most crucial feature of word equations is missing, which is the possibility of having variables on both sides.

The aim of this paper is to transfer the knowledge and respective techniques of the matching problem to variants of the solvability problem for word equations that are *not* just variants of the matching problem. In particular, we investigate whether the structural restrictions mentioned above, which are beneficial for the matching problem, can be extended, with a comparable positive impact, to classes of word equations that have variables on both sides. We pay special attention to *regular constraints*, i. e., each variable x is accompanied by a regular language L_x from which $h(x)$ must be selected in a solution h. While Makanin's algorithm still works in the presence of regular constraints, it turns out that for more restricted classes of equations, the addition of regular constraints can drastically increase the complexity of the solvability problem.

2   Definitions

Let Σ be a finite alphabet of *constants* and let $X = \{x_1, x_2, x_3, \ldots\}$ be an enumerable set of *variables*. For any word $w \in (\Sigma \cup X)^*$ and $z \in \Sigma \cup X$,

we denote by $|w|_z$ the number of occurrences of z in w, by $\mathsf{var}(w)$ the set of variables occurring in w and, for every i, $1 \leq i \leq |w|$, $w[i]$ denotes the symbol at position i in w. A morphism $h : (\Sigma \cup X)^* \to \Sigma^*$ with $h(a) = a$ for every $a \in \Sigma$ is called a *substitution*. A *word equation* is a tuple $(\alpha, \beta) \in (\Sigma \cup X)^+ \times (\Sigma \cup X)^+$ (for the sake of convenience, we also write $\alpha = \beta$) and a *solution* to a word equation (α, β) is a substitution h with $h(\alpha) = h(\beta)$, where $h(\alpha)$ is the *solution word* (*of h*). A word equation is *solvable* if there exists a solution for it and the *solvability problem* is to decide for a given word equation whether or not it is solvable.

Let $\alpha \in (\Sigma \cup X)^*$. We say that α is *regular*[1], if, for every $x \in \mathsf{var}(\alpha)$, $|\alpha|_x = 1$; e. g., $ax_1bax_2cx_3bcax_4ax_5bb$ is regular. The word α is *non-cross* if between any two occurrences of the same variable x no other variable different from x occurs, e. g., $ax_1bax_1x_2ax_2x_2x_3x_3bx_4$ is non-cross, whereas $x_1bx_1x_2bax_3x_3x_4x_4bcx_2$ is not. A word equation (α, β) is regular or non-cross, if both α and β are regular or both α and β are non-cross, respectively. An equation (α, β) is *variable disjoint* if $\mathsf{var}(\alpha) \cap \mathsf{var}(\beta) = \emptyset$.

For a word equation $\alpha = \beta$ and an $x \in \mathsf{var}(\alpha\beta)$, a *regular constraint* (*for x*) is a regular language L_x and a solution h for $\alpha = \beta$ *satisfies* the regular constraint L_x if $h(x) \in L_x$. The solvability problem for word equations with regular constraints is to decide on whether an equation $\alpha = \beta$ with regular constraints L_x, $x \in \mathsf{var}(\alpha\beta)$, given as NFA, has a solution that satisfies all regular constraints. The *size* of the regular constraints is the sum of the number of states of the NFA. If the regular constraints are all of the form Γ^*, for some $\Gamma \subseteq \Sigma$, then we call them *word equations with individual alphabets*.

A word equation $\alpha = \beta$ along with an $m \in \mathbb{N}$ is a *bounded word equation*. The problem of *solving a bounded word equation* is then to decide on whether there exists a solution h for $\alpha = \beta$ with $|h(x)| \leq m$ for every $x \in \mathsf{var}(\alpha\beta)$.

For an $\alpha \in (\Sigma \cup X)^*$, $L(\alpha) = \{h(\alpha) \mid h \text{ is a substitution}\}$ is the *pattern language of α*.

3   Regular and Non-cross Word Equations

For the matching problem, the restriction of regularity implies that every variable has only one occurrence in the equation, which makes the solvability problem trivial (in fact, it boils down to the membership problem for a very simple regular language). However, word equations in which both sides are regular can still have repeated variables, although the maximum number of occurrences per variable is 2 (i. e., regular equations are restricted variants of quadratic equations (see, e. g., [14])) and these two occurrences must occur on different sides. Unfortunately, we are neither able to show NP-hardness nor to find a polynomial-time algorithm for the solvability problem of regular word equations.

---

[1] The use of the term regular in this context has historical reasons: the matching problem has been first investigated in terms of so-called *pattern languages*, i. e., the set of all words that match a given pattern $\alpha \in (\Sigma \cup X)^*$, which are regular languages if α is regular.

Open Problem 1. *Can regular word equations be solved in polynomial-time?*

As we shall see later, solving a system of two regular equations is NP-hard (Corollary 6), solving regular equations with regular constraints is even PSPACE-complete (Theorem 7), and solving bounded regular equations or regular equations with individual alphabets is NP-hard (Corollaries 17 and 19, respectively), as well.

On the positive side, it can be easily shown that regular word equations can be solved in polynomial-time, if we additional require them to be variable disjoint (which simply means that no variable is repeated in the whole equation). More precisely, in this case, we only have to check emptiness for the intersection of the pattern languages described by the two sides of the equations (which are regular languages).

Next, we show the stronger result that polynomial-time solvability is still possible if at most one variable is repeated, and each side contains at least one of the non-repeating variables.

Theorem 2. *Word equations with only one repeated variable, and each side containing at least one non-repeating variable, can be solved in polynomial time.*

If we allow an arbitrary number of occurrences of each variable, but require them to be sorted on both sides on the equation, where the sorting order might be different on the two sides, then we arrive at the class of non-cross word equations. As for the class of regular patterns, also for non-cross patterns the matching problem can be solved efficiently. However, as we shall see next, for non-cross equations, the solvability problem becomes NP-hard.

Theorem 3. *Solving non-cross word equations is NP-hard.*

We prove this theorem by a reduction[2] from a graph problem, for which we first need the following definition.

Let $\mathcal{G} = (V, E)$ be a graph with $V = \{t_1, t_2, \ldots, t_n\}$. A vertex s is the *neighbour* of a vertex t if $\{t, s\} \in E$ and the set $N_{\mathcal{G}}[t] = \{s \mid \{t, s\} \in E\} \cup \{t\}$ is called the (*closed*) *neighbourhood* of t. If, for some $k \in \mathbb{N}$, every vertex of $\mathcal{G}$ has exactly k neighbours, then $\mathcal{G}$ is *k-regular*. A *perfect code* for $\mathcal{G}$ is a subset $C \subseteq V$ with the property that, for every $t \in V$, $|N_{\mathcal{G}}[t] \cap C| = 1$. Next, we define the problem to decide whether or not a given 3-regular graph has a perfect code, which is NP-complete (see [9]):

3-REGULAR PERFECT CODE (3RPERCODE)
Instance: A 3-regular graph $\mathcal{G}$.
Question: Does $\mathcal{G}$ contain a perfect code?

We now define a reduction from 3RPERCODE. To this end, let $\mathcal{G} = (V, E)$ be a 3-regular graph with $V = \{t_1, t_2, \ldots, t_n\}$ and, for every i, $1 \leq i \leq n$, N_i is the neighbourhood of t_i. Since the neighbourhoods play a central role, we shall define them in a more convenient way. For every r, $1 \leq r \leq 4$, we use

---

[2] We will also use minor modifications later on of this reduction in order to conclude corollaries of Theorem 3.

a mapping $\wp_r : \{1, 2 \dots, n\} \to \{1, 2 \dots, n\}$ that maps an $i \in \{1, 2 \dots, n\}$ to the index of the rth vertex of neighbourhood N_i, i.e., for every i, $1 \leq i \leq n$, $N_i = \{t_{\wp_1(i)}, t_{\wp_2(i)}, t_{\wp_3(i)}, t_{\wp_4(i)}\}$. Obviously, the mappings $\wp_r$, $1 \leq r \leq 4$, imply a certain order on the vertices in the neighbourhoods, but, since our constructions are independent of this actual order, any order is fine.

We transform $\mathcal{G}$ into a word equation with variables $\{x_{i,j} \mid 1 \leq i, j \leq n\} \cup \{y_i, y'_i \mid 1 \leq i \leq n\}$ and constants from $\Sigma = \{\star, \diamond, \overline{\diamond}, \odot, \#, \mathsf{a}\}$. For every i, j, $1 \leq i, j \leq n$, the variable $x_{i,j}$ represents $t_i \in N_j$. For every i, $1 \leq i \leq n$, we define

$$\alpha_i = x_{\wp_1(i),i} \dots x_{\wp_4(i),i}, \qquad \alpha'_i = \# \, \mathsf{a}^8 \, \# \, \# \, ,$$
$$\beta_i = \mathsf{a}, \qquad \beta'_i = y_i \, \# (x_{i,\wp_1(i)})^2 \dots (x_{i,\wp_4(i)})^2 \, \# \, y'_i$$

and

$$
\begin{array}{ccccccccccccc}
u & = & \alpha_1 & \star & \dots & \star & \alpha_n & \star & \odot & \overline{\diamond} & \alpha'_1 & \diamond & \dots & \diamond & \alpha'_n, \\
v & = & \beta_1 & \star & \dots & \star & \beta_n & \star & \odot & \overline{\diamond} & \beta'_1 & \diamond & \dots & \diamond & \beta'_n.
\end{array}
$$

Proposition 4. *The words u and v are non-cross and can be constructed from $\mathcal{G}$ in polynomial time.*

Lemma 5. *The graph $\mathcal{G}$ has a perfect code if and only if (u, v) has a solution.*

Proof. For the sake of convenience, let $u = u_1 \odot u_2$ and $v = v_1 \odot v_2$. We start with the *only if* direction. For a perfect code C of $\mathcal{G}$, we construct a substitution h with $h(u) = h(v)$ in the following way. For every i, $1 \leq i \leq n$, we define $h(x_{i,\wp_r(i)}) = \mathsf{a}$, $1 \leq r \leq 4$, if $t_i \in C$, and $h(x_{i,\wp_r(i)}) = \varepsilon$, otherwise. Thus, for every i, $1 \leq i \leq n$, $h((x_{i,\wp_1(i)})^2 \dots (x_{i,\wp_4(i)})^2) \in \{\mathsf{a}^8, \varepsilon\}$, which implies that $h(y_i)$ and $h(y'_i)$ can be defined such that $h(\beta'_i) = h(\alpha'_i)$. Consequently, $h(v_2) = h(u_2)$. Since C is a perfect code, for every i, $1 \leq i \leq n$, there is an r, $1 \leq r \leq 4$, such that $t_{\wp_r(i)} \in C$ and $t_{\wp_{r'}(i)} \notin C$, $1 \leq r' \leq 4$, $r \neq r'$. Therefore, $h(x_{\wp_1(i),i} x_{\wp_2(i),i} x_{\wp_3(i),i} x_{\wp_4(i),i}) = h(x_{\wp_r(i),i}) = \mathsf{a}$, which means that $h(\alpha_i) = h(\beta_i)$. Since this particularly implies $h(u_1) = h(v_1)$, we can conclude $h(u) = h(v)$.

In order to prove the *if* direction, we assume that there exists a solution h.

Claim: If $h(u_1) = h(v_1)$ and $h(u_2) = h(v_2)$, then $\mathcal{G}$ has a perfect code.

Proof of Claim: From $h(u_1) = h(v_1)$, we can directly conclude that, for every i, $1 \leq i \leq n$, $h(\alpha_i) = \beta_i$, which means that exactly one of the variables $x_{\wp_1(i),i}, x_{\wp_2(i),i}, x_{\wp_3(i),i}, x_{\wp_4(i),i}$ is mapped to a, while the others are mapped to ε. From $h(v_2) = h(u_2)$ it follows that, for every i, $1 \leq i \leq n$, $h(\beta'_i) = \alpha'_i$. Next, we observe that, for every i, $1 \leq i \leq n$, due to the symbols $\#$ in β'_i and α'_i, $h((x_{i,\wp_1(i)})^2 \dots (x_{i,\wp_4(i)})^2) \in \{\mathsf{a}^8, \varepsilon\}$. Since each of the variables $x_{i,\wp_1(i)}, x_{i,\wp_2(i)}, x_{i,\wp_3(i)}, x_{i,\wp_4(i)}$ are mapped to either a or ε, this implies that either all of these variables are erased or all of them are mapped to a. Let C be the set of exactly the vertices $t_i \in V$ for which $h(x_{i,\wp_1(i)}) = h(x_{i,\wp_2(i)}) = h(x_{i,\wp_3(i)}) = h(x_{i,\wp_4(i)}) = \mathsf{a}$. For every neighbourhood $V_j = \{t_{\wp_1(j)}, t_{\wp_2(j)}, t_{\wp_3(j)}, t_{\wp_4(j)}\}$,

$1 \leq j \leq n$, $h(x_{\wp_1(j),j}\, x_{\wp_2(j),j}\, x_{\wp_3(j),j}\, x_{\wp_4(j),j})$ is mapped to a, which implies that for some r, $1 \leq r \leq 4$, $h(x_{\wp_r(j),j}) = \mathsf{a}$; thus, $t_{\wp_r(j)} \in C$. Furthermore, $h(x_{\wp_{r'}(j),j}) = \varepsilon$, $1 \leq r' \leq 4$, $r \neq r'$, which means that $t_{\wp_{r'}(j)} \notin C$, $1 \leq r' \leq 4$, $r \neq r'$. Consequently, C is a perfect code. $\hfill$ *(Claim)* $\quad\square$

It remains to show that a solution h necessarily satisfies $h(u_1) = h(v_1)$ and $h(u_2) = h(v_2)$. Let w be the solution word of h. We first recall that, since $v_1, u_2 \in \Sigma^*$, $h(v_1) = v_1$ and $h(u_2) = u_2$, which particularly means that $v_1 \odot$ is a prefix and $\odot u_2$ is a suffix of w. If $|w|_\odot = 1$, then $w = v_1 \odot u_2$ and therefore $h(u_1) = h(v_1)$ and $h(u_2) = h(v_2)$. If, on the other hand, $|w|_\odot \geq 2$, then $w = v_1 \odot \gamma \odot u_2$. If $\gamma = \varepsilon$, then $w = v_1 \odot \odot u_2$, which is a contradiction, since w must contain the factor $\star \odot \overline{\odot}$. From $h(u_2) = u_2$ and $h(v_1) = v_1$ it follows that $h(u_1) = v_1 \odot \gamma =$ and $h(v_2) = \gamma \odot u_2$. The factor v_2 starts with an occurrence of $\overline{\odot}$ and since γ is a non-empty prefix of $h(v_2)$, this means that $|\gamma|_{\overline{\odot}} = k \geq 1$. Moreover, γ is also a suffix of $h(u_1)$ and since $|u_1|_{\overline{\odot}} = 0$, this implies that there are variables $z_1, z_2, \ldots, z_\ell \in \mathsf{var}(u_1)$, $1 \leq \ell \leq k$, with $\sum_{i=1}^{\ell} |h(z_i)|_{\overline{\odot}} \geq k$. Since each of these variables z_i, $1 \leq i \leq \ell$, is repeated twice in v_2 and since $|v_2|_{\overline{\odot}} = 1$, we can conclude that $|h(v_2)|_{\overline{\odot}} \geq 2k + 1$. In the suffix $\odot u_2$ of $h(v_2)$, there is only one occurrence of $\overline{\odot}$, which implies that $|\gamma|_{\overline{\odot}} \geq 2k$. Since $k \geq 1$, this is clearly a contradiction to $|\gamma|_{\overline{\odot}} = k$. $\hfill\square$

The equation obtained by the reduction from above has the form $u_1 \odot u_2 = v_1 \odot v_2$, where in a solution h, $h(u_1) = h(v_1)$ and $h(u_2) = h(v_2)$. In order to achieve this synchronisation between the two left parts and between the two right parts, we need to repeat variables in v_2. However, we can as well represent $u_1 \odot u_2 = v_1 \odot v_2$ as a system of two equations $u_1 = v_1$ and $u_2 = v_2$ and, since the synchronisation of the left parts and the right parts is now enforced by the fact that we regard them as two separate equations, we can get rid of the repeated variables in v_2, which makes the two equations regular.

Corollary 6. *The problem of checking solvability of a system of 2 regular word equations $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$ with β_1, $\beta_2 \in \Sigma^*$ is* NP-*hard.*

We conclude this section by stressing the fact that the non-cross equation from the reduction above is "almost regular", i. e., one side is regular, while for the other the maximum number of occurrences per variable is 2. However, we were not able to get rid of these repeated variables, which suggests that a hardness reduction for the regular case needs to be substantially different or regular word equations can be solved in polynomial-time.

4   Word Equations with Regular Constraints

In practical scenarios, it seems rather artificial that we only want to find just any solution for a word equation and we are fine with whatever sequence of symbols the variables will be substituted with. It is often more realistic that the variables have a well-defined domain from which we want the solution to select the words.

This motivates the addition of regular constraints to word equations, as defined in Sect. 2, for which we investigate the solvability problem in this section.

As mentioned in Sect. 1, regular constraints can be easily incorporated into algorithms for the general solvability problem. However, while it is open whether solving general word equations is hard for PSPACE, for word equations with regular constraints, this can be easily shown, even for regular equations.

Theorem 7. *Solving word equations with regular constraints is* PSPACE-*complete, even for regular equations.*

Proof. We can reduce the PSPACE-hard intersection emptiness problem for NFA, i.e., deciding for given NFA M_i, $1 \leq i \leq n$, whether or not $\bigcap_{i=1}^{n} L(M_i) = \emptyset$. To this end, let $M_1, \ldots, M_n$ be NFA over some alphabet Σ with $\# \notin \Sigma$. We define $\alpha = x_1 \# x_2 \# \ldots \# x_{n-1}$ and $\beta = x_2 \# x_3 \# \ldots \# x_n$, and we define the regular constraints $L_{x_i} = L(M_i)$. We note that the equation $\alpha = \beta$ is regular.

If there exists a word $w \in \bigcap_{i=1}^{n} L(M_i)$, then h with $h(x_i) = w$, $1 \leq i \leq n$, is a solution for $\alpha = \beta$, since $h(\alpha) = (w\#)^{n-2}w = h(\beta)$, and, furthermore, h satisfies the regular constraints. Let h be a solution for $\alpha = \beta$ that satisfies the regular constraints. This implies that $h(x_1)\#h(x_2)\#\ldots\#h(x_{n-1}) = h(x_2)\#h(x_3)\#\ldots\#h(x_n)$ and, since $|h(x_i)|_\# = 0$, $1 \leq i \leq n$, $h(x_1) = h(x_2) = \ldots = h(x_n)$ follows. Thus, $h(x_1) \in \bigcap_{i=1}^{n} L(M_i)$. □

Recall that we mentioned in Sect. 3 that word equations without repeated variables can be solved in polynomial time. This also holds for word equations with regular constraints.

Theorem 8. *Solving word equations with regular constraints and without repeated variables can be done in polynomial time.*

Word equations with only one variable can be solved in linear time (see Jeż [7]). If we add regular constraints to equations with only one variable, then the solvability problem is still in P.

Theorem 9. *Solving word equations with regular constraints and with only one variable can be done in polynomial time.*

Word equations with two variables can be solved in polynomial-time (see [2]). We shall see next that for word equations with regular constraints this is no longer the case (assuming $P \neq NP$). More precisely, solving equations with two variables and regular constraints is NP-hard, even if only one variable is repeated and the equations are variable disjoint. Moreover, we can show that the existence of an algorithm solving word equations with two variables and with regular constraints in time $2^{o(n+m)}$ (where n is the length of the equation and m is the size of the regular constraints) is very unlikely, since it would refute the well-known *exponential-time hypothesis* (ETH, for short).

We conduct a linear reduction from 3-SAT to the problem of solving word equations with regular constraints.[3] Let $C = \{c_1, c_2, \ldots, c_m\}$ be a Boolean formula in conjunctive normal form (CNF) with 3 literals per clause over the variables $\{v_1, v_2, \ldots, v_n\}$. We first transform C into a CNF C' such that C is satisfiable if and only if C' has an assignment that satisfies exactly one literal per clause (in the following, we call such an assignment a 1-*in*-3 *assignment*). For every i, $1 \leq i \leq m$, we replace $c_i = \{y_1, y_2, y_3\}$ by 5 new clauses

$$\{y_1, z_1, z_2\}, \{y_2, z_2, z_3\}, \{z_1, z_3, z_4\}, \{z_2, z_5, z_6\}, \{y_3, z_5\},$$

where z_i, $1 \leq i \leq 6$, are new variables.[4] We note that C' has $5m$ clauses and $n + 6m$ variables. Next, we obtain C'' from C' by adding, for every i, $1 \leq i \leq n$, a new clause $\{v_i, \widehat{v_i}\}$, where $\widehat{v_i}$ is a new variable, and we replace all occurrences of $\overline{v_i}$ (i.e., the variable v_i in negated form) by $\widehat{v_i}$.

The following proposition can be easily verified.

Proposition 10. *There is a satisfying assignment for C if and only if C'' has a 1-in-3 assignment. Furthermore, C'' has no negated variables, C'' has $5m + n$ clauses and $2n + 6m$ variables.*

For the sake of convenience, we set $n' = 2n + 6m$, $m' = 5m + n$, $C'' = \{c'_1, c'_2, \ldots, c'_{m'}\}$ and let $\{v'_1, v'_2, \ldots, v'_{n'}\}$ be the variables of C''. Furthermore, for every i, $1 \leq i \leq n'$, let k_i be the number of occurrences of variable v'_i in C''.

Next, we transform C'' into a word equation with regular constraints as follows. Let $\Sigma = \{v'_1, v'_2, \ldots, v'_{n'}, \#\}$ and let the equation $\alpha = \beta$ be defined by $\alpha = (x_1 \#)^{n'-1} x_1$ and $\beta = x_2$. For the variables x_1 and x_2, we define the following regular constraints over Σ:

$$L_{x_1} = \{w \mid |w| = m', w[i] \in c'_i, 1 \leq i \leq m'\},$$
$$L_{x_2} = \{u_1 \# u_2 \# \ldots \# u_{n'} \mid u_i \in (\Sigma \setminus \{\#\})^*, |u_i|_{v'_i} \in \{k_i, 0\}, 1 \leq i \leq n'\}.$$

Proposition 11. *There are DFA M_{x_1} and M_{x_2} accepting the languages L_{x_1} and L_{x_2}, respectively, with $5m + n + 2$ and $21m + 5n + 1$ states, respectively.*

By definition, only NFA are required to represent the regular constraints, but our use of DFA here points out that the following hardness result (and the ETH lower bound) also holds for the case that we require the regular constraints to be represented by DFA. So the hardness of the problem does not result from the fact that NFA can be exponentially smaller than DFA.

Lemma 12. *The Boolean formula C'' has a 1-in-3 assignment if and only if $\alpha = \beta$ has a solution that satisfies the regular constraints L_{x_1} and L_{x_2}.*

---

[3] In order to prove NP-hardness, a simpler production would suffice, but we need a linear reduction in order to obtain the ETH lower bound.

[4] Note that this is just the reduction used by Schaefer [15] in order to reduce 3-SAT to 1-IN-3 3-SAT. We recall it here to observe that this reduction is linear.

Proof. We start with the *only if* direction. To this end, let $\pi : \{v_1', v_2', \ldots, v_n'\} \to \{0, 1\}$ be a 1-in-3 assignment for C'', where, for every i, $1 \leq i \leq m'$, y_i is the unique variable with $y_i \in c_i'$ and $\pi(y_i) = 1$. Let h be a substitution defined by $h(x_1) = y_1 y_2 \ldots y_{m'}$ and $h(x_2) = (h(x_1) \#)^{n-1} h(x_1)$. Obviously, h is a solution for $\alpha = \beta$, $h(x_1) \in L_{x_1}$ and, since every v_i' has either 0 occurrences in $h(x_1)$ (in case that $\pi(v_i') = 0$) or k_i occurrences (in case that $\pi(v_i') = 1$), also $h(x_2) \in L_{x_2}$.

For the *if* direction, let h be a solution for $\alpha = \beta$ that satisfies the regular constraints. Consequently, $h(x_1) = y_1 y_2 \ldots y_{m'}$, where $y_i \in c_i'$, $1 \leq i \leq m'$, and, furthermore, for every i, $1 \leq i \leq n$, $|h(x_2)|_{v_i'} \in \{k_i, 0\}$. This directly implies that $\pi : \{v_1', v_2', \ldots, v_n'\} \to \{0, 1\}$, defined by $h(v_i') = 1$ if $|h(x_2)|_{v_i'} = k_i$ and $h(v_i') = 0$ if $|h(x_2)|_{v_i'} = 0$, is a 1-in-3 assignment for C''.   □

The exponential-time hypothesis, mentioned above, roughly states that 3-SAT cannot be solved in subexponential-time. For more informations on the ETH, the reader is referred to Chapter 14 of the textbook [1]. For our application of the ETH, it is sufficient to recall the following result.

Theorem 13 (Impagliazzo et al. [6]). *Unless ETH fails, 3-SAT cannot be solved in time $2^{o(n+m)}$, where n is the number of variables and m is the number of clauses.*

The reduction from above implies that a subexponential algorithm for solving word equations with two variables and regular constraints can be easily turned into a subexponential algorithm for 3-SAT; thus, the existence of such an algorithm contradicts ETH.

Theorem 14. *Solving word equations with two variables and with regular constraints is NP-hard, even if only one variable is repeated and the equations are variable disjoint. Furthermore, unless ETH fails, such word equations cannot be solved in time $2^{o(n+m)}$ (where n is the length of the equation and m is the size of the regular constraints).*

4.1   Bounded Word Equations

We first note that bounded word equations can be considered as a special case of word equations with regular constraints, since the bound m functions as regular constraints of the form $\{w \in \Sigma^* \mid |w| \leq m\}$ for every variable. However, there is an important difference: the length of a binary encoding of m is logarithmic in the size of an NFA for $\{w \in \Sigma^* \mid |w| \leq m\}$; thus, NP-hardness of a class of bounded word equations does not necessarily carry over to word equations with regular constraints. As usual, we call the solvability problem for a class of bounded word equations NP-*hard in the strong sense*, if the NP-hardness remains if the bound m is given in unary.

Theorem 15. *Solving bounded word equations is NP-hard (in the strong sense), even for equations $\alpha = \beta$ satisfying $|\mathsf{var}(\alpha)| = 1$, $\mathsf{var}(\alpha) \cap \mathsf{var}(\beta) = \emptyset$ and β is regular.*

Proof. We reduce from the shortest common superstring problem, i. e., deciding for given $k \in \mathbb{N}$ and strings $v_1, v_2, \ldots, v_n \in \Sigma^*$ whether there is a string u with $|u| \leq k$ that contains each v_i as a factor. Let $v_1, v_2, \ldots, v_n \in \Sigma^*$, $k \in \mathbb{N}$ be an instance of the shortest common superstring problem. Furthermore, let $\#$ be a new symbol, i. e., $\# \notin \Sigma$. We construct a word equation $\alpha = \beta$, where

$$
\begin{aligned}
\alpha &= x \quad \# \quad x \quad \# \quad \ldots \quad \# \quad x\,, \\
\beta &= y_1 v_1 y_1' \quad \# \quad y_2 v_2 y_2' \quad \# \quad \ldots \quad \# \quad y_n v_n y_n'\,.
\end{aligned}
$$

Furthermore, we let k be the upper bound on the substitution word lengths.

If there exists a word $w \in \Sigma^*$ with $|w| \leq k$ and, for every i, $1 \leq i \leq n$, $w = u_i v_i u_i'$, then we define a substitution h by $h(x) = w$, $h(y_i) = u_i$ and $h(y_i') = u_i'$, $1 \leq i \leq n$. Obviously, h satisfies the length bound and, for every i, $1 \leq i \leq n$, $h(x) = h(y_i v_i y_i')$; thus, $h(\alpha) = h(\beta)$.

Let h be a solution for $\alpha = \beta$ that satisfies the length bound. We observe that since $h(\beta)$ contains every v_i as a factor, also $h(\alpha) = h(x) \# h(x) \# \ldots \# h(x)$ contains every v_i as a factor and, furthermore, since $|v_i|_\# = 0$, $1 \leq i \leq n$, every v_i is also a factor of $h(x)$. Consequently, $|h(x)| \leq k$ and $h(x)$ contains every v_i, $1 \leq i \leq n$, as a factor.

For the shortest common superstring problem, we can assume that $k \leq \sum_{i=1}^{n} |v_i|$, since otherwise $v_1 v_2 \ldots v_n$ would also be a solution. Consequently, we can assume that k is given in unary, which means that solving bounded word equations of the form mentioned in the statement of the theorem is NP-hard in the strong sense.

Due to the strong NP-hardness in Theorem 15, we can conclude the following.

Corollary 16. *Solving word equations with regular constraints is NP-hard, even for equations $\alpha = \beta$ satisfying $|\operatorname{var}(\alpha)| = 1$, $\operatorname{var}(\alpha) \cap \operatorname{var}(\beta) = \emptyset$ and β is regular.*

By using 1 as the bound on the substitution words and by a minor modification of the reduction for Theorem 3, we can obtain a hardness reduction for bounded regular word equations.

Corollary 17. *Solving bounded regular word equations is NP-hard.*

4.2   Individual Alphabets

The least restrictive regular constraints are probably constraint languages of the form Γ^* for some $\Gamma \subseteq \Sigma$, i. e., word equations with individual alphabets, which we shall investigate in this section.

We first note that if $|\Sigma| = 1$, then general word equations and word equations with individual alphabets coincide and, furthermore, the solvability problem for word equations can be solved in polynomial-time, if $|\Sigma| = 1$.

Theorem 18. *Solving word equations can be done in polynomial time if $|\Sigma| = 1$.*

However, if $\Sigma = \{a, b\}$ and $\{a\}$ is used as individual alphabet for all variables, then solving word equations becomes NP-hard again, simply because the matching problem is already NP-hard for this case (as can be easily concluded from the reduction of Lemma 5 in [5]).

By using individual alphabets, the reduction for Theorem 3 can be easily transformed to a hardness reduction for the solvability problem of regular equations with individual alphabets.

Corollary 19. *Solving regular word equations with individual alphabets is* NP-*hard.*

5   Conclusions

We conclude this work by summarising our main results and by suggesting some further research directions.

First of all, the polynomial-time decidability of the matching problem for non-cross patterns does not carry over to non-cross equations (which also means that the concept of the scope coincidence degree, briefly mentioned in Sect. 1, will not help, since it is a generalisation of the non-cross concept), while for regular equations, this is still open (see Open Problem 1), which constitutes the most important question left open in this work.

As soon as we allow regular constraints, it is possible to prove hardness results for strongly restricted variants of the solvability problem, often including the regular case. More precisely, for general regular constraints, the solvability problem is PSPACE-complete, even for regular equations (Theorem 7), and NP-hard for variable disjoint equations with only one repeated variable and two variables in total (Theorem 14). Especially this latter result, for which we can also obtain an ETH lower bound, points out a drastic difference in terms of complexity between general word equations and equations with regular constraints: both the tractable cases of equations with only two variables or with only one repeated variable and at least one non-repeated variable on both sides (Theorem 2) become NP-hard if we allow regular constraints.[5] Moreover, the case with only one repeated variable remains intractable, even if the constraints are only bounding the length of the substitution words (Theorem 15). In particular, even if it turns out that, for some k, $k \geq 3$, or even for all constant k, general word equations with at most k variables can be solved in polynomial-time, Theorem 14 severely limits their practical application, since it shows that these polynomial-time algorithms cannot cope with regular constraints (unless P = NP).

As for regular equations, allowing a system of only two equations (and no further constraints), allowing bounds on the substitution words or allowing individual alphabets is enough to make the solvability problem NP-hard.

Our choice of restrictions for word equations is motivated by polynomial-time solvable cases of the matching problem. In order to obtain tractable classes

---

[5] For the latter case, note that in the reduction of Theorem 14, we can add a non-repeated variable with regular constraint $\emptyset$ to the left side.

of word equations, it might be worthwhile to strengthen the concept of non-cross and regularity by requiring $\alpha\beta$ to be regular or non-cross, instead of only requiring this for α and β separately. Another possible further restriction would be to require the order of the variables on the left and on the right side to be the same (e. g., $x_1\mathsf{ab}x_2\mathsf{c}x_3 = x_1\mathsf{c}x_3$ is *ordered regular*, while $x_1\mathsf{ab}x_2\mathsf{c}x_3 = x_3\mathsf{c}x_2$ is not). In this regard, it is interesting to note that the patterns produced by the reduction of Theorem 3 are not ordered non-cross (and not ordered regular for the corresponding corollaries), while Theorem 7, the PSPACE-completeness of solving word equations with regular constraints, also holds for ordered regular equations. Additionally requiring $\mathsf{var}(\alpha) = \mathsf{var}(\beta)$ for ordered regular equations would be a further restriction that might be useful.

References

1. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer International Publishing AG, Cham (2015)
2. Dąbrowski, R., Plandowski, W.: Solving two-variable word equations. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 408–419. Springer, Heidelberg (2004)
3. Fernau, H., Manea, F., Mercaş, R., Schmid, M.L.: Pattern matching with variables: fast algorithms and new hardness results. In: Proceedings of 32nd Symposium on Theoretical Aspects of Computer Science, STACS 2015, Leibniz International Proceedings in Informatics (LIPIcs), vol. 30, pp. 302–315 (2015)
4. Fernau, H., Schmid, M.L.: Pattern matching with variables: a multivariate complexity analysis. Inf. Comput. **242**, 287–305 (2015)
5. Fernau, H., Schmid, M.L., Villanger, Y.: On the parameterised complexity of string morphism problems. Theory of Computing Systems (2015). http://dx.doi.org/10.1007/s00224-015-9635-3
6. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**, 512–530 (2001)
7. Jeż, A.: One-variable word equations in linear time. Algorithmica **74**, 1–48 (2016)
8. Jeż, A.: Recompression: a simple and powerful technique for word equations. J. ACM **63**(1), 4:1–4:51 (2016)
9. Kratochvíl, J., Křivánek, M.: On the computational complexity of codes in graphs. In: Chytil, M.P., Koubek, V., Janiga, L. (eds.) MFCS 1988. LNCS, vol. 324, pp. 396–404. Springer, Heidelberg (1988)
10. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
11. Makanin, G.: The problem of solvability of equations in a free semigroup. Matematicheskii Sbornik **103**, 147–236 (1977)
12. Plandowski, W.: An efficient algorithm for solving word equations. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC 2006, pp. 467–476 (2006)

13. Reidenbach, D., Schmid, M.L.: Patterns with bounded treewidth. Inf. Comput. **239**, 87–99 (2014)
14. Robson, J.M., Diekert, V.: On quadratic word equations. STACS 1999. LNCS, vol. 1563, pp. 217–226. Springer, Heidelberg (1999)
15. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of 10th Annual ACM Symposium on Theory of Computing, STOC 1978, pp. 216–226. ACM (1978)