Word Equations in the Context of String Solving

Joel D. Day(✉)

Loughborough University, Loughborough, UK
J.Day@lboro.ac.uk

Abstract. String solvers are tools for automatically reasoning about words over some finite alphabet. They are commonly used to perform analyses of string manipulating programs. A fundamental problem which string solvers need to address is solving word equations, usually in combination with additional constraints involving e.g. string lengths or regular languages. In this article, a survey of results on the topic of word equations is presented with an emphasis on recent results which are relevant to the theoretical foundations of string solvers.

Keywords: Word equations · String constraints · String solving

1 Introduction

Describing one object as a combination others, whether expressed concretely or abstractly, is one of the most fundamental things we do in mathematics. Naturally, we can also do this for words w over some alphabet Σ. By introducing a set X of variables, we can express w as the concatenation of smaller words, some of which are not known explicitly. For example, if $x, y \in X$ are variables and $a, b \in \Sigma$, we can express that w is a word containing an occurrence of aba (so consisting of an unknown word followed by aba followed by another unknown word) by writing w as $xabay$.

Word equations arise when we have multiple ways of expressing the same word in this way. For example, we might describe a word containing both ab and ba via the word equation $x_1 aby_1 \doteq x_2 bay_2$. Formally, a word equation is a pair $(U, V) \in (X \cup \Sigma)^*$ which we usually write as $U \doteq V$. Its solutions are substitutions of the variables for words in Σ^* which identify the two sides. Formally, solutions are modelled by morphisms $h : (X \cup \Sigma)^* \to \Sigma^*$ satisfying $h(a) = a$ for all $a \in \Sigma$ and such that $h(U) = h(V)$.

Perhaps unsurprisingly given their fundamental nature, many natural and well-studied problems related to words can be expressed in terms of word equations. We have already seen how to express the *pattern matching problem*, namely the property that a concrete word (aba) occurs as a factor of some larger word $w \in \Sigma^*$ via the equation $w \doteq xabay$. Expressing that a concrete word occurs as a (scattered) subsequence of w can be done by adding further variables $z_1, z_2, \ldots$ between each of the letters. In the case of aba, we get the equation

$w \doteq xaz_1bz_2ay$. More generally, word equations of the form $w \doteq V$ where $w \in \Sigma^*$ and $V \in (X \cup \Sigma)^*$ correspond exactly to the *membership problem for pattern languages* (also called *pattern matching with variables*). For a constant $k \in \mathbb{N}$, the relation "x is a length-k scattered subsequence of w" where x is a variable can also be expressed, meaning that the k-spectra (see, e.g. [67]) of words can also be expressed as solution-sets to word equations. On the other hand, it follows from [41] that the property "x is a scattered subsequence of y" where both x and y are variables and therefore not of bounded length cannot be expressed via word equations.

The problems mentioned above have been well studied independently from word equations and are substantial areas of research in their own right. Arguably the most interesting and complex cases of word equations are when the variables occur in such a way that they form cyclic dependencies. For example, the equation $xy \doteq yx$ for variables x and y, often referred to as the commutation equation, is solved by a substitution if and only if x and y are substituted for repetitions of the same word w [61]. The equations $xyz = zux$ and $xaby \doteq ybax$ are examples whose solution-sets' descriptions are much more involved, and which have strong connections to the Fibonacci, and Standard and Sturmian words respectively [20,49].

The explicit study of word equations (equivalently equations in a free monoid or semigroup) can be traced back as far as A. A. Markov in the 1940s, although connections to Diophantine equations mean that in some sense, they have been studied indirectly for much longer [32]. Originally it was hoped that the satisfiability problem - whether or not a given word equation has a solution - might provide an means of connecting Diophantine equations with the computations of Turing machines in such a way as to allow for a proof that solving the former is undecidable and thus settling Hilbert's famous 10th problem. However this turned out not to be feasible with Makanin famously providing an algorithm for the satisfiability problem for word equations in 1977 [64]. Since then, several further algorithms have been presented by Plandowski and Rytter [76], Plandowski [74] and Jeż [52]. The latter, based on the *Recompression* technique resulted in a considerably simpler proof of correctness and has since been use to improve the PSPACE complexity upper bound given by Plandowski's algorithm to non-deterministic linear space [53].

The positive result of Makanin became highly influential in combinatorial group theory: in a series of results ultimately resolving Tarski's Conjectures, it was first adapted to work for equations in free groups by Makanin himself [63,65], before being used by Razborov to provide a method for describing all solutions to equations in free groups via so-called Makanin-Razborov diagrams [78]. More recently, algorithms for solving word equations have been extended to work in a range of algebraic structures such as hyperbolic groups [19,23] and partially commutative groups [35].

There has also been much interest in the free monoid case (on which we concentrate in the current paper) from several perspectives. Constructions exist (see e.g. [55,61] for reducing the satisfiability of Boolean combinations of word equations to the satisfiability of a single word equation. Consequently, Makanin's result extends to the existential theory of a free semigroup or monoid. Further results on logics involving words/concatenation can be found e.g. in Büchi and Senger [14] where the notion of definability is considered, and Quine [77] and Durnev [36] who show undecidability when quantifier alternations are allowed. Further undecidability results considering logics involving words and additional predicates (such as "is a scattered subsequence of" and "is abelian-equivalent to") can be found in [26,41]. The expressive power of word equations in defining relations and languages via (projections of) their solution-sets is considered in detail in [55] where some powerful tools are given for showing inexpressibility based on the notion of a *synchronising factorisation*.

In the field of combinatorics on words, constant-free word equations (equations of the form $U \doteq V$ where $U, V \in X^*$) have been studied extensively. In addition to solving specific equations or families of equations [21,43,46,70,80], a major topic of ongoing research involves *independent systems* of word equations - systems such that removing any one of the equations leads to a strictly larger set of solutions [20,42,48,56,71]. A connection between constant-free word equations and the general case is shown in [81].

From an algorithms and complexity perspective, it is easily seen that the satisfiability problem for word equations is NP-hard. Indeed, the result follows directly from NP-completeness of the membership problem for pattern languages [8]. On the other hand, it remains a long-standing open problem as to what the true complexity of the problem is, and in particular whether or not it is NP-complete. Plandowski and Rytter [76] showed that long solutions are highly compressible, and consequently, even an exponential bound on the length of the shortest solution, when one exists, would imply inclusion in NP. Nevertheless, the best known bound remains double-exponential [52,73].

Open Problem 1 ([76]). *Is the satisfiability problem for word equations contained in NP?*

2 String Solving

In recent years, a whole new community has formed with the goal of developing automated reasoning tools capable of proving or disproving statements involving words called *string solvers*, motivated primarily by applications in formal methods. Simply put, string solvers take as input a *string constraint* which can be thought of as a (usually quantifier-free) formula comprising Boolean combinations of atomic constraints involving:

- constants from Σ^* for some finite alphabet Σ,
- variables whose potential values range over Σ^*,
- common relations and operations on words such as concatenation, equality, length-comparisons, regular language membership, and typical string-manipulation functions such as `Replace_All()` and `Index_Of()`, string-number conversion, etc.

Their task is then to automatically determine the satisfiability of that formula, so, whether or not values for the variables can be found such that the formula becomes true. A comprehensive list of common relations and operations occurring as string constraints can be found in [7]. A standardised language for specifying string constraints in string solvers based on the *Satisfiability Modulo Theories* framework has also been implemented as part of the SMT-LIB format [2,11]. Most currently available string-solvers do not cater for the whole SMT-LIB standard, but focus rather on specific subsets to which their approaches or target applications are best suited.

Growing interest in string solving is possibly due to a steady increase in string-manipulating programs which are vulnerable to exploitation or attack (e.g. due to being publicly accessible on the web) in combination with substantial improvements in string solvers' own performance when employed in static analysis tasks aimed at improving security and reliability of software. There are many ways string solvers can be deployed in the context of software analysis. We list a few below (see e.g. [5,12,15,60]):

Path Feasibility. A common task in static analysis is to break down the possible executions of a piece of software into finitely many cases (paths). The problem of *path feasibility* involves working out which combinations of conditions on internal values result in paths which might actually occur in a real execution, and conversely which combinations of conditions are contradictory and so don't need to be considered further. Path feasibility analyses are also particularly useful in automated test-case generation.

Sanitisation and Validation. Cross-site scripting (XSS) and SQL injection attacks have both regularly been listed on the OWASP list of *Top 10 Web Application Security Risks* in recent years [1]. Although the two categories have been merged into a single one: "injection" in the 2021 list, it remains in a prominent position at #3. Such injection attacks involve tricking a system to execute malicious code. In the case of SQL injection, it might be that textual input is given which later forms part of an SQL database query constructed as a string. Maliciously designed entries, when not properly sanitised, can then influence the structure and meaning of the query, allowing the user to access or erase data. In XSS attacks, a similar effect can be achieved by getting a user's browser to execute malicious code e.g. in a link. Errors in input sanitisation and validation are not uncommon and automated analysis of (parts of) software handling externally generated data can help to reduce such errors [15].

Dynamically Generated Code. It is becoming increasingly common in programming languages to be able to dynamically load code such as functions and classes from string variables meaning that the executed code depends on the values of those string variables at runtime. While extremely powerful, this is also dangerous if the strings are not constructed safely and correctly. Again, both static and dynamic analyses performed with the help of string solvers can mitigate to some extent these risks.

The are also many potential areas of application for string solvers beyond software analysis and formal methods more generally, for example in database theory [10,37,38] or as automated proof assistants for areas such as combinatorics on words (see e.g. [47]).

Many string solving tools are now available employing a wide range of strategies, including CVC5 [9], Z3Str4 [68], Norn [3], Z3-Trau [4], OSTRICH [17], Sloth [45], Woorpje [24,28], CertiStr [54] and HAMPI [57]. Some, such as CVC5 and Z3str4 are designed to be more general, while others are developed with more specific tasks in mind. Many benchmarks exist, and in addition to the MOSCA (meeting on String Constraints and Applications) workshop [40], there is also now a string track at the annual SMT competition. Several meta-tools have been developed for comparing string solvers [58], automatically producing or altering test cases and benchmarks (fuzzing) [13] and very recently also for analysing large and often opaque sets of benchmarks [27].

Nevertheless, many challenges remain, of which one is obtaining a better understanding of the theoretical foundations involving word equations in combination with other combinations of constraints. When atomic constraints are restricted to involve only concatenation and equality comparisons, string solving can be reduced to solving word equations. Several other types of constraint, like regular language membership and linear arithmetic involving string-lengths are well understood in isolation and can be tackled in practice using highly optimised tools. However a common feature of string solving applications is that types of constraints often have to satisfied in combination.

It has been known since the 1990s that satisfiability of word equations with regular language membership constraints on the variables is decidable [33,82] (see also [61]). However, many other combinations quickly lead to undecidability: for example in the presence of a `Replace_All()` operator (modelled formally as finite transducers) [60] or in the presence of functions which count the numbers of a letter occurring in a word [14,26]. Since concatenation, equality (and thus word equations), regular language membership and linear (in)equalities over lengths of variables are particularly prominent types of string constraints (which additionally can in combination be used to model several other common constraints such as `Index_Of()`), a particularly important open problem remains:

Open Problem 2. *Is the satisfiability problem decidable for quantifier-free formulas combining word equations with regular language membership and linear (in)equalities over the lengths of variables?*

Formally, the satisfiability problem for quantifier-free formulas is a natural extension of the satisfiability problem for word equations: can we find substitutions for the variables which make the formula true when evaluated according to the natural semantics?

We call atomic constraints consisting of regular language membership *regular constraints*. They have the form $x \in L$ where x is a variable and L is a regular language given e.g. by an NFA or regular expression[1]. Similarly, we call atomic constraints consisting of linear (in)equalities over lengths of variables *length constraints*. We use $|x|$ to denote the length of a word x, including the case that x is a variable and the length is then an unknown number. Length constraints can be formalised e.g. as having the form $c_1|x_1| + c_2|x_2| \dots c_k|x_k| \oplus d_1|x_1| + d_2|x_2| \dots d_k|x_k|$ where $\oplus \in \{>, =\}$, and for $1 \leq i \leq k$, $c_i, d_i \in \mathbb{Z}$ and x_i are variables. Using this terminology, and recalling that Boolean combinations of word equations can be rewritten as a single equation, Open Problem 2 asks whether the satisfiability problem for word equations with regular and length constraints is decidable. An example of a string constraint involving word equations, regular constraints and length constraints is given below (x, y, z are variables and $a, b \in \Sigma$.

$$(x \doteq yabz \wedge |y| > |z|) \vee (x \doteq ybaz \wedge z \in a^*)$$

Both regular constraints and length constraints can be subsumed by language-membership constraints involving *visibly pushdown languages* (VPLs) [6]. VPLs generalise regular languages while retaining many of the desirable closure and algorithmic properties. Nevertheless, it is shown in [25] that combining word equations with VPL membership constraints results in an undecidable satisfiability problem.

Theorem 1 **([25]).** *For every recursively enumerable language L, there exists a quantifier free formula f combining word equations and VPL membership constraints and a variable x occurring in f, such that*

$$L = \{w \mid x \textit{ may be substituted by } w \textit{ as part of a satisfying assignment for } f\}.$$

It follows that the satisfiability problem for such formulas is undecidable.

Since VPLs share many properties with regular languages, VPL-membership constraints can be viewed as only a minor generalisation of regular (and length) constraints. This leads us to the following open problem, for which a negative would also yield a negative answer to Open Problem 2.

Open Problem 3. *Does there exist a recursively enumerable language L which is not expressible as the set*

$$\{w \mid x \textit{ may be substituted by } w \textit{ as part of a satisfying assignment for } f\}$$

for some quantifier free formula f combining word equations, regular constraints and length constraints?

A careful application of Greibach's theorem reveals that we cannot in general decide whether a property of words expressed by solutions to a string constraint

[1] Since it is easy to simulate the intersection of regular languages via conjunctions of regular constraints, the satisfiability problem for formulas containing regular constraints is automatically PSPACE-hard. Therefore, we can convert between any reasonable choices for specifying regular languages without affecting the computational complexity.

combining word equations, regular constraints, and length constraints can also be expressed using a combination of word equations and regular constraints alone.

Theorem 2 ([25]). *The following problem is undecidable: given a quantifier free formula f combining of word equations, regular constraints and length constraints, and a variable x occurring in f, does there exist a quantifier free formula f' containing only word equations and regular constraints and a variable y in f' such that $S_1 = S_2$ where:*

$$S_1 = \{w \mid x \textit{ may be substituted by } w \textit{ as part of a satisfying assignment for } f\}$$

and

$$S_2 = \{w \mid x \textit{ may be substituted by } w \textit{ as part of a satisfying assignment for } f'\}?$$

A weaker version of Open Problem 2, where regular constraints are omitted, is also a long standing open problem in the field of word equations.

Open Problem 4. *Is the satisfiability problem decidable for quantifier-free formulas combining word equations and linear (in)equalities over the lengths of variables?*

The difficulty of dealing with word equations in combination with length constraints is highlighted in [62], where they show that the set $\{(|h(x)|, |h(y)|) \mid h \text{ is a solution to } xaby \doteq ybax\}$, where $|w|$ denotes the length of the word w, is not definable in Presburger arithmetic.

3 A Closer Look at Solution-Sets

One of the most natural ways to try to improve our understanding of word equations, and in particular to improve techniques for solving them in combination with other constraints in practice, is to look closer at the structure of their solution-sets. Indeed, if when we are given a string constraint consisting, for example, of word equations, regular language memberships and length (in)equalities, one could try to solve the overall constraint by first providing a description of the set of solutions to the system of word equations without the additional constraints, and subsequently use that description to reason whether a solution exists satisfying the additional constraints. For example, given the string constraint

$$xy \doteq yx \wedge |y| > |x| \wedge x \in \{ab\}^* \wedge y \notin a\{a,b\}^+$$

one could first notice that a pair of words x, y is a solution to the word equation $xy \doteq yx$ if and only if they are both repetitions of the same word, or more formally, if there exist $w \in \Sigma^*$ and $p, q \in \mathbb{N}_0$ such that $x = w^p$ and $y = w^q$ where w^0 is the empty word and $w^{i+1} = w^i w$ [61]. With such an explicit description to hand, it is then not difficult to observe that since $x \in \{ab\}^+$, x and y must necessarily both be repetitions of $w = ab$. From $y \notin a\{a,b\}^*$ we further conclude that y is the empty word, and since this implies that $|y| > |x|$ cannot be met, that this string constraint is unsatisfiable.

3.1 Parametric Solutions

The description of solutions x, y to the equation $xy \doteq yx$ given in terms of an unknown word w and numbers p, q is called a *parametric solution*. Parametric solutions are particularly useful because they are a very explicit description of the solution-set. Following [20], they are defined formally as follows.

Definition 1. *Let Δ, Γ be alphabets. We call elements of Δ* word parameters *and elements of Γ* numerical parameters*. Parametric words are defined inductively as follows.*

- *Each element of Δ is a parametric word,*
- *if δ is a parametric word and $k \in \Gamma$ is a numerical parameter, then δ^k is a parametric word,*
- *if δ_1, δ_2 are parametric words, then their concatenation $\delta_1\delta_2$ is also a parametric word.*

Every assignment φ of words in Σ^* to the word parameters and numbers from $\mathbb{N}_0$ to the numerical parameters maps a parametric word to a unique concrete word $\varphi(\delta) \in \Sigma^*$ called the value*.*

Given a word equation E over n variables, a parametric solution *for E is an n-tuple of parametric words such that every assignment φ induces a solution. Moreover, E is said to be* parametrizable *if the solution-set is exactly described by finitely many parametric solutions.*

The definition above is given in context of constant-free word equations. For this reason, it does not include provision for constants occurring in parametric solutions. However, it is very natural to extend Definition 1 to the more general case by simply adding the axiom that each element of $\Sigma \cup \{\varepsilon\}$ is also a parametric word, where ε denotes the empty word.

Unfortunately, although it was shown by Hmelevskii [44] that every constant-free equation with at most three variables is parametrizable, the same does not hold when a fourth variable is introduced. A more concise proof of the latter fact is given by Czeizler in [20].

Theorem 3 (Czeizler [20], Hmelevskii [44]). *Let x, y, z, v be variables. Then the word equation $xyz \doteq zvx$ is not parametrizable.*

Of course the negative parametrizability result also carries through to the general case in which constants are also allowed. In fact, it follows from [49] that the word equation $xaby \doteq ybax$ with only two variables x and y is not parametrizable.

3.2 Graph Representations of Solution-Sets

A key question then, is how to represent solution-sets to word equations if not by parametric words? One answer can be found in approaches for algorithmically solving word equations which have been extended to produce descriptions of the full solution-set.

Decision procedures for solving word equations usually revolve around some non-deterministic search for solutions, made necessary by the fact that the satisfiability problem is NP-hard. This search can often be formalised in terms of iteratively applying transformation rules (e.g. to a possibly extended representation of the equation, or a solution to it[2]) with the aim of eventually reaching some trivial case signifying that a solution exists. Such an approach yields a (possibly infinite) graph which, with the correct setup, provides a complete description of the set of solutions by virtue of accounting for the search across all possibilities. Guaranteeing that the graph is finite presents more of a challenge, although there are now several different approaches which achieve this.

In an early example of this approach, Makanin's algorithm for solving equations in a free group was used by Razborov in developing an algorithmic representation of all solutions to systems of equations in a free group [78]. In [75], Plandowski adapted his algorithm for solving word equations to produce a finite graph representing all solutions, and more recently, the same was achieved in a simpler manner using the Recompression technique of Jeż [52].

Diekert, Jeż and Plandowski generalised the Recompression approach to work in the presence of both regular membership constraints and involution [34]. This combination is significant because it allows for the extension of methods from the free monoid to the free group setting. Shortly after, Ciobanu, Diekert and Elder [18] provided a simpler representation in terms of EDT0L languages.

Theorem 4 ([18]). *Solution-sets to word equations with regular membership constraints and involution (and hence also equations in free groups) are EDT0L languages. In particular, they are also indexed languages.*

Indexed languages are a subset of the context-sensitive languages. EDT0L (Extended Deterministic Table 0-Interaction Lindermayer) languages are languages defined by a specific variety of so-called L-systems, which generate words by iterated applications of morphisms to some initial "seed" word. They are strictly contained in the indexed languages and are incomparable to context-free languages. Despite their verbose name, EDT0L languages are a natural class with a simple intuition, corresponding to the case when the application of the morphisms is constrained by some NFA-like control.

Definition 2. *Let A be an alphabet and $L \subseteq A^*$. Then L is an EDT0L language if there exists an alphabet C with $A \subseteq C$, a word $w \in C^*$ and a rational set R of morphisms $h : C^* \to C^*$ such that $L = \{h(w) \mid h \in R\}$.*

Given an equation $U \doteq V$ with variables $x_1, x_2, \ldots, x_n$, the construction from [18] essentially equates the set of solutions

$$\{(g(x_1), g(x_2), \ldots, g(x_n)) \mid g(U) = g(V)\}$$

with the set $\{(h(c_1), h(c_2), \ldots, h(c_n)) \mid h \in R\}$ for some rational set of morphisms R and additional letters $c_1, c_2, \ldots, c_k$. The latter set is then easily encoded as an EDT0L language using a separator symbol # as the set

[2] These two viewpoints are not mutually exclusive as a word equation can be thought of as a compact representation of a solution.

$\{h(c_1\#c_2\#\dots\#c_n) \mid h \in R\}$. The set R, when represented by the underlying NFA, provides a graph representation of all solutions. This graph representation facilitates algorithmic solutions to problems other than just the satisfiability problem. In particular (in)finiteness can be determined by looking for cycles in the NFA defining R.

Corollary 1 ([18]). *It is decidable whether the solution-set to a (system of) word equations with regular language membership constraints is finite.*

Unfortunately, these graph representations are not as well suited to other canonical decision problems. Indeed, negative results can be inferred from [36, 39], both of which provide proofs of the fact that deciding the truth of logical sentences of the form

$$\forall x \exists y_1, y_2, \dots, y_n. \, \varphi$$

where φ is a Boolean combination of word equations is undecidable. In particular, we get the following.

Theorem 5 ([36,39]). *It is undecidable whether or not, given a word equation E containing a variable x (and possibly others), the set $\{h(x) \mid h$ is a solution to $E\}$ is exactly Σ^*.*

Moreover, we note the following negative result from [25].

Theorem 6 ([25]). *It is undecidable, given a word equation E containing a variable x (and possibly others), the set $\{h(x) \mid h$ is a solution to $E\}$ is a regular language.*

Consequently, we cannot expect that any reasonable (computable) representation of solution-sets to word equations is sufficiently descriptive as to allow for inference of all interesting properties.

3.3 Nielsen Transformations

A disadvantage of the graph representations discussed in the previous section is that, even in for those that are guaranteed to be finite, the edge relations are complex (or, at least in the case of Recompression, highly non-deterministic) meaning it is difficult to study their structure in detail. Quadratic word equations, in which each variable occurs at most twice, offer a much simpler means of producing a finite graph describing all solutions via a rewriting process based on a well-known type of morphism called Nielsen transformations.

The rewriting relation, which we denote $\Rightarrow_{NT}$, is combinatorially simple at a local level. When applied iteratively to a given word equation E, it induces a graph $\mathcal{G}_E^{\Rightarrow_{NT}}$ describing all solutions which in the general case is usually infinite, but in the quadratic case is guaranteed to be finite. $\mathcal{G}_E^{\Rightarrow_{NT}}$ has as vertices word equations (including E) and its edges are labelled with morphisms $\psi : (X \cup \Sigma)^* \rightarrow (X \cup \Sigma)^*$. Solutions to E are obtained by composing the morphisms occurring as edge-labels on walks[3] in the graph starting at E and finishing at the trivial equation $\varepsilon \doteq \varepsilon$. The underlying idea comes from a basic fact concerning semigroups called Levi's lemma, stated as follows.

Lemma 1 (Levi's Lemma). *Let $u, v, x, y \in \Sigma^*$ be words such that $uv = xy$. Then there exists w such that either:*

- *$u = xw$ and $wv = y$, or*
- *$x = uw$ and $wy = v$.*

Levi's lemma applies to word equations in the following way: given a word equation $xU \doteq yV$ where $x, y \in X \cup \Sigma$ are the leftmost symbols on each side of the equation and $U, V \in (X \cup \Sigma)^*$ are the remaining parts, we have three possibilities for a non-erasing solution[4] h: either

- $h(x) = h(y)$ and $h(U) = h(V)$ (this corresponds to the case that $w = \varepsilon$ in Levi's lemma), or
- $h(x) = h(y)w$ and $wh(U) = h(V)$ for some $w \in \Sigma^+$
- $h(x)w = h(y)$ and $h(U) = wh(V)$ for some $w \in \Sigma^+$.

In the first case, if $x, y \in \Sigma$ with $x \neq y$, no solutions exist. If $x = y \in \Sigma$ then solutions to E are exactly solutions to $U \doteq V$. Otherwise, solutions h to E can be found by looking for solutions h' to the equation $U' \doteq V'$ obtained by replacing x everywhere by y in U and V respectively if x is a variable (or vice-versa if y is a variable).

In the second case, if x is not a variable, then no solutions exist for this case. Otherwise, by introducing a new variable z (intended to account for w, so that $h(z) = w$) we can find solutions h to E in terms of solutions h' to the equation $zU' \doteq V'$ obtained by replacing all occurrences of x by yz in U and V respectively. The third case is symmetrical to the second.

Thus, overall, solutions to E can be reduced to solutions to (at most) three further equations derived by cancelling some symbols from the left and performing a replacement of the form $x \to yz$ or $x \to y$. Notice that when performing a replacement $x \to yz$, we actually remove all occurrences of x from the equation, and so we might as well re-use the variable x in place of z to get $x \to yx$ (and similarly for $y \to xz$ we might as well use y in place of z to get $y \to xy$). These replacements can be performed via the application of morphisms from a set Ψ defined as follows: for $x \in X \cup \Sigma$ and $y \in X$, $\psi_{(x,y)}, \hat{\psi}_{(x,y)} : (X \cup \Sigma)^* \to (X \cup \Sigma)^*$ belong to Ψ such that:

$$\psi_{(x,y)}(y) = xy \qquad \hat{\psi}_{(x,y)}(y) = x$$
$$\psi_{(x,y)}(z) = z \text{ for } z \neq y, \qquad \hat{\psi}_{(x,y)}(z) = z \text{ for } z \neq y.$$

[3] Paths in which both vertices and edges may be repeated.

[4] Non-erasing solutions are solutions for which $h(x)$ is not the empty word for any variable x. The general case can be reduced to the non-erasing case by simply guessing in advance which variables should be mapped to the empty word and removing them from the word equation(s).

The morphisms $\psi_{(x,y)}$ are called *Nielsen transformations*, hence the name of this approach. We denote by $\Rightarrow_{NT}$ the relation consisting of pairs (E_1, E_2) such that E_2 may be derived from E_1 according to one of the three cases above.

Now suppose we have a quadratic equation E and consider E' such that $E \Rightarrow_{NT} E'$. Then the removal of the leftmost symbols means that $|E| \geq |E'|$. Similarly, the number of occurrences of each variable in E is at least as high as in E', and no new symbols are introduced. It follows that there are only finitely many equations E'' such that $E \Rightarrow_{NT}^{*} E''$ where $\Rightarrow_{NT}^{*}$ denotes the reflexive transitive closure of $\Rightarrow_{NT}$.

Let $\mathcal{G}_E^{\Rightarrow_{NT}}$ be the graph whose vertices are equations E'' reachable from E by iteratively applying $\Rightarrow_{NT}$, and whose edges are given by $\Rightarrow_{NT}$, labelled with the appropriate corresponding morphisms. Given a word equation E' occurring as a vertex in this graph, for each solution h' to E' there exists an edge in the graph from E' to a (not necessarily distinct) equation E'' labelled with a morphism ψ, such that $h' = h'' \circ \psi$ for some strictly shorter solution[5] h'' to E''. For this reason, all solutions to the original equation E can be obtained by composing the morphisms occurring on a walk in the graph from the original equation to the trivial equation $\varepsilon \doteq \varepsilon$ as mentioned previously. If the equation $\varepsilon \doteq \varepsilon$ is not present in the graph, no solutions exist.

As an example, consider the equation E given by $Xabaa \doteq baaXa$ over the variable X and constants $a, b \in \Sigma$. The graph $\mathcal{G}_E^{\Rightarrow_{NT}}$ (with labels) is shown in Fig. 1.

Treating the graph as an NFA $\mathcal{A}_E$ over the alphabet of morphisms Ψ whose accepting state is $\varepsilon \doteq \varepsilon$ and whose initial state is E, we obtain a rational set $L(\mathcal{A}_E)$ of morphisms exactly describing the set of solutions to E. For example one solution is given by $h = \hat{\psi}_{(a,X)} \circ \psi_{(b,X)} \circ \psi_{(a,X)} \circ \psi_{(a,X)} \circ \psi_{(b,X)}$ (note that the composition occurs in the opposite order from left to right to the "word" from Ψ^* accepted by $\mathcal{A}$), which is the substitution h given by $h(X) = baaba$, $h(a) = a$ and $h(b) = b$.

$$\begin{aligned}
h(X) &= \hat{\psi}_{(a,X)} \circ \psi_{(b,X)} \circ \psi_{(a,X)} \circ \psi_{(a,X)} \circ \psi_{(b,X)}(X) \\
&= \hat{\psi}_{(a,X)} \circ \psi_{(b,X)} \circ \psi_{(a,X)} \circ \psi_{(a,X)}(bX) \\
&= \hat{\psi}_{(a,X)} \circ \psi_{(b,X)} \circ \psi_{(a,X)}(baX) \\
&= \hat{\psi}_{(a,X)} \circ \psi_{(b,X)}(baaX) \\
&= \hat{\psi}_{(a,X)}(baabX) \\
&= baaba
\end{aligned}$$

The simplicity of this approach based on Nielsen transformations and Levi's lemma, along with the fact that it is easily adapted for use with regular language membership constraints and length constraints means it is a good candidate for practical implementations. As such it has been used in the string solving tool Woorpje [28], and other string solvers make use of similar ideas. It also has

[5] Where the length of the solution is measured in terms of the word obtained by applying it to one side of the equation.

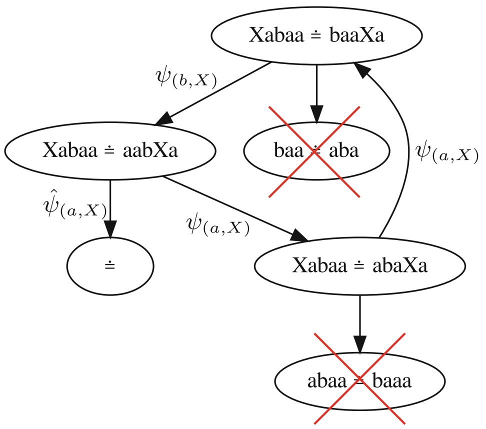

Fig. 1. The graph $\mathcal{G}_E^{\Rightarrow NT}$ in the case that E is the equation $Xabaa \doteq baaXa$. Trivially unsolvable equations in the graph are crossed out and their ingoing edge labels omitted.

several advantages from a theoretical point of view: the structural properties of the graph $\mathcal{G}_E^{\Rightarrow NT}$ provide information about the solution-set of E. For example if it is a directed acyclic graph (DAG), then it is straightforward to show that E is parametrizable. Actually, a slightly stronger statement, namely that $\mathcal{G}_E^{\Rightarrow NT}$ does not have two distinct cycles sharing a vertex, is sufficient to guarantee that E is parametrizable [72]. Similar restrictions on the structure of $\mathcal{G}_E^{\Rightarrow NT}$ have been used to identify cases where satisfiability remains decidable even when length constraints are added [62]. Several case where the graph is guaranteed to be finite even when the underlying equation E is not quadratic are considered in [69]. Moreover the simplicity of the rewriting transformations make the graphs obtained via Nielsen transformations much more accessible for more detailed combinatorial analyses such as the one given in [29].

3.4 Restricted Word Equations

In the absence of positive answers to Open Problems 1, 2 and 4, it is natural to consider them also in the context of syntactically restricted subclasses of word equations. For simplicity, we concentrate in this section on single word equations. Some care is needed when generalising to Boolean combinations: while in general it is no restriction to do so due to the constructions e.g. in [55], these constructions are not guaranteed to respect the syntactic restrictions and so cannot be used directly to generalise the results in this section. However, in most cases equivalent results hold at least for systems (conjunctions) of equations.

Solution-sets to word equations containing only one variable have a particularly restricted form and are well understood:

Theorem 7 ([59,71]). *Let x be a variable, and let $U \doteq V$ be a word equation such that $U, V \in \{x\} \cup \Sigma^*$. Then one of the following holds:*

1. *The set of solutions for $U \doteq V$ is finite and has cardinality at most three, or*
2. *There exist words $u, v \in \Sigma^*$ such that uv is primitive*[6] *the set of solutions for $U \doteq V$ has the form $\{h : (\{x\} \cup \Sigma)^* \rightarrow \Sigma^* \mid h(x) \in (uv)^* u\}$.*

Corollary 2. *Word equations with exactly one variable are parametrizable. Moreover, the satisfiability problem for word equations with one variable and with length constraints and regular constraints is decidable.*

The satisfiability problem for word equations is solvable in deterministic linear time [51]. Similarly, word equations with two variables are also solvable in polynomial time [16,22,49]. However, we have already seen that solution-sets to word equations with two variables are not necessarily parametrizable, and Open Problems 2 and 4 remain open in this case.

Further cases can be derived from restricted classes of string constraints. The notions of *solved form* [39], *acyclic* [5] and *straight-line* [60] constraints are all syntactic restrictions designed such that cyclic dependencies between the variables are avoided. As such, the solution-sets to constraints adhering to these definitions are generally parametrizable, and satisfiability for constraints involving word equations, length constraints and regular constraints (and even `Replace_All()` in the case of the straight-line fragment) become decidable.

We have already mentioned in the previous section that quadratic word equations - equations which contain each variable at most twice (although the number of variables is unconstrained) - possess the desirable property that the simple Nielsen transformation algorithm for producing a graph representation of all solutions is guaranteed to terminate. Nevertheless, Open Problems 1, 2 and 4 all remain open even in the quadratic case. It was shown in [79] that the satisfiability problem remains NP-hard in the quadratic case. On the other hand, word equations in which variables occur at most once are trivially parametrizable, and it is easily seen that the satisfiability problem remains decidable in the presence of various additional constraints, including length constraints and regular constraints.

In [66], the class of regular word equations was proposed as a natural subclass of the quadratic word equations. The initial idea was to consider classes of equations $U \doteq V$ for which the satisfiability problem remains NP-hard, even when the two sides U and V constitute patterns for which the membership problem can be solved in polynomial time. Regular word equations derive their name from regular patterns [50] in which each variable occurs at most once. Consequently, each variable may occur twice overall, but not twice on the same side of the equation.

Definition 3. *A word equation $U \doteq V$ is* regular *if each variable occurs at most once in U and at most once in V.*

Surprisingly, even severely restricted subclasses of regular word equations have an NP-hard satisfiability problem. The class of regular-ordered word equations (ROWEs) is the class of regular word equations for which the variables

[6] A word is primitive if it cannot be written as the repetition of a strictly shorter word.

occur in the same order from left to right on both sides of the equation (some variables may still occur on only one side). So, for example, the equation $x_1ax_2bx_3 \doteq x_1babax_3$ is regular-ordered, but $x_1ax_2bx_3 \doteq x_3babax_1$ is not.

Theorem 8 **([30]).** *The satisfiability problem for ROWEs is NP-complete.*

Moreover, it was shown in [62] that the satisfiability problem for ROWEs with length constraints is decidable, extending a weaker result from [26].

In [31], Theorem 8 was extended slightly to cover regular-reversed word equations (RRWEs): equations $U \doteq V$ in which the order of variables in V is exactly the reverse of the order of the variables in U. A much more comprehensive result is given in [29], which describes in detail the structure of the graphs $\mathcal{G}_E^{\Rightarrow NT}$ obtained as the result of the Nielsen transformation algorithm described in Sect. 3.3 in the case of all regular word equations. A consequence of this description is that the minimal path between any two vertices (when it exists) has length bounded by a polynomial in the length of the original equation.

Theorem 9 **([29]).** *Let E be a regular word equation. Let v_1, v_2 be vertices in the graph $\mathcal{G}_E^{\Rightarrow NT}$ such that there exists a path from v_1 to v_2. Then the shortest such path has length at most $O(|E|^{12})$.*

Consequently, in order to (non-deterministically) check whether a solution exists to a regular word equation E, it suffices to guess the equations occurring along the path in $\mathcal{G}_E^{\Rightarrow NT}$ from E to $\varepsilon \doteq \varepsilon$. Each equation on that path will be no larger than E, and it is easily verifiable in polynomial time that there is indeed an edge between each successive pair of equations. Thus, it follows that satisfiability for regular word equations is in NP. Combined with the hardness result from [30], we get the following.

Corollary 3 **([29,30]).** *The Satisfiability problem for regular word equations is NP-complete.*

Of course one of the most natural open problems remains whether the techniques of [29] can be extended to work for quadratic word equations more generally.

Open Problem 5. *Does Theorem 9 also hold for all quadratic word equations?*

Moreover, the detailed analysis in [29] would be a good basis from which to try to resolve Open Problems 4 and 2 in the sub-case of regular word equations. In particular, several structures in the graphs $\mathcal{G}_E^{\Rightarrow NT}$ are described which might provide new insights into how complex the set of lengths of solutions to (regular) word equations can be.

Open Problem 6. *What is the decidability status for the satisfiability problem for regular word equations with length constraints? What about for regular word equations with length constraints and regular constraints?*

4 Conclusions

The study of word equations has yielded many significant and influential results over the past half-century, and is of interest in a variety of areas, including logic, formal languages, combinatorics on words, and combinatorial group theory. More recently, there has been substantial interest in the topic from the within the formal methods community, specifically in relation to string solvers, which aim to solve problems involving words which often incorporate word equations alongside other constraints such as regular language membership and length-comparisons. The two fields are mutually beneficial: theoretical results on word equations and related topics can provide insights and ideas for more efficient, powerful and ultimately practical algorithms implemented in string solvers while on the other hand, a better understanding of the problems that string solvers must tackle can reveal new open problems and directions to be explored in the theory. Recent results focusing on the structure and properties of solution-sets for restricted classes of word equations provide a basis from which we can hope to make progress on long standing open problems which remain central to both the theory and practice.

References

1. OWASP top ten web application security risks. https://owasp.org/www-project-top-ten/. Accessed 15 Mar 2022
2. SMT-LIB standard for unicode strings. https://smtlib.cs.uiowa.edu/theories-UnicodeStrings.shtml. Accessed 15 Mar 2022
3. Abdulla, P.A., et al.: Norn: an SMT solver for string constraints. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 462–469. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_29
4. Abdulla, P.A., et al.: Efficient handling of string-number conversion. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 943–957 (2020)
5. Abdulla, P.A., et al.: String constraints for verification. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 150–166. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_10
6. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proceedings of the 36th ACM Symposium on Theory of Computing (STOC), STOC 2004, pp. 202–211 (2004)
7. Amadini, R.: A survey on string constraint solving. ACM Comput. Surv. (CSUR) **55**(1), 1–38 (2021)
8. Angluin, D.: Finding patterns common to a set of strings. In: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, pp. 130–141 (1979)
9. Barbosa, H., et al.: cvc5: a versatile and industrial-strength SMT solver. In: Fisman, D., Rosu, G. (eds) TACAS 2022. LNCS, vol. 13243, pp. 415–442. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99524-9_24
10. Barceló, P., Muñoz, P.: Graph logics with rational relations: the role of word combinatorics. ACM Trans. Comput. Logic (TOCL) **18**(2), 1–41 (2017)
11. Barrett, C., Stump, A., Tinelli, C., et al.: The SMT-LIB standard: Version 2.0. In: Proceedings of the 8th International Workshop on Satisfiability Modulo Theories, Edinburgh, England, vol. 13, p. 14 (2010)

12. Bjørner, N., Tillmann, N., Voronkov, A.: Path feasibility analysis for string-manipulating programs. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 307–321. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00768-2_27
13. Blotsky, D., Mora, F., Berzish, M., Zheng, Y., Kabir, I., Ganesh, V.: StringFuzz: a fuzzer for string solvers. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10982, pp. 45–51. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96142-2_6
14. Büchi, J.R., Senger, S.: Definability in the existential theory of concatenation and undecidable extensions of this theory. In: Mac Lane, S., Siefkes, D. (eds.) The Collected Works of J. Richard Büchi, pp. 671–683. Springer, New York (1990). https://doi.org/10.1007/978-1-4613-8928-6_37
15. Bultan, T., Yu, F., Alkhalaf, M., Aydin, A.: String Analysis for Software Verification and Security, vol. 10. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68670-7
16. Charatonik, W., Pacholski, L.: Word equations with two variables. In: Abdulrab, H., Pécuchet, J.-P. (eds.) IWWERT 1991. LNCS, vol. 677, pp. 43–56. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-56730-5_30
17. Chen, T., Hague, M., Lin, A.W., Rümmer, P., Wu, Z.: Decision procedures for path feasibility of string-manipulating programs with complex operations. Proc. ACM Program. Lang. **3**(POPL), 1–30 (2019)
18. Ciobanu, L., Diekert, V., Elder, M.: Solution sets for equations over free groups are EDT0L languages. Internat. J. Algebra Comput. **26**(05), 843–886 (2016)
19. Ciobanu, L., Elder, M.: Solutions sets to systems of equations in hyperbolic groups are EDT0L in PSPACE. arXiv preprint arXiv:1902.07349 (2019)
20. Czeizler, E.: The non-parametrizability of the word equation xyz = zvx: a short proof. Theoret. Comput. Sci. **345**(2–3), 296–303 (2005)
21. Czeizler, E., Holub, Š, Karhumäki, J., Laine, M.: Intricacies of simple word equations: an example. Int. J. Found. Comput. Sci. **18**(06), 1167–1175 (2007)
22. 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). https://doi.org/10.1007/978-3-540-27836-8_36
23. Dahmani, F., Guirardel, V.: Foliations for solving equations in groups: free, virtually free, and hyperbolic groups. J. Topol. **3**(2), 343–404 (2010)
24. Day, J.D., et al.: On solving word equations using SAT. In: Filiot, E., Jungers, R., Potapov, I. (eds.) RP 2019. LNCS, vol. 11674, pp. 93–106. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30806-3_8
25. Day, J.D., Ganesh, V., Grewal, N., Manea, F.: Formal languages via theories over strings: What's decidable? Unpublished manuscript
26. Day, J.D., Ganesh, V., He, P., Manea, F., Nowotka, D.: The satisfiability of word equations: decidable and undecidable theories. In: Potapov, I., Reynier, P.-A. (eds.) RP 2018. LNCS, vol. 11123, pp. 15–29. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00250-3_2
27. Day, J.D., Kröger, A., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: BASC: benchmark analysis for string constraints. Unpublished manuscript
28. Day, J.D., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: Rule-based word equation solving. In: Proceedings of the 8th International Conference on Formal Methods in Software Engineering, pp. 87–97 (2020)
29. Day, J.D., Manea, F.: On the structure of solution-sets to regular word equations. In: Theory of Computing Systems, pp. 1–78 (2021)

30. Day, J.D., Manea, F., Nowotka, D.: The hardness of solving simple word equations. In: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
31. Day, J.D., Manea, F., Nowotka, D.: Upper bounds on the length of minimal solutions to certain quadratic word equations. In: 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
32. Diekert, V.: More than 1700 years of word equations. In: Maletti, A. (ed.) CAI 2015. LNCS, vol. 9270, pp. 22–28. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23021-4_2
33. Diekert, V., Gutierrez, C., Hagenah, C.: The existential theory of equations with rational constraints in free groups is PSPACE-complete. Inf. Comput. **202**(2), 105–140 (2005)
34. Diekert, V., Jeż, A., Plandowski, W.: Finding all solutions of equations in free groups and monoids with involution. Inf. Comput. **251**, 263–286 (2016)
35. Diekert, V., Muscholl, A.: Solvability of equations in free partially commutative groups is decidable. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 543–554. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-48224-5_45
36. Durnev, V.G.: Undecidability of the positive $\forall\exists^3$-theory of a free semigroup. Sib. Math. J. **36**(5), 917–929 (1995)
37. Freydenberger, D.D.: A logic for document spanners. Theory Comput. Syst. **63**(7), 1679–1754 (2019)
38. Freydenberger, D.D., Peterfreund, L.: The theory of concatenation over finite models. In: 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2021)
39. Ganesh, V., Minnes, M., Solar-Lezama, A., Rinard, M.: Word equations with length constraints: what's decidable? In: Biere, A., Nahir, A., Vos, T. (eds.) HVC 2012. LNCS, vol. 7857, pp. 209–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39611-3_21
40. Hague, M.: Strings at MOSCA. ACM SIGLOG News **6**(4), 4–22 (2019)
41. Halfon, S., Schnoebelen, P., Zetzsche, G.: Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In: 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–12. IEEE (2017)
42. Harju, T., Nowotka, D.: On the independence of equations in three variables. Theoret. Comput. Sci. **307**(1), 139–172 (2003)
43. Harju, T., Nowotka, D.: On the equation $x^k = z_1^{k_1} z_2^{k_2} \cdots z_n^{k_n}$ in a free semigroup. Theoret. Comput. Sci. **330**(1), 117–121 (2005)
44. Hmelevskii, J.I.: Equations in free semigroups, volume 107 of Am. Math. Soc. Transl. Proc. Steklov and Insti. Mat (1976)
45. Holik, L., Janku, P., Lin, A.W., Rümmer, P., Vojnar, T.: String constraints with concatenation and transducers solved efficiently. In: Proceedings of the ACM on Programming Languages, vol. 2, pp. 1–32. ACM Digital Library (2018)
46. Holub, Š, Kortelainen, J.: On systems of word equations with simple loop sets. Theoret. Comput. Sci. **380**(3), 363–372 (2007)
47. Holub, Š., Starosta, Š.: Formalization of basic combinatorics on words. In: 12th International Conference on Interactive Theorem Proving (ITP 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2021)
48. Holub, Š, Žemlička, J.: Algebraic properties of word equations. J. Algebra **434**, 283–301 (2015)

49. Ilie, L., Plandowski, W.: Two-variable word equations. RAIRO-Theoret. Inform. Appl. **34**(6), 467–501 (2000)
50. Jain, S., Ong, Y.S., Stephan, F.: Regular patterns, regular languages and context-free languages. Inf. Process. Lett. **110**(24), 1114–1119 (2010)
51. Jeż, A.: One-variable word equations in linear time. Algorithmica **74**(1), 1–48 (2016)
52. Jeż, A.: Recompression: a simple and powerful technique for word equations. J. ACM (JACM) **63**(1), 1–51 (2016)
53. Jeż, A.: Word equations in non-deterministic linear space. J. Comput. Syst. Sci. **123**, 122–142 (2022)
54. Kan, S., Lin, A.W., Rümmer, P., Schrader, M.: CertiStr: a certified string solver. In: Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, pp. 210–224 (2022)
55. Karhumäki, J., Mignosi, F., Plandowski, W.: The expressibility of languages and relations by word equations. J. ACM (JACM) **47**(3), 483–505 (2000)
56. Karhumäki, J., Saarela, A.: On maximal chains of systems of word equations. Proc. Steklov Inst. Math. **274**(1), 116–123 (2011)
57. Kiezun, A., Ganesh, V., Artzi, S., Guo, P.J., Hooimeijer, P., Ernst, M.D.: HAMPI: a solver for word equations over strings, regular expressions, and context-free grammars. ACM Trans. Softw. Eng. Methodol. (TOSEM) **21**(4), 1–28 (2013)
58. Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: ZaligVinder: a generic test framework for string solvers. J. Softw. Evol. Process, e2400 (2021)
59. Laine, M., Plandowski, W.: Word equations with one unknown. Int. J. Found. Comput. Sci. **22**(02), 345–375 (2011)
60. Lin, A.W., Barceló, P.: String solving with word equations and transducers: towards a logic for analysing mutation XSS. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 123–136 (2016)
61. Lothaire, M.: Algebraic Combinatorics on Words, vol. 90. Cambridge University Press, Cambridge (2002)
62. Majumdar, R., Lin, A.W.: Quadratic word equations with length constraints, counter systems, and Presburger arithmetic with divisibility. Log. Meth. Comput. Sci. **17** (2021)
63. Makanin, G.S.: Decidability of the universal and positive theories of a free group. Math. USSR-Izvestiya **25**(1), 75 (1985)
64. Makanin, G.S.: The problem of solvability of equations in a free semigroup. Matematicheskii Sbornik **145**(2), 147–236 (1977)
65. Makanin, G.S.: Equations in a free group. Math. USSR-Izvestiya **21**(3), 483 (1983)
66. Manea, F., Nowotka, D., Schmid, M.L.: On the complexity of solving restricted word equations. Int. J. Found. Comput. Sci. **29**(05), 893–909 (2018)
67. Maňuch, J.: Characterization of a word by its subwords. In: Developments in Language Theory: Foundations, Applications, and Perspectives, pp. 210–219. World Scientific (2000)
68. Mora, F., Berzish, M., Kulczynski, M., Nowotka, D., Ganesh, V.: Z3str4: a multi-armed string solver. In: Huisman, M., Păsăreanu, C., Zhan, N. (eds.) FM 2021. LNCS, vol. 13047, pp. 389–406. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90870-6_21
69. Nepeivoda, A.: Program specialization as a tool for solving word equations. In: Electronic Proceedings in Theoretical Computer Science, EPTCS, pp. 42–72 (2021)
70. Nowotka, D., Saarela, A.: One-variable word equations and three-variable constant-free word equations. Int. J. Found. Comput. Sci. **29**(05), 935–950 (2018)

71. Nowotka, D., Saarela, A.: An optimal bound on the solution sets of one-variable word equations and its consequences. SIAM J. Comput. **51**(1), 1–18 (2022)
72. Petre, E.: An elementary proof for the non-parametrizability of the equation $xyz=zvx$. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 807–817. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28629-5_63
73. Plandowski, W.: Satisfiability of word equations with constants is in NEXPTIME. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 721–725 (1999)
74. Plandowski, W.: Satisfiability of word equations with constants is in PSPACE. J. ACM (JACM) **51**(3), 483–496 (2004)
75. Plandowski, W.: An efficient algorithm for solving word equations. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 467–476 (2006)
76. Plandowski, W., Rytter, W.: Application of Lempel-Ziv encodings to the solution of word equations. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 731–742. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055097
77. Quine, W.V.: Concatenation as a basis for arithmetic. J. Symbolic Logic **11**(4), 105–114 (1946)
78. Razborov, A.A.: On systems of equations in free groups. In: Combinatorial and Geometric Group Theory, pp. 269–283 (1993)
79. Robson, J.M., Diekert, V.: On quadratic word equations. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 217–226. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49116-3_20
80. Saarela, A.: Word equations with kth powers of variables. J. Comb. Theory Ser. A. **165**, 15–31 (2019)
81. Saarela, A.: Hardness results for constant-free pattern languages and word equations. In: 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
82. Schulz, K.U.: Makanin's algorithm for word equations-two improvements and a generalization. In: Schulz, K.U. (ed.) IWWERT 1990. LNCS, vol. 572, pp. 85–150. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55124-7_4