A Remark About Quadratic Trace Equations

Volker Diekert Manfred Kufleitner

Universität Stuttgart
Institut für Formale Methoden der Informatik
Breitwiesenstr. 20–22, D–70565 Stuttgart, Germany

Abstract. We present a simple PSPACE-algorithm for computing the most general solution of a system of quadratic trace equations with involution. We extend the known linear time algorithm for quadratic word equations with length constraints to cope with involutions. Finally, we show that the same linear-time result cannot be expected for trace equations. We obtain an NP-hardness result.

1 Introduction

A string graph is the intersection graph of a finite set of curves on a surface. Recently the string graph recognition problem in the plane has been shown to be NP-complete via a surprising polynomial time reduction to word equations with involution by Schaefer, Sedgwick and Štefankovič [9]. The NP-completeness result came quite unexpected because for many years, even decidability was not known. After the reduction in [9] one finds a system of word equations and the NP-algorithm guesses in a first step the lengths in the solution. Starting from string graphs these lengths are bounded by a single exponential function, so in binary notation the NP-algorithm can indeed guess them. Then it is possible to apply the techniques developed by Plandowski and Rytter in [7] for solving word equations in polynomial time when lengths are specified. However, inspecting the reduction from string graph recognition to word equations, one sees that only quadratic word equations appear and this simplifies the situation, so that as an alternative one can also apply the method for solving quadratic equations of [8] in linear time when lengths are specified. The only problem is that the authors of [8] did not consider the presence of an involution. But this is not essential as it is shown in Sect. 3.

For surfaces of higher genus the reduction of [9] leads to quadratic systems of trace equations, again with involution. Trace monoids are free monoids with a partial commutation between letters. (The partial commutation corresponds to the crossings of the curves.) It turns out that these quadratic trace equations can be solved rather straightforwardly, too: We present a simple PSPACE-algorithm for computing the most general solution when the input is a partial commutation relation with a system of quadratic trace equations with involution. By this result[1] the string graph recognition problem for surfaces of higher genus is in PSPACE.

In contrast to the situation in the plane, there is no a priori bound known on the lengths in a solution. Nevertheless, it is natural to ask what happens, if we encounter as input a system of quadratic trace equations where the lengths in a solution are specified in binary. We show in Sect. 4 that still we cannot expect any polynomial time algorithm in the trace case. The satisfiability problem for single quadratic matching equation remains NP-hard, even when the length information is given. For the NP-hardness result the trace monoid can be fixed.

2 Quadratic trace equations with involution

Free partially commutative monoids are commonly denoted as *trace monoids* and well-studied in mathematics and computer science [3]. A trace monoid is a free monoid modulo a partial commutation between generators. Motivated by extending results to free partially commutative groups (*graph groups*) one has been led to consider trace monoids with involution. The involution corresponds to taking inverse elements. In the group case the involution is usually without fixed points on generators since in the free case there is no non-trivial element of order two. But when starting the discussion by string graphs then the involution is reading crossing points from right-to-left, so all letters are fixed points. The best is therefore to make no special assumption how the underlying involution behaves on letters. The central object here are trace monoids with involution and quadratic equations for them.

We let Γ be a finite alphabet which is equipped with an involution $^{-} : \Gamma \to \Gamma$. An *involution* is a mapping such that $\overline{\overline{a}} = a$ of all $a \in \Gamma$. By Ω we denote the set of variables and by $\overline{\Omega}$ its disjoint copy, so $\Omega \cup \overline{\Omega}$ is a set with an involution without fixed points. The involution is extended to $(\Gamma \cup \Omega \cup \overline{\Omega})^*$ by letting $\overline{x_1 \cdots x_n} = \overline{x_n} \cdots \overline{x_1}$. Note that, in contrast to variables, we allow fixed points on letters (constants). As a special case we have $\overline{a} = a$ for all $a \in \Gamma$.

An *equation* $U = V$ is a pair (U, V) with $U, V \in (\Gamma \cup \Omega \cup \overline{\Omega})^*$, and a *system of equations* is a finite collections $\{U_1 = V_1, \ldots, U_k = V_k\}$ of equations. It is a *quadratic* system if every variable $x \in \Omega$ appears at most twice in the system. A variable may appear either as x or in the form $\overline{x}$, but both forms are counted for x.

By $I \subseteq \Gamma \times \Gamma$ we denote an *independence relation* which is an irreflexive and symmetric relation. Moreover, we demand that I is compatible with the involution, i.e., we have $(a, b) \in I$ if and only if $(a, \overline{b}) \in I$. Since I is irreflexive, this implies $(a, \overline{a}) \notin I$ for all $a \in \Gamma$.

By $M(\Gamma, I)$ we denote the *trace monoid* which is defined by $M(\Gamma, I) = \Gamma^* / \{ab = ba \mid (a, b) \in I\}$. Thus each trace $t \in M(\Gamma, I)$ can be represented by a word $a_1 \cdots a_n$ with $a_i \in \Gamma$. Moreover, we can define its *length* $|t|$ by n, its *alphabet* $\mathrm{alph}(t)$ by $\{a_1, \ldots, a_n\} \subseteq \Gamma$, and we define $\overline{t}$ by the trace $\overline{a_n} \cdots \overline{a_1}$.

[1] Compare also [9] with its journal version to appear in the Journal of Computer and System Sciences and which will contain this PSPACE-statement, too.

Note that all three notions are well-defined. Hence, the trace monoid $M(\Gamma, I)$ is also equipped with an involution.

We shall use a basic tool from trace theory which is called Levi's Lemma. For a proof see e.g. [5, Thm 1.3.4].

Lemma 1. *Let $x, y, u, v \in M(\Gamma, I)$ be traces. Then the following assertions are equivalent:*

(1) We have $xy = yv$.
(2) There are $p, q, r, s \in M(\Gamma, I)$ such that $\mathrm{alph}(r) \times \mathrm{alph}(s) \subseteq I$ *and*

$$x = pr, \quad y = sq, \quad u = ps, \quad v = rq.$$

By σ we mean a mapping $\sigma : \Omega \to M(\Gamma, I)$. The mapping is extended to a homomorphism $\sigma : (\Gamma \cup \Omega \cup \overline{\Omega})^* \to M(\Gamma, I)$ by defining $\sigma(a) = a$ for $a \in \Gamma$ and $\sigma(\overline{x}) = \overline{\sigma(x)}$ for $x \in \Omega$.

We say that σ is a *solution* of a system $\{U_1 = V_1, \ldots, U_k = V_k\}$, if $\sigma(U_1) = \sigma(V_1), \ldots, \sigma(U_k) = \sigma(V_k)$ are true in $M(\Gamma, I)$. Since solvability is asked in $M(\Gamma, I)$ we speak henceforth about a system of trace equations with involution.

By [2] we know how to solve such a system in PSPACE, if (Γ, I) is not part of the input. Nevertheless the algorithm in [2] is very complex: It is a quite involved reduction to word equations with involution. It relies on Plandowski's method [6] and on the result in [1]. Here we consider the uniform problem where (Γ, I) is part of the input (and no PSPACE bound is known, in general). But we restrict ourselves to quadratic systems (which are much easier to handle): We obtain a very simple PSPACE-algorithm. The algorithm is an analogue to the classical PSPACE-algorithm for quadratic word equations in [4].

Theorem 1. *The following problem can be solved in* PSPACE.

Input:	*A structure $(\Gamma, \overline{}, I)$ and a quadratic system of trace equations with involution over this structure.*
Question:	*Does the system have a solution?*

Proof. After a preprocessing step we may assume that all equations are triangulated, i.e., they have the form $z = xy$ with $x, y, z \in \Gamma \cup \Omega \cup \overline{\Omega}$. Note that if e.g. there were an equation $x = \overline{x}$ for $x \in \Omega$, then this equation was simply removed, since $x = \overline{x}$ is satisfiable and neither x nor $\overline{x}$ appeared elsewhere in the system. Next we guess for each variable $x \in \Omega$ an alphabet $\alpha(x) \subseteq \Gamma$. We let $\alpha(a) = \{a\}$ for $a \in \Gamma$ and $\alpha(\overline{x}) = \overline{\alpha(x)}$ where $\overline{\{a_1, \ldots, a_n\}}$ is the set $\{\overline{a_1}, \ldots, \overline{a_n}\}$. The idea behind $\alpha(x)$ is that for a solution σ we have $\mathrm{alph}(\sigma(x)) = \alpha(x)$ for all $x \in \Gamma \cup \Omega \cup \overline{\Omega}$.

For all equations $z = xy$ we must have $\alpha(z) = \alpha(x) \cup \alpha(y)$. If we write $(r, s) \in I$ we must satisfy $\alpha(r) \times \alpha(s) \subseteq I$. If alphabetic constraints as above cannot be satisfied, we stop the following procedure without success.

In another preprocessing step we remove all variables x with $\alpha(x) = \emptyset$. As a consequence some more variables and equations may disappear. For each equation $z = xy$ we define its weight by the product $|\alpha(x)| \cdot |\alpha(y)|$. The total weight of the system is the sum of the weights of all equations. We fix this weight as the space bound for the algorithm. The algorithm performs Levi's Lemma and removes equations of weight zero as soon as they are created. This procedure is described by four rules. The first three remove equations of weight zero. We must apply them as long as possible before we are allowed to apply the fourth rule which is Levi's Lemma.

Here are four the rules:

Rule 1: If there is an equation $z = xy$ with $z \in \Gamma$, then we must have $|\alpha(x)| \cdot |\alpha(y)| = 0$ (weight zero) and $|\alpha(x)| + |\alpha(y)| = 1$. We remove the equation and we replace all occurrences of x (or of y resp.) by the corresponding constant. (The output is here $x := a$ (or $y := a$ resp.), if $z = a \in \Gamma$. We omit mostly the discussion of the output in the following.)

Rule 2: We may assume (by Rule 1 and by replacing $\overline{z} = xy$ by $z = \overline{y}\,\overline{x}$) that all equations $z = xy$ satisfy $z \in \Omega$. If there is an equation $z = xy$ of weight zero, then we can remove it. More precisely, if, say $|\alpha(y)| = 0$, then we replace z in the remaining equations by x, so we have also one variable less. Note that exactly here we may encounter a situation $z = \overline{z}$, which is a satisfiable constraint if and only if $\alpha(z)$ is closed under involution. The condition $z = \overline{z}$ has to be respected when computing a most general solution.

Rule 3: By applying Rules 1 and 2 as long as possible we may assume that $\alpha(x) \neq \emptyset$ for all remaining variables. If there is an equation $z = xy$ and z appears only once, then we simply remove this equation. (It can be viewed as a definition $z := xy$ moving to the output.)

Rule 4: After Rule 3 there must be a doubly defined variable or the system is empty. Otherwise the system is not solvable. So assume we have two equations:

$$z = xy,$$
$$z = uv.$$

We apply Levi's Lemma. Thus we replace these two equations by four new equations with four new variables:

$$x = pr, \quad y = sq, \quad u = ps, \quad v = rq.$$

We then guess the alphabetical constraints for the new variables p, r, s and q such that we satisfy (or we have to stop):

$$\begin{aligned} \alpha(x) &= \alpha(p) \cup \alpha(r), \\ \alpha(y) &= \alpha(s) \cup \alpha(q), \\ \alpha(u) &= \alpha(p) \cup \alpha(s), \\ \alpha(v) &= \alpha(r) \cup \alpha(q), \\ \alpha(r) \times \alpha(s) &\subseteq I. \end{aligned}$$

The last condition implies $\alpha(r) \cap \alpha(s) = \emptyset$ and this is enough to verify that the total weight did not increase. In fact, no pair $(a, b) \in \Gamma \times \Gamma$ is counted more than twice in the weight for the new equations and if it is counted twice, then it has been counted twice before. If a pair $(a, b) \in \Gamma \times \Gamma$ is counted exactly once in the weight of the new equations, then it was counted once before. There are also some cases, where the weight actually decreases.

Before we allow to apply Rule 4 again, we have to use Rules 1 to 3 as long as possible since some of the four new equations may have weight zero.

It is clear that the procedure can be implemented in PSPACE. If the empty system is calculated, then the original system is satisfiable.

We now consider the converse: Let us assume that the system is satisfiable. Then there is a solution σ and we may define a length by $|x| = 2|\sigma(x)|$ for $x \in \Omega$ and $|a| = 1$ for a constant $a \in \Gamma$. We may define a length of the system by a pair (m, n) where m the maximal length $|x|$ for some variable in the system and n is sum of the lengths of all right-hand sides of the equations. Then we can apply the rules in such a form that at each step at least one component of the length decreases and no component increases. This proves termination and the number of necessary steps is at most linear in the length of the original system. □

Remark 1. From the algorithm described above, we can deduce a doubly exponential upper bound for the length of a minimal solution.

Remark 2. Alphabetic constraints as used in the algorithm above are special instances of recognizable constraints. Assume that in addition, the input contains a list of recognizable trace languages $L(x) \subseteq M(\Gamma, I)$ for all $x \in \Omega$ (each specified, say, by some NFA for all representing words of $L(x)$ in Γ^*). A solution has now to respect the recognizable constraints, too. It is possible to extend the PSPACE-algorithm in such a way that it copes with the more general problem as well. Details are left to the reader.

3 Quadratic word equations with length constraints

In this section, we will describe a linear-time algorithm for solving a system of quadratic word equations with involution. This is essentially the very same algorithm and result as in [8]. The only minor modification is due to the involution.

Theorem 2. *There is a linear time algorithm to solve the following problem (on a unit cost RAM).*

Input: *A quadratic system of word equations E over the structure $(\Gamma, \overline{})$ and a list of natural numbers $(\ell_x)_{x \in \Omega}$, written in binary.*

Output: *If there exists a solution σ with $|\sigma(x)| = \ell_x$ for every $x \in \Omega$ then return an equivalent system with no doubly defined variables, else if there is no such solution: Stop.*

Proof. (The proof is nearly identical to the one in [8, Thm. 2]. However we explain the entire algorithm for convenience of the reader.) The first step is to split every equation into equations of the form $z = xy$, $z \in \Omega$, $x, y \in \Gamma \cup \Omega \cup \overline{\Omega}$. After each step we eliminate variables x with $\ell_x = 0$ and verify that $\ell_z = \ell_x + \ell_y$, where $\ell_{\overline{x}} = \ell_x$ for $x \in \Omega$ and $\ell_a = 1$ for $a \in \Gamma$. In addition, we can output a definition for the eliminated variables after each step. If all variables in the system E are at most defined once, i.e. occur at most once on a left-hand side, the whole systems consists of definitions. In this case we can output E and we are done. Let in the following be $c = 0.55$.

Consider a doubly defined variable z and its two equations:

$$z = xy, \tag{1}$$

$$z = uv. \tag{2}$$

If $\ell_x = \ell_u$, then we either have an immediate contradiction or we can eliminate at least one variable. Therefore, without restriction, $0 < \ell_u < \ell_x$.

If $\ell_x - \ell_u > c \cdot \ell_z$, we have to distinguish different cases. First if $x = v$, then u and y are conjugates. Since x has no other occurrences we can replace (1) and (2) by

$$u = rs, \quad y = sr,$$

where r and s are new variables with $\ell_r = \ell_x \pmod{\ell_u}$ and $\ell_s = \ell_u - \ell_r$.

The next case (and new case w.r.t. [8]) is $\overline{x} = v$. We first have to check whether there exists a word s with $s = \overline{s}$ and $|s| = \ell_x - \ell_u$. If $\ell_x - \ell_u$ is even we can always find such an s and if it is odd this is possible if and only if the involution $\overline{}$ has a fixed point, i.e. there is an $a \in \Gamma$ with $a = \overline{a}$. If no such s exists, we have an immediate contradiction, else we have $u = \overline{y}$ since $\ell_u < \ell_x$. So we can remove (1) and (2) and the variables z, x and u from E. Note, that in a solution we then have $x = us = \overline{y}\,\overline{s}$ and $z = us\overline{u}$. In particular, $z = \overline{z}$.

We will next handle the case $\ell_x - \ell_u > c \cdot \ell_z$, $x \neq v$, $\overline{x} \neq v$ and x or v has a second definition. By symmetry we may assume that there is an equation

$$x = pq. \tag{3}$$

If $\ell_p \geq c \cdot \ell_z$ we introduce a new variable r with $\ell_r = \ell_z - \ell_p$ and replace (1) and (3) by

$$z = pr, \quad r = qy.$$

If $\ell_p < c \cdot \ell_z$ we have to distinguish the sub-cases $\ell_u = \ell_p$, $\ell_u < \ell_p$ and $\ell_u > \ell_p$. If $\ell_u = \ell_p$, then again, either there is an immediate contradiction or we can eliminate a variable. The next sub-case is $\ell_u < \ell_p$. We now introduce new variables w and r with $\ell_w = \ell_x - \ell_u$ and $\ell_r = \ell_p - \ell_u$ and replace (1), (2) and (3) by

$$p = ur, \quad w = rq, \quad v = wy.$$

The last of our three sub-cases is $\ell_u > \ell_p$. In analogy to the previous sub-case we introduce new variables w and r with $\ell_w = \ell_x - \ell_u$ and $\ell_r = \ell_u - \ell_p$ and replace (1), (2) and (3) by

$$u = pr, \quad q = rw, \quad v = wy.$$

If $\ell_x - \ell_u \leq c \cdot \ell_z$ or if we are in none of the cases above, i.e. $x \neq v$, $\overline{x} \neq v$ and neither x nor y has a second definition, we replace (1) and (2) by

$$x = uw, \quad v = wy,$$

where w is a new variable with $\ell_w = \ell_x - \ell_u$.

Let $\mathrm{dd}(E)$ be the number of doubly defined variables in E and k be some value such that $k \geq -3/\ln(c) \approx 5.01$. We define the weight of E as follows: $W(E) = |\Omega| + \mathrm{dd}(E) + k\sum_{x\in\Omega} \ln \ell_x$. This weight is linear in the size of the input. The algorithm reduces the weight after each transformation by at least one: In the case $\ell_x - \ell_u > c \cdot \ell_z$ and $\overline{x} = v$, the overall effect is that z, v $(= \overline{x})$ and u $(= \overline{y})$ have been removed and the number of doubly defined variables did not increase, since we eliminated (1) and (2) but u may now be doubly defined. Hence, we have reduced $W(E)$ by at least 3. The calculations for the other cases can be found in [8]. Since a single transformation needs only constant time this yields a linear time algorithm. □

4 Quadratic trace equations with length constraints

In this section we show that one cannot hope for a polynomial time algorithm to solve quadratic trace equations with length constraints. This happens for a simple matching problem and when underlying trace monoid is generated by three letters only; this number of letters is minimal.

Theorem 3. *Let $M(\Gamma, I) = \{a,b\}^* \times \{c\}^*$. The following problem is* NP*-hard.*

Input: *A single quadratic trace equation where the right-hand side is a trace, and a list of natural numbers $(\ell_x)_{x\in\Omega}$, where Ω denotes the set of variables.*

Question: *Does the system have a solution σ satisfying $|\sigma(x)| = \ell_x$ for every $x \in \Omega$?*

Proof. In [8] it has been shown that (using a reduction the from $3-\mathrm{SAT}$) the following matching problem is NP-hard.
Input: A quadratic word equation $L = r$ over the alphabet $\{a,b\}$ with a word $r \in \{a,b\}^*$.
Question: Is there a solution $\sigma : \Omega \to \{a,b\}^*$?

We will use this problem for a reduction to a quadratic equation over the trace monoid $\{a,b\}^* \times \{c\}^*$.

Let $w_0x_1w_1x_2w_2\cdots x_kw_k = r$ be a word equation where the w_i, $0 \leq i \leq k$, and r are words over $\{a,b\}$ and the x_i, $1 \leq i \leq k$, are variables. Without restriction, we can assume $k \geq 1$. An upper bound for every solution of the x_i is $\ell = |r| - \sum_{i=0}^{k} |w_i|$. If ℓ is negative, the above equation has no solution. We transform the equation into the following trace equation over $\{a,b\}^* \times \{c\}^*$:

$$w_0x_1w_1x_2w_2\cdots x_kw_k = rc^{(k-1)\cdot\ell}. \tag{4}$$

We set $\ell_{x_i} = \ell$ for $1 \leq i \leq k$.

Clearly, the original system has a solution if and only if (4) has a solution satisfying the length constraints. □

Conclusion

We have seen that quadratic trace equations seem to behave as good (or as bad) as quadratic word equations. However, if lengths constraints are considered then the word case is simpler and we have no analogue to Theorem 2 for traces. We conjecture however that for every trace monoid there is a polynomial time algorithm for solving quadratic trace equations, when, as part of the input, for every variable $x \in \Omega$ and for every letter $a \in \Gamma$ it is specified by some number $\ell_{x,a}$ written in binary how many a occur in the solution for x.

References

1. V. Diekert, C. Gutiérrez, and C. Hagenah. The existential theory of equations with rational constraints in free groups is PSPACE-complete. In A. Ferreira and H. Reichel, editors, *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01), Dresden (Germany), 2001*, number 2010 in Lecture Notes in Computer Science, pages 170–182. Springer, 2001.
2. V. Diekert and A. Muscholl. Solvability of equations in free partially commutative groups is decidable. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP'01)*, number 2076 in Lecture Notes in Computer Science, pages 543–554, Berlin-Heidelberg-New York, 2001. Springer.
3. V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.
4. Yu. Matiyasevich. A connection between systems of word and length equations and Hilbert's Tenth Problem. *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 8:132–144, 1968. In Russian. English translation in: *Sem. Math. V. A. Steklov, 8*, 61–67, 1970.
5. A. Mazurkiewicz. Introduction to trace theory. In V. Diekert and G. Rozenberg, editors, *The Book of Traces*, chapter 1, pages 3–41. World Scientific, Singapore, 1995.
6. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Proc. 40th Ann. Symp. on Foundations of Computer Science, FOCS'99*, pages 495–500. IEEE Computer Society Press, 1999.
7. W. Plandowski and W. Rytter. Application of Lempel-Ziv encodings to the solution of word equations. In K. G. Larsen et al., editors, *Proc. 25th International Colloquium on Automata, Languages and Programming (ICALP'98), Aalborg (Denmark), 1998*, number 1443 in Lecture Notes in Computer Science, pages 731–742, Berlin-Heidelberg-New York, 1998. Springer.
8. J. M. Robson and V. Diekert. On quadratic word equations. In C. Meinel and S. Tison, editors, *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99), Trier (Germany), 1999*, number 1563 in Lecture Notes in Computer Science, pages 217–226. Springer-Verlag, 1999.
9. M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. In *Proceedings 34th Annual ACM Symposium on Theory of Computing, STOC'2002*, pages 1–6, New York, 2002. ACM Press.