Noname manuscript No.
(will be inserted by the editor)

On the Parameterised Complexity of String Morphism Problems

Henning Fernau · Markus L. Schmid · Yngve Villanger

Received: date / Accepted: date

Abstract Given a source string u and a target string w, to decide whether w can be obtained by applying a string morphism on u (i. e., uniformly replacing the symbols in u by strings) constitutes an $\mathcal{NP}$-complete problem. We present a multivariate analysis of this problem (and its many variants) from the viewpoint of parameterised complexity theory, thereby pinning down the sources of its computational hardness. Our results show that most parameterised variants of the string morphism problem are fixed parameter intractable and, apart from some very special cases, tractable variants can only be obtained by considering a large part of the input as parameters, namely the length of w and the number of different symbols in u.

Keywords String Problems · String Morphisms · Parameterised Complexity · Exponential Time Hypothesis · Pattern Languages

1 Introduction

Many of the typical string problems are concerned with special kinds of string operations like, e. g., concatenating strings with each other, deleting symbols from or inserting symbols into a string or replacing symbols by other symbols or even by other strings. Among the most prominent of these string problems are *string-to-string correction*, *sequence alignment* as well as the *longest*

A preliminary version [15] of this paper was presented at the conference FSTTCS 2013.

Henning Fernau, Markus L. Schmid
Fachbereich IV – Abteilung Informatikwissenschaften,
Universität Trier, D-54296 Trier, Germany
E-mail: {Fernau, MSchmid}@uni-trier.de

Yngve Villanger
Department of Informatics, University of Bergen, Bergen, Norway
E-mail: yngve.villanger@gmail.com

common subsequence and *shortest common supersequence problem*. The complexity of these problems have been intensely studied, both in the classical sense as well as in the parameterised setting (see, e. g., [1, 11, 27, 30]).

In this work, we investigate string problems that arise from a less well-known operation on strings, i. e., mapping a *source string* u to a *target string* w by uniformly (i. e., by a mapping) replacing the symbols of u by strings. For example, we can turn the source string $u := \texttt{abba}$ into the target string $w := \texttt{bbaaaaabba}$ by replacing $\texttt{a}$ and $\texttt{b}$ of u by the strings $\texttt{bba}$ and $\texttt{aa}$, respectively. On the other hand, $w' := \texttt{abaaaaaabb}$ cannot be obtained from u in a similar way. The *string morphism problem*[1] (denoted by StrMorph) is to decide for two given strings u and w, whether or not w can be obtained from u by this kind of operation. Due to its simple definition, variants of this $\mathcal{NP}$-complete problem can be found in many different areas of theoretical computer science. In fact, many respective results are scattered throughout the literature without pointers to each other and consulting the existing literature suggests that variants of the string morphism problem have emerged and have been investigated in different contexts without knowledge of other related work.

A Brief History of the String Morphism Problem

The origin of the string morphism problem is usually traced back to the 1979 paper by Angluin [4], in which she introduced the model of *pattern languages* (the membership problem of which is essentially the string morphism problem), but, independently and at the same time, it has also been studied by Ehrenfeucht and Rozenberg in [12]. Garey and Johnson [19], by referring to a private communication with Aho and Ullman from 1977, report the $\mathcal{NP}$-completeness of the problem Regular Expression Substitution, for which, on close inspection, the string morphism problem turns out to be a natural subproblem.

Since their introduction, Angluin's pattern languages have been intensely studied in the context of learning theory and formal language theory. While questions of learnability as well as language theoretical properties were the main focus of research, results regarding the complexity of their membership problem (except of its general $\mathcal{NP}$-completeness) were sparse and only appeared as by-products (see, e. g., [4, 20, 22, 32]).

In the pattern matching community, independent of Angluin's work, the pattern matching problem described above has been rediscovered in a series of papers. This development starts with [5] in which Baker introduces so-called *parameterised pattern matching*, where a pattern with parameters and a text is given and the text is searched for factors that can be obtained by uniformly replacing the parameters of the pattern by single symbols. Furthermore, different parameters must be replaced by different symbols. Amir et al. [2] generalise this problem to *function matching* by dropping this injectivity condition and Amir and Nor [3] as well as Clifford et al. [9] consider function matching where

[1] We choose this name, since, mathematically speaking, it is the problem to decide on the existence of a morphism between two strings.

parameters can be substituted by words instead of single symbols, which leads to patterns as introduced by Angluin. In [3], motivations for this kind of pattern matching can be found from such diverse areas as software engineering, image searching, DNA analysis, poetry and music analysis, or author validation.

The string morphism problem can also be seen as the solvability problem for word equations where one side does not contain variables (for more details on word equations, see Mateescu and Salomaa [26]). Combinatorial properties of the operation of uniformly replacing the symbols in a string by other strings are investigated in numerous other areas of theoretical computer science and discrete mathematics, such as (un-)avoidable patterns (cf. Jiang et al. [24]), the ambiguity of morphisms (cf. Freydenberger et al. [18]) and equality sets (cf. Harju and Karhumäki [21]). Last but not least, the string morphism problem can also be found in practical applications. More precisely, it constitutes a special case of the matchtest for *regular expressions with backreferences* (see, e. g., Câmpeanu et al. [6]), which nowadays are a standard element of most text editors and programming languages.

Our Contribution

A systematic study of the computational complexity of StrMorph has been taken on just recently. In [28, 29, 31], several possibilities are presented of how to restrict the structure of the source strings, such that StrMorph can be solved in polynomial time, and in [14], the $\mathcal{NP}$-completeness of a large number of strongly restricted versions of StrMorph is shown.

In the literature mentioned above, different *variants* of StrMorph are considered, each tailored to different aspects and research questions. The most common variants arise from whether or not we allow symbols to be *erased* (i. e., replaced by the empty string), whether or not we allow *constants* (also called *terminals*) in the source string, which cannot be replaced and whether or not the replacement function needs to be *injective* (i. e., different symbols cannot be replaced by the same string). While the subtle difference of whether or not symbols can also be erased has a substantial impact on decidability questions for pattern languages, the differentiation of StrMorph in an injective and a non-injective version is motivated by pattern matching tasks. In addition to these different variants of StrMorph, we can observe many natural *parameters* (in the definition of the following parameters, let u be a source string over the alphabet A and let w be a target string over the alphabet B).

– p_1: The cardinality of A.
– p_2: The length of w.
– p_3: The cardinality of B.
– p_4: The maximum length of the strings substituted for the symbols in u.
– p_5: The maximum number of occurrences of any symbol in u.

In [14], for every variant of StrMorph and for every subset of the above mentioned parameters, it is shown whether the chosen parameters can be bounded by constants such that the resulting version of StrMorph can be solved in polynomial time or whether it is still $\mathcal{NP}$-complete. For example, if the cardinality of A is bounded by a constant, then all variants of StrMorph can be solved in polynomial time. This trivially follows from the fact that there is a simple brute-force algorithm that runs in time that is exponential only in p_1. Close inspection reveals that StrMorph can also be solved in time that is exponential only in p_2 (for details, the reader is referred to [20] and also [14]). Consequently, in terms of parameterised complexity, StrMorph parameterised by p_1 or p_2 is in XP and the question arises whether these problems are in FPT. Unfortunately, as a main result of this work, we report that both of the above parametrisations are $W[1]$-hard, even if additional parameters are taken into account. More precisely, we show that the following versions of StrMorph are $W[1]$-hard:

1. *all variants* of StrMorph parameterised by (p_1, p_3, p_5),
2. *all variants* of StrMorph (except the nonerasing ones) parameterised by (p_2, p_3, p_4, p_5).

For obtaining the first result, we extend a result by Stephan et al. [33], who showed the $W[1]$-hardness for a special variant of StrMorph parameterised by p_1 and in order to prove the second result, we devise a new reduction from k-Multicoloured-Clique. With respect to the parameters defined above, this leaves only very few parameterised variants of StrMorph that could possibly be in FPT. In this regard, we report the fixed parameter tractability of the following versions of StrMorph:

3. *all variants* of StrMorph parameterised by (p_1, p_2),
4. *all variants* of StrMorph parameterised by (p_1, p_4),
5. *all nonerasing variants* of StrMorph parameterised by p_2,
6. *the nonerasing, injective variant* of StrMorph parameterised by (p_3, p_4).

The above results 1 to 6 (in conjunction with results from [14]) completely settle the fixed parameter tractability of all possible parameterised variants of StrMorph, with respect to the parameters p_1 to p_5. We complement these results by showing for all the $W[1]$-hard cases $W[1]$-membership or $W[P]$-membership.

We conclude the paper by demonstrating the unlikeness of a subexponential algorithm for an important variant of the string morphism problem by applying the *Exponential Time Hypothesis.*

2 Preliminaries

Let $\mathbb{N} := \{1, 2, 3, \ldots\}$. For an arbitrary alphabet A, a *string* (*over* A) is a finite sequence of symbols from A, and ε is the *empty string.* The notation A^+ refers to the set of all non-empty strings over A, and $A^* := A^+ \cup \{\varepsilon\}$. For the *concatenation* of two strings w_1, w_2 we write $w_1 w_2$. We say that a string $v \in A^*$ is a *substring* (or *factor*) of a string $w \in A^*$ if there are $u_1, u_2 \in A^*$

such that $w = u_1\, v\, u_2$. The powers w^i, $i \in \mathbb{N}$, of a string w are inductively defined by $w^1 := w$ and $w^i = w\, w^{i-1}$. The notation $|K|$ stands for the size of a set K or the length of a string K. By $|w|_b$, we denote the number of occurrences of $b \in A$ in w, by $w[i, j]$, we denote the factor from position i to j in w and $w[i] := w[i, i]$.

Let A and B be alphabets. A mapping $h : A^* \to B^*$ with $h(u\, w) = h(u)\, h(w)$, for every $u, w \in A^*$, is a *morphism*. It can be easily verified that a morphism is uniquely defined by the images $h(b)$, $b \in A$. If, for every $b \in A$, $h(b) \neq \varepsilon$, then h is said to be *nonerasing*. If, for all $b, c \in A$ with $b \neq c$, $h(b) \neq \varepsilon$ and $h(c) \neq \varepsilon$ implies $h(b) \neq h(c)$, then h is E-*injective* and h is *injective* if it is E-injective and nonerasing. The *size* of a morphism h is defined by $|h| := \max\{|h(b)| \mid b \in A\}$. Let A and B be alphabets with $B \subseteq A$. A morphism $h : A^* \to B^*$ that satisfies $h(b) = b$, for every $b \in B$, is a *substitution*.

Example 1 Let $u_1 := x\,x\,y\,z\,y$, $u_2 := x\,\mathtt{a}\,y\,\mathtt{b}\,y\,x\,y$, $w_1 :=$ `ababababab`, $w_2 :=$ `bacbabbacb` and $w_3 :=$ `abaabbababab`. The nonerasing morphism h_1 defined by $h_1(x) := h_1(y) := h_1(z) :=$ `ab` satisfies $h_1(u_1) = w_1$. However, u_1 cannot be mapped to $w_1\mathtt{b} =: w_1'$ by a nonerasing morphism. If, on the other hand, we do not restrict ourselves to nonerasing morphisms, then $h_2(u_1) = w_1'$, where h_2 is defined by $h_2(x) := h_2(y) := \varepsilon$ and $h_2(z) := w_1'$. It can be verified that $g_1(u_2) = w_2$, where g_1 is a substitution defined by $g_1(x) :=$ `bacb`, $g_1(y) := \varepsilon$ and $g_2(u_2) = w_3$, where g_2 is a substitution defined by $g_2(x) := g_2(y) :=$ `ab`. Furthermore, u_2 cannot be mapped to w_2 by a nonerasing substitution and u_2 cannot be mapped to w_3 by an E-injective or injective substitution.

Next, we briefly recall some of the main concepts of parameterised complexity theory (in our definitions and notations, we follow the textbook [16] by Flum and Grohe, to which we also refer for all terms and concepts not explicitly explained here). As it is common in complexity theory, we consider decision problems as languages over some alphabet Γ (if the instances are tuples of several elements, then we assume that they are encoded as single strings over Γ). A *parameterisation* (*of* Γ) is a polynomial time computable mapping $\kappa : \Gamma^* \to \mathbb{N}$ and a *parameterised problem* is a pair (Q, κ), where Q is a problem (over Γ) and κ is a parameterisation of Γ. A parameterised problem (Q, κ) is *fixed-parameter tractable* if there is an algorithm that decides whether $x \in Q$ in time $\mathrm{O}(f(\kappa(x)) \times p(|x|))$, where $f : \mathbb{N} \to \mathbb{N}$ is a recursive function and $p : \mathbb{N} \to \mathbb{N}$ is a polynomial, and such an algorithm is called an *fpt-algorithm*. The class of fixed parameter tractable problems is denoted by FPT. In order to argue for the fixed parameter *intractability* of a parameterised problem, we need the notion of an *fpt-reduction* from a parameterised problem (Q_1, κ_1) (over the alphabet Γ_1) to a parameterised problem (Q_2, κ_2) (over the alphabet Γ_2), which is a function $R : \Gamma_1^* \to \Gamma_2^*$ with the following properties. For every $x \in \Gamma_1^*$, $x \in Q_1$ if and only if $R(x) \in Q_2$, the function R is computable in time $\mathrm{O}(f(\kappa_1(x)) \times p(|x|))$, where f is a recursive function and p is a polynomial, and there is a recursive function $g : \mathbb{N} \to \mathbb{N}$ such that $\kappa_2(R(x)) \leq g(\kappa_1(x))$. The framework of parameterised complexity provides the classes of the so-called W-hierarchy, for which the hard problems are considered fixed parameter intractable. For a detailed definition of the W-hierarchy, we refer to the textbooks [10, 16]; in this paper, all intractability results are $W[1]$-hardness results.

The classes of polynomial and nondeterministically polynomial time solvable problems are denoted by $\mathcal{P}$ and $\mathcal{NP}$, respectively.

Now, we define the string morphisms problems that are investigated in this work. We start with the following most general version of the string morphism problem, which shall serve as a base for the definitions of all the further restricted versions.

StrMorph
Instance: Two strings u and w over some alphabets A and B.
Question: Does there exist a morphism $h : A^* \to B^*$ with $h(u) = w$?

By StrSubst, we denote the version of StrMorph, where instead for a morphism we are looking for a substitution. By adding the prefixes NE, Inj and Ne-Inj, we denote the variants of the problems StrMorph and StrSubst, where the morphism (the substitution) needs to be nonerasing, E-injective and nonerasing injective, respectively. Let SMP be the class containing exactly these 8 variants of the string morphism problem, i. e.,

$$\text{SMP} := \{Z\text{-}\textsc{StrMorph}, Z\text{-}\textsc{StrSubst} \mid Z \in \{\varepsilon, \textsc{Ne}, \textsc{Inj}, \textsc{Ne-Inj}\}\} .$$

Next, we fix some notation that shall be used throughout the paper. For an instance (u, w) of one of the string morphism problems defined above, u is called the *source string*, w is called the *target string* and the respective alphabets A and B with $u \in A^*$ and $w \in B^*$ are called the *source* and *target alphabet*, respectively. From now on, the target alphabet is always denoted by Σ and the source alphabet is X for string morphism problems and $(X \cup \Sigma)$ for string substitution problems, where $X \subseteq \{x_1, x_2, x_3, \ldots\}$. The symbols in Σ are called *terminals* and the symbols in X are called *variables.* For any string $u \in (\Sigma \cup X)^*$, by $\text{var}(u)$ we refer to the set of variables occurring in u and $|u|_{\text{var}}$ is the maximum number of occurrences of a variable in u, i. e., $|u|_{\text{var}} := \max\{|u|_x \mid x \in \text{var}(u)\}$. For the problems in SMP, we consider the following parameters:

Parameter	Description
$\lvert\text{var}(u)\rvert$	the number of variables in the source string
$\lvert\Sigma\rvert$	the cardinality of the target alphabet
$\lvert w\rvert$	the length of the target string
$\lvert u\rvert_{\text{var}}$	the maximum number of occurrences of a variable
$\lvert h\rvert$	the size of the morphism or substitution

A *list of parameters* is a tuple $[\rho_1, \ldots, \rho_k]$, where $1 \leq k \leq 5$, $\{\rho_1, \ldots, \rho_k\} \subseteq \{|\text{var}(u)|, |\Sigma|, |w|, |u|_{\text{var}}, |h|\}$. For example, $[|\text{var}(u)|, |\Sigma|, |h|]$ is a list of parameters. For every $K \in$ SMP and every list L of parameters, by L-K, we denote the problem K parameterised by the sum of the parameters in L, e. g., $[|\Sigma|, |u|_{\text{var}}]$-Ne-StrMorph is the parameterised problem (Ne-StrMorph, κ), where the parameterisation κ is defined by $\kappa(u,w) := |\Sigma| + |u|_{\mathrm{var}}$. As a convention, whenever we consider parameterised problems L-K, $K \in \mathrm{SMP}$, where L contains $|h|$, we assume that the parameter $|h|$ is explicitly given as input along with the source and target string.

In Section 4, in order to argue for the unlikeness of a subexponential algorithm, we shall apply the *Exponential Time Hypothesis* (ETH) by Impagliazzo, Paturi, and Zane [23], which, informally speaking, is the conjecture that 3SAT cannot be solved in time $2^{o(n)}$. For an introduction to ETH, the reader is referred to [17, 25].

3 The Parameterised Complexity of String Morphism Problems

In this section, we show for *every* list of parameters L and for *every* $K \in \mathrm{SMP}$, whether or not L-K is fixed parameter tractable or $W[1]$-hard. We start with the hardness results and then present fpt-algorithms for all the other cases. The section is concluded by showing $W[1]$-membership and $W[P]$-membership for the $W[1]$-hard cases.

3.1 $W[1]$-Hardness

As explained in Section 1, it can be easily seen that all variants of StrMorph can be solved in polynomial time if $|\mathrm{var}(u)|$ or $|w|$ is bounded by a constant. Furthermore, as shall be explained in Section 3.2, it also follows trivially that all variants of StrMorph are in FPT if parameterised by $|\mathrm{var}(u)|$ *and* $|w|$ at the same time. Hence, the most interesting question is whether this also holds if either $|\mathrm{var}(u)|$ or $|w|$ is a parameter. We show that this is very unlikely, since the corresponding parameterised versions of the string morphism problems are $W[1]$-hard.

The Parameter $|\mathrm{var}(u)|$

First, we consider the case that $|\mathrm{var}(u)|$ is a parameter and $|w|$ is not a parameter, for which we can show $W[1]$-hardness, even if $|u|_{\mathrm{var}}$ and $|\Sigma|$ are parameters, too.

Theorem 1 *For every* $K \in \mathrm{SMP}$*,* $[|\mathrm{var}(u)|, |\Sigma|, |u|_{\mathrm{var}}]$-$K$ *is* $W[1]$*-hard.*

Theorem 1 can be proven by a reduction from the canonical parameterised variant of the clique problem:

k-Clique
Instance: A graph $\mathcal{G}$ and an integer k.
Parameter: $\kappa(\mathcal{G}, k) = k$.
Question: Does $\mathcal{G}$ has a clique of size $\kappa(\mathcal{G}, k)$?

It is a well-known fact that k-Clique is complete for $W[1]$ (with respect to parameterised reductions) [16]. In order to prove Theorem 1, we modify a reduction from Stephan et al. [33]. More precisely, in [33] a reduction from k-Clique is used in order to prove the $W[1]$-hardness of $[|\mathrm{var}(u)|, |\Sigma|, |u|_{\mathrm{var}}]$-Ne-StrSubst and we shall modify this reduction in such a way that it works for all problems $[|\mathrm{var}(u)|, |\Sigma|, |u|_{\mathrm{var}}]$-$K$, $K \in$ SMP.

To this end, we define a mapping Φ that maps a given graph $\mathcal{G} = (V, E)$ with $V := \{p_1, p_2, \ldots, p_n\}$ and integer k to a source string $u \in (\Sigma \cup X)^+$ and a target string $w \in \Sigma^+$, where $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$. First, we define π to be the mapping that maps a vertex p_i, $1 \leq i \leq n$, to its binary respresentation over $\{\mathtt{a}, \mathtt{b}\}$, e. g., if $n = 25$, then $\pi(p_3) = \mathtt{aaabb}$, $\pi(p_{13}) = \mathtt{abbab}$ and $\pi(p_{25}) = \mathtt{bbaab}$. For the sake of convenience, in the following definition of u, by z we always denote an occurrence of some variable with only a single occurrence in u. Now we define

$$\overline{u} := z\,\mathtt{c}\,x_1\,\mathtt{c}\,z\,z\,\mathtt{c}\,x_2\,\mathtt{c}\,z\,z\,\mathtt{c}\,x_3\,\mathtt{c}\,z \ldots z\,\mathtt{c}\,x_k\,\mathtt{c}\,z\,,$$
$$\overline{w} := \mathtt{c}^3\,\pi(p_1)\,\mathtt{c}^4\,\pi(p_2)\,\mathtt{c}^5 \ldots \mathtt{c}^{n+2}\,\pi(p_n)\,\mathtt{c}^{n+3}\,.$$

The string $\overline{u}$ is an enumeration of k vertices (represented by the variables $x_1, x_2, \ldots, x_k$) and $\overline{w}$ is an enumeration of all the vertices of $\mathcal{G}$, encoded as binary strings over $\{\mathtt{a}, \mathtt{b}\}$. The idea is that by mapping $\overline{u}$ to $\overline{w}$, we pick exactly k vertices of $\mathcal{G}$ by mapping the variables $x_1, x_2, \ldots, x_k$ to exactly k encodings of vertices. We now have to construct another gadget that makes sure that these k selected vertices form a k-clique of $\mathcal{G}$. To this end, we first encode the clique property, i. e., the property that between each two selected vertices there is an edge in $\mathcal{G}$, as a string $\widehat{u}$:

$$\begin{aligned}\widehat{u} := {} & z\,\mathtt{d}\,x_1\,x_2\,\mathtt{d}\,z\,\mathtt{d}\,x_1\,x_3\,\mathtt{d}\,z\,\mathtt{d}\,x_1\,x_4\,\mathtt{d}\,z \ldots z\,\mathtt{d}\,x_1\,x_k\,\mathtt{d}\\ & z\,\mathtt{d}\,x_2\,x_3\,\mathtt{d}\,z\,\mathtt{d}\,x_2\,x_4\,\mathtt{d}\,z \ldots z\,\mathtt{d}\,x_2\,x_k\,\mathtt{d}\\ & \vdots\\ & z\,\mathtt{d}\,x_{k-2}\,x_{k-1}\,\mathtt{d}\,z\,\mathtt{d}\,x_{k-1}\,x_k\,\mathtt{d}\\ & z\,\mathtt{d}\,x_{k-1}\,x_k\,\mathtt{d}\,z\,.\end{aligned}$$

Before we can define the counterpart of $\widehat{u}$, which will be a string representing the complete structure of the graph $\mathcal{G}$, we need the following definition. For every i, $1 \leq i \leq n-1$, let $e_{i,1}, e_{i,2}, \ldots, e_{i,l_i}$ be an enumeration of exactly the edges $\{p_i, p_j\} \in E$, with $i < j$ and let this enumeration be in ascending order with respect to j, e. g., $l_3 = 4$ and $e_{3,1} = \{p_3, p_1\}, e_{3,2} = \{p_3, p_5\}, e_{3,3} = \{p_3, p_7\}, e_{3,4} = \{p_3, p_8\}$. Furthermore, we extend the mapping π to edges by defining $\pi(\{p_i, p_j\}) = \pi(p_i)\,\pi(p_j)$ for every edge $\{p_i, p_j\}$. Now, we define

$$\begin{aligned}\widehat{w} := {} & \mathtt{d}^3\,\pi(e_{1,1})\,\mathtt{d}^4\,\pi(e_{1,2})\,\mathtt{d}^5 \ldots \mathtt{d}^{l_1+2}\,\pi(e_{1,l_1})\\ & \mathtt{d}^{l_1+3}\,\pi(e_{2,1})\,\mathtt{d}^{l_1+4}\,\pi(e_{2,2})\,\mathtt{d}^{l_1+5} \ldots \mathtt{d}^{l_1+l_2+2}\,\pi(e_{2,l_2})\\ & \vdots\\ & \mathtt{d}^{l_1+\ldots+l_{n-2}+3}\,\pi(e_{n-1,1})\,\mathtt{d}^{l_1+\ldots+l_{n-2}+4} \ldots \mathtt{d}^{l_1+\ldots+l_{n-1}+2}\,\pi(e_{n-1,l_{n-1}})\end{aligned}$$

Finally, we define $u := \overline{u}\,\mathtt{e}\,\widehat{u}$, $w := \overline{w}\,\mathtt{e}\,\widehat{w}$ and $\Phi(\mathcal{G}, k) := (u, w)$.

The proof of the following lemma is partly due to Stephan et al. [33].

Lemma 1 *Let $\mathcal{G}$ be a graph, let k be an integer, and let $(u, w) := \Phi(\mathcal{G}, k)$. The following statements are equivalent:*

1. *There is a clique of size k in $\mathcal{G}$.*
2. *There is a substitution h such that $h(u) = w$.*
3. *There is an* E*-injective substitution h such that $h(u) = w$.*
4. *There is a nonerasing substitution h such that $h(u) = w$.*
5. *There is a nonerasing injective substitution h such that $h(u) = w$.*

Proof We first note that, by definition, (5) implies (4), (4) implies (3) and (3) implies (2). Next, we show that (2) implies (5), which means that (2) - (5) are equivalent. To this end, we assume that there exists a substitution h with $h(u) = w$. Clearly, $h(\overline{u}) = \overline{w}$ and $h(\widehat{u}) = \widehat{w}$. From $h(\overline{u}) = \overline{w}$ and the fact that there is no occurrence of $\mathtt{c}$ in $\widehat{w}$, but at least one occurrence of every x_i, $1 \leq i \leq k$, in $\widehat{u}$, we can conclude that $h(x_i) \in \{\pi(p_1), \pi(p_2), \dots, \pi(p_n)\}$. Furthermore, if any of the variables z in $\overline{u}$ is mapped to ε, then $h(u)$ starts with a single occurrence of $\mathtt{c}$ followed by a symbol different from $\mathtt{c}$ or in $h(u)$ there is an occurrence of a factor $\mathtt{cc}$ delimited by symbols different from $\mathtt{c}$, which is a contradiction, since this is not the case for w. Analogously, the assumption that any of the variables z in $\widehat{u}$ is mapped to ε leads to a contradiction as well. Next, we observe that all the variables x_i, $1 \leq i \leq k$, as well as the variables z, are mapped to different strings. Thus, if there exists a substitution h with $h(u) = w$, then h is nonerasing injective, which implies the statement (5).

In order to conclude the proof, we first show that (1) implies (2) and then that (5) implies (1).

We assume that there is a clique of size k in $\mathcal{G}$. We define a substitution h with $h(u) = w$ by allocating exactly the vertices of the k-clique to the variables x_i, $1 \leq i \leq k$. More precisely, if, e. g., $k = 4$ and $\{p_2, p_4, p_8, p_9\}$ is the k-clique, then we define $h(x_1) := \pi(p_2)$, $h(x_2) := \pi(p_4)$, $h(x_3) := \pi(p_8)$ and $h(x_4) := \pi(p_9)$. Since the variables x_i, $1 \leq i \leq 4$, appear in $\overline{u}$ in the order x_1, x_2, x_3, x_4 and the factors $\pi(p_i)$, $i \in \{2, 4, 8, 9\}$ appear in $\overline{w}$ in the order $\pi(p_2), \pi(p_4), \pi(p_8), \pi(p_9)$, the variables z can be substituted in such a way that $h(\overline{u}) = \overline{w}$ holds. Next, we note that $\widehat{u}$ contains the factors $x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4$ in exactly this order and $\widehat{w}$ contains the factors $\pi(\{p_2, p_4\})$, $\pi(\{p_2, p_8\})$, $\pi(\{p_2, p_9\})$, $\pi(\{p_4, p_8\})$, $\pi(\{p_4, p_9\})$, $\pi(\{p_8, p_9\})$ in exactly this order; thus, the variables z can be substituted in such a way that $h(\widehat{u}) = \widehat{w}$ holds. This implies that $h(u) = w$.

Next, we assume that there exists a nonerasing injective substitution h with $h(u) = w$. In the same way as above, this implies that, for every i, $1 \leq i \leq k$, $h(x_i) \in \{\pi(p_1), \pi(p_2), \dots, \pi(p_n)\}$. Informally speaking, this means that the variables x_i, $1 \leq i \leq k$, necessarily pick k vertices. Since, for every i, j, $1 \leq i < j \leq k$, the factor $x_i\,x_j$ occurs in $\widehat{u}$ and $h(\widehat{u}) = \widehat{w}$, in $\widehat{w}$ factor $h(x_1)\,h(x_2)$ must occur, which represents the edge between the vertex picked by x_i and the vertex picked by x_j. We recall that $\widehat{w}$ contains exactly the edges of the graph $\mathcal{G}$, which implies that for all i, j, $1 \leq i < j \leq k$, in the graph $\mathcal{G}$ there is an edge between the vertex picked by x_i and the vertex picked by x_j. Thus, the vertices picked by variables x_i form a clique in $\mathcal{G}$. This concludes the proof. □

In order to use the reduction Φ to conclude the $W[1]$-hardness results claimed in Theorem 1, it is important to note that Φ is a parameterised reduction with respect to the parameters $|\mathrm{var}(u)|$, $|\Sigma|$ and $|u|_{\mathrm{var}}$.

Proposition 1 *Let $\mathcal{G}$ be a graph, let k be an integer, and let $(u, w) := \Phi(\mathcal{G}, k)$. Then $|\mathrm{var}(u)|$ and $|u|_{\mathrm{var}}$ are bounded by a function in k and $|\Sigma| = 5$.*

Proof Obviously, there are k variables x_i, $1 \leq i \leq k$. In $\overline{u}$, there are $k+1$ variables z and in $\widehat{u}$ there are $\mathrm{O}(k^2)$ variables z. The alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ has a cardinality of 5. Every variable x_i, $1 \leq i \leq k$, has one occurrence in $\overline{u}$ and $k-1$ occurrences in $\widehat{u}$; thus, $|u|_{x_i} = k$, $1 \leq i \leq k$. □

So far, we have proven the statement of Theorem 1 only for the problems Z-StrSubst, $Z \in \{\textsc{Ne}, \textsc{Inj}, \textsc{Ne-Inj}, \varepsilon\}$. In order to conclude the proof, we show that the above transformation Φ can be extended in such a way that it works for the problems Z-StrMorph, $Z \in \{\textsc{Ne}, \textsc{Inj}, \textsc{Ne-Inj}, \varepsilon\}$ as well. To this end, let $\mathcal{G} := (V, E)$ be a graph and let k be some integer. Furthermore, let $(u, w) := \Phi(\mathcal{G}, k)$. We define

$$u' := x_{\mathtt{e}}\, x_{\mathtt{c}}\, x_{\mathtt{d}}\, (u'')^2\,,$$
$$w' := \mathtt{e}\,\mathtt{c}\,\mathtt{d}\,(w)^2\,,$$

where $x_{\mathtt{c}}$, $x_{\mathtt{d}}$ and $x_{\mathtt{e}}$ are new variables and u'' is obtained from u by substituting every occurrence of $\mathtt{c}$, $\mathtt{d}$ and $\mathtt{e}$ by an occurrence of the new variables $x_{\mathtt{c}}$, $x_{\mathtt{d}}$ and $x_{\mathtt{e}}$, respectively. We note that $u' \in X^*$.

We can now prove an analogue of Lemma 1 with respect to this modified reduction:

Lemma 2 *Let u' and w' be defined as above. The following statements are equivalent:*

1. *There is a clique of size k in $\mathcal{G}$.*
2. *There is a morphism h' such that $h(u') = w'$.*
3. *There is an* E*-injective morphism h' such that $h'(u') = w'$.*
4. *There is a nonerasing morphism h such that $h'(u') = w'$.*
5. *There is a nonerasing injective morphism h such that $h(u') = w'$.*

Proof We assume that (1) holds. By Lemma 2, this implies that there exists an injective nonerasing substitution h with $h(u) = w$ (recall that $(u, w) = \Phi(\mathcal{G}, k)$). The substitution h can be transformed into an injective nonerasing morphism that maps u' to w' by defining $h(x_{\mathtt{c}}) := \mathtt{c}$, $h(x_{\mathtt{d}}) := \mathtt{d}$ and $h(x_{\mathtt{e}}) := \mathtt{e}$, which implies (5) and, by definition, also (2), (3) and (4).

Next, we assume that (2) holds, i.e., there exists a morphism h' with $h'(u') = w'$. If $h'(x_{\mathtt{c}}) = \mathtt{c}$, $h'(x_{\mathtt{d}}) = \mathtt{d}$ and $h'(x_{\mathtt{e}}) = \mathtt{e}$, then $h'(u'') = w$;

thus, $h'(u) = w$ if h' is interpreted as a substitution. If $h(x_{\mathtt{e}}x_{\mathtt{c}}x_{\mathtt{d}})$ is a proper prefix of $\mathtt{ecd}$, then h' must map $(u'')^2$ to a word that is not a square, which is not possible. If neither $h'(x_{\mathtt{e}}x_{\mathtt{c}}x_{\mathtt{d}})$ is a proper prefix of $\mathtt{ecd}$ nor $h'(x_{\mathtt{e}}) = \mathtt{e}$, $h'(x_{\mathtt{c}}) = \mathtt{c}$, $h'(x_{\mathtt{d}}) = \mathtt{d}$, then for at least one $x \in \{x_{\mathtt{c}}, x_{\mathtt{d}}, x_{\mathtt{e}}\}$, $h'(x)$ contains the factor $\mathtt{ec}$, $\mathtt{cd}$ or $\mathtt{dc}$. Since x has at least 2 occurrences in u', but the factors $\mathtt{ec}$, $\mathtt{cd}$ or $\mathtt{dc}$ have only one occurrence in w', this is a contradiction. Consequently, $h'(u') = w'$ implies $h'(x_{\mathtt{c}}) = \mathtt{c}$, $h'(x_{\mathtt{d}}) = \mathtt{d}$ and $h'(x_{\mathtt{e}}) = \mathtt{e}$ and therefore, as explained above, there exists a substitution h with $h(u) = w$. By Lemma 2, this means that (1) is implied. □

It can be easily verified that an analogue of Proposition 1 holds for the modified reduction described above, which means that it is a parameterised reduction with respect to the parameters $|\text{var}(u)|$, $|\Sigma|$ and $|u|_{\text{var}}$. This concludes the proof of Theorem 1.

The Parameter $|w|$

Next, we consider the case where $|w|$ is a parameter instead of $|\text{var}(u)|$. In this regard, we can state a rather strong result, i. e., the $W[1]$-hardness for all (but the nonerasing)[2] variants of string morphism problems parameterised by all the considered parameters except $|\text{var}(u)|$.

Theorem 2 *For every $Z \in \{\text{Inj}, \varepsilon\}$ and $K \in \{\text{StrMorph}, \text{StrSubst}\}$, the problem $[|w|, |\Sigma|, |u|_{\text{var}}, |h|]$-$Z$-$K$ is $W[1]$-hard.*

The reductions that have been used in order to prove Theorem 1 are of no use for proving Theorem 2, since they produce target strings whose lengths depend on the size of the graph and not on the size of the clique. For the proof of Theorem 2, we utilise the following variant of k-Clique:

k-Multicoloured-Clique
Instance: A graph $\mathcal{G} := (V, E)$, a partition $V_1, V_2, \ldots, V_k$ of V, such that every V_i is an independent set.
Parameter: $\kappa(\mathcal{G}, V_1, V_2, \ldots, V_k) = k$.
Question: Does $\mathcal{G}$ has a clique of size $\kappa(\mathcal{G}, V_1, V_2, \ldots, V_k)$?

Obviously, an instance $(\mathcal{G}, V_1, \ldots, V_k)$ of k-Multicoloured-Clique is a positive instance if and only if $\mathcal{G}$ has a k-clique with exactly one element from each V_i, $1 \leq i \leq k$. It is a well-known fact that k-Multicoloured-Clique is complete for $W[1]$ (with respect to parameterised reductions) [13].

We now define a mapping Φ that maps a given graph $\mathcal{G} := (V, E)$ and a partition $V_1, V_2, \ldots, V_k$ of V to a source string $u \in (\Sigma \cup X)^+$ and a target string $w \in \Sigma^+$, where $\Sigma := \{\mathtt{a}_{\{i,j\}} \mid 1 \leq i \leq j \leq k, i \neq j\} \cup \{\$\}$ and $X := \{x_e \mid e \in E\}$. For the sake of concreteness, we define, for every i, $1 \leq i \leq k$, $V_i := \{v_{i,1}, v_{i,2}, \ldots, v_{i,t_i}\}$. Note that, for every i, j, $1 \leq i \leq j \leq k$,

[2] It shall be explained later in Section 3.2 that the nonerasing variants of the string morphism problem are trivially fixed parameter tractable if parameterised by $|w|$.

$i \neq j$, the symbols $\mathtt{a}_{\{i,j\}}$ and $\mathtt{a}_{\{j,i\}}$ are considered identical. For every i, j, $1 \leq i < j \leq k$, we define

$$\begin{aligned} \overline{u}_{i,j} &:= \$\, x_{e^i_{j,1}}\, x_{e^i_{j,2}} \ldots x_{e^i_{j,l_{i,j}}}\, \$\,, \\ \overline{w}_{i,j} &:= \$\, \mathtt{a}_{\{i,j\}}\, \$\,, \end{aligned}$$

where $e^i_{j,1}, e^i_{j,2}, \ldots, e^i_{j,l_{i,j}}$ is an enumeration of exactly the edges between V_i and V_j. We recall that in the reduction used for proving Theorem 1, we use the variables in order to pick k vertices from the graph. Here, we apply a similar idea, but with respect to the edges of the graph. More precisely, if we map $\overline{u}_{i,j}$ to $\overline{w}_{i,j}$, then exactly one $x_{e^i_{j,q}}$, $1 \leq q \leq l_{i,j}$, is mapped to $\mathtt{a}_{\{i,j\}}$, which means that we pick the edge $e^i_{j,q}$ as the one that serves as the connection between V_i and V_j in the clique we are looking for. All these factors $\overline{u}_{i,j}$ and $\overline{w}_{i,j}$ are appended in the following way:

$$\begin{aligned} \overline{u} &:= \overline{u}_{1,2}\, \overline{u}_{1,3} \ldots \overline{u}_{1,k}\, \overline{u}_{2,3}\, \overline{u}_{2,4} \ldots \overline{u}_{2,k} \ldots \overline{u}_{k-2,k-1}\, \overline{u}_{k-2,k}\, \overline{u}_{k-1,k}\,, \\ \overline{w} &:= \overline{w}_{1,2}\, \overline{w}_{1,3} \ldots \overline{w}_{1,k}\, \overline{w}_{2,3}\, \overline{w}_{2,4} \ldots \overline{w}_{2,k} \ldots \overline{w}_{k-2,k-1}\, \overline{w}_{k-2,k}\, \overline{w}_{k-1,k}\,. \end{aligned}$$

It now remains to define a gadget that makes sure that the selected edges are in fact the edges of a k-clique. To achieve this, it is sufficient to ensure that, for every i, $1 \leq i \leq k$, all the $k - 1$ selected edges that are connected to some vertex in V_i are all connected to exactly the same vertex in V_i. To this end, for every i, $1 \leq i \leq k$, we define a gadget $(\widetilde{u}_i, \widetilde{w}_i)$ in the following way. For every j, p, $1 \leq p \leq t_i$, $1 \leq j \leq k$, $i \neq j$, we define

$$\widehat{u}_{i,p,j} := x_{e^p_{j,1}}\, x_{e^p_{j,2}} \ldots x_{e^p_{j,s_{p,j}}}\,,$$

where $e^p_{j,1}, e^p_{j,2}, \ldots, e^p_{j,s_{p,j}}$ is an enumeration of exactly the edges between vertex $v_{i,p}$ and some vertex of V_j. Next, for every p, $1 \leq p \leq t_i$, we define

$$\widehat{u}_{i,p} := \widehat{u}_{i,p,1}\, \widehat{u}_{i,p,2} \ldots \widehat{u}_{i,p,i-1}\, \widehat{u}_{i,p,i+1}\, \widehat{u}_{i,p,i+2} \ldots \widehat{u}_{i,p,k}\,.$$

This means that $\widehat{u}_{i,p}$ is an enumeration of all edges adjacent to the vertex $v_{i,p}$ in the following order: first, we list all edges connecting $v_{i,p}$ with some vertex in V_1, then all edges connecting $v_{i,p}$ with some vertex in V_2 and so on. Our goal is to enforce that, for every i, $1 \leq i \leq k$, there is exactly one q, $1 \leq q \leq t_i$, such that $\widehat{u}_{i,q}$ is mapped to the string $\mathtt{a}_{\{i,1\}}\mathtt{a}_{\{i,2\}} \ldots \mathtt{a}_{\{i,i-1\}}\mathtt{a}_{\{i,i+1\}}\mathtt{a}_{\{i,i+2\}} \ldots \mathtt{a}_{\{i,k\}}$ (which represents the selected edges between V_i and all other V_j, $1 \leq j \leq k$, $i \neq j$) and all other $\widehat{u}_{i,q'}$, $1 \leq q' \leq t_i$, $q \neq q'$, are mapped to the empty word. To this end, we define

$$\begin{aligned} \widetilde{u}_i &:= \widehat{u}^2_{i,1}\, \widehat{u}^2_{i,2} \ldots \widehat{u}^2_{i,t_i}\,, \\ \widetilde{w}_i &:= \left(\mathtt{a}_{\{i,1\}}\, \mathtt{a}_{\{i,2\}} \ldots \mathtt{a}_{\{i,i-1\}}\, \mathtt{a}_{\{i,i+1\}}\, \mathtt{a}_{\{i,i+2\}} \ldots \mathtt{a}_{\{i,k\}}\right)^2. \end{aligned}$$

Furthermore, we define

$$\begin{aligned} \widetilde{u} &:= \$\, \widetilde{u}_1\, \$\, \widetilde{u}_2\, \$ \ldots \$\, \widetilde{u}_k\, \$\,, \\ \widetilde{w} &:= \$\, \widetilde{w}_1\, \$\, \widetilde{w}_2\, \$ \ldots \$\, \widetilde{w}_k\, \$\,, \end{aligned}$$

and, finally, $u := \overline{u}\,\widetilde{u}$, $w := \overline{w}\,\widetilde{w}$ and $\Phi(\mathcal{G}, V_1, V_2, \ldots, V_k) := (u, w)$.

The next lemma states that the function Φ defined above is a reduction from k-Multicoloured-Clique to Inj-StrSubst:

Lemma 3 *Let $\mathcal{G} := (V, E)$ be a graph, let $V_1, V_2, \ldots, V_k$ be a partition of V, such that every V_i is an independent set and let $(u, w) := \Phi(\mathcal{G}, V_1, V_2, \ldots, V_k)$. There exists a clique of size k in $\mathcal{G}$ if and only if there exists an* E*-injective substitution h of size* 1 *with $h(u) = w$ if and only if there exists a substitution h' of size* 1 *with $h'(u) = w$.*

Proof We first note that, for every substitution h, if $h(u) = w$, then h has property $(*)$: for every i, j, $1 \leq i \leq j \leq k$, $i \neq j$, where $e^i_{j,1}, e^i_{j,2}, \ldots, e^i_{j,l_{i,j}}$ are exactly the edges between V_i and V_j, there exists an s, $1 \leq s \leq l_{i,j}$, such that $h(x_{e^i_{j,s}}) = \mathtt{a}_{\{i,j\}}$ and, for every s', $1 \leq s' \leq l_{i,j}$, $s \neq s'$, $h(x_{e^i_{j,s}}) = \varepsilon$. This is due to the fact that $h(u) = w$ implies $h(\overline{u}_{i,j}) = \overline{w}_{i,j}$, for every i, j, $1 \leq i \leq j \leq k$, $i \neq j$, and $\overline{u}_{i,j}$ is a listing (delimited by the symbol \$) of variables corresponding to exactly the edges $e^i_{j,1}, e^i_{j,2}, \ldots, e^i_{j,l_{i,j}}$ and $\overline{w}_{i,j}$ equals $\$\,\mathtt{a}_{\{i,j\}}\,\$$. We further observe that if a substitution has property $(*)$, then, since every variable occurrs in some $\overline{u}_{i,j}$, it is necessarily E-injective and of size 1. This means that in order to prove the statement of the lemma, it is sufficient to show that there exists a clique of size k in $\mathcal{G}$ if and only if there exists a substitution h with $h(u) = w$ and h has property $(*)$. Obviously, a substitution with property $(*)$ can be interpreted as selecting a subset of the edges of $\mathcal{G}$, i. e., for every $e \in E$, if $h(x_e) \neq \varepsilon$, then we say that h selects the edge e.

We start with the *if* direction and assume that there exists a substitution h with property $(*)$ and $h(u) = w$. Let C_E be the set of edges selected by h and let C_V be the set of corresponding vertices, i. e., the set of all vertices that are adjacent to some edge of C_E. Since, for every i, j, $1 \leq i \leq j \leq k$, $i \neq j$, $h(\overline{u}_{i,j}) = \overline{w}_{i,j}$, we can conclude that in C_E there is exactly one edge between V_i and V_j, which, in particular, implies that $V_i \cap C_V \neq \emptyset$ and $V_j \cap C_V \neq \emptyset$.

We note that, due to the occurrences of symbols \$ in $\widetilde{u}$ and $\widetilde{w}$, for every i, $1 \leq i \leq k$,

$$h(\widetilde{u}_i) = h(\widehat{u}^2_{i,1}\,\widehat{u}^2_{i,2} \ldots \widehat{u}^2_{i,t_i}) = \\ (\mathtt{a}_{\{i,1\}}\,\mathtt{a}_{\{i,2\}} \ldots \mathtt{a}_{\{i,i-1\}}\,\mathtt{a}_{\{i,i+1\}}\,\mathtt{a}_{\{i,i+2\}} \ldots \mathtt{a}_{\{i,k\}})^2 = \widetilde{w}_i\,.$$

Now let p, $1 \leq p \leq t_i$, be such that $h(\widehat{u}_{i,p})$ contains some $\mathtt{a}_{i,j}$, $1 \leq j \leq k$, $j \neq i$, as a factor. This directly implies that $h((\widehat{u}_{i,p})^2)$ contains

$$\mathtt{a}_{\{i,j\}}\,\mathtt{a}_{\{i,j+1\}} \ldots \mathtt{a}_{\{i,i-1\}}\,\mathtt{a}_{\{i,i+1\}}\,\mathtt{a}_{\{i,i+2\}} \ldots \mathtt{a}_{\{i,k\}}\,\mathtt{a}_{\{i,1\}}\,\mathtt{a}_{\{i,2\}} \ldots \mathtt{a}_{\{i,j\}}$$

if $j < i$ and

$$\mathtt{a}_{\{i,j\}}\,\mathtt{a}_{\{i,j+1\}} \ldots \mathtt{a}_{\{i,k\}}\,\mathtt{a}_{\{i,1\}}\,\mathtt{a}_{\{i,2\}} \ldots \mathtt{a}_{\{i,i-1\}}\,\mathtt{a}_{\{i,i+1\}}\,\mathtt{a}_{\{i,i+2\}} \ldots \mathtt{a}_{\{i,j\}}$$

if $i < j$, as a factor. Now assume that $h((\widehat{u}_{i,p})^2) \neq \widetilde{w}_i$. This means that a non-empty prefix q of $\mathtt{a}_{i,1}\,\mathtt{a}_{i,2} \ldots \mathtt{a}_{i,j-1}$ is generated by $\widehat{u}^2_{i,1}\,\widehat{u}^2_{i,2} \ldots \widehat{u}^2_{i,p-1}$ or a non-empty suffix q' of $\mathtt{a}_{i,j+1}\,\mathtt{a}_{i,j+2}\dots\mathtt{a}_{i,k}$ is generated by $\widehat{u}^2_{i,p+1}\,\widehat{u}^2_{i,p+2}\dots\widehat{u}^2_{i,t_i}$, which is a contradiction, since every variable in the strings $\widehat{u}^2_{i,1}\,\widehat{u}^2_{i,2}\dots\widehat{u}^2_{i,p-1}$ and $\widehat{u}^2_{i,p+1}\,\widehat{u}^2_{i,p+2}\dots\widehat{u}^2_{i,t_i}$ is repeated, whereas every symbol in q and q' occurs only once. Thus, for some p, $1 \leq p \leq t_i$, $h((\widehat{u}_{i,p})^2) = \widetilde{w}_i$, which implies that

$$h(\widehat{u}_{i,p}) = \mathtt{a}_{\{i,1\}}\,\mathtt{a}_{\{i,2\}}\dots\mathtt{a}_{\{i,i-1\}}\,\mathtt{a}_{\{i,i+1\}}\,\mathtt{a}_{\{i,i+2\}}\dots\mathtt{a}_{\{i,k\}}\,.$$

We recall that

$$\widehat{u}_{i,p} = \widehat{u}_{i,p,1}\,\widehat{u}_{i,p,2}\dots\widehat{u}_{i,p,i-1}\,\widehat{u}_{i,p,i+1}\,\widehat{u}_{i,p,i+2}\dots\widehat{u}_{i,p,k}\,,$$

where, for every j, $1 \leq j \leq k$, $i \neq j$, $\widehat{u}_{i,p,j}$ contains exactly these variables that correspond to edges between vertex $v_{i,p}$ and some vertex in V_j. Now if, for some j, $1 \leq j \leq k$, $i \neq j$, $|h(\widehat{u}_{i,p,j})| \geq 2$, then there are two edges $e_1, e_2 \in C_E$ that both connect $v_{i,p}$ with some vertex in V_j, which is a contradiction to the fact that h has property $(*)$. Thus, for every j, $1 \leq j \leq k$, $i \neq j$, $h(\widehat{u}_{i,p,j}) = \mathtt{a}_{\{i,j\}}$. Consequently, for every i, $1 \leq i, j \leq k$, all the $k-1$ edges in C_E, which connect V_i with each of the sets V_j, $1 \leq j \leq k$, $i \neq j$, are adjacent to the same vertex $v_{i,p}$ in V_i. This directly implies that C_V has cardinality of k, which means that C_V is a clique of size k of $\mathcal{G}$.

In order to prove the *only if* direction, we assume that there exists a clique of size k in $\mathcal{G}$. We define a substitution h that maps x_e to $\mathtt{a}_{\{i,j\}}$ if and only if e connects V_i and V_j and its adjacent vertices are members of the clique. It can be easily verified that h has property $(*)$ and, for every i, j, $1 \leq i \leq j \leq k$, $i \neq j$, $h(\overline{u}_{i,j}) = \overline{w}_{i,j}$. Moreover, since, for every i, $1 \leq i \leq k$, there is a p, $1 \leq p \leq t_i$, such that $v_{i,p}$, the clique member from V_i, is connected to exactly one clique member from every V_j, $1 \leq j \leq k$, $i \neq j$, we can conclude that

$$h(\widehat{u}_{i,p}) = \mathtt{a}_{\{i,1\}}\,\mathtt{a}_{\{i,2\}}\dots\mathtt{a}_{\{i,i-1\}}\,\mathtt{a}_{\{i,i+1\}}\,\mathtt{a}_{\{i,i+2\}}\dots\mathtt{a}_{\{i,k\}}\,.$$

Furthermore, all the other vertices $v_{i,p'}$, $1 \leq p \leq t_i$, $p \neq p'$, of V_i, are no clique members, which implies that $h(\widehat{u}_{i,p'}) = \varepsilon$. Hence, $h(\widetilde{u}_i) = \widetilde{w}_i$, for every i, $1 \leq i \leq k$, and therefore, $h(\widetilde{u}) = \widetilde{w}$ and $h(u) = w$. This concludes the proof. $\square$

It follows from Lemma 3 that Φ is and fpt-reduction with respect to the parameter $|h|$. Next, we note that Φ is also an fpt-reduction with respect to the parameters $|w|$, $|\Sigma|$ and $|u|_{\text{var}}$.

Proposition 2 *Let $\mathcal{G}$ be a graph, let $V_1, V_2, \dots, V_k$ be a partition of V and let $(u, w) := \Phi(\mathcal{G}, V_1, V_2, \dots, V_k)$. Then $|w|$ and $|\Sigma|$ are bounded by a function of k and $|u|_{\text{var}} = 3$.*

Proof For every i, j, $1 \leq i \leq j \leq k$, $i \neq j$, $|\overline{w}_{i,j}| = 3$; thus, $|\overline{w}| = \mathrm{O}(k^2)$. Furthermore, for every i, $1 \leq i \leq k$, $|\widetilde{w}_i| = 2(k-1)$; thus, $|\widetilde{w}| = \mathrm{O}(k^2)$. Consequently, $|w| = \mathrm{O}(k^2)$. Every variable $x \in \mathrm{var}(u)$ occurs once in $\overline{u}$ and twice in $\widetilde{u}$. Thus, $|u|_{\text{var}} = 3$. Finally, $|\Sigma| = \mathrm{O}(k^2)$. $\square$

However, the reduction Φ only works for the problems Z-StrSubst, $Z \in \{\textsc{Inj}, \varepsilon\}$, but it can be extended to the problems Z-StrMorph, $Z \in \{\textsc{Inj}, \varepsilon\}$, as well, i. e., to a mapping Φ' that maps a k-Multicoloured-Clique instance to a source string $u' \in X^+$ and a target string $w' \in \Sigma^+$. To this end let $\mathcal{G} := (V, E)$ be a graph and let $V_1, V_2, \ldots, V_k$ be a partition of V, such that every V_i is an independent set. Since Φ' is very similar to Φ, we shall only point out in which regards they differ. The main difference is that for Φ', instead of using occurrences of the symbol \$ in u, we use an occurrence of a new variable per each occurrence of \$. Furthermore, in order to maintain the E-injectivity, each of these new variables has to match its own individual symbol in w. More formally, for every i, j, $1 \leq i < j \leq k$, we define

$$\overline{u}_{i,j} := z_{¢_{i,j}}\, x_{e_{i,j,1}}\, x_{e_{i,j,2}} \ldots x_{e_{i,j,l_{i,j}}}\, z_{¢_{i,j}}\,,$$
$$\overline{w}_{i,j} := ¢_{i,j}\, \mathtt{a}_{\{i,j\}}\, ¢_{i,j}$$

and

$$\widetilde{u} := z_{\$_1}\, \widetilde{u}_1\, z_{\$_2}\, \widetilde{u}_2\, z_{\$_3} \ldots z_{\$_k}\, \widetilde{u}_k\, z_{\$_{k+1}}\,,$$
$$\widetilde{w} := \$_1\, \widetilde{w}_1\, \$_2\, \widetilde{w}_2\, \$_3 \ldots \$_k\, \widetilde{w}_k\, \$_{k+1}\,,$$

where the factors $\widetilde{u}_i$ and $\widetilde{w}_i$, $1 \leq i \leq k$, are defined as in the definition of Φ. Furthermore, analogously to the definition of Φ, we define $\overline{u}$ and $\overline{w}$ to be the concatenations of the factors $\overline{u}_{i,j}$ and $\overline{w}_{i,j}$, respectively. Finally, we define

$$u' := z_{\%}\, z_{\widehat{\%}}\, z_{\%}\, s^2\, z_{\widehat{\%}}\, \overline{u}\, z_{\widehat{\%}}\, \widetilde{u}\,,$$
$$w' := \%\, \widehat{\%}\, \%\, r^2\, \widehat{\%}\, \overline{w}\, \widehat{\%}\, \widetilde{w}\,,$$

and $\Phi'(\mathcal{G}, V_1, V_2, \ldots, V_k) := (u', w')$, where s is a concatenation of all the new variables of form $z_{¢_{i,j}}$ and $z_{\$_i}$ and r is the corresponding concatenation of the symbols $¢_{i,j}$ and $\$_i$.

For the reduction Φ', we can prove analogous results to Lemma 3 and Proposition 2, which concludes the proof of Theorem 2:

Lemma 4 *Let $\mathcal{G} := (V, E)$ be a graph, let $V_1, V_2, \ldots, V_k$ be a partition of V, such that every V_i is an independent set and let $(u', w') := \Phi'(\mathcal{G}, V_1, V_2, \ldots, V_k)$. There exists a clique of size k in $\mathcal{G}$ if and only if there exists an* E*-injective morphism h of size* 1 *with $h(u') = w'$ if and only if there exists a morphism h' of size* 1 *with $h'(u') = w'$.*

Proof Let $(u, w) := \Phi(\mathcal{G}, V_1, V_2, \ldots, V_k)$. If there exists a clique of size k in $\mathcal{G}$, then, by Lemma 3, there exists a E-injective substitution g of size 1 with $g(u) = w$. We can now obtain an E-injective substitution h of size 1 with $h(u') = w'$ by simply mapping each variable $z_{\$_i}$, $z_{¢_{i,j}}$, $z_{\%}$ and $z_{\widehat{\%}}$ to $\$_i$, $¢_{i,j}$, $\%$ and $\widehat{\%}$, respectively. By definition, this means that there exists a morphism h' of size 1 with $h'(u') = w'$.

Next, we assume that there exists a morphism h of size 1 for u' such that $h(u') = w'$. If $h(z_{\%}) = \%$, then $h(s^2) = r^2$, $h(\overline{u}) = \overline{w}$ and $h(\widetilde{u}) = \widetilde{w}$.

Furthermore, since $|s^2| = |r^2|$, if a variable of s^2 is mapped to the empty word, then this means that another variable of s^2 must be mapped to a word of length at least 2, which contradicts to the assumption $|h| = 1$; thus, every variable $z_{¢_{i,j}}$ and $z_{\$_i}$ is mapped to $¢_{i,j}$ and $\$_i$, which implies that h can be interpreted as a substitution of size 1 for u such that $h(u) = w$. By Lemma 3, this directly implies that there exists a clique of size k in $\mathcal{G}$. Furthermore, in the proof of Lemma 3, it has been shown that a substitution of size 1 that maps u to w is necessarily E-injective, which means that the morphism h is E-injective as well. Hence, in order to conclude the proof, it only remains to show that every morphism h of size 1 for u' with $h(u') = w'$ necessarily satisfies $h(z_{\%}) = \%$. To this end, we assume that $h(z_{\%}) \neq \%$. Thus, $h(z_{\%}) = \varepsilon$, which implies that there is some other variable $y \in \mathrm{var}(u')$, such that $h(y) = \%$. If $y \in \mathrm{var}(u)$, then, by Proposition 2, y has 3 occurrences and if $y \notin \mathrm{var}(u)$, then either y equals $z_{\widehat{\%}}$ and therefore it has 3 occurrences or y has 2 occurrences in s^2 (which explains why we use the square of s and r in the strings u' and w') and at least 1 occurrence in $\overline{u}\widetilde{u}$ and therefore it has at least 3 occurrences, too. Since there are only two occurrences of % in w', this is a contradiction. □

In order to conclude the proof of Theorem 2, it only remains to observe that the reduction Φ' is still a parameterised reduction with respect to the parameters $|w'|$, $|\Sigma|$ and $|u'|_{\mathrm{var}}$.

Proposition 3 *Let $\mathcal{G}$ be a graph, let $V_1, V_2, \ldots, V_k$ be a partition of V and let $(u', w') := \Phi'(\mathcal{G}, V_1, V_2, \ldots, V_k)$. Then $|w'|$ and $|\Sigma|$ are bounded by a function of k and $|u'|_{\mathrm{var}} = 4$.*

Proof In order to prove the statement of the proposition, let u and w be the source and target string produced by Φ. Every variable in $\mathrm{var}(u') \setminus \mathrm{var}(u)$ is a variable $z_{\$_i}$, $z_{¢_{i,j}}$, $z_{\%}$ or $z_{\widehat{\%}}$; the variables $z_{\%}$ and $z_{\widehat{\%}}$ have 2 and 3 occurrences, respectively, the variables $z_{\$_i}$ have 2 occurrences in s^2 and 1 occurrence in $\widetilde{u}$ and the variables $z_{¢_{i,j}}$ have 2 occurrences in s^2 and 2 occurrence in $\overline{u}$. Furthermore, every variable in $\mathrm{var}(u') \cap \mathrm{var}(u)$ has as many occurrences in u' as in u. Thus, by Proposition 2, we can conclude that $|u'|_{\mathrm{var}} = 4$. Obviously, $|w'| \leq 2|w|$, thus, since $|w|$ is bounded by a function of k, $|w'|$ is bounded by a function of k, too. Let A and B be the sets of terminal symbols that occur in w' and w, respectively. By definition, $B \subseteq A$. Furthermore, for every occurrence of \$ in w, we use a new individual terminal symbol, which implies $|A \setminus B| \leq |w|$. Again, by Proposition 2, we can conclude that $|A|$ is bounded by a function of k. □

3.2 Fixed Parameter Tractability

In the previous section, the $W[1]$-hardness of a large number of string morphism problems is shown. In this section, we prove that the remaining variants are fixed parameter tractable. To this end, we now present two brute-force algorithms for the string morphism problems.

The algorithm BF-$1_{\textsc{StrMorph}}$, presented as Algorithm 1, solves the problem StrMorph by enumerating all possible substitutions for the source string. It is straightforward to generalise BF-$1_{\textsc{StrMorph}}$ to algorithms BF-1_K, which solve the problems $K \in$ SMP. We only have to make sure that, depending on the problem K, we only enumerate m-tuples of factors of w that induce an injective, a nonerasing or an injective nonerasing morphism (or substitution).

```
Input  : A source string $u$ and a target string $w$
Output: YES iff $(u, w) \in$ StrMorph
for every tuple $(w_1, w_2, \ldots, w_{|\text{var}(u)|})$ of factors of $w$ do
    for $i = 1$ to $|\text{var}(u)|$ do
        $h(x_i) := w_i$;
    if $h(u) = w$ then return YES;
return NO;
```

Algorithm 1: BF-$1_{\textsc{StrMorph}}$

Proposition 4 *Let $K \in$ SMP. The running time of* BF-$1_K(u, w)$ *is* $O(|u| \times |w| \times (|w|^2)^{|\text{var}(u)|})$.

Proof Since there exist $O(|w|^2)$ different factors of w, the main loop of the algorithm is executed $O((|w|^2)^{|\text{var}(u)|})$ times. In every iteration of the loop, we have to construct h and check whether $h(u) = w$, which can be done in time $O(|h(u)|) = O(|u| \times |w|)$. Thus, the total running time of the algorithm BF-$1_K(u, w)$ is $O(|u| \times |w| \times (|w|^2)^{|\text{var}(u)|})$. □

By slightly changing Algorithm 1, we can define the algorithm BF-$2_{\textsc{StrMorph}}$, presented as Algorithm 2. In a similar way as done for Algorithm 1, for every $K \in$ SMP, we can extend BF-$2_{\textsc{StrMorph}}$ to BF-2_K, which solves K.

```
Input  : Source string $u$, target string $w$ over an alphabet $\Sigma$ and $k \in \mathbb{N}$
Output: YES iff there exists a morphism $h$ with $|h| \leq k$ and $h(u) = w$
for every tuple $(w_1, \ldots, w_{|\text{var}(u)|}) \in (\Sigma^*)^{|\text{var}(u)|}$, $|w_i| \leq k$, $1 \leq i \leq |\text{var}(u)|$ do
    for $i = 1$ to $|\text{var}(u)|$ do
        $h(x_i) := w_i$;
    if $h(u) = w$ then return YES;
return NO;
```

Algorithm 2: BF-$2_{\textsc{StrMorph}}$

Proposition 5 *Let $K \in$ SMP. The running time of* BF-$2_K(u, w, \Sigma, k)$ *is* $O(|u| \times k \times (k \times |\Sigma|^k)^{|\text{var}(u)|})$.

Proof The main loop is executed $O((\sum_{i=0}^{k} |\Sigma|^i)^{|\text{var}(u)|}) = O((k \times |\Sigma|^k)^{|\text{var}(u)|})$ times. In every iteration of the loop, we have to construct h and check whether $h(u) = w$, which can be done in time $O(|h(u)|) = O(|u| \times k)$. Thus, the total running time of the algorithm is $O(|u| \times k \times (k \times |\Sigma|^k)^{|\text{var}(u)|})$. □

By applying the brute-force algorithms from above for solving the string morphism problems, we can conclude the following fixed parameter tractability results:

Theorem 3

1. *For every* $K \in \text{SMP}$*,* $[|\text{var}(u)|, |w|]$*-*K *is in FPT.*
2. *For every* $Z \in \{\text{NE}, \text{NE-INJ}\}$ *and* $K \in \{\text{STRMORPH}, \text{STRSUBST}\}$*,* $[|w|]$*-*Z*-*K *is in FPT.*
3. *For every* $K \in \{\text{STRMORPH}, \text{STRSUBST}\}$*,* $[|h|, |\Sigma|]$*-NE-INJ-*K *is in FPT.*
4. *For every* $K \in \text{SMP}$*,* $[|\text{var}(u)|, |h|]$*-*K *is in FPT.*

Proof Obviously, BF-1_K is an fpt-algorithm for $[|\text{var}(u)|, |w|]$-K, $K \in \text{SMP}$, which proves A. For the NE variants of the string morphism problems, we can assume $|w| \geq |u| \geq |\text{var}(u)|$; thus, for every $Z \in \{\text{NE}, \text{NE-INJ}\}$ and $K \in \{\text{STRMORPH}, \text{STRSUBST}\}$, BF-$1_{Z\text{-}K}(u, w)$ has a running time of $\text{O}(|u| \times |w| \times (|w|^2)^{|w|}))$, which proves B.

We shall now prove part C, but only show the case $K = \text{STRMORPH}$, since the case $K = \text{STRSUBST}$ can be handled analogously. Let u be the source string, let $w \in \Sigma^*$ be the target string and let $k \in \mathbb{N}$ be the parameter $|h|$. We check whether there exists an injective nonerasing morphism h of size at most k with $h(u) = w$ in the following way. First, we note that there are $\sum_{i=1}^{k} |\Sigma|^i \leq k \times |\Sigma|^k$ different non-empty strings over Σ with a length of at most k. Hence, if $|\text{var}(u)| > k \times |\Sigma|^k$, which can be checked in time $\text{O}(k \times |\Sigma|^k)$, then every morphism h of size at most k with $h(u) = w$ is necessarily non-injective. If, on the other hand $|\text{var}(u)| \leq k \times |\Sigma|^k$, then we can use the algorithm BF-$2_{\text{NE-INJ-STRMORPH}}(u, w, \Sigma, k)$ in order to check whether there exists an injective nonerasing morphism h of size at most k with $h(u) = w$ and $|h(x)| \leq k$, $x \in \text{var}(u)$, in time

$$\text{O}\left(|u| \times k \times \left(k \times |\Sigma|^k\right)^{\left(k \times |\Sigma|^k\right)}\right) .$$

For part D, we again only show the case $K = \text{STRSUBST}$, since all the other cases $K \in \text{SMP} \setminus \{\text{STRSUBST}\}$ can be handled analogously. Let u be the source string, let $w \in \Sigma^*$ be the target string and let $k \in \mathbb{N}$ be the parameter $|h|$. Let Γ be exactly the symbols from Σ which have more occurrences in w than in u. Obviously, these are the only symbols that need to occur in the images of the substitution. Now if $|\Gamma| > k \times |\text{var}(u)|$, then there does not exist a substitution h of size at most k that satisfies $h(u) = w$, since every variable in $\text{var}(u)$ can generate at most k new symbols. If, on the other hand, $|\Gamma| \leq k \times |\text{var}(u)|$, then we can use the algorithm BF-$2_{\text{STRSUBST}}(u, w, \Gamma, k)$, which has a running time of

$$\text{O}\left(|u| \times k \times \left(k \times (k \times |\text{var}(u)|)^k\right)^{|\text{var}(u)|}\right) .$$

□

Problems	$\lvert\text{var}(u)\rvert$	$\lvert w\rvert$	$\lvert u\rvert_{\text{var}}$	$\lvert h\rvert$	$\lvert\Sigma\rvert$	Compl.	Ref.
SMP	p	p	–	–	–	FPT	Thm. 3.1
{Ne, Ne - Inj}-{StrMorph, StrSubst}	–	p	–	–	–	FPT	Thm. 3.2
Ne - Inj -{StrMorph, StrSubst}	–	–	–	p	p	FPT	Thm. 3.3
SMP	p	–	–	p	–	FPT	Thm. 3.4
SMP	p	–	p	–	6	$W[1]$-h.	Thm. 1
$\{\varepsilon$, Inj}-{StrMorph, StrSubst}	–	p	3	1	p	$W[1]$-h.	Thm. 2

Table 1 A summary of parameterised complexity results.

We conclude this section by pointing out that the results presented in Section 3.1 and 3.2 completely settle the fixed parameter tractability of all possible parameterised variants of string morphism problems, with respect to the parameters considered in the context of this work. In order to verify this claim, we recall these results in Table 1. In this table, an entry p means that the problems denoted in the row are parameterised by the parameter in the column and an integer entry constitutes a constant bound for this parameter. We note that all the cases parameterised by both $|\text{var}(u)|$ and $|w|$ are settled by row 1. Furthermore, all the cases parameterised by $|w|$, but not by $|\text{var}(u)|$ are settled by rows 2 and 6, and all the cases parameterised by $|\text{var}(u)|$, but not by $|w|$ are settled by rows 4 and 5. In order to see that all the cases parameterised neither by $|\text{var}(u)|$ nor by $|w|$ are settled as well, we need to take a closer look.

From row 5, we can only conclude that as long as $|h|$ is not a parameter, then all variants are $W[1]$-hard. However, for the cases where $|h|$ is a parameter, we can only conclude from row 6 the $W[1]$-hardness for all but the Ne and Ne - Inj variants, and, in addition to that, from row 3 we can conclude the FPT-membership for the Ne - Inj variant, where $|\Sigma|$ is a parameter, too. Consequently, for every $K \in \{\text{StrMorph}, \text{StrSubst}\}$ and $Z \in \{\text{Ne}, \text{Ne - Inj}\}$, the following cases are open: (1) $[|h|, |u|_{\text{var}}, |\Sigma|]$-Ne-$K$, (2) $[|h|, |\Sigma|]$-Ne-K, (3) $[|h|, |u|_{\text{var}}]$-Z-K and (4) $[|h|]$-Z-K. In [14] it has been shown that the problems Ne-K are $\mathcal{NP}$-complete even if the parameters $|h|$, $|u|_{\text{var}}$ and $|\Sigma|$ are bounded by constants, which implies that, unless $\mathcal{P} = \mathcal{NP}$, the problems of cases (1) and (2) are not in XP; thus, they are not in FPT. The same holds for the problems Z-K with respect to parameters $|h|$ and $|u|_{\text{var}}$, which, in a similar way, implies that the problems of cases (3) and (4) are not in FPT.

3.3 $W[1]$-Membership and $W[P]$-Membership

In this section, we investigate the $W[1]$-membership and $W[P]$-membership for those variants of the string morphism problem that are shown to be $W[1]$-hard in Section 3.1. So, we try to find their proper place within the W-hierarchy of parameterised complexity. More precisely, we first show that, for every $K \in$ SMP, the problems $[|\text{var}(u)|, |u|_{\text{var}}]$-$K$ and $[|w|]$-K are in $W[1]$. Unfortunately, for the problems $[|\text{var}(u)|]$-K, $K \in$ SMP, we are only able to show a weaker result, i. e., their $W[P]$-membership. Our $W[1]$-membership results shall be proven by reductions to the following $W[1]$-complete problem, hence employing the "Turing way" to the W-hierarchy as proved by Cesati [7].

Short-NTM-Comp
Instance: A nondeterministic Turing machine M and a string w over the input alphabet of M and a $k \in \mathbb{N}$.
Parameter: $\kappa(M, w, k) = k$.
Question: Does M have an accepting computation for w with at most k steps?

We wish to point out that the problem variant that fixes the input w to the empty string characterises $W[1]$, too, and, for the sake of convenience, we shall employ this problem to prove our $W[1]$-membership results.

On the other hand, the result that all problems $[|\text{var}(u)|]$-K are in $W[P]$ is obtained by using the following characterisation of the class $W[P]$:

Proposition 6 (Flum and Grohe [16]) *A parameterised problem (Q, κ) with $Q \in \mathcal{NP}$ is in $W[P]$ if and only if there is a computable function $h : \mathbb{N} \to \mathbb{N}$, a polynomial p and a nondeterministic Turing machine M deciding Q such that on every run with input x the machine M performs at most $p(|x|)$ steps, at most $h(\kappa(x)) \times \log(|x|)$ of them being nondeterministic.*

We are now ready to present our results and we start with the problem $[|\text{var}(u)|, |u|_{\text{var}}]$-StrSubst:

Theorem 4 *Let $K \in$ SMP. The problem $[|\text{var}(u)|, |u|_{\text{var}}]$-$K$ is in $W[1]$.*

Proof In order to prove the statement of the theorem, we define a parameterised reduction from $[|\text{var}(u)|, |u|_{\text{var}}]$-StrSubst to Short-NTM-Comp. We shall then show how this reduction can be extended to reductions from all the problems $[|\text{var}(u)|, |u|_{\text{var}}]$-$K$, $K \in$ SMP, to Short-NTM-Comp.

Let $u := u_0\, y_1\, u_1\, y_2\, u_2\, y_3 \dots y_n\, u_n$, $y_i \in X$, $1 \leq i \leq n$, $u_j \in \Sigma^*$, $0 \leq j \leq n$, be a source string and let $w \in \Sigma^*$ be a target string. For every i, j, k, l, $1 \leq i \leq j < k \leq l \leq |w|$, we define $P_{i,j,k,l} := 1$ if $w[i, j] = w[k, l]$ and $P_{i,j,k,l} := 0$ if $w[i, j] \neq w[k, l]$. Furthermore, for every i, j, k, l, $1 \leq i < j \leq n$, $1 \leq k \leq l \leq |w|$, we define $Q_{i,j,k,l} := 1$ if $u_i\, u_{i+1} \dots u_{j-1} = w[k+1, l-1]$ and $Q_{i,j,k,l} := 0$ if $u_i\, u_{i+1} \dots u_{j-1} \neq w[k+1, l-1]$. Next, we define a nondeterministic Turing machine $M_{u,w}$, which accepts the empty string if and only if $h(u) = w$ for some substitution h. The tape alphabet of $M_{u,w}$ is $\tau := X \cup \tau'$, where $\tau' := \{T^{(i)}_{j,k} \mid 1 \leq i \leq n, 1 \leq j \leq k \leq |w|\} \cup \{T_\varepsilon\}$. The states and transition function of $M_{u,w}$ is implicitly given by the following description of how $M_{u,w}$ works:

1. $M_{u,w}$ writes $y_1\, y_2 \dots y_n$ on the working tape (in any of the following instructions, by u' we denote the current content of the working tape).
2. $M_{u,w}$ guesses $S \subseteq \text{var}(u')$ and, for every i, $1 \leq i \leq n$,
   (a) if $y_i \in S$, then y_i is replaced by T_ε,
   (b) if $y_i \notin S$, then y_i is replaced by $T^{(i)}_{j,k}$, for some j, k, $1 \leq j \leq k \leq |w|$.

3. $M_{u,w}$ checks, for every factor $T^{(i_1)}_{j_1,k_1}\, T^l_\varepsilon\, T^{(i_2)}_{j_2,k_2}$, $0 \leq l \leq n-2$, of u', whether $Q_{i_1,i_2,k_1,j_2} = 1$ is satisfied.
4. $M_{u,w}$ checks, for every i_1, i_2, $1 \leq i_1 < i_2 \leq n$, with $u'[i_1] = T^{(i_1)}_{j_1,k_1}$, $u'[i_2] = T^{(i_2)}_{j_2,k_2}$ and $y_{i_1} = y_{i_2}$, whether $P_{j_1,k_1,j_2,k_2} = 1$ is satisfied.
5. If conditions 3 and 4 are satisfied, then $M_{u,w}$ accepts and rejects otherwise.

The following three claims, which can be verified with moderate effort, show that the transformation of a source string u and a target string w into the nondeterministic Turing machine $M_{u,w}$ constitutes a parameterised reduction from $[|\mathrm{var}(u)|, |u|_{\mathrm{var}}]$-StrSubst to Short-NTM-Comp.

Claim 1. Let u be a source string and let $w \in \Sigma^*$ be a target string. There exists a substitution h with $h(u) = w$ if and only if $M_{u,w}$ accepts the empty string.

Claim 2. Let u be a source string and let $w \in \Sigma^*$ be a target string. The Turing machine $M_{u,w}$ can be constructed in time $\mathrm{O}(g(|u|, |w|))$, for a polynomial g.

Claim 3. Let u be a source string and let $w \in \Sigma^*$ be a target string. Every computation of the Turing machine $M_{u,w}$ halts after $\mathrm{O}(g(|u|_{\mathrm{var}}, |\mathrm{var}(u)|))$ steps, where g is a polynomial.

It is straightforward to modify the construction of $M_{u,w}$ in such a way that a Turing machine $M_{u,w,K}$, $K \in \mathrm{SMP} \setminus \{\text{StrSubst}\}$, is constructed, which accepts the empty string if and only if (u, w) is a positive instance of problem $[|\mathrm{var}(u)|, |u|_{\mathrm{var}}]$-$K$. More precisely, in order to define $M_{u,w,K}$ for the nonerasing versions of string morphism problems, we have to make sure that in step 2 we set $S := \emptyset$ and in order to define $M_{u,w,K}$ for the injective versions of string morphism problems, we need an additional step similar to step 4 in which we check, for every i_1, i_2, $1 \leq i_1 < i_2 \leq n$, with $u'[i_1] = T^{(i_1)}_{j_1,k_1}$, $u'[i_2] = T^{(i_2)}_{j_2,k_2}$ and $y_{i_1} \neq y_{i_2}$, whether $P_{j_1,k_1,j_2,k_2} = 0$ is satisfied. This concludes the proof of Theorem 4. □

Next, we consider the case where instead of $|\mathrm{var}(u)|$, the length of the target string, $|w|$, is a parameter.

Theorem 5 *Let $K \in \mathrm{SMP}$. The problem $[|w|]$-K is in $W[1]$.*

Proof We proceed analogously to the proof of Theorem 4, i.e., we define a parameterised reduction from $[|w|]$-StrSubst to Short-NTM-Comp and we shall later explain how this reduction can be extended to all the other problems $K \in \mathrm{SMP}$.

Let u be a source string and let $w \in \Sigma^*$ be a target string. We define a nondeterministic Turing machine $N_{u,w}$ in the following way. The tape alphabet of $N_{u,w}$ is $\tau := \Sigma \cup \{1, 2, \ldots, |u|\}$. The states and transition function of $N_{u,w}$ is implicitly given by the following description of how $N_{u,w}$ works:

1. $N_{u,w}$ nondeterministically initialises a variable $c := i$, $1 \leq i \leq |u|$, and writes w on the working tape (in any of the following instructions, by w' we denote the current content of the working tape).

2. $N_{u,w}$ scans over w' from left to right and every single occurrence of a terminal symbol $b \in \Sigma$ that is encountered is replaced by c and then c is nondeterministically set to a value j, $c \leq j \leq |u|$.
3. $N_{u,w}$ checks, for every i, $1 \leq i \leq |u|$, with $u[i] \in \Sigma$, whether there is exactly one j, $1 \leq j \leq |w'|$, with $w'[j] = i$ and, furthermore, $w[j] = u[i]$.
4. $N_{u,w}$ checks, for every i_1, i_2, $1 \leq i_1 < i_2 \leq |u|$, and for every j_1, k_1, j_2, k_2, $1 \leq j_1 \leq k_1 < j_2 \leq k_2 \leq |w'|$, with $w'[j_1, k_1] = i_1^{(k_1 - j_1 + 1)}$, $w'[j_1 - 1] < i_1 < w'[k_1 + 1]$, $w'[j_2, k_2] = i_2^{(k_2 - j_2 + 1)}$ and $w'[j_2 - 1] < i_2 < w'[k_2 + 1]$, whether or not $u[i_1] = u[i_2]$ implies $w[j_1, k_1] = w[j_2, k_2]$.
5. If conditions 3 and 4 are satisfied, then $N_{u,w}$ accepts and rejects otherwise.

The following claims show that the construction of $N_{u,w}$ described above is a parameterised reduction from $[|w|]$-StrSubst to Short-NTM-Comp.

Claim 1. Let u be a source string and let $w \in \Sigma^*$ be a target string. There exists a substitution h with $h(u) = w$ if and only if $N_{u,w}$ accepts the empty string.

Claim 2. Let u be a source string and let $w \in \Sigma^*$ be a target string. The Turing machine $N_{u,w}$ can be constructed in time $\mathrm{O}(g(|u|, |w|))$, for a polynomial g.

Claim 3. Let u be a source string and let $w \in \Sigma^*$ be a target string. Every computation of the Turing machine $N_{u,w}$ halts after $\mathrm{O}(g(|w|))$ steps, where g is a polynomial.

We can again note that it is straightforward to extend the construction of $N_{u,w}$ from above in such a way that a Turing machine $N_{u,w,K}$, $K \in \mathrm{SMP} \setminus \{\text{StrSubst}\}$, is constructed, which accepts the empty string if and only if (u, w) is a positive instance of problem K. More precisely, in order to define $N_{u,w,K}$ for the injective versions of string morphism problems, we need to check in step 4, whether or not the substitution induced by the replacements done in step 2 satisfies the injectivity or E-injectivity condition, and in order to define $N_{u,w,K}$ for the nonerasing versions of string morphism problems, we only have to make sure that in step 2, for every c, at least 1 terminal symbol is replaced by c. This concludes the proof of Theorem 5. □

We have now established the $W[1]$-membership of all $[|\mathrm{var}(u)|, |u|_{\mathrm{var}}]$-$K$ and $[|w|]$-K, $K \in \mathrm{SMP}$. Furthermore, since $W[1]$-membership is preserved if we add more parameters, Theorems 4 and 5 imply $W[1]$-completeness for all the $W[1]$-hard versions of string morphism problems, except $[|\mathrm{var}(u)|, |\Sigma|]$-$K$ and $[|\mathrm{var}(u)|]$-K, $K \in \mathrm{SMP}$, for which $W[1]$-membership does not follow, since in the reduction defined above, we need $|u|_{\mathrm{var}}$ as a parameter, too. However, for the problems $[|\mathrm{var}(u)|]$-K, $K \in \mathrm{SMP}$, we can show a weaker result, i. e., their $W[P]$-membership (which then also carries over to the problems $[|\mathrm{var}(u)|, |\Sigma|]$-$K$, $K \in \mathrm{SMP}$).

Theorem 6 *For every* $K \in \mathrm{SMP}$*,* $[|\mathrm{var}(u)|]$-$K \in W[P]$.

Proof In order to prove the statement of the Theorem, we define a Turing machine $M_{\text{Ne-StrMorph}}$, which solves $[|\mathrm{var}(u)|]$-Ne-StrMorph and satisfies the conditions given in Proposition 6. Later on, we explain how this definition can be extended to Turing machines M_K for every $K \in \text{SMP}$.

Let $u = y_1 y_2 \dots y_m$, $y_i \in \text{var}(u)$, $1 \leq i \leq m$, be a source string and let $w \in \Sigma^*$ be a target string. The states and transition function of $M_{\text{NE-STRMORPH}}$ is implicitly given by the following description of how $M_{\text{NE-STRMORPH}}$ works. We assume that the input is given in the form $u \# w$:

1. $M_{\text{NE-STRMORPH}}$ nondeterministically guesses numbers $i_1, j_1, i_2, j_2, \dots, i_m, j_m \in \mathbb{N}$, such that $1 = i_1 \leq j_1 \leq i_2 \leq j_2 \leq \dots \leq i_m \leq j_m = |w|$ and, for every l, $1 \leq l \leq m-1$, $j_l + 1 = i_{l+1}$.
2. $M_{\text{NE-STRMORPH}}$ accepts if, for every p, q, $1 \leq p < q \leq |u|$, $y_p = y_q$ implies $w[i_p, j_p] = w[i_q, j_q]$, and rejects otherwise.

The following claims establish that the Turing machine $M_{\text{NE-STRMORPH}}$ solves the problem $[|\text{var}(u)|]$-NE-STRMORPH and satisfies the conditions given in Proposition 6.

Claim 1. The Turing machine $M_{\text{NE-STRMORPH}}$ solves NE-STRMORPH.

Claim 2. Let u be a source string and let $w \in \Sigma^*$ be a target string. The Turing machine $M_{\text{NE-STRMORPH}}$, on input $u\#w$, performs $O(|\text{var}(u)| \times \log(|w|))$ nondeterministic steps.

Claim 3. Let u be a source string and let $w \in \Sigma^*$ be a target string. Any computation of the Turing machine $M_{\text{NE-STRMORPH}}$ on input $u\#w$ is polynomial.

Proposition 6 and the claims from above imply that NE-STRMORPH is in $W[P]$. Next, we observe that the Turing machine $M_{\text{NE-STRMORPH}}$ can be extended to Turing machines M_K, $K \in \text{SMP} \setminus \{\text{NE-STRMORPH}\}$, which solves K in the following way. If the source string contains terminal symbols, then we have to make sure that the numbers guessed in step 1 cater for this situation, e. g., if there are 5 terminal symbols between y_p and y_{p+1}, then $j_p + 6 = i_{p+1}$ must hold. If we are concerned with erasing substitutions or morphisms, then, before performing step 1, the Turing machine first guesses a subset of variables which are erased from u. This can be done with $O(|\text{var}(u)|)$ nondeterministic steps. Moreover, the claims from above still hold for these extended versions of $M_{\text{NE-STRMORPH}}$. Thus, we can conclude the statement of the theorem. □

4 A Lower Bound

For most string problems, it is a natural assumption that the alphabet Σ is fixed (in fact, it often has very small cardinality as, e. g., 2 if we are dealing with binary numbers or 4 in the case of DNA sequences). Furthermore, if we use strings with variables (i. e., source strings) for specifying a class of similar string objects (which is a typical application of strings with variables), then, for many applications, there are only finitely many string objects that can replace the variables. Hence, the problems $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$, $K \in \text{SMP}$, $k_1, k_2 \in \mathbb{N}$ (i. e., the parameters $|h|$ and $|\Sigma|$ are bounded by k_1 and k_2, respectively), are of special interest. We recall that Proposition 5 demonstrates that, for every constants $k_1, k_2 \in \mathbb{N}$ and for every $K \in$ SMP, $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$ can be solved in time $\mathrm{O}(|u| \times k_1 \times (k_1 \times k_2^{k_1})^{|\mathrm{var}(u)|}) = |u| \times 2^{\mathrm{O}(|\mathrm{var}(u)|)}$. In this section, we show that if $k_2 \geq 2$, then, for every $K \in \{\textsc{StrMorph}, \textsc{StrSubst}\}$, it is very unlikely that a subexponential algorithm for $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$ exists.

One common way to argue for the unlikeness of a subexponential algorithm is to use the *Exponential Time Hypothesis* (ETH) by Impagliazzo, Paturi, and Zane. For an introduction to ETH, see [17, 25]. By the observation that each variable of a Boolean formula is used at least once, and by the Sparsification Lemma [23], ETH can be expanded as follows:

Exponential Time Hypothesis (ETH) [23]: There is a positive real s such that 3-SAT instances on n variables and m clauses cannot be solved in time $2^{sn}(n+m)^{O(1)}$. In particular: there is a real $s' > 0$ such that 3-SAT instances on m clauses cannot be solved in time $2^{s'(n+m)}(n+m)^{O(1)}$.

Obviously, if there exists some algorithm solving 3-SAT in time $2^{\mathrm{o}(n+m)}(n+m)^{O(1)}$, this would contradict ETH.

In the following, we show that if $k_2 \geq 2$, then, for every $K \in \{\textsc{StrMorph}, \textsc{StrSubst}\}$, there does not exist an algorithm that solves $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$ in time $(|u||w|)^{\mathrm{O}(1)} \times 2^{\mathrm{o}(|\mathrm{var}(u)|)}$, unless ETH fails.

To this end, we define a reduction Φ from 3SAT to $\textsc{StrSubst}$. Let $C := \{c_1, c_2, \dots, c_m\}$ be a set of three-literal-clauses with variables $v_1, v_2, \dots, v_n$ (the negation of a variable v_i is denoted by $\neg v_i$). We define

$$\begin{aligned} \overline{u} &:= \cent\, x_1\, \overline{x}_1\, \cent\, x_2\, \overline{x}_2\, \cent \dots \cent\, x_n\, \overline{x}_n\, \cent\,, \\ \overline{w} &:= \cent\, (\mathtt{a}\, \cent)^n\,, \end{aligned}$$

and, for every clause $c_i := \{p_{i_1}, p_{i_2}, p_{i_3}\}$, $1 \leq i \leq m$, we define

$$\begin{aligned} \widehat{u}_i &:= \cent\, y_{i_1}\, y_{i_2}\, y_{i_3}\, z_{i,1}\, z_{i,2}\, \cent\, z_{i,1}\, z_{i,2}\, z'_{i,1}\, z'_{i,2}\, \cent\,, \\ \widehat{w}_i &:= \cent\, \mathtt{a}\,\mathtt{a}\,\mathtt{a}\, \cent\, \mathtt{a}\,\mathtt{a}\, \cent\,, \end{aligned}$$

where $y_{i_j} := x_{i_j}$ if $p_{i_j} = v_{i_j}$ and $y_{i_j} := \overline{x}_{i_j}$ if $p_{i_j} = \neg v_{i_j}$, $1 \leq j \leq 3$. Finally, $u := \overline{u}\, \widehat{u}_1\, \widehat{u}_2 \dots \widehat{u}_m$, $w := \overline{w}\, \widehat{w}_1\, \widehat{w}_2 \dots \widehat{w}_m$ and $\Phi(C) := (u, w)$.

Lemma 5 *Let C be a 3CNF formula and let $(u, w) := \Phi(C)$. The formula C is satisfiable if and only if there exists a substitution h of size 1 with $h(u) = w$.*

Proof We first prove the *only if* direction and assume that there exists a satisfying assignment $\pi : \{v_1, v_2, \dots, v_n\} \to \{\mathtt{true}, \mathtt{false}\}$ for C. We define a substitution for u in the following way. For every i, $1 \leq i \leq n$, we define $h(x_i) := \mathtt{a}$ and $h(\overline{x}_i) := \varepsilon$, if $\pi(v_i) = \mathtt{true}$ and $h(x_i) := \varepsilon$ and $h(\overline{x}_i) := \mathtt{a}$, if $\pi(v_i) = \mathtt{false}$. This implies $h(\overline{u}) = \overline{w}$. Furthermore, since π is satisfying, for every i, $1 \leq i \leq m$, $h(y_{i_1}\, y_{i_2}\, y_{i_3}) \in \{\mathtt{a}, \mathtt{a}\,\mathtt{a}, \mathtt{a}\,\mathtt{a}\,\mathtt{a}\}$. Thus, we define

– $h(z_{i,1}) := \mathtt{a}$, $h(z_{i,2}) := \mathtt{a}$, $h(z'_{i,1}) := \varepsilon$ and $h(z'_{i,2}) := \varepsilon$, if $h(y_{i_1}\, y_{i_2}\, y_{i_3}) = \mathtt{a}$,

– $h(z_{i,1}) := \mathtt{a}$, $h(z_{i,2}) := \varepsilon$, $h(z'_{i,1}) := \mathtt{a}$, $h(z'_{i,2}) := \varepsilon$, if $h(y_{i_1}\, y_{i_2}\, y_{i_3}) = \mathtt{a}\,\mathtt{a}$,
– $h(z_{i,1}) := \varepsilon$, $h(z_{i,2}) := \varepsilon$, $h(z'_{i,1}) := \mathtt{a}$, $h(z'_{i,2}) := \mathtt{a}$, if $h(y_{i_1}\, y_{i_2}\, y_{i_3}) = \mathtt{a}\,\mathtt{a}\,\mathtt{a}$.

This directly implies that, for every i, $1 \leq i \leq m$, $h(\widehat{u}_i) = \widehat{w}_i$. Hence, $h(u) = w$ and h is of size 1.

In order to prove the *if* direction, we assume that there exists a substitution h of size 1 with $h(u) = w$, which implies $h(\overline{u}) = \overline{w}$ and, for every i, $1 \leq i \leq m$, $h(\widehat{u}_i) = \widehat{w}_i$. From $h(\overline{u}) = \overline{w}$ we can directly conclude that, for every i, $1 \leq i \leq n$, $h(x_i) = \mathtt{a}$ and $h(\overline{x}_i) = \varepsilon$ or $h(x_i) = \varepsilon$ and $h(\overline{x}_i) = \mathtt{a}$. Furthermore, $h(\widehat{u}_i) = \widehat{w}_i$, $1 \leq i \leq m$, implies that $h(y_{i_1}\, y_{i_2}\, y_{i_3}\, z_{i,1}\, z_{i,2}) = \mathtt{a}\,\mathtt{a}\,\mathtt{a}$ and $h(z_{i,1}\, z_{i,2}\, z'_{i,1}\, z'_{i,2}) = \mathtt{a}\,\mathtt{a}$. This particularly implies that, for every i, $1 \leq i \leq m$, $h(y_{i_1}\, y_{i_2}\, y_{i_3}) \neq \varepsilon$, since otherwise $h(z_{i,1}\, z_{i,2}) = \mathtt{a}\,\mathtt{a}\,\mathtt{a}$, which is a contradiction to $h(z_{i,1}\, z_{i,2}\, z'_{i,1}\, z'_{i,2}) = \mathtt{a}\,\mathtt{a}$. Consequently, if we assign every v_i with $h(x_i) = \mathtt{a}$ to `true`, then at least one variable in every clause is assigned to `true`, which means that C is satisfied. □

We note that $\Phi(C)$ produces a source string u with $|u| = 3n+1+12m$ and a target string $w \in \{\mathtt{a}, ¢\}^*$ with $|w| = 2n + 1 + 8m$, where n is the number of Boolean variables and m is the number of clauses of C. This implies that, for every $k_1, k_2 \in \mathbb{N}$, $k_2 \geq 2$, if $[|h| \leq k_1, |\Sigma| \leq k_2]$-StrSubst can be solved in time $(|u||w|)^{O(1)} \times 2^{o(|\text{var}(u)|)}$, then 3SAT can be solved in time $(m+n)^{O(1)} \times 2^{o(m+n)}$.

Furthermore, the reduction Φ can be extended to morphisms, i. e., to a reduction Φ' that maps a Boolean formula C with n variables and m clauses to a source string u that only contains variables, i. e., $u \in X^*$, with $|u| = O(n+m)$. To this end, let C be a set of m three-literal-clauses with n variables and let $(u, w) := \Phi(C)$. First, we obtain a source string $u' \in X^*$ from u by substituting every occurrence of ¢ by an occurrence of the new variable $y_¢$. Next, we define $u'' := y_¢\, y_¢\, (u')^2$ and $w'' := ¢\,¢\,(w)^2$. Obviously, $|u''| = 2|u| + 2 = O(n + m)$. In order to prove that Φ' is a valid reduction, it is sufficient to show that there exists a substitution h of size 1 with $h(u) = w$ if and only if there exists a morphism g of size 1 with $g(u'') = w''$. The *only if* direction is obvious, since if $h(u) = w$, then $g(u'') = w''$, where $g(x) := h(x)$, $x \in \text{var}(u)$, and $g(y_¢) := ¢$. For the *if* direction, we observe that if $g(u'') = w''$ and $g(y_¢) = ¢$, then $g(u) = w$ holds as well. If, on the other hand, $g(y_¢) = \varepsilon$, then $g(u^2) = w''$, which is a contradiction, since w'' is not a square.

Hence, for every $k_1, k_2 \in \mathbb{N}$, $k_2 \geq 2$, if we can solve $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$, $K \in \{$StrMorph, StrSubst$\}$, in time $(|u||w|)^{O(1)} \times 2^{o(|\text{var}(u)|)}$, then 3SAT can be solved in time $(m+n)^{O(1)} \times 2^{o(m+n)}$, which is a contradiction to ETH.

Theorem 7 *For every $K \in \{$StrMorph, StrSubst$\}$ and $k_1, k_2 \in \mathbb{N}$, $k_2 \geq 2$, $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$ cannot be solved in time $(|u||w|)^{O(1)} \times 2^{o(|\text{var}(u)|)}$, unless ETH fails.*

Proof Let C be a 3CNF formula with n variables and m clauses, let $(u, w) := \Phi(C)$ and let $(u', w') := \Phi'(C)$, where Φ and Φ' are the transformations from 3SAT to StrSubst and StrMorph, respectively. We observe that $|u| = 3n+1+12m = O(n+m)$, $|u'| = 2|u|+2 = O(n+m)$, $|w| = 2n+1+8m = O(n+m)$

and $|w'| = 2|w| + 2 = \mathrm{O}(n+m)$. In particular, this means that $|\mathrm{var}(u)| = \mathrm{O}(n+m)$ and $|\mathrm{var}(u')| = \mathrm{O}(n+m)$. We assume that there exists an algorithm χ_K, $K \in \{\textsc{StrSubst}, \textsc{StrMorph}\}$, that solves $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$ in time $(|u||w|)^{\mathrm{O}(1)} \times 2^{\mathrm{o}(|\mathrm{var}(u)|)}$. We can now solve the 3SAT instance C as follows. In case $K = \textsc{StrSubst}$, we perform the reduction Φ, which constructs u and w, and in case $K = \textsc{StrMorph}$, we perform the reduction Φ', which constructs u' and w'. Then we apply χ_K on input (u, w) or (u', w'), respectively, which has a running time of

$$\begin{aligned}
(|u||w|)^{\mathrm{O}(1)} \times 2^{\mathrm{o}(|\mathrm{var}(u)|)} &= (m+n)^{\mathrm{O}(1)} \times 2^{\mathrm{o}(m+n)} = \\
2^{\mathrm{o}(m+n)+\log^{\mathrm{O}(1)}(m+n)} &= 2^{\mathrm{o}(m+n)} .
\end{aligned}$$

As pointed out by Lemma 5, Φ and Φ' are reductions from 3SAT to $[|h| \leq k_1, |\Sigma| \leq k_2]$-$K$, $K \in \{\textsc{StrSubst}, \textsc{StrMorph}\}$, as long as $k_1 \geq 1$ and $k_2 \geq 2$. Hence, C is a positive 3SAT instance if and only if (u, v) is a positive $[|h| \leq k_1, |\Sigma| \leq k_2]$-$\textsc{StrSubst}$ instance (if and only if (u', v') is a positive $[|h| \leq k_1, |\Sigma| \leq k_2]$-$\textsc{StrMorph}$ instance, respectively). This is a contradiction to ETH as formulated before. □

5 Conclusions

In this paper, we investigate 8 variants of the string morphism problem (i. e., the problems in SMP) with respect to the 5 parameters $|\mathrm{var}(u)|$, $|\Sigma|$, $|w|$, $|u|_{\mathrm{var}}$ and $|h|$. From our results, we can conclude either $W[1]$-hardness or FPT-membership for *all* of these 256 parameterised problems.

Our results can be summarised as follows. The string morphism problems become fixed parameter tractable if parameterised by $|\mathrm{var}(u)|$ and $|w|$, but, if at most one of those is a parameter, then almost all problem variants are $W[1]$-hard and for the few cases that are still in FPT, this is due to the fact that the considered parameters implicitly bound $|\mathrm{var}(u)|$ and $|w|$ as well. A natural question that arises is whether there are better fpt-algorithms for the problems $[|\mathrm{var}(u)|, |w|]$-K, $K \in$ SMP than the brute-force algorithm. Furthermore, it might be interesting to measure the kernel sizes for these problems.

For the $W[1]$-hard variants of the string morphism problems, we show $W[1]$-membership for the cases that $|w|$ is a parameter and both $|\mathrm{var}(u)|$ and $|u|_{\mathrm{var}}$ are parameters, whereas for the case that only $|\mathrm{var}(u)|$ is a parameter, we are only able to show $W[P]$-membership. Hence, the exact location of this problem variant in the $W[1]$-hierarchy remains open. In this regards, it is worth mentioning that examples for problems that are complete for $W[t]$ with $t \geq 3$ are rare (see Chen and Zhang [8]).

References

1. Abu-Khzam, F.N., Fernau, H., Langston, M.A., Lee-Cultura, S., Stege, U.: A fixed-parameter algorithm for string-to-string correction. Discrete Optimization **8**, 41–49 (2011)

2. Amir, A., Aumann, Y., Cole, R., Lewenstein, M., Porat, E.: Function matching: Algorithms, applications, and a lower bound. In: Proc. 30th Int. Coll. on Automata, Languages and Programming, ICALP 2003, *LNCS*, vol. 2719, pp. 929–942 (2003)
3. Amir, A., Nor, I.: Generalized function matching. Journal of Discrete Algorithms **5**, 514–523 (2007)
4. Angluin, D.: Finding patterns common to a set of strings. In: Proc. 11th Annual ACM Symposium on Theory of Computing, STOC 1979, pp. 130–141 (1979)
5. Baker, B.S.: Parameterized pattern matching: Algorithms and applications. Journal of Computer and System Sciences **52**, 28–42 (1996)
6. Câmpeanu, C., Salomaa, K., Yu, S.: A formal study of practical regular expressions. International Journal of Foundations of Computer Science **14**, 1007–1018 (2003)
7. Cesati, M.: The Turing way to parameterized complexity. Journal of Computer and System Sciences **67**, 654–685 (2003)
8. Chen, J., Zhang, F.: On product covering in 3-tier supply chain models: Natural complete problems for W[3] and W[4]. Theoretical Computer Science **363**(3), 278–288 (2006)
9. Clifford, R., Harrow, A.W., Popa, A., Sach, B.: Generalised matching. In: Proc. 16th International Symposium on String Processing and Information Retrieval, SPIRE 2009, *LNCS*, vol. 5721, pp. 295–301 (2009)
10. Downey, R., Fellows, M., Kapron, B., Hallett, M., Wareham, H.: Parameterized complexity of some problems in logic and linguistics (extended abstract). In: Proc. 2nd Workshop on Structural Complexity and Recursion-theoretic Methods in Logic Programming, *LNCS*, vol. 813, pp. 89–101 (1994)
11. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
12. Ehrenfeucht, A., Rozenberg, G.: Finding a homomorphism between two words is NP-complete. Information Processing Letters **9**, 86–88 (1979)
13. Fellows, M.R., Hermelin, D., Rosamond, F., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theoretical Computer Science **401**, 53–61 (2009)
14. Fernau, H., Schmid, M.L.: Pattern matching with variables: A multivariate complexity analysis. In: Proc. 24th Annual Symposium on Combinatorial Pattern Matching, CPM 2013, *LNCS*, vol. 7922, pp. 83–94 (2013)
15. Fernau, H., Schmid, M.L., Villanger, Y.: On the parameterised complexity of string morphism problems. In: Proc. 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 24, pp. 55–66 (2013)
16. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
17. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical Computer Science. Springer (2010)
18. Freydenberger, D.D., Reidenbach, D., Schneider, J.C.: Unambiguous morphic images of strings. International Journal of Foundations of Computer Science **17**, 601–628 (2006)
19. Garey, M.R., Johnson, D.S.: Computers And Intractability. W. H. Freeman and Company (1979)
20. Geilke, M., Zilles, S.: Learning relational patterns. In: Proc. 22nd International Conference on Algorithmic Learning Theory, ALT 2011, *LNCS*, vol. 6925, pp. 84–98 (2011)
21. Harju, T., Karhumäki, J.: Morphisms. In: G. Rozenberg, A. Salomaa (eds.) Handbook of Formal Languages, vol. 1, chap. 7, pp. 439–510. Springer (1997)
22. Ibarra, O., Pong, T.C., Sohn, S.: A note on parsing pattern languages. Pattern Recognition Letters **16**, 179–182 (1995)
23. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences **63**, 512–530 (2001)
24. Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., Yu, S.: Pattern languages with and without erasing. International Journal of Computer Mathematics **50**, 147–163 (1994)
25. Lokshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the Exponential Time Hypothesis. EATCS Bulletin **105**, 41–72 (2011)
26. Mateescu, A., Salomaa, A.: Finite degrees of ambiguity in pattern languages. RAIRO Informatique théoretique et Applications **28**, 233–253 (1994)

27. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. Journal of Computer and System Sciences **67**(4), 757–771 (2003)
28. Reidenbach, D., Schmid, M.L.: A polynomial time match test for large classes of extended regular expressions. In: Proc. 15th International Conference on Implementation and Application of Automata, CIAA 2010, *LNCS*, vol. 6482, pp. 241–250 (2011)
29. Reidenbach, D., Schmid, M.L.: Patterns with bounded treewidth. In: Proc. 6th International Conference on Language and Automata Theory and Applications, LATA 2012, *LNCS*, vol. 7183, pp. 468–479 (2012)
30. Rinaudo, P., Ponty, Y., Barth, D., Denise, A.: Tree decomposition and parameterized algorithms for rna structure-sequence alignment including tertiary interactions and pseudoknots — (extended abstract). In: B.J. Raphael, J. Tang (eds.) Algorithms in Bioinformatics — 12th International Workshop, WABI, *LNCS*, vol. 7534, pp. 149–164. Springer (2012)
31. Schmid, M.L.: On the membership problem for pattern languages and related topics. Ph.D. thesis, Dept. of Computer Science, Loughborough University (2012)
32. Shinohara, T.: Polynomial time inference of pattern languages and its application. In: Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, pp. 191–209 (1982)
33. Stephan, F., Yoshinaka, R., Zeugmann, T.: On the parameterised complexity of learning patterns. In: Proc. 26th International Symposium on Computer and Information Sciences, ISCIS 2011, pp. 277–281