Pattern Matching with Variables: A Multivariate Complexity Analysis
(Extended Abstract)

Henning Fernau and Markus L. Schmid*

Fachbereich 4 – Abteilung Informatik, Universität Trier, D-54286 Trier, Germany
{Fernau,MSchmid}@uni-trier.de

Abstract. In the context of this paper, a pattern is a string that contains variables and terminals. A pattern α matches a terminal word w if w can be obtained by uniformly substituting the variables of α by terminal words. It is a well-known fact that deciding whether a given terminal word matches a given pattern is an NP-complete problem. In this work, we consider numerous parameters of this problem and for all possible combinations of these parameters, we investigate the question whether or not the variant obtained by bounding these parameters by constants can be solved efficiently.

Keywords: Parameterised Pattern Matching, Function Matching, NP-Completeness, Membership Problem for Pattern Languages, Morphisms.

1 Introduction

In the present work, a detailed complexity analysis of a computationally hard pattern matching problem is provided. The *patterns* considered in this context are strings containing *variables* from $\{x_1, x_2, x_3, \ldots\}$ and *terminal symbols* from a finite alphabet Σ, e. g., $\alpha := x_1\,\mathtt{a}\,x_1\,\mathtt{b}\,x_2\,x_2$ is a pattern, where $\mathtt{a}, \mathtt{b} \in \Sigma$. We say that a word w over Σ *matches* a pattern α if and only if w can be derived from α by uniformly substituting the variables in α by terminal words. The respective pattern matching problem is then to decide for a given pattern and a given word, whether or not the word matches the pattern. For example, the pattern α from above is matched by the word $u := \mathtt{bacaabacabbaba}$, since substituting x_1 and x_2 in α by $\mathtt{baca}$ and $\mathtt{ba}$, respectively, yields u. On the other hand, α is not matched by the word $v := \mathtt{cbcabbcbbccbc}$, since v cannot be obtained by substituting the variables of α by some words.

To the knowledge of the authors, this kind of pattern matching problem first appeared in the literature in 1979 in form of the membership problem for Angluin's *pattern languages* [3, 4] (i. e., the set of all words that match a certain pattern) and, independently, it has been studied by Ehrenfeucht and Rozenberg in [9], where they investigate the more general problem of deciding on the

* Corresponding author.

J. Fischer and P. Sanders (Eds.): CPM 2013, LNCS 7922, pp. 83–94, 2013.

existence of a morphism between two given words (which is equivalent to the above pattern matching problem, if the patterns are *terminal-free*, i. e., they only contain variables).

Since their introduction by Angluin, pattern languages have been intensely studied in the learning theory community in the context of inductive inference (see, e. g., Angluin [4], Shinohara [28], Reidenbach [22, 23] and, for a survey, Ng and Shinohara [20]) and, furthermore, their language theoretical aspects have been investigated (see, e. g., Angluin [4], Jiang et al. [17], Ohlebusch and Ukkonen [21], Freydenberger and Reidenbach [10], Bremer and Freydenberger [6]). However, a detailed investigation of the complexity of their membership problem, i. e., the above described pattern matching problem, has been somewhat neglected. Some of the early work that is worth mentioning in this regard is by Ibarra et al. [15], who provide a more thorough worst case complexity analysis, and by Shinohara [29], who shows that matching patterns with variables can be done in polynomial time for certain special classes of patterns. Recently, Reidenbach and Schmid [24, 25] identify complicated structural parameters of patterns that, if bounded by a constant, allow the corresponding matching problem to be performed in polynomial time (see also Schmid [27]).

In the pattern matching community, independent from Angluin's work, the above described pattern matching problem has been rediscovered by a series of papers. This development starts with [5] in which Baker introduces so-called *parameterised pattern matching*, where a text is not searched for all occurrences of a specific factor, but for all occurrences of factors that satisfy a given pattern with parameters (i. e., variables). In the original version of parameterised pattern matching, the variables in the pattern can only be substituted by single symbols and, furthermore, the substitution must be injective, i. e., different variables cannot be substituted by the same symbol. Amir et al. [1] generalise this problem to *function matching* by dropping the injectivity condition and in [2], Amir and Nor introduce *generalized function matching*, where variables can be substituted by words instead of single symbols and "don't care" symbols can be used in addition to variables. In 2009, Clifford et al. [8] considered generalised function matching as introduced by Amir and Nor, but without "don't care" symbols, which leads to patterns as introduced by Angluin.

In [2], 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. Another motivation arises from the observation that the problem of matching patterns with variables constitutes a special case of the matchtest for *regular expressions with backreferences* (see, e. g., Câmpeanu et al. [7]), which nowadays are a standard element of most text editors and programming languages (cf. Friedl [12]). Due to its simple definition, the above described pattern matching paradigm also has connections to numerous other areas of theoretical computer science and discrete mathematics, such as (un-)avoidable patterns (cf. Jiang et al. [16]), word equations (cf. Mateescu and Salomaa [19]), the ambiguity of morphisms (cf. Freydenberger et al. [11]) and equality sets (cf. Harju and Karhumäki [14]).

It is a well-known fact that – in its general sense – pattern matching with variables is an NP-complete problem; a result that has been independently reported several times (cf. Angluin [4], Ehrenfeucht and Rozenberg [9], Clifford et al. [8]). However, there are many different versions of the problem, tailored to different aspects and research questions. For example, in Angluin's original definition, variables can only be substituted by non-empty words and Shinohara soon afterwards complemented this definition in [28] by including the empty word as well. This marginal difference, as pointed out by numerous results, can have a substantial impact on learnability and decidability questions of the corresponding classes of *nonerasing* pattern languages on the one hand and *erasing* pattern languages on the other. If we turn from the languages point of view of patterns to the respective pattern matching task, then, at a first glance, this difference whether or not variables can be erased seems negligible. However, in the context of pattern matching, other aspects are relevant, which for pattern languages are only of secondary importance. For example, requiring variables to be substituted in an *injective* way is a natural assumption for most pattern matching tasks and bounding the maximal length of these terminal words by a constant (which would turn pattern languages into finite languages) makes sense for special applications (cf. Baker [5]). Hence, there are many variants of the above described pattern matching problem, each with its individual motivation, and the computational hardness of all these variants cannot directly be concluded from the existing NP-completeness results.

For a systematic investigation, we consider the following natural parameters: the number of different variables in the pattern, the maximal number of occurrences of the same variable in the pattern, the length of the terminal word, the maximum length of the substitution words for variables and the cardinality of the terminal alphabet. For all combinations of these parameters, we answer the question whether or not the parameters can be bounded by (preferably small) constants such that the resulting variant of the pattern matching problem is still NP-complete. In addition to this, we also study the differences between the erasing and nonerasing case, between the injective and non-injective case and between the case where patterns may contain terminal symbols and the terminal-free case.

Due to space constraints, the formal proofs for most of the results presented in this paper are omitted.

2 Definitions

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

Let $X := \{x_1, x_2, x_3, \ldots\}$ and every $x \in X$ is a *variable*. Let Σ be a finite alphabet of *terminals*. Every $\alpha \in (X \cup \Sigma)^+$ is a *pattern* and every $w \in \Sigma^*$ is a (*terminal*) *word*. For any pattern α, we refer to the set of variables in α as $\mathrm{var}(\alpha)$ and, for any variable $x \in \mathrm{var}(\alpha)$, $|\alpha|_x$ denotes the number of occurrences of x in α.

Let α be a pattern. A *substitution* (*for* α) is a mapping $h : \mathrm{var}(\alpha) \to \Sigma^*$. For every $x \in \mathrm{var}(\alpha)$, we say that x *is substituted by* $h(x)$ and $h(\alpha)$ denotes the word obtained by substituting every occurrence of a variable x in α by $h(x)$ and leaving the terminals unchanged. If, for every $x \in \mathrm{var}(\alpha)$, $h(x) \neq \varepsilon$, then h is *nonerasing* (h is also called *erasing* if it is not non-erasing). If, for all $x, y \in \mathrm{var}(\alpha)$, $x \neq y$ and $h(x) \neq \varepsilon \neq h(y)$ implies $h(x) \neq h(y)$, then h is E-*injective*[1] and h is called *injective* if it is E-injective and, for at most one $x \in \mathrm{var}(\alpha)$, $h(x) = \varepsilon$.

Example 1. Let $\beta := x_1\,\mathtt{a}\,x_2\,\mathtt{b}\,x_2\,x_1\,x_2$ be a pattern, let $u := \mathtt{bacbabbacb}$ and let $v := \mathtt{abaabbababab}$. It can be verified that $h(\beta) = u$, where $h(x_1) = \mathtt{bacb}$, $h(x_2) = \varepsilon$ and $g(\beta) = v$, where $g(x_1) = g(x_2) = \mathtt{ab}$. Furthermore, β cannot be mapped to u by a nonerasing substitution and β cannot be mapped to v by an injective substitution.

If the type of substitution is clear from the context, then we simply say that a word w *matches* α to denote that there exists such a substitution h with $h(\alpha) = w$. We can now formally define the *pattern matching problem with variables*, denoted by PMV, which has informally been described in Section 1:

PMV
Instance: A pattern α and a word $w \in \Sigma^*$.
Question: Does there exist a substitution h with $h(\alpha) = w$?

As explained in Section 1, the above problem exists in various contexts with individual terminology. Since we consider the problem in a broader sense, we term it pattern matching problem with variables in order to distinguish it – and all its variants to be investigated in this paper – from the classical pattern matching paradigm without variables.

Next, we define several parameters of PMV. To this end, let α be a pattern, let w be a word and let h be a substitution for α.

- $\rho_{|\,\mathrm{var}(\alpha)|} := |\,\mathrm{var}(\alpha)|$,
- $\rho_{|\alpha|_x} := \max\{|\alpha|_x \mid x \in \mathrm{var}(\alpha)\}$,
- $\rho_{|w|} := |w|$,
- $\rho_{|\Sigma|} := |\Sigma|$,
- $\rho_{|h(x)|} := \max\{|h(x)| \mid x \in \mathrm{var}(\alpha)\}$.

The restricted versions of the problem PMV are now defined by P-$[Z, I, T]$-PMV, where P is a list of parameters that are bounded, $Z \in \{\mathrm{E}, \mathrm{NE}\}$ denotes whether we are considering the erasing or nonerasing case, $T \in \{\mathrm{tf}, \text{n-tf}\}$ denotes whether or not we require the patterns to be terminal-free and $I \in \{\mathrm{inj}, \text{n-inj}\}$ denotes whether or not we require the substitution to be injective (more precisely, if $Z = \mathrm{NE}$, then $I = \mathrm{inj}$ denotes injectivity, but if $Z = \mathrm{E}$, then $I = \mathrm{inj}$ denotes E-injectivity). Hence, $[\rho^{c_1}_{|\alpha|_x}, \rho^{c_2}_{|\Sigma|}, \rho^{c_3}_{|h(x)|}]$-[NE, n-inj, tf]- PMV denotes the problem to decide for a given *terminal-free* pattern α and a given word $w \in \Sigma^*$ with $\max\{|\alpha|_x \mid x \in \mathrm{var}(\alpha)\} \leq c_1$ and $|\Sigma| \leq c_2$, whether or not there exists a

[1] We use E-injectivity, since if an erasing substitution is injective in the classical sense, then it is "almost" nonerasing, i. e., only one variable can be erased.

nonerasing substitution h (possibly *non-injective*) that satisfies $\max\{|h(x)| \mid x \in \operatorname{var}(\alpha)\} \leq c_3$ and $h(\alpha) = w$, where c_1, c_2 and c_3 are some constants.

The contribution of this paper is to show for each of the 256 individual problems P-$[Z, I, T]$-PMV whether or not there exist constants such that if the parameters in P are bounded by these constants, this version of PMV is still NP-complete or whether it can be solved in polynomial time. To this end, we first summarise all the respective known results from the literature and then we close the remaining gaps.

3 Known Results and Preliminary Observations

In this section, we briefly summarise those variants of PMV, for which NP-completeness or membership in P has already been established. To this end, we first informally describe an obvious and simple brute-force algorithm that solves the pattern matching problem with variables. For some instance (α, w) of PMV with $m := |\operatorname{var}(\alpha)|$, we simply enumerate all tuples $(u_1, u_2, \ldots, u_m)$, where, for every i, $1 \leq i \leq m$, u_i is a factor of w. Then, for each such tuple $(u_1, u_2, \ldots, u_m)$, we check whether $h(\alpha) = w$, where h is defined by $h(x_i) := u_i$, $1 \leq i \leq m$. This procedure can be performed in time exponential only in m and, furthermore, it is generic in that it works for any variant of PMV. This particularly implies that every version of PMV, for which $\rho_{|\operatorname{var}(\alpha)|}$ is restricted, can be solved in polynomial time.

Next, we note that in the nonerasing case, a restriction of $\rho_{|w|}$ implicitly bounds $\rho_{|\operatorname{var}(\alpha)|}$ as well and, thus, all the corresponding versions of the pattern matching problem with variables can be solved efficiently. Moreover, in [13], Geilke and Zilles note that if $\rho_{|w|} \leq c$, for some constant c, then this particularly implies that the number of variables that are *not* erased is bounded by c as well. As demonstrated in [13], this means that also for the erasing case PMV can be solved in polynomial time if the length of the input word is bounded by a constant. Consequently, every version of PMV, for which $\rho_{|\operatorname{var}(\alpha)|}$ or $\rho_{|w|}$ is restricted, can be solved in polynomial time; thus, in the following, we shall neglect these two parameters and focus on the remaining 3 parameters $\rho_{|\alpha|_x}$, $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$.

In the next table, we briefly summarise those variants of PMV, for which NP-completeness has already been established. A numerical entry denotes the constant bound of a parameter and "–" means that a parameter is unrestricted.

	E / NE	inj / n-inj	tf / n-tf	$\lvert h(x)\rvert$	$\lvert\alpha\rvert_x$	$\lvert\Sigma\rvert$	Complexity
1	NE	n-inj	n-tf	**3**	–	**2**	**NP-C** [4]
2	E, NE	n-inj	tf	**3**	–	**2**	**NP-C** [9]
3	NE	n-inj	tf	**2**	–	**2**	**NP-C** [8]
4	NE	inj	tf	–	–	**2**	**NP-C** [8]
5	NE	inj	tf	**2**	–	–	**NP-C** [8]
6	E	n-inj	n-tf	–	**2**	**2**	**NP-C** [26]

The main contribution of this paper is to close the gaps that are left open in the above table. Before we present our main results in this regard, we conclude this section by taking a closer look at the parameters $\rho_{|\alpha|_x}$ and $\rho_{|\Sigma|}$. As indicated by rows 1 to 4 and row 6, restricting $\rho_{|\Sigma|}$ does not seem to help to solve PMV efficiently. In [26] it is shown that even if we additionally require the number of occurrences per variable to be bounded by 2, then PMV is still NP-complete. However, regarding these two parameters, we seem to have reached the boundary between P and NP-completeness, since it can be easily shown that PMV can be solved in polynomial time if parameter ρ_Σ or $\rho_{|\alpha|_x}$ is bounded by 1 (see, e. g., Schmid [27]).

4 Main Results

In this section, we investigate the complexity of all the variants of the pattern matching problem with variables that are not already covered by the table presented in the previous section. Most of these variants turn out to be NP-complete. The general proof technique used to establish these results is illustrated in Section 5. We shall now first consider the non-injective case, i. e., we consider the problems P-$[Z, \text{n-inj}, T]$-PMV first and the problems P-$[Z, \text{inj}, T]$-PMV later on.

4.1 The Non-injective Case

All the results of this section are first presented in a table of the form already used in the previous section and then we discuss them in more detail.

E / NE	inj / n-inj	tf / n-tf	$\lvert h(x)\rvert$	$\lvert\alpha\rvert_x$	$\lvert\Sigma\rvert$	Complexity
E	n-inj	n-tf	**1**	**2**	**2**	**NP-C**
NE	n-inj	n-tf	**3**	**2**	**2**	**NP-C**
E	n-inj	tf	**1**	**8**	**2**	**NP-C**
NE	n-inj	tf	**3**	**3**	**4**	**NP-C**

As mentioned in Section 3, Clifford et al. show in [8] that the nonerasing, terminal-free and non-injective case of the pattern matching problem with variables is NP-complete, even if additionally the parameters $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$ are bounded. By the rows 2 and 4 of the above table, we strengthen this result by stating that the NP-completeness is preserved, even if in addition also $\rho_{|\alpha|_x}$ is bounded and this holds both for the terminal-free and non-terminal-free case. However, we are only able to prove that these results hold if the parameter $\rho_{|\alpha|_x}$ is bounded by 3 and the case where $\rho_{|\alpha|_x}$ is bounded by 2 is left open.

With respect to the erasing case, i. e., rows 1 and 3 of the above table, we observe a surprising situation that deserves to be discussed in a bit more detail. To this end, we introduce a special case of a substitution. A substitution h (for a pattern α) is called a *renaming* if every variable of α is either erased by h or substituted by a single symbol, i. e., for every $x \in \text{var}(\alpha)$, $|h(x)| \leq 1$. Now row 1 shows that the erasing, non-injective and non-terminal-free version of the pattern

matching problem with variables remains NP-complete, even if both $\rho_{|\alpha|_x}$ and $\rho_{|\Sigma|}$ are bounded by 2 and the substitution needs to be a renaming. This is a very restricted version of the pattern matching problem with variables, which seems to be located directly on the border between NP-completeness and polynomial time solvability, since the nonerasing version of this problem is trivially solvable in linear time, the parameter $\rho_{|h(x)|}$ is already bounded in the strongest possible sense, if $\rho_{|\alpha|_x}$ or $\rho_{|\Sigma|}$ is bounded by 1 instead of 2, then, as mentioned in Section 3, the problem becomes polynomial time solvable and, in the next section, we shall see that the injective version is in P as well.

With respect to the terminal-free case (row 3 of the table), we are only able to show NP-completeness if the parameter $\rho_{|\alpha|_x}$ is bounded by 8 instead of 2. This version of the pattern matching problem with variables can be rephrased as a more general problem on strings: given two strings u and v, can u be transformed into v in such a way that every symbol of w is either erased, substituted by $\mathtt{a}$ or substituted by $\mathtt{b}$? This problem is NP-complete, even if every symbol in u occurs at most 8 times. It is open, however, whether it is still NP-complete if at most two occurrences per symbol are allowed.

We conclude this section by pointing out that the pattern matching problem that Baker considers in [5], and for which she presents efficient algorithms, in fact relies on the problem of finding a renaming between two words. However, in [5] only nonerasing and injective renamings are considered and with our above result we can conclude that Baker's pattern matching problem most likely cannot be solved in polynomial time if it is generalised to erasing and non-injective renamings.

4.2 The Injective Case

A main difference between the complexity of the injective and non-injective cases is that in the injective case, bounding $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$ already yields polynomial time solvability (see Theorem 1 below), whereas the non-injective case remains NP-complete, even if we additionally bound $\rho_{|\alpha|_x}$ (as stated in Section 4.1). Informally speaking, this is due to the fact that if $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$ are bounded by some constants, then the number of words variables can be substituted with is bounded by some constant, say c, as well. Now if we additionally require injectivity, then the number of variables that are substituted with non-empty words is bounded by c, too, which directly implies the polynomial time solvability for the nonerasing case. In order to extend this result to the erasing case, we apply a technique similar to the one used by Geilke and Zilles in [13].

Theorem 1. *Let $k_1, k_2 \in \mathbb{N}$, let $Z \in \{\mathrm{E}, \mathrm{NE}\}$ and let $T \in \{\mathrm{tf}, \text{n-tf}\}$. The problem $[\rho^{k_1}_{|\Sigma|}, \rho^{k_2}_{|h(x)|}]$-$[Z, \mathrm{inj}, T]$-PMV is in P.*

Proof. Since the case $Z = \mathrm{E}$ implies the case $Z = \mathrm{NE}$, we shall only prove the former.

Let α be a pattern and let w be a word over $\Sigma := \{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{k_1}\}$. Let S be an arbitrary subset of $\mathrm{var}(\alpha)$. We say that S satisfies condition $(*)$ if and only

if there exists an E-injective substitution h with $h(\alpha) = w$, $1 \leq |h(x)| \leq k_2$, for every $x \in S$, and $h(x) = \varepsilon$, for every $x \in \mathrm{var}(\alpha) \setminus S$. For any set $S \subseteq \mathrm{var}(\alpha)$, it can be checked in time exponential in $|S|$, whether S satisfies condition $(*)$. More precisely, this can be done in the following way. First, we obtain a pattern β from α by erasing all variables in $\mathrm{var}(\alpha) \setminus S$. Then we use a brute-force algorithm to check whether or not there exists an injective nonerasing substitution h with $h(\beta) = w$ and $1 \leq |h(x)| \leq k_2$, $x \in \mathrm{var}(\beta)$, which can be done in time $\mathrm{O}(k_2^{|S|})$.

For the sake of convenience, we define $k' := k_2 \times k_1^{k_2}$. We observe that there are $\mathrm{O}(k')$ non-empty words over $\{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{k_1}\}$ of length at most k_2. This implies that every substitution h that maps more than k' variables to non-empty words of length at most k_2 is necessarily not E-injective. So, for every set $S \subseteq \mathrm{var}(\alpha)$, if $|S| > k'$, then S does not satisfy condition $(*)$. Consequently, there exists an E-injective substitution h with $h(\alpha) = w$, $|h(x)| \leq k_2$, for every $x \in \mathrm{var}(\alpha)$, if and only if there exists a set $S \subseteq \mathrm{var}(\alpha)$ with $|S| \leq k'$ and S satisfies the condition $(*)$.

We conclude that we can solve the problem stated in the theorem by enumerating all possible sets $S \subseteq \mathrm{var}(\alpha)$ with $|S| \leq k'$ and, for each of these sets, checking whether they satisfy condition $(*)$. Since the number of sets $S \subseteq \mathrm{var}(\alpha)$ with $|S| \leq k'$ is

$$\sum_{i=0}^{k'} \binom{|\mathrm{var}(\alpha)|}{i} \leq \sum_{i=0}^{k'} |\mathrm{var}(\alpha)|^i \leq (k'+1)\,\mathrm{var}(\alpha)^{k'} = \mathrm{O}(|\mathrm{var}(\alpha)|^{k'})\,,$$

the runtime of this procedure is exponential only in k'; thus, since k' is a constant, it is polynomial. □

On the other hand, as pointed out by the following table, for all other possibilities to bound some of the parameters $\rho_{|\Sigma|}$, $\rho_{|\alpha|_x}$ and $\rho_{|h(x)|}$, without bounding both $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$ at the same time, we can show NP-completeness:

E / NE	inj / n-inj	tf / n-tf	$\lvert h(x)\rvert$	$\lvert\alpha\rvert_x$	$\lvert\Sigma\rvert$	Complexity
E	inj	n-tf	**5**	**2**	–	**NP-C**
NE	inj	n-tf	**19**	**2**	–	**NP-C**
E, NE	inj	n-tf	–	**2**	**2**	**NP-C**
E, NE	inj	tf	**19**	**4**	–	**NP-C**
E, NE	inj	tf	–	**9**	**5**	**NP-C**

With respect to the injective case (and in contrast to the non-injective case), we are not able to conclude any results about renamings. In particular, the most interesting open question in this regard is whether or not the following problem is NP-complete:

Instance: A pattern α and a word $w \in \Sigma^*$.
Question: Does there exist an E-injective renaming h with $h(\alpha) = w$?

We conjecture that this question can be answered in the affirmative.

In order to conclude this section, we wish to point out that for every variant of the pattern matching problem with variables that is not explicitly mentioned in the above tables, either NP-completeness or membership in P is directly implied by one of the results presented in this section or Section 3.

5 Proof Techniques

In this section, we give a sketch of the main proof technique for the hardness results presented in Section 4. To this end, we first define a graph problem, which is particularly suitable for our purposes.

Let $\mathcal{G} = (V, E)$ be a graph with $V := \{v_1, v_2, \ldots, v_n\}$. The *neighbourhood* of a vertex $v \in V$ is the set $N_{\mathcal{G}}(v) := \{u \mid \{v, u\} \in E\}$ and $N_{\mathcal{G}}[v] := N_{\mathcal{G}}(v) \cup \{v\}$ is called the *closed neighbourhood* of v. If, for some $k \in \mathbb{N}$, $|N_{\mathcal{G}}(v)| = k$, for every $v \in V$, then $\mathcal{G}$ is *k-regular*. A *perfect code* for $\mathcal{G}$ is a subset $C \subseteq V$ with the property that, for every $v \in V$, $|N_{\mathcal{G}}[v] \cap C| = 1$. Next, we define the problem to decide whether or not a given 3-regular graph has a perfect code:

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

In [18], Kratochvíl and Křivánek prove the problem 3R-PERFECT-CODE to be NP-complete:

Theorem 2 (Kratochvíl and Křivánek [18]). 3R-PERFECT-CODE *is NP-complete.*

All the NP-completeness results of Section 4 can be proved by reducing 3R-PERFECT-CODE to the appropriate variant of PMV. However, these reductions must be individually tailored to these different variants. As an example, we give a reduction from 3R-PERFECT-CODE to $[\rho^5_{|h(x)|}, \rho^2_{|\alpha|_x}]$-[E, inj, n-tf]-PMV, which implies the result stated in row 1 of the table presented in Section 4.2.

Let $\mathcal{G} = (V, E)$ with $V := \{v_1, v_2, \ldots, v_n\}$ be a 3-regular graph and, for every i, $1 \leq i \leq n$, let N_i be the closed neighbourhood of v_i. We transform the graph $\mathcal{G}$ into a pattern α and a word w over $\Sigma := \{\mathtt{a}_i, ¢_i, \#_j \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$, such that, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 2$. Now, for any i, $1 \leq i \leq n$, let $N_{j_1}, N_{j_2}, N_{j_3}, N_{j_4}$ be exactly the closed neighbourhoods that contain vertex v_i. We transform vertex v_i into the pattern variables $x_{i,j_1}, x_{i,j_2}, x_{i,j_3}, x_{i,j_4}$; thus, our interpretation shall be that variable $x_{i,j}$ refers to vertex v_i in the closed neighbourhood of vertex v_j.

For every i, $1 \leq i \leq n$, the closed neighbourhood $N_i := \{v_{j_1}, v_{j_2}, v_{j_3}, v_{j_4}\}$ is transformed into

$$\beta_i := x_{j_1,i}\, x_{j_2,i}\, x_{j_3,i}\, x_{j_4,i} \text{ and}$$
$$u_i := \mathtt{a}_i\,.$$

Furthermore, for every i, $1 \leq i \leq n$, we define

$$\gamma_i := z_i \, ¢_i \, x_{i,j_1} \, x_{i,j_2} \, x_{i,j_3} \, x_{i,j_4} \, ¢_i \, z_i' \text{ and}$$
$$v_i := ¢_i \, ¢_i \, \mathtt{a}_{j_1} \, \mathtt{a}_{j_2} \, \mathtt{a}_{j_3} \, \mathtt{a}_{j_4} \, ¢_i \, ,$$

where $N_i = \{v_{j_1}, v_{j_2}, v_{j_3}, v_{j_4}\}$. Finally, we define

$$\alpha := \beta_1 \, \#_1 \, \beta_2 \, \#_2 \cdots \#_{n-1} \, \beta_n \, \#_n \, \gamma_1 \, \#_{n+1} \, \gamma_2 \, \#_{n+2} \cdots \#_{2n-1} \, \gamma_n \text{ and}$$
$$w := u_1 \, \#_1 \, u_2 \, \#_2 \cdots \#_{n-1} \, u_n \, \#_n \, v_1 \, \#_{n+1} \, v_2 \, \#_{n+2} \cdots \#_{2n-1} \, v_n \, .$$

Every variable z_i, z_i', $1 \leq i \leq n$, has only one occurrence in α. For every i, $1 \leq i \leq n$, and every j with $v_j \in N_i$, variable $x_{j,i}$ has exactly one occurrence in β_i and exactly one occurrence in γ_j. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 2$.

In order to see that the existence of an E-injective substitution h for α with $h(\alpha) = w$ and $|h(x)| \leq 5$ implies the existence of a perfect code for $\mathcal{G}$, we first observe that, for any substitution h with $h(\alpha) = w$, $h(\beta_i) = u_i$ and $h(\gamma_i) = v_i$, $1 \leq i \leq n$, is satisfied. This implies that, for every i, $1 \leq i \leq n$, exactly one of the variables $x_{j_l,i}$, $1 \leq l \leq 4$, where $N_i = \{v_{j_1}, v_{j_2}, v_{j_3}, v_{j_4}\}$, is mapped to $\mathtt{a}_i$ and the other three variables are erased. Furthermore either each of the variables x_{i,j_l}, $1 \leq l \leq 4$, is mapped to $\mathtt{a}_{j_l}$ or all these variables are erased. This directly translates into the situation that it is possible to pick exactly one vertex from each neighbourhood. The converse statement, i. e., the existence of a perfect code implies the existence of such a substitution h, follows from the observation that for the variables $x_{i,j}$ we can define h as induced by the perfect code and, for every i, $1 \leq i \leq n$, either $h(z_i) := ¢_i$ and $h(z_i') := \varepsilon$ or $h(z_i) := \varepsilon$ and $h(z_i') := \mathtt{a}_{j_1} \, \mathtt{a}_{j_2} \, \mathtt{a}_{j_3} \, \mathtt{a}_{j_4} \, ¢_i$, depending on whether or not vertex v_i is a member of the perfect code.

We wish to point out that the above reduction strongly relies on the possibility to erase variables and to have terminal symbols in the pattern; thus, as pointed out by the following explanations, converting it to the nonerasing or the terminal-free case is non-trivial. The general idea of extending our reduction to the terminal-free case is that instead of using terminals # in the pattern, we use variables that are forced to be substituted by #. Especially for the erasing case, this is technically challenging and, furthermore, if we use an unbounded number of occurrences of the same terminal symbol in the pattern, then it is difficult to maintain the restriction on the number of variable occurrences and injectivity at the same time. In the above reduction, we also use the possibility of having an unbounded number of terminal symbols. Hence, if parameter $\rho_{|\Sigma|}$ is bounded, then instead of using arbitrarily many different symbols $\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n$, we either have to use only one symbol $\mathtt{a}$ for different variables, which destroys the injectivity, or we have to encode a single symbol $\mathtt{a}_i$ by a string $\mathtt{ba}^i\mathtt{b}$, which breaks the bound on parameter $\rho_{|h(x)|}$.

6 Future Research Directions

In this paper, for every variant P-$[Z, I, T]$-PMV of the pattern matching problem with variables, we either show that bounding the parameters by *any* constants

leads to polynomial time solvability or that the parameters can be bounded by *some* constants, such that P-$[Z, I, T]$-PMV is NP-complete. Although for the results of the latter type we are mostly able to present rather small constants, we do not provide a full dichotomy result for the class of problems P-$[Z, I, T]$-PMV.

As pointed out in Section 4.1, $[\rho^1_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, n-tf]-PMV is an example for an NP-complete version of the pattern matching problem with variables for which we provable know that any further restriction – except to the terminal-free case, which is open – makes the problem polynomial time solvable. On the other hand, we do not know when exactly the problem $[\rho^3_{|h(x)|}, \rho^3_{|\alpha|_x}, \rho^4_{|\Sigma|}]$-[NE, n-inj, tf]-PMV shifts from NP-completeness to polynomial time solvability when the constants are decreased.

Consequently, possible further research is to completely determine these borderlines between NP-completeness and P with respect to the pattern matching problem with variables.

References

1. Amir, A., Aumann, Y., Cole, R., Lewenstein, M., Porat, E.: Function matching: Algorithms, applications, and a lower bound. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 929–942. Springer, Heidelberg (2003)
2. Amir, A., Nor, I.: Generalized function matching. Journal of Discrete Algorithms 5, 514–523 (2007)
3. 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)
4. Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62 (1980)
5. Baker, B.S.: Parameterized pattern matching: Algorithms and applications. Journal of Computer and System Sciences 52, 28–42 (1996)
6. Bremer, J., Freydenberger, D.D.: Inclusion problems for patterns with a bounded number of variables. In: Gao, Y., Lu, H., Seki, S., Yu, S. (eds.) DLT 2010. LNCS, vol. 6224, pp. 100–111. Springer, Heidelberg (2010)
7. Câmpeanu, C., Salomaa, K., Yu, S.: A formal study of practical regular expressions. International Journal of Foundations of Computer Science 14, 1007–1018 (2003)
8. Clifford, R., Harrow, A.W., Popa, A., Sach, B.: Generalised matching. In: Karlgren, J., Tarhio, J., Hyyrö, H. (eds.) SPIRE 2009. LNCS, vol. 5721, pp. 295–301. Springer, Heidelberg (2009)
9. Ehrenfeucht, A., Rozenberg, G.: Finding a homomorphism between two words is NP-complete. Information Processing Letters 9, 86–88 (1979)
10. Freydenberger, D.D., Reidenbach, D.: Bad news on decision problems for patterns. Information and Computation 208, 83–96 (2010)
11. Freydenberger, D.D., Reidenbach, D., Schneider, J.C.: Unambiguous morphic images of strings. International Journal of Foundations of Computer Science 17, 601–628 (2006)
12. Friedl, J.E.F.: Mastering Regular Expressions, 3rd edn. O'Reilly, Sebastopol (2006)
13. Geilke, M., Zilles, S.: Learning relational patterns. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 84–98. Springer, Heidelberg (2011)

14. Harju, T., Karhumäki, J.: Morphisms. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, ch. 7, pp. 439–510. Springer (1997)
15. Ibarra, O., Pong, T.-C., Sohn, S.: A note on parsing pattern languages. Pattern Recognition Letters 16, 179–182 (1995)
16. 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)
17. Jiang, T., Salomaa, A., Salomaa, K., Yu, S.: Decision problems for patterns. Journal of Computer and System Sciences 50, 53–63 (1995)
18. Kratochvíl, J., Křivánek, M.: On the computational complexity of codes in graphs. In: Koubek, V., Janiga, L., Chytil, M.P. (eds.) MFCS 1988. LNCS, vol. 324, pp. 396–404. Springer, Heidelberg (1988)
19. Mateescu, A., Salomaa, A.: Finite degrees of ambiguity in pattern languages. RAIRO Informatique Théoretique et Applications 28, 233–253 (1994)
20. Ng, Y.K., Shinohara, T.: Developments from enquiries into the learnability of the pattern languages from positive data. Theoretical Computer Science 397, 150–165 (2008)
21. Ohlebusch, E., Ukkonen, E.: On the equivalence problem for E-pattern languages. Theoretical Computer Science 186, 231–248 (1997)
22. Reidenbach, D.: A non-learnable class of E-pattern languages. Theoretical Computer Science 350, 91–102 (2006)
23. Reidenbach, D.: Discontinuities in pattern inference. Theoretical Computer Science 397, 166–193 (2008)
24. Reidenbach, D., Schmid, M.L.: A polynomial time match test for large classes of extended regular expressions. In: Domaratzki, M., Salomaa, K. (eds.) CIAA 2010. LNCS, vol. 6482, pp. 241–250. Springer, Heidelberg (2011)
25. Reidenbach, D., Schmid, M.L.: Patterns with bounded treewidth. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 468–479. Springer, Heidelberg (2012)
26. Schmid, M.L.: A note on the complexity of matching patterns with variables. Information Processing Letters (Submitted)
27. Schmid, M.L.: On the Membership Problem for Pattern Languages and Related Topics. PhD thesis, Department of Computer Science, Loughborough University (2012)
28. Shinohara, T.: Polynomial time inference of extended regular pattern languages. In: Goto, E., Furukawa, K., Nakajima, R., Nakata, I., Yonezawa, A. (eds.) RIMS 1982. LNCS, vol. 147, pp. 115–127. Springer, Heidelberg (1983)
29. 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)