Information and Computation 242 (2015) 287–305

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Pattern matching with variables: A multivariate complexity analysis ☆

Henning Fernau, Markus L. Schmid *

Fachbereich 4 – Abteilung Informatik, Universität Trier, D-54286 Trier, Germany

ARTICLE INFO

Article history:
Received 11 September 2013
Received in revised form 25 June 2014
Available online 23 March 2015

Keywords:
Parameterised pattern matching
Function matching
NP-completeness
Membership problem for pattern languages
Morphisms

ABSTRACT

A pattern α, i.e., a string that contains variables and terminals, matches a terminal word w if w can be obtained by uniformly substituting the variables of α by terminal words. Deciding whether a given terminal word matches a given pattern is NP-complete and this holds for several natural variants of the problem that result from whether or not variables can be erased, whether or not the patterns are required to be terminal-free or whether or not the mapping of variables to terminal words must be injective. We consider numerous parameters of this problem (i.e., number of variables, length of w, length of the words substituted for variables, number of occurrences per variable, cardinality of the terminal alphabet) and for all possible combinations of the parameters (and variants described above), we answer the question whether or not the problem is still NP-complete if these parameters are bounded by constants.

© 2015 Elsevier Inc. All rights reserved.

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 [10], where they investigate the more general problem of deciding on the 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).

☆ A preliminary version [11] of this paper was presented at the conference CPM 2013.

* Corresponding author.
E-mail addresses: Fernau@uni-trier.de (H. Fernau), MSchmid@uni-trier.de (M.L. Schmid).

http://dx.doi.org/10.1016/j.ic.2015.03.006
0890-5401/© 2015 Elsevier Inc. All rights reserved.

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 [32], Reidenbach [24,25] and, for a survey, Ng and Shinohara [22]) and, furthermore, their language theoretical aspects have been investigated (see, e.g., Angluin [4], Jiang et al. [19], Ohlebusch and Ukkonen [23], Freydenberger and Reidenbach [12], Bremer and Freydenberger [7]). 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. [17], who provide a more thorough worst case complexity analysis, and by Shinohara [33], who shows that matching patterns with variables can be done in polynomial time for certain special classes of patterns. Recently, Reidenbach and Schmid [26,27] 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 [29]).

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 *generalised 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. [9] 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. [8], Schmid [30]), which nowadays are a standard element of most text editors and programming languages (cf. Friedl [14]). 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. [18]), word equations (cf. Mateescu and Salomaa [21]), the ambiguity of morphisms (cf. Freydenberger et al. [13]), equality sets (cf. Harju and Karhumäki [16]) and database theory (cf. Barceló et al. [6]).

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 [10], Clifford et al. [9]). 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 [32] 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 *non-erasing* 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. Hence, there are 2^5 different combinations of parameters and coupling these with the 2^3 variants of the pattern matching problem with variables results in 256 individual problems. For each of these problems, the question arises 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 this paper, by giving a brief overview of all the existing related results that we are aware of, we show that answers to many of these questions can already be concluded from the literature. Nevertheless, several important cases have not yet been settled and our main contribution is to close all these remaining gaps, such that we obtain an answer for *all* of these 256 questions. In this regard, we provide dichotomy results (in terms of Schaefer [28]) for the pattern matching problem with variables with respect to the above mentioned parameters.

The main motivation for undertaking such a research task is the following. While many NP-complete problems naturally occur in theory and also in practical situations, it is almost never the case that the actual problem that has to be solved is exactly the one for which NP-completeness has been established. It is rather the case that we face a subproblem of an NP-complete problem, tailored to the context in which it is encountered. This is also the reason why NP-completeness results are usually accompanied with statements like "This problem is still NP-complete, even if …". As an example, consider the intensive research that is done on proving NP-completeness for graph problems on restricted classes of graphs as planar graphs, graphs with constant degree and so on or the classical results that the satisfiability problem for Boolean formulas is NP-complete even if every clause contains at most 3 literals. Similarly, if some kind of pattern matching problem with

variables is encountered, this is most likely not exactly one of the versions for which NP-completeness has been established in the literature, but, as we believe, it is very likely to be covered by one of the 256 restricted subproblems described above. Therefore, a thorough analysis of the complexity of these problems is a worthwhile research task.

This paper is organised as follows. In Section 2, we recall some general concepts and we formally define the pattern matching problem with variables. In Section 3, we first provide an overview of all the related results that exist in the literature and that we are aware of and, furthermore, we outline the proof technique that shall be used in order to establish our new results. The main part of the paper is represented by Sections 4 and 5. More precisely, in Section 4 we only consider the non-injective variants of the pattern matching problem with variables and in Section 5, the injective variants are investigated. In each of these parts, we shall first consider the erasing case and then the non-erasing one. Finally, in Section 6, we summarise our results and state some related open problems.

2. Preliminaries

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* with $\Sigma \cap X = \emptyset$. 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 *non-erasing* (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 non-erasing 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 are now ready to 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 and w be a pattern and a word, respectively, over Σ 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 bounded parameters, $Z \in \{\mathrm{E}, \mathrm{NE}\}$ denotes whether we are considering the erasing or non-erasing 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). For example, $[\rho_{|\alpha|_x}^{c_1}, \rho_{|\Sigma|}^{c_2}, \rho_{|h(x)|}^{c_3}]$-[NE, n-inj, tf]-PMV denotes the problem to decide for a given *terminal-free* pattern α with

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

Table 1
Overview of known tractability results.

	E/NE	inj/n-inj	tf/n-tf	$\lvert\mathrm{var}(\alpha)\rvert$	$\lvert w\rvert$	$\lvert\Sigma\rvert$	$\lvert\alpha\rvert_x$	Ref.
1	E, NE	inj, n-inj	tf, n-tf	$\mathbb{N}$	–	–	–	–
2	E, NE	inj, n-inj	tf, n-tf	–	$\mathbb{N}$	–	–	[15]
3	E, NE	inj, n-inj	tf, n-tf	–	–	**1**	–	–
4	E, NE	inj, n-inj	tf, n-tf	–	–	–	**1**	–

Table 2
Overview of known NP-completeness results.

	E/NE	inj/n-inj	tf/n-tf	$\lvert h(x)\rvert$	$\lvert\alpha\rvert_x$	$\lvert\Sigma\rvert$	Ref.
1	NE	n-inj	n-tf	**3**	–	**2**	[4]
2	E, NE	n-inj	tf	**3**	–	**2**	[10]
3	NE	n-inj	tf	**2**	–	**2**	[9]
4	NE	inj	tf	–	–	**2**	[9]
5	NE	inj	tf	**2**	–	–	[9]
6	E	n-inj	n-tf	–	**2**	**2**	[31]

$\max\{|\alpha|_x \mid x \in \mathrm{var}(\alpha)\} \leq c_1$ and a given word $w \in \Sigma^*$ with $|\Sigma| \leq c_2$, whether or not there exists a *non-erasing* substitution h (possibly *non-injective*) that satisfies $\max\{|h(x)| \mid x \in \mathrm{var}(\alpha)\} \leq c_3$ and $h(\alpha) = w$, where c_1, c_2 and c_3 are some constants.

The role of parameter $\rho_{|h(x)|}$ is somewhat special and should therefore be explained in more detail. While bounding any of the parameters $\rho_{|\mathrm{var}(\alpha)|}$, $\rho_{|\alpha|_x}$, $\rho_{|w|}$ or $\rho_{|\Sigma|}$ by a constant constitutes a restriction of the instance comprising of a pattern α and a word w, this is not the case for parameter $\rho_{|h(x)|}$. Therefore, for all variants of PMV where $\rho_{|h(x)|}$ is bounded, we assume that the actual bound of $\rho_{|h(x)|}$ is given as part of the instance. In particular, this means that NP-completeness of a variant of PMV where $\rho_{|h(x)|}$ is upper bounded by constant c is preserved if instead $\rho_{|h(x)|}$ is upper bounded by constant $c' > c$ and, analogously, tractability is preserved if the constant bound is decreased. For the other four parameters this is obviously true. Moreover, since we can always assume that $c \leq |w|$, adding the possible bound c on $\rho_{|h(x)|}$ to the input does not asymptotically increase the input size, regardless of whether this is done in a unary or binary way. This also justifies that, for presentational reasons, in our hardness reductions, we do not explicitly mention the bound c.

The contribution of this paper is to show for each of the 256 individual problems P-$[Z, I, T]$-PMV (either by citing known results from the literature or by giving an original proof) 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. We wish to point out that the constant bounds in our NP-completeness results, although often fairly small, are not necessarily the smallest possible.

3. The hardness of pattern matching with variables

We shall now briefly summarise those variants of PMV, for which NP-completeness or membership in P has already been established. This shall be done in form of tables, in which a numerical entry denotes a constant bound of a parameter, the entry "–" means that a parameter is unrestricted and $\mathbb{N}$ denotes that the result holds for any constant bound. The known tractability results, which are presented in Table 1, are fairly scarce. In order to see that all versions of PMV can be solved efficiently if the number of variables is bounded, we informally describe an obvious and simple brute-force algorithm. For some instance (α, w) of PMV with $m := |\mathrm{var}(\alpha)|$, 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 implies Row 1 of Table 1.

In the non-erasing case, a restriction of $\rho_{|w|}$ implicitly bounds $\rho_{|\mathrm{var}(\alpha)|}$ as well and, thus, all the corresponding versions of the pattern matching problem with variables can be solved efficiently. Moreover, in [15], Geilke and Zilles show 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 [15], 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, which implies Row 2 of Table 1. Finally, it can be easily shown that PMV can be solved in polynomial time if either of the parameters $\rho_{|\Sigma|}$ or $\rho_{|\alpha|_x}$ is bounded by 1 (see, e.g., Schmid [29]).

Next, in Table 2, we briefly summarise those variants of PMV, for which NP-completeness has already been established.

As indicated by Rows 1 to 4 and Row 6, restricting $\rho_{|\Sigma|}$ does not seem to help solve PMV efficiently. In [31] 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, as pointed out by Rows 3 and 4 of Table 1.

Since every version of PMV, for which $\rho_{|\mathrm{var}(\alpha)|}$ or $\rho_{|w|}$ is restricted, can be solved in polynomial time, 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 section, we investigate the complexity of all the variants of the pattern matching problem with variables that are not already covered by Table 2. Most of these variants turn out to be NP-complete. A summary of all the following results can be found in Table 3.

The general proof technique to establish these results is to conduct a polynomial reduction from the problem of checking whether or not a given graph has a so-called perfect code, which shall be defined more formally in the following.

Let $\mathcal{G} = (V, E)$ be a graph with $V := \{t_1, t_2, \ldots, t_n\}$.[2] A vertex s is the *neighbour* of a vertex t if $\{t, s\} \in E$ and the set $N_{\mathcal{G}}[t] := \{s \mid \{t, s\} \in E\} \cup \{t\}$ is called the (*closed*) *neighbourhood* of t. If, for some $k \in \mathbb{N}$, every vertex of $\mathcal{G}$ has exactly k neighbours, then $\mathcal{G}$ is *k-regular*. A *perfect code* for $\mathcal{G}$ is a subset $C \subseteq V$ with the property that, for every $t \in V$, $|N_{\mathcal{G}}[t] \cap C| = 1$. Next, we define the problem to decide whether or not a given 3-regular graph has a perfect code:

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

In [20], Kratochvíl and Křivánek prove the problem 3RPerCode to be NP-complete:

Theorem 2. *(See Kratochvíl and Křivánek [20].) The problem* 3RPerCode *is NP-complete.*

As mentioned above, all the following NP-completeness results can be proved by reducing 3RPerCode to the appropriate variant of PMV. However, since these reductions must cater for the different variants of the problem (i.e., erasing or non-erasing, injective or non-injective, with terminals or terminal-free) as well as for the choice of the restricted parameters, it can be challenging to adapt an NP-completeness proof that works for one version to a seemingly only slightly different version. In order to illustrate these difficulties in a bit more detail, we shall outline the general idea of our reductions.

For the remainder of the entire paper, we define now $\mathcal{G} = (V, E)$ to be a 3-regular graph, where $V := \{t_1, t_2, \ldots, t_n\}$ and, for every i, $1 \leq i \leq n$, let $N_i \subseteq V$ be the neighbourhood of t_i. In order to define a reduction from 3RPerCode to the pattern matching problem with variables, we can simply use a variable x_i for every $t_i \in V$ and represent a neighbourhood as the concatenation of the appropriate variables, e.g., $N_7 = \{t_2, t_6, t_7, t_{12}\}$ is represented by the factor $\beta_7 := x_2x_6x_7x_{12}$. Then, since we want to select exactly one vertex from every N_i, we align each of these β_i with a single occurrence of the symbol $\mathtt{a}$, i.e., we construct a pattern $\alpha := \beta_1\#\beta_2\#\ldots\#\beta_n$ and a word $w := \mathtt{a}\#\mathtt{a}\#\ldots\#\mathtt{a}$, where # is a symbol used as a separator. It can be easily verified that if w matches α, then, since every β_i matches $\mathtt{a}$, in each β_i exactly one variable is mapped to $\mathtt{a}$ and all the others are mapped to the empty word. This directly translates to a perfect code of $\mathcal{G}$. Equivalently, a perfect code in $\mathcal{G}$ implies a substitution that maps α to w.

In our terminology, the above sketched reduction shows the NP-completeness of [E, n-inj, n-tf]-PMV. This is due to the fact that we use the terminal symbol # in the pattern (so the pattern is not terminal-free), we need the ability to erase variables and, as several variables are mapped to the same word $\mathtt{a}$, the mapping of variables is non-injective. A closer look further reveals that in fact $[\rho^1_{|h(x)|}, \rho^4_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, n-tf]-PMV is NP-complete, since we only use two different terminal symbols, every variable is substituted by either $\mathtt{a}$ or the empty word and, as every vertex is contained in exactly 4 neighbourhoods, every variable x_i occurs at most 4 times in α.

It is not straightforward to adapt this reduction to other variants of PMV. For example, if we want to adapt the reduction to the terminal-free case, then we cannot use the symbol # as a separator in α anymore, which plays an important role in the above reduction, since it forces certain parts of the pattern to be matched to certain parts of the word. An idea to make α terminal-free would be to replace every occurrence of # in α by a new variable $y_\#$ that is somehow forced to match the single symbol #. However, this leads to other problems: we lose the bound on the number of occurrences per variable. If we cater for that by using an individual variable per occurrence of #, then we lose the injectivity. Furthermore, if we try to maintain the injectivity by using different separator symbols $\#_i$, then we blow up the alphabet size and, finally, restricting the alphabet size by using binary encodings of the symbols $\#_i$ requires an unbounded length of the substitution words.

Consequently, adapting the simple reduction sketched above to individual reductions from 3RPerCode to *each* NP-complete version of PMV involves considerable technical difficulties.

In the following, in Section 4, we first investigate the complexity of all non-injective variants of PMV for which the question of NP-completeness is still open and then, in Section 5, we take a closer look at the injective variants. In both of these sections, we shall first consider the erasing and then the non-erasing case.

4. The non-injective case

In this section we show that for all non-injective variants of PMV, we can find rather small constants such that if we bound the parameters $\rho_{|h(x)|}$, $\rho_{|\alpha|_x}$ and $\rho_{|\Sigma|}$ by these constants, then the problem is still NP-complete. We first consider the erasing case, for which we can present slightly stronger results, and then we take a closer look at the non-erasing case.

[2] Usually, vertices in a graph are denoted by $v_1, v_2, \ldots$, but we reserve the characters u, v and w for words and factors.

4.1. The erasing case

In the following, we show that the problem $[\rho^1_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, tf]-PMV is NP-complete. This problem can be rephrased in terms of renamings. In this regard, a *renaming* is simply a mapping $h : \Sigma \to \Gamma$, where Σ and Γ are alphabets. The problem to decide for a given word $u_1 u_2 \dots u_n$, $u_i \in \Sigma$, $1 \leq i \leq n$, and a word $v \in \Gamma^*$, whether there exists a renaming $h : \Sigma \to \Gamma$, such that $h(u_1)h(u_2)\dots h(u_n) = v$ can be easily solved in linear time. We simply align u and v (which must have the same length) and check position by position whether the symbol in u (and all its occurrences in u) can be turned into the corresponding one in v in such a way that all the changed symbols agree in both words. The NP-completeness result mentioned above implies that this problem of finding a renaming becomes NP-complete, only if we are looking for an *erasing renaming* $h : \Sigma \to (\Gamma \cup \{\varepsilon\})$, even if $|\Gamma| = \{\mathtt{a}, \mathtt{b}\}$ and every symbol in u has at most 2 occurrences. This is surprising, since the simple formulation of the problem leads us to believe that it might be solvable in polynomial time by some kind of dynamic programming strategy. Especially the fact that every symbol u_i, $1 \leq i \leq n$, has to be either turned into $\mathtt{a}$, $\mathtt{b}$ or the empty word and there exists at most only one other occurrence of the symbol u_i makes it sound simpler than it apparently is.

We can further note that the problem $[\rho^1_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, tf]-PMV is arguably the strongest restricted NP-complete variant of PMV, since any further restriction makes it trivially solvable in polynomial time. More precisely, as mentioned above, the non-erasing version of this problem is solvable in linear time. 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 the parameter $\rho_{|h(x)|}$ is already bounded in the strongest possible sense. Furthermore, we shall show that its injective version is in P as well (see Theorem 9).

Theorem 3. $[\rho^1_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, tf]-PMV *is NP-complete.*

Obviously, the NP-completeness of a terminal-free variant of PMV carries over to the non-terminal-free case as well; thus, Theorem 3 shows that the non-terminal-free case is NP-complete as well. In order to prove Theorem 3, we devise a reduction from 3RPerCode to $[\rho^1_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, n-inj, tf]-PMV. For the sake of a clearer presentation, we shall first define a reduction to the non-terminal-free case, which is then extended to the terminal-free case.

We recall that in all the following constructions $\mathcal{G} = (V, E)$ is a 3-regular graph with $V := \{t_1, t_2, \dots, t_n\}$ and, for every i, $1 \leq i \leq n$, N_i is the neighbourhood of t_i. Since the neighbourhoods play a central role, we shall define them in a more convenient way. For every r, $1 \leq r \leq 4$, we use a mapping $\wp_r : \{1, 2 \dots, n\} \to \{1, 2 \dots, n\}$ that maps an $i \in \{1, 2 \dots, n\}$ to the index of the rth vertex of neighbourhood N_i, i.e., for every i, $1 \leq i \leq n$, $N_i = \{t_{\wp_1(i)}, t_{\wp_2(i)}, t_{\wp_3(i)}, t_{\wp_4(i)}\}$. Obviously, the mappings $\wp_r$, $1 \leq r \leq 4$, imply a certain order on the vertices in the neighbourhoods, but, since our constructions are independent of this actual order, any definition of these mappings is fine.

Let $\Sigma := \{\mathtt{a}, \#, \mathtt{¢}\}$ be a terminal alphabet. We define a mapping Φ_{E} that maps $\mathcal{G}$ to a pattern α and a word $w \in \Sigma^*$, such that, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 2$. We shall later show how this mapping can be modified in such a way that a binary alphabet is sufficient. Since the basic idea of all the following reductions is similar, it is beneficial to invest some more time here to illustrate it in an informal way. We basically use the idea described in Section 3, but in order to bound the number of occurrences per variable by 2, we represent each single occurrence of a vertex in a neighbourhood by an individual variable. More precisely, the occurrence of vertex t_i in the neighbourhood of vertex t_j is represented by variable $x_{i,j}$. This requires a gadget that synchronises the 4 variables x_{i,j_1}, x_{i,j_2}, x_{i,j_3}, x_{i,j_4} that correspond to t_i, i.e., we have to make sure that either all of these 4 variables are mapped to $\mathtt{a}$ or none of them is.

The formal definition of α and w is as follows. For every i, $1 \leq i \leq n$, let

$$\beta_i := x_{\wp_1(i),i}\, x_{\wp_2(i),i}, x_{\wp_3(i),i}, x_{\wp_4(i),i} \text{ and}$$
$$u_i := \mathtt{a}.$$

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

$$\gamma_i := y_i\, ¢\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, ¢\, z_{i,1}\, z_{i,2}\, z_{i,3}\, z_{i,4}\, y'_i \text{ and}$$
$$v_i := ¢¢\mathtt{a}^4¢.$$

Finally, we define

$$\alpha := \beta_1 \# \beta_2 \# \cdots \# \beta_n \# \gamma_1 \# \gamma_2 \# \cdots \# \gamma_n,$$
$$w := u_1 \# u_2 \# \cdots \# u_n \# v_1 \# v_2 \# \cdots \# v_n,$$

and $\Phi_{\mathrm{E}}(\mathcal{G}) := (\alpha, w)$. We note that every variable of form $z_{i,j}$, y_i or y'_i has exactly one occurrence in α. Every variable of form $x_{i,j}$ has one occurrence in each β_j and γ_i. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 2$. The following lemma states that Φ_{E} is in fact a reduction.

Lemma 4. *Let $(\alpha, w) := \Phi_E(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists a substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 1$.*

Proof. We begin with the *if* direction and assume that there exists a substitution h for α, such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 1$. Since in α there are as many occurrences of # as in w, we can conclude that, for every i, $1 \leq i \leq n$, $h(\beta_i) = u_i$ and $h(\gamma_i) = v_i$. Consequently, for every i, $1 \leq i \leq n$, $h(x_{\wp_l(i),i}) \in \{\mathtt{a}, \varepsilon\}$, $1 \leq l \leq 4$, and there is exactly one l, $1 \leq l \leq 4$, with $h(x_{\wp_l(i),i}) = \mathtt{a}$. Furthermore, for every i, $1 \leq i \leq n$,

$$\text{case 1: } h(\mathtt{c}\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathtt{c}) = \mathtt{c}\, \mathtt{a}^4\, \mathtt{c} \text{ or}$$

$$\text{case 2: } h(\mathtt{c}\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathtt{c}) = \mathtt{c}\, \mathtt{c}.$$

This is due to the fact that $h(\mathtt{c}\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathtt{c}) = \mathtt{c}\, \mathtt{c}\, \mathtt{a}^4\, \mathtt{c}$ implies that in $h(\beta_1 \# \beta_2 \# \cdots \# \beta_n)$ there is an occurrence of $\mathtt{c}$, which is a contradiction. If case 2 applies, then $h(x_{i,\wp_l(i)}) = \varepsilon$, $1 \leq l \leq 4$. If, on the other hand, case 1 applies, then we can conclude that $h(x_{i,\wp_l(i)}) = \mathtt{a}$, $1 \leq l \leq 4$.

We can now construct a set C of vertices of $\mathcal{G}$ in the following way. For every i, $1 \leq i \leq n$, and for every $t_j \in N_i$, if $h(x_{j,i}) \neq \varepsilon$, then we add t_j to C. As explained above, for every i, $1 \leq i \leq n$, there exists exactly one l, $1 \leq l \leq 4$, such that $h(x_{\wp_l(i),i}) = \mathtt{a}$, which implies that $|C \cap N_i| \geq 1$. Now assume that, for some i, $|C \cap N_i| \geq 2$, i.e., there exist j, j', $1 \leq j < j' \leq 4$, such that $h(x_{j,i}) = h(x_{j',i}) = \mathtt{a}$. This is a contradiction to $h(\beta_i) = u_i$. Thus, C is a perfect code for $\mathcal{G}$.

Next, we prove the *only if* direction. Let C be a perfect code for $\mathcal{G}$. We now construct a substitution h for α with $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 1$. Since C is a perfect code, for every i, $1 \leq i \leq n$, there is exactly one p_i, $1 \leq p_i \leq n$, such that $\{t_{p_i}\} = N_i \cap C$ (i.e., for every i, $1 \leq i \leq n$, t_{p_i} is the vertex that the perfect code selects from N_i). Hence, for every i, $1 \leq i \leq n$, we define $h(x_{p_i,i}) = \mathtt{a}$ and $h(x) = \varepsilon$ for all the other variables $x_{i,j}$. We note that this directly implies that, for every i, $1 \leq i \leq n$, $h(\beta_i) = u_i$ is satisfied.

Now, for every i, $1 \leq i \leq n$, it only remains to define $h(y_i)$, $h(y_i')$ and $h(z_{i,l})$, l, $1 \leq l \leq 4$, in such a way that $h(\gamma_i) = v_i$ is satisfied. To this end, let i, $1 \leq i \leq n$, be arbitrarily chosen. If $t_i \in C$, then this implies that, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \mathtt{a}$. Thus, $h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}) = \mathtt{a}^4$. Therefore, we define $h(y_i) := \mathtt{c}$, $h(y_i') := \varepsilon$ and $h(z_{i,l}) := \varepsilon$, $1 \leq l \leq 4$. Consequently,

$$\begin{aligned} h(\gamma_i) &= h(y_i)\, \mathtt{c}\, h(x_{i,\wp_1(i)} \cdots x_{i,\wp_4(i)})\, \mathtt{c}\, h(z_{i,1} \cdots z_{i,4})\, h(y_i') \\ &= h(y_i)\, \mathtt{c}\, \mathtt{a}^4\, \mathtt{c}\, h(z_{i,1} \cdots z_{i,4})\, h(y_i') \\ &= \mathtt{c}\, \mathtt{c}\, \mathtt{a}^4\, \mathtt{c} \\ &= v_i. \end{aligned}$$

If, on the other hand, $t_i \notin C$, then this implies that, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \varepsilon$. Thus, $h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}) = \varepsilon$. Therefore, we define $h(y_i) := \varepsilon$, $h(y_i') := \mathtt{c}$ and $h(z_{i,l}) := \mathtt{a}$, $1 \leq l \leq 4$. Consequently,

$$\begin{aligned} h(\gamma_i) &= h(y_i)\, \mathtt{c}\, h(x_{i,\wp_1(i)} \cdots x_{i,\wp_4(i)})\, \mathtt{c}\, h(z_{i,1} \cdots z_{i,4})\, h(y_i') \\ &= h(y_i)\, \mathtt{c}\, \mathtt{c}\, h(z_{i,1} \cdots z_{i,4})\, h(y_i') \\ &= \mathtt{c}\, \mathtt{c}\, \mathtt{a}^4\, \mathtt{c} \\ &= v_i. \end{aligned}$$

We conclude that, for every i, $1 \leq i \leq n$, $h(\gamma_i) = v_i$, which implies $h(\alpha) = w$.

We note that by the above definitions, $|h(x)| \leq 1$, $x \in \mathrm{var}(\alpha)$, is satisfied. This concludes the proof. □

We can observe that in the definition of Φ_E, all occurrences of symbol # can be substituted by $\mathtt{a}\, \mathtt{c}^4\, \mathtt{a}$ and the lemma above still holds. This is due to the fact that then in α there are as many occurrences of factor $\mathtt{a}\, \mathtt{c}^4\, \mathtt{a}$ as in w, which implies that every substitution h that maps α to w must map β_i to u_i and γ_i to v_i, for every i, $1 \leq i \leq n$. Thus, we can conclude in the same way as in the proof above that there must exist a perfect code for the corresponding graph. In the following, let $\overline{\Phi}_E$ denote this refined version of the reduction.

We shall now extend the reduction $\overline{\Phi}_E$ to a reduction Φ_E' that maps a given 3-regular graph $\mathcal{G}$ to a *terminal-free* pattern α and a word $w \in \{\mathtt{c}, \mathtt{a}\}^*$, such that, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 2$. Since variations on the following construction shall also be used for later results, we spend some more time in order to informally explain it. The basic idea is that we modify the reduction for the non-terminal-free case in such a way that all terminal symbols in the pattern, which mainly act as separators, are substituted by special separator-variables. The difficulty is that we have to enforce that a substitution that maps the pattern to the word also maps the separator-variables to the corresponding separator symbols, since then they play the same role as the separators, which allows us to argue in a similar way as before. Especially in the erasing case this can be difficult to achieve, which leads to a more complicated structure of the pattern and the word. Furthermore, since we want to maintain the bound on the number of occurrences per variable, we have to use an individual separator-variable for each occurrence of a separator. We shall now give the formal definitions of this construction.

Let $(\alpha', w') := \overline{\Phi}_{\mathrm{E}}(\mathcal{G})$. First, we observe that there are $2n$ occurrences of $\mathtt{¢}$ in the part $\gamma_1\gamma_2\dots\gamma_n$ of α' and $2n-1$ distinct occurrences of factor $\# = \mathtt{a}\,\mathtt{¢}^4\,\mathtt{a}$ in α'. Hence, there are $m := 2n + 8n - 4$ occurrences of $\mathtt{¢}$ and $m' := 4n-2$ occurrences of $\mathtt{a}$ in α'. We now obtain α'' from α' by substituting, for every i, $1 \le i \le m$, the ith occurrence of $\mathtt{¢}$ by a single occurrence of a new variable $y_{\mathtt{¢},i}$ and, for every i, $1 \le i \le m'$, the ith occurrence of $\mathtt{a}$ by a single occurrence of a new variable $y_{\mathtt{a},i}$. Finally, we define

$$\begin{aligned}\alpha &:= y_{\mathtt{¢},1}y_{\mathtt{¢},2}\dots y_{\mathtt{¢},m}\cdot\widehat{z}\cdot z_{\mathtt{a},1}z_{\mathtt{a},2}\dots z_{\mathtt{a},|\alpha''|}\\ &\qquad y_{\mathtt{a},1}y_{\mathtt{a},2}\dots y_{\mathtt{a},m'}\cdot\widehat{z}\cdot\alpha''z_{\mathtt{a},|\alpha''|}z_{\mathtt{a},|\alpha''|-1}\dots z_{\mathtt{a},1},\\ w &:= \mathtt{¢}^m\,\mathtt{¢}\,\mathtt{a}^{|\alpha''|}\,\mathtt{a}^{m'}\,\mathtt{¢}\,w'\,\mathtt{a}^{|\alpha''|},\end{aligned}$$

and $\Phi'_{\mathrm{E}}(\mathcal{G}) := (\alpha, w)$, where the $z_{\mathtt{a},i}$, $1 \le i \le |\alpha''|$, and $\widehat{z}$ are new variables. We observe that $w \in \{\mathtt{a},\mathtt{¢}\}^*$, for every variable $x \in \mathrm{var}(\alpha')$, $|\alpha'|_x = |\alpha|_x$ and, for every $x \in \mathrm{var}(\alpha) \setminus \mathrm{var}(\alpha')$, $|\alpha|_x = 2$.

The variables $y_{\mathtt{¢},i}$ and $y_{\mathtt{a},i}$ are the separator-variables and the variables $z_{\mathtt{a},i}$ and $\widehat{z}$ are auxiliary variables that are used in order to force the separator-variables to be mapped to the corresponding separator symbols.

Lemma 5. *Let $(\alpha, w) := \Phi'_{\mathrm{E}}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists a substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \le 1$.*

Proof. Let $(\alpha', w') := \overline{\Phi}_{\mathrm{E}}(\mathcal{G})$. By using Lemma 4, we can prove this lemma by showing that there exists a substitution h for α with $|h(x)| \le 1$, $x \in \mathrm{var}(\alpha)$, and $h(\alpha) = w$ if and only if there exists a substitution g for α' with $|h(x)| \le 1$, $x \in \mathrm{var}(\alpha')$, and $g(\alpha') = w'$. The *if* direction is obvious, since if $g(\alpha') = w'$, then $h(\alpha) = w$, where $h(x) := g(x)$, $x \in \mathrm{var}(\alpha')$, and $h(y_{\mathtt{¢},i}) := \mathtt{¢}$, $1 \le i \le m$, $h(\widehat{z}) := \mathtt{¢}$, $h(y_{\mathtt{a},i}) := \mathtt{a}$, $1 \le i \le m'$, $h(z_{\mathtt{a},i}) := \mathtt{a}$, $1 \le i \le |\alpha''|$.

In order to prove the *only if* direction, we assume that there exists a substitution for α with $h(\alpha) = w$, and, furthermore, $|h(x)| \le 1$ for every $x \in \mathrm{var}(\alpha)$ is satisfied, which is crucial for the following argument. First, we observe that if $|h(y_{\mathtt{¢},1}\dots y_{\mathtt{¢},m}\widehat{z})| < m+1$, then there exists an i, $1 \le i \le |\alpha''|$, such that $h(z_{\mathtt{a},1}\dots z_{\mathtt{a},i-1}) = \varepsilon$ and $h(z_{\mathtt{a},i}) = \mathtt{¢}$. This is due to the fact that if all variables $z_{\mathtt{a},i}$, $1 \le i \le |\alpha''|$, are erased, then

$$h(y_{\mathtt{¢},1}\dots y_{\mathtt{¢},m}\widehat{z}\,y_{\mathtt{a},1}\dots y_{\mathtt{a},m'}\widehat{z}\,\alpha'') = w,$$

which is a contradiction, since

$$\begin{aligned}|y_{\mathtt{¢},1}\dots y_{\mathtt{¢},m}\widehat{z}\,y_{\mathtt{a},1}\dots y_{\mathtt{a},m'}\widehat{z}\,\alpha''| &= m+1+m'+1+|\alpha''|\\ &< m+1+m'+|\alpha''|+1+|w'|+|\alpha''| = |w|.\end{aligned}$$

We wish to point out here, that this is only a contradiction because $|h(x)| \le 1$, $x \in \mathrm{var}(\alpha)$. However, if, for some i, $1 \le i \le |\alpha''|$, $h(z_{\mathtt{a},1}\dots z_{\mathtt{a},i-1}) = \varepsilon$ and $h(z_{\mathtt{a},i}) = \mathtt{¢}$, then $h(\alpha)$ ends with symbol $\mathtt{¢}$, which is a contradiction, since w ends with $\mathtt{a}$. Consequently, none of the variables $y_{\mathtt{¢},i}$ is erased, which means, since they are mapped to a word of length at most 1, $h(y_{\mathtt{¢},i}) = \mathtt{¢}$, $1 \le i \le m$, and $h(\widehat{z}) = \mathtt{¢}$. This means that $h(z_{\mathtt{a},1}\dots z_{\mathtt{a},|\alpha''|}\,y_{\mathtt{a},1}\dots y_{\mathtt{a},m'}) = \mathtt{a}^{|\alpha''|}\,\mathtt{a}^{m'}$; thus, $h(z_{\mathtt{a},i}) = \mathtt{a}$, $1 \le i \le |\alpha''|$, and $h(y_{\mathtt{a},i}) = \mathtt{a}$, $1 \le i \le m'$. Thus, $h(\alpha'') = w'$ is implied, where h maps every variable $y_{\mathtt{¢},i}$, $1 \le i \le m$, to $\mathtt{¢}$ and every variable $y_{\mathtt{a},i}$, $1 \le i \le m'$, to $\mathtt{a}$. Therefore, we can conclude that $h(\alpha') = w'$ holds as well. □

It is straightforward to see that the mapping Φ'_{E} can be computed in polynomial time; thus, from Lemma 5, we can now conclude Theorem 3. For all our further reductions it will be easy to see that they can be computed in polynomial time, so we shall not explicitly mention this anymore.

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 PMV, where the length of the substitution words is bounded by 1. However, in [5] only non-erasing and injective substitutions are considered and with Theorem 3 we can conclude that Baker's pattern matching problem most likely cannot be solved in polynomial time if it is generalised to the erasing and non-injective case.

4.2. The non-erasing case

As mentioned in Section 3, Clifford et al. show in [9] that the non-erasing, 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. 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 these results for the case that parameter $\rho_{|\alpha|_x}$ is bounded by 3 and the case where $\rho_{|\alpha|_x}$ is bounded by 2 is left open (recall that in the non-erasing case, if $\rho_{|\alpha|_x}$ is bounded by 1, then PMV can be easily solved in polynomial time). Furthermore, for the terminal-free case we have slightly larger constant bounds on the parameters $\rho_{|\alpha|_x}$ and $\rho_{|\Sigma|}$. Therefore, in the non-erasing case we can show an analogue of Theorem 3, but the constants are a bit larger.

Theorem 6.

1. $[\rho^3_{|h(x)|}, \rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[NE, n-inj, n-tf]-PMV *is NP-complete.*
2. $[\rho^3_{|h(x)|}, \rho^3_{|\alpha|_x}, \rho^4_{|\Sigma|}]$-[NE, n-inj, tf]-PMV *is NP-complete.*

The proof of Theorem 6 proceeds similar as the one for Theorem 3, i.e., we first give a reduction that proves statement 1 of Theorem 6 and then we extend this reduction to the terminal-free case. However, adapting the reduction to the terminal-free case increases the constant bounds of 2 for parameters $\rho_{|\alpha|_x}$ and $\rho_{|\Sigma|}$ to 3 and 4, respectively.

We shall now extend the reduction Φ_{E} to a reduction Φ_{NE}, which works for the non-erasing case. The basic idea is the following. We use the same alphabet $\Sigma = \{\mathtt{a}, \mathtt{¢}, \#\}$ as for Φ_{E}, but instead of representing whether or not a vertex is selected as a member of the perfect code by mapping the corresponding variable to either $\mathtt{a}$ or ε, it is mapped to either $\mathtt{aa}$ or $\mathtt{a}$, i.e., the β_i are mapped to $\mathtt{a}^5$. This means that all the variables that represent different occurrences of the same vertex in different neighbourhoods must be synchronised by mapping them to either $\mathtt{a}^4$ or $\mathtt{a}^8$, which requires more sophisticated synchronisation gadgets (γ_i, v_i) with more auxiliary variables in γ_i and a more complicated form of v_i.

We now formally define Φ_{NE} by pointing out in which regards it differs from Φ_{E}. The β_i parts are defined in exactly the same way and $u_i := \mathtt{a}^5$, $1 \le i \le n$. Furthermore, for every i, $1 \le i \le n$,

$$\gamma_i := z_{i,0}\, z_{i,1}\, z_{i,2}\, z_{i,3}\, z_{i,4}\, \mathtt{¢}\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathtt{¢}\, z'_{i,0}\, z'_{i,1}\, z'_{i,2}\, z'_{i,3} z'_{i,4} \text{ and}$$
$$v_i := \mathtt{¢}^5\, \mathtt{¢}\, \mathtt{a}^8\, \mathtt{¢}\, \mathtt{a}^4\, \mathtt{¢}\, \mathtt{¢}^5.$$

Finally, $\Phi_{\mathrm{NE}}(\mathcal{G}) := (\alpha, w)$, where α and w are constructed by concatenating the β_i, γ_i, u_i and v_i with occurrences of # as separators in the same way as it is done in the definition of Φ_{E}. Obviously, every variable of form $z_{i,j}$ or $z'_{i,j}$ has exactly one occurrence in α. Every variable of form $x_{i,j}$ has one occurrence in each β_j and γ_i. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \le 2$ still holds.

Lemma 7. *Let $(\alpha, w) := \Phi_{\mathrm{NE}}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists a non-erasing substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \le 3$.*

Proof. We begin with the *if* direction and assume that there exists a non-erasing substitution h for α with $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \le 3$. Since in α there are as many occurrences of # as in w, we can conclude that, for every i, $1 \le i \le n$, $h(\beta_i) = u_i$ and $h(\gamma_i) = v_i$. Thus, for every i, $1 \le i \le n$, $h(\beta_i) = \mathtt{a}^5$, which directly implies that, for every l, $1 \le l \le 4$, $h(x_{\wp_l(i),i}) \in \{\mathtt{a}, \mathtt{aa}\}$.

Now we assume that, for some i, $1 \le i \le n$, $h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)})$ is not of the form $\mathtt{a}^8$ or $\mathtt{a}^4$. This implies that there is a factor $\mathtt{¢}\,\mathtt{a}^p\,\mathtt{¢}$ in $h(\gamma_i)$, where $5 \le p \le 7$. This is a contradiction, since $h(\gamma_i) = v_i$ and there is no such factor in $v_i = \mathtt{¢}^5\,\mathtt{¢}\,\mathtt{a}^8\,\mathtt{¢}\,\mathtt{a}^4\,\mathtt{¢}\,\mathtt{¢}^5$. Consequently, for every i, $1 \le i \le n$, either $h(x_{i,\wp_l(i)}) = \mathtt{aa}$ for every l, $1 \le l \le 4$, or $h(x_{i,\wp_l(i)}) = \mathtt{a}$ for every l, $1 \le l \le 4$. By collecting all t_i with $h(x_{i,\wp_l(i)}) = \mathtt{aa}$, l, $1 \le l \le 4$, this translates into a perfect code in the same way as in the proof of Lemma 4, which concludes the *if* direction.

Next, we prove the *only if* direction. To this end, we assume that there exists a perfect code C for $\mathcal{G}$. We construct now a non-erasing substitution h for α with $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \le 3$. Now let i, $1 \le i \le n$, be arbitrarily chosen. If $t_i \in C$, then, for every l, $1 \le l \le 4$, we define $h(x_{i,\wp_l(i)}) := \mathtt{aa}$ and if, on the other hand, $t_i \notin C$, then, for every l, $1 \le l \le 4$, we define $h(x_{i,\wp_l(i)}) := \mathtt{a}$. Since, for every i, $1 \le i \le n$, $|C \cap N_i| = 1$, we can conclude that $h(\beta_i) = u_i$. It only remains to define, for every i, $1 \le i \le n$, and every l, $0 \le l \le 4$, $h(z_{i,l})$ and $h(z'_{i,l})$ in such a way that $h(\gamma_i) = v_i$ is satisfied. This is achieved in the following way.

Let i with $1 \le i \le n$ be arbitrarily chosen. If $h(x_{i,\wp_l(i)}) = \mathtt{aa}$, $1 \le l \le 4$, referred to as case 1, then we define h such that $h(z_{i,0}\, z_{i,1} \cdots z_{i,4}) = \mathtt{¢}^5$ and $h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) = \mathtt{a}^4\,\mathtt{¢}\,\mathtt{¢}^5$. If, on the other hand, $h(x_{i,\wp_l(i)}) = \mathtt{a}$, $1 \le l \le 4$, referred to as case 2, then we define h such that $h(z_{i,0}\, z_{i,1} \cdots z_{i,4}) = \mathtt{¢}^5\,\mathtt{¢}\,\mathtt{a}^8$ and $h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) = \mathtt{¢}^5$. We note that the definitions above can be done in such a way that, for every l, $0 \le l \le 4$, $|h(z_{i,l})| \le 3$ and $|h(z'_{i,l})| \le 3$. We conclude that in case 1

$$\begin{aligned} h(\gamma_i) &= h(z_{i,0}\, z_{i,1} \cdots z_{i,4})\, \mathtt{¢}\, h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)})\, \mathtt{¢}\, h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) \\ &= h(z_{i,0}\, z_{i,1} \cdots z_{i,4})\, \mathtt{¢}\, \mathtt{a}^8\, \mathtt{¢}\, h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) \\ &= \mathtt{¢}^5\, \mathtt{¢}\, \mathtt{a}^8\, \mathtt{¢}\, \mathtt{a}^4\, \mathtt{¢}\, \mathtt{¢}^5 \end{aligned}$$

and in case 2

$$\begin{aligned} h(\gamma_i) &= h(z_{i,0}\, z_{i,1} \cdots z_{i,4})\, \mathtt{¢}\, h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)})\, \mathtt{¢}\, h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) \\ &= h(z_{i,0}\, z_{i,1} \cdots z_{i,4})\, \mathtt{¢}\, \mathtt{a}^4\, \mathtt{¢}\, h(z'_{i,0}\, z'_{i,1} \cdots z'_{i,4}) \\ &= \mathtt{¢}^5\, \mathtt{¢}\, \mathtt{a}^8\, \mathtt{¢}\, \mathtt{a}^4\, \mathtt{¢}\, \mathtt{¢}^5. \end{aligned}$$

Consequently, for every i, $1 \le i \le n$, $h(\gamma_i) = v_i$ is satisfied, which implies that $h(\alpha) = w$.

We observe that, for every i, $1 \leq i \leq n$ and every l, $1 \leq l \leq 4$, $h(x_{i,\wp l(i)}) \in \{\mathtt{aa}, \mathtt{a}\}$ and, as explained above, for every l, $0 \leq l \leq 4$, $|h(z_{i,j})| \leq 3$ and $|h(z'_{i,j})| \leq 3$. This implies that, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 3$, which concludes the proof. □

In a way similar as before, we can observe that if we encode the symbol # by $\mathtt{a}\,\mathtt{¢}^7\,\mathtt{a}$, then Lemma 7 still holds. This proves Theorem 6.1.

Next, we show how Φ_{NE} can be extended to the terminal-free case. To this end, similar as in the erasing case, we apply the idea of substituting the terminal symbols in the pattern by separator-variables. However, in the non-erasing case it is easier to enforce the separator-variables to be mapped to the corresponding separator symbols, since we do not have to take care of the situation that separator-variables are erased.

Let $(\alpha', w') := \Phi_{\mathrm{NE}}(\mathcal{G})$, where Φ_{NE} is the mapping that uses the alphabet $\{\mathtt{a}, \#, ¢\}$, i.e., the reduction without the encoding of symbol # by $\mathtt{a}\,\mathtt{¢}^7\,\mathtt{a}$. We now define a terminal-free pattern α and a word w by modifying α' and w'. Firstly, all occurrences of symbols ¢ or # in α' are substituted by new individual symbols (i.e., we use $|\alpha'|_¢ + |\alpha'|_\#$ new symbols) and for every new such symbol b, by $\pi(b)$ we denote the original symbol that has been substituted by b. We now assume that after this modification in α' there occur exactly the symbols $¢_1, ¢_2, \ldots, ¢_m$. Then, for every j, $1 \leq j \leq m$, we substitute each $¢_j$ by a new variable $y_{¢_j}$ and we denote this modified version of α' by α''. Next, we define

$$\alpha := y_\% \, y_\% \, y_{¢_1} \, y_{¢_2} \cdots y_{¢_m} \, y_\% \, \alpha'',$$
$$w := \%\%\pi(¢_1)\,\pi(¢_2)\cdots\pi(¢_m)\,\%\,w',$$

and $\Phi'_{\mathrm{NE}}(\mathcal{G}) := (\alpha, w)$, where % is a new symbol and $y_\%$ is a new variable. We observe that α is terminal-free, for every $x \in (\mathrm{var}(\alpha') \cap \mathrm{var}(\alpha))$, $|\alpha'|_x = |\alpha|_x$ and, for every $x \in (\mathrm{var}(\alpha) \setminus \mathrm{var}(\alpha'))$, $|\alpha|_x \leq 3$. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 3$. Furthermore, $w \in \{\mathtt{a}, \#, ¢, \%\}^*$. The following lemma shows that Φ'_{NE} is a valid reduction.

Lemma 8. *Let $(\alpha, w) := \Phi'_{\mathrm{NE}}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists a non-erasing substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 3$.*

Proof. Let $(\alpha', w') := \Phi_{\mathrm{NE}}(\mathcal{G})$. Again, we prove the lemma by using Lemma 7, i.e., we show that there exists a substitution h for α' such that $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha')$, $|h(x)| \leq 3$, if and only if there exists a substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 3$. The *only if* direction of this statement follows trivially from the fact that if $h(\alpha') = w'$, then h can be extended to g with $g(\alpha) = w$ by mapping the separator-variables to the corresponding separator symbols.

It remains to prove the *if* direction. To this end, we assume that there exists a substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 3$. We observe that $g(y_\%) = \%$ holds, since otherwise $g(y_\%) = \%\%\cdots$, which implies that there are at least 6 occurrences of symbol % in $g(\alpha)$, a contradiction to $g(\alpha) = w$. In particular, this also means that $g(\alpha'') = w'$ and $g(y_{¢_1}\, y_{¢_2} \cdots y_{¢_m}) = \pi(¢_1)\,\pi(¢_2)\cdots\pi(¢_m)$; thus, for every i, $1 \leq i \leq m$, $g(y_{¢_i}) = \pi(¢_i)$. This directly implies $g(\alpha') = w'$. □

From Lemma 8, the second statement of Theorem 6 follows, which concludes the proof of Theorem 6.

5. The injective case

In the remainder of the main part of this paper, we shall focus on the injective variants of PMV. A main difference between the injective and the non-injective cases is that in the injective case, bounding $\rho_{|\Sigma|}$ and $\rho_{|h(x)|}$ already yields polynomial time solvability (see Theorem 9 below), whereas the non-injective case remains NP-complete, even if we additionally bound $\rho_{|\alpha|_x}$ (as stated in Section 4). 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 non-erasing 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 [15].

Theorem 9. *Let $k_1, k_2 \in \mathbb{N}$, let $Z \in \{\mathrm{E}, \mathrm{NE}\}$ and let $T \in \{\mathrm{tf}, \mathrm{n\text{-}tf}\}$. The problem $[\rho^{k_1}_{|\Sigma|}, \rho^{k_2}_{|h(x)|}]$-$[Z, \mathrm{inj}, T]$-PMV is in P.*

Proof. Since the case $Z = \mathrm{NE}$ can be proved analogously to the case $Z = \mathrm{E}$, we shall only prove the latter.

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 non-erasing 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 less than 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 $S \subseteq \operatorname{var}(\alpha)$, 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 $x \in \operatorname{var}(\alpha)$, if and only if there exists a set $S \subseteq \operatorname{var}(\alpha)$ with $|S| \leq k'$ and S satisfying the condition $(*)$.

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

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

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

Theorem 9 implies that if P ≠ NP, then an analogue of Theorems 3 and 6 does not exist in the injective case. However, it remains to investigate all the injective variants where either $\rho_{|h(x)|}$ and $\rho_{|\alpha|_x}$ are bounded, but $\rho_{|\Sigma|}$ is unbounded or $\rho_{|\Sigma|}$ and $\rho_{|\alpha|_x}$ are bounded, but $\rho_{|h(x)|}$ is unbounded. For all these variants, we can prove NP-completeness, but for some of the results we need more complicated reductions than the ones used for the non-injective case. Furthermore, the specific constant bounds in our results are larger, especially for the parameter $\rho_{|h(x)|}$. Again, we start with the erasing case and take care of the non-erasing case later on.

5.1. The erasing case

We first show the NP-completeness for the situation that the parameters $\rho_{|h(x)|}$ and $\rho_{|\alpha|_x}$ are bounded.

Theorem 10.

1. $[\rho^5_{|h(x)|}, \rho^2_{|\alpha|_x}]$-[E, inj, n-tf]-PMV *is NP-complete.*
2. $[\rho^5_{|h(x)|}, \rho^4_{|\alpha|_x}]$-[E, inj, tf]-PMV *is NP-complete.*

As before, our proof strategy consists again in adapting the reduction Φ_{E} (see Lemma 4) first to the injective case and then to the injective and terminal-free case. In the reduction Φ_{E}, all variables $x_{i,j}$ that are not erased are mapped to the same word $\mathtt{a}$, which is the reason why this reduction does not work for the injective case. Therefore, for the injective case, we use several individual symbols $\mathtt{a}_i$, $1 \leq i \leq n$, in order to ensure that the substitution that maps the pattern to the word is injective. More precisely, we use the alphabet $\Sigma := \{\mathtt{a}_i, \mathtt{c}_i, \#_j \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$, the size of which depends on the size of the graph $\mathcal{G}$, i.e., the alphabet size is not bounded by a constant, which we can afford since the parameter $\rho_{|\Sigma|}$ is unbounded. The parts β_i that represent the neighbourhoods are defined as usual, but, for every i, $1 \leq i \leq n$, the corresponding factors u_i equal the symbol $\mathtt{a}_i$ instead of $\mathtt{a}$. In the v_i, instead of having 4 occurrences of $\mathtt{a}$, we have to list all 4 different symbols that correspond to the 4 neighbourhoods in which t_i occurs. More precisely, for every i, $1 \leq i \leq n$, we define

$$\gamma_i := z_i\, \mathtt{c}_i\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathtt{c}_i\, z'_i \text{ and}$$

$$v_i := \mathtt{c}_i\, \mathtt{c}_i\, \mathtt{a}_{\wp_1(i)}\, \mathtt{a}_{\wp_2(i)}\, \mathtt{a}_{\wp_3(i)}\, \mathtt{a}_{\wp_4(i)}\, \mathtt{c}_i.$$

Finally, we construct α and w as usual, but we use individual separators, i.e., we define

$$\alpha := \beta_1 \#_1 \beta_2 \#_2 \cdots \#_{n-1} \beta_n \#_n \gamma_1 \#_{n+1} \gamma_2 \#_{n+2} \cdots \#_{2n-1} \gamma_n,$$

$$w := u_1 \#_1 u_2 \#_2 \cdots \#_{n-1} u_n \#_n v_1 \#_{n+1} v_2 \#_{n+2} \cdots \#_{2n-1} v_n,$$

and $\Psi_{\mathrm{E},1}(\mathcal{G}) := (\alpha, w)$.

The use of individual separators $\#_i$, $1 \leq i \leq 2n-1$, is not really necessary here, but it helps us to extend this reduction to the terminal-free case later on. We note that 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 $t_j \in N_i$, variable $x_{j,i}$ has exactly one occurrence in β_i and exactly one occurrence in γ_j. Thus, for every $x \in \operatorname{var}(\alpha)$, $|\alpha|_x \leq 2$.

Lemma 11. *Let $(\alpha, w) := \Psi_{\mathrm{E},1}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an* E*-injective substitution h for α such that $h(\alpha) = w$ and, for every $x \in \operatorname{var}(\alpha)$, $|h(x)| \leq 5$.*

Proof. We begin with the *if* direction and assume that there exists an E-injective substitution h for α, such that $h(\alpha) = w$ and, for every $x \in \operatorname{var}(\alpha)$, $|h(x)| \leq 5$. By the use of the symbols $\#_i$, $1 \leq i \leq 2n-1$, we can conclude that, for every i,

$1 \leq i \leq n$, $h(\beta_i) = u_i$ and $h(\gamma_i) = v_i$. Consequently, for every i, $1 \leq i \leq n$ there is exactly one l, $1 \leq l \leq 4$, with $h(x_{\wp_l(i),i}) = \mathtt{a}_i$ and $h(x_{\wp_{l'}(i),i}) = \varepsilon$, $1 \leq l' \leq 4$, $l \neq l'$. Furthermore, since, for every i, $1 \leq i \leq n$, the factor $\mathfrak{c}_i x_{i,\wp_1(i)} x_{i,\wp_2(i)} x_{i,\wp_3(i)} x_{i,\wp_4(i)} \mathfrak{c}_i$ of γ_i must be mapped to a factor of v_i that is delimited by occurrences of $\mathfrak{c}_i$, we can conclude

$$\text{case 1: } h(\mathfrak{c}_i\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathfrak{c}_i) = \mathfrak{c}_i\, \mathtt{a}_{\wp_1(i)}\, \mathtt{a}_{\wp_2(i)}\, \mathtt{a}_{\wp_3(i)}\, \mathtt{a}_{\wp_4(i)}\, \mathfrak{c}_i \text{ or}$$

$$\text{case 2: } h(\mathfrak{c}_i\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathfrak{c}_i) = \mathfrak{c}_i\, \mathfrak{c}_i.$$

This is due to the fact that if $h(\mathfrak{c}_i\, x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}\, \mathfrak{c}_i)$ equals the word $\mathfrak{c}_i\, \mathfrak{c}_i\, \mathtt{a}_{\wp_1(i)}\, \mathtt{a}_{\wp_2(i)}\, \mathtt{a}_{\wp_3(i)}\, \mathtt{a}_{\wp_4(i)}\, \mathfrak{c}_i$, then in $h(\beta_1 \#_1 \beta_2 \#_2 \cdots \#_{n-1} \beta_n)$ there is an occurrence of $\mathfrak{c}_i$, which is a contradiction. If case 2 applies, then $h(x_{i,\wp_l(i)}) = \varepsilon$, $1 \leq l \leq 4$. If, on the other hand, case 1 applies, then we can conclude that $h(x_{i,\wp_l(i)}) = \mathtt{a}_{\wp_l(i)}$, $1 \leq l \leq 4$. From this observation, we can conclude in the same way as in the proof of Lemma 4 that we can construct a perfect code for $\mathcal{G}$.

In order to prove the *only if* direction, let C be a perfect code for $\mathcal{G}$. We now construct an E-injective substitution h for α with $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 5$. Since C is a perfect code, for every i, $1 \leq i \leq n$, there is exactly one p_i, $1 \leq p_i \leq n$, such that $t_{p_i} \in N_i \cap C$ (i.e., for every i, $1 \leq i \leq n$, t_{p_i} is the vertex that is chosen from N_i as a member of the perfect code). Hence, for every i, $1 \leq i \leq n$, we define $h(x_{p_i,i}) = \mathtt{a}_i$ and $h(x) = \varepsilon$ for all the other variables $x_{i,j}$. We note that this directly implies that, for every i, $1 \leq i \leq n$, $h(\beta_i) = u_i$ is satisfied.

Now it only remains, for every i, $1 \leq i \leq n$, to define $h(z_i)$ and $h(z_i')$ in such a way that $h(\gamma_i) = v_i$ is satisfied. To this end, let i, $1 \leq i \leq n$, be arbitrarily chosen. If $t_i \in C$, then this implies that, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \mathtt{a}_{\wp_l(i)}$. Thus, $h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}) = \mathtt{a}_{\wp_1(i)}\, \mathtt{a}_{\wp_2(i)}\, \mathtt{a}_{\wp_3(i)}\, \mathtt{a}_{\wp_4(i)}$. Therefore, we define $h(z_i) = \mathfrak{c}_i$ and $h(z_i') = \varepsilon$. It can be easily verified that this implies $h(\gamma_i) = v_i$. If, on the other hand, $t_i \notin C$, then, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \varepsilon$. Thus, $h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}) = \varepsilon$. Hence, we define $h(z_i) = \varepsilon$ and $h(z_i') = \mathtt{a}_{\wp_1(i)}\, \mathtt{a}_{\wp_2(i)}\, \mathtt{a}_{\wp_3(i)}\, \mathtt{a}_{\wp_4(i)}\, \mathfrak{c}_i$ and again this means that $h(\gamma_i) = v_i$ is satisfied. We conclude that, for every i, $1 \leq i \leq n$, $h(\gamma_i) = v_i$, which implies $h(\alpha) = w$.

Next, we can observe that, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 5$. It remains to prove that h is E-injective, i.e., each two variables that are not erased are mapped to different words. For all the variables $x_{i,j}$ this is obviously true. Next, we observe that, for every i, $1 \leq i \leq n$, if $h(z_i)$ or $h(z_i')$ is not the empty word, then $h(z_i)$ or $h(z_i')$, respectively, contains $\mathfrak{c}_i$. This implies that, for every i, j, $1 \leq i < j \leq n$, and every $\widehat{z} \in \{z_i, z_i'\}$, $\widetilde{z} \in \{z_j, z_j'\}$, if $h(\widehat{z}) \neq \varepsilon$ and $h(\widetilde{z}) \neq \varepsilon$, then $h(\widehat{z}) \neq h(\widetilde{z})$. Furthermore, for the same reasons, for every i, i', $1 \leq i, i' \leq n$, every j with $t_i \in N_j$ and every $z \in \{z_{i'}, z_{i'}'\}$, if $h(x_{i,j}) \neq \varepsilon$ and $h(z) \neq \varepsilon$, then $h(x_{i,j}) \neq h(z)$. Finally, since, for every i, $1 \leq i \leq n$, either $h(z_i)$ or $h(z_i')$ is empty, we can conclude that h is E-injective. □

Lemma 11 implies the first statement of Theorem 10.

Next, we extend the reduction $\Psi_{\mathrm{E},1}$ to a reduction $\Psi'_{\mathrm{E},1}$ that works for the terminal-free case. The general strategy is again to introduce separator-variables, which, due to the required injectivity, need to be mapped to individual separator symbols. This is the reason why we already use such individual separator symbols in the definition of $\Psi_{\mathrm{E},1}$. Now let α' and w' be the pattern and the word given by the mapping $\Psi_{\mathrm{E},1}$. Firstly, for every i, $1 \leq i \leq 2n-1$, we substitute every occurrence of $\#_i$ in α' by a new variable $y_{\#_i}$ and, for every i, $1 \leq i \leq n$, we substitute every occurrence of $\mathfrak{c}_i$ in α' by a new variable $y_{\mathfrak{c}_i}$. Thus, we obtain a terminal-free pattern, which we denote by α''. Next, we define

$$\alpha := (y_{\#_1}\, y_{\#_2} \cdots y_{\#_{2n-1}}\, y_{\mathfrak{c}_1}\, y_{\mathfrak{c}_2} \cdots y_{\mathfrak{c}_n})^2\, \alpha''\, \alpha'',$$
$$w := (\#_1\, \#_2 \cdots \#_{2n-1}\, \mathfrak{c}_1\, \mathfrak{c}_2 \cdots \mathfrak{c}_n)^2\, w'\, w',$$

and $\Psi'_{\mathrm{E},1}(\mathcal{G}) := (\alpha, w)$. We note that all the variables in α that also occur in α' have twice as many occurrences in α as in α' and all the variables $\{y_{\mathfrak{c}_i}, y_{\#_j} \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$ have at most 4 occurrences. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 4$.

Lemma 12. *Let $(\alpha, w) := \Psi'_{\mathrm{E},1}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an* E*-injective substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 5$.*

Proof. Let $(\alpha', w') := \Psi_{\mathrm{E},1}(\mathcal{G})$. By using Lemma 11, we can prove the statement of the lemma by showing that there exists an E-injective substitution h for α' such that $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha')$, $|h(x)| \leq 5$, if and only if there exists an E-injective substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 5$. The *only if* direction follows in the usual way, i.e., we can extend h such that the separator variables are mapped to the corresponding separator symbols.

It remains to prove the *if* direction. To this end, we assume that there exists an E-injective substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 5$. If $g(y_{\#_i}) := \#_i$, $1 \leq i \leq 2n-1$, and $g(y_{\mathfrak{c}_i}) := \mathfrak{c}_i$, $1 \leq i \leq n$, then $g(\alpha') = w'$ holds as well. Hence, in the following, we assume that this is not satisfied.

If, for some $z \in \{y_{\mathfrak{c}_i}, y_{\#_j} \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$, $|g(z)| \geq 2$, then $g(z)$ contains a factor $u \in \{\#_i\, \#_{i+1}, \#_{2n-1}\, \mathfrak{c}_1, \mathfrak{c}_i\, \mathfrak{c}_{i+1}, \mathfrak{c}_n\, \#_1, \mathfrak{c}_n\, \mathtt{a}_1\}$. In α there are at least 3 occurrences of z, but there are at most 2 occurrences of factor u in w. Thus, we can conclude that, for every $z \in \{y_{\mathfrak{c}_i}, y_{\#_j} \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$, $|g(z)| \leq 1$. Next, we observe that if, for some $z \in \{y_{\mathfrak{c}_i}, y_{\#_j} \mid 1 \leq i \leq n, 1 \leq j \leq 2n-1\}$, $g(z) = \varepsilon$, then $|g((y_{\#_1}\, y_{\#_2} \cdots y_{\#_{2n-1}}\, y_{\mathfrak{c}_1}\, y_{\mathfrak{c}_2} \cdots y_{\mathfrak{c}_n})^2)| < 2(2n-1+n)$. This means that $g(\alpha''\alpha'') = u\, w'\, w'$, where u is a non-empty suffix of $(\#_1\, \#_2 \cdots \#_{2n-1}\, \mathfrak{c}_1\, \mathfrak{c}_2 \cdots \mathfrak{c}_n)^2$. This is a contradiction, since $g(\alpha''\alpha'')$ is a square, but $u\, w'\, w'$ is not a square. Consequently, we can conclude that, for every i, $1 \leq i \leq 2n-1$, $g(y_{\#_i}) = \#_i$ and, for every i, $1 \leq i \leq n$, $g(y_{\mathfrak{c}_i}) = \mathfrak{c}_i$. This particularly implies that $g(\alpha') = w'$, which concludes the proof. □

From Lemma 12 the second statement of Theorem 10 follows, which concludes the proof of Theorem 10.

The question is now whether or not, in the injective and erasing case, the NP-completeness is preserved if instead of $\rho_{|h(x)|}$, the parameter $\rho_{|\Sigma|}$ is bounded by a constant. We can answer this question in the affirmative.

Theorem 13.

1. $[\rho^2_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[E, inj, n-tf]-PMV *is NP-complete.*
2. $[\rho^9_{|\alpha|_x}, \rho^5_{|\Sigma|}]$-[E, inj, tf]-PMV *is NP-complete.*

Both statements of Theorem 13 can be proved by modifying the reduction $\Psi_{E,1}$ in such a way that the many different symbols $\{\mathtt{a}_i, \mathfrak{c}_i, \#_j \mid 1 \le i \le n, 1 \le j \le 2n-1\}$ used in $\Psi_{E,1}$ are encoded as long binary words, i.e., we trade in the bound of the length of the substitution words for a bound on the alphabet size. When proving that the thus obtained reduction is correct, we can generally proceed as in the proof of Lemma 11, but we have to be a bit more careful, since we also have to show that it is impossible for a substitution to map variables to proper factors of the binary code words for the original symbols. More precisely, the reduction $\Psi_{E,2}$ is defined in the following way. For every i, $1 \le i \le n$, we substitute all occurrences of $\mathtt{a}_i$ in α and w by $\mathtt{a}\mathtt{b}^i\mathtt{a}$, all occurrences of $\mathfrak{c}_i$ by $\mathtt{a}\mathtt{b}^{n+i}\mathtt{a}$ and all occurrences of $\#_j$ by $\mathtt{a}\mathtt{b}^{2n+j}\mathtt{a}$. Let α and w be the pattern and the word produced by this mapping, which is denoted by $\Psi_{E,2}$. We observe that $w \in \{\mathtt{a},\mathtt{b}\}^*$ and, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \le 2$.

Lemma 14. *Let $(\alpha, w) := \Psi_{E,2}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an* E*-injective substitution h for α such that $h(\alpha) = w$.*

Proof. The *only if* direction follows from the proof of Lemma 11. We let $(\alpha', w') = \Psi_{E,1}(\mathcal{G})$. If there exists a perfect code in (G), then there exists a substitution g with $g(\alpha') = w'$, according to Lemma 11. This implies that $h(\alpha) = w$, where h is obtained from g by replacing all the symbols $\mathtt{a}_i, \mathfrak{c}_i$, $1 \le i \le n$, and $\#_j$, $1 \le j \le 2n-1$, in the images of g by the corresponding code words defined above. Furthermore, the E-injectivity of g is preserved.

It remains to prove the *if* direction. To this end, we first note that if there exists an E-injective substitution h for α, such that $h(\alpha) = w$, then, since the binary encoded separators work in the same way as before, we can conclude that, for every i, $1 \le i \le n$, $h(\beta_i) = u_i$ and $h(\gamma_i) = v_i$. This particularly implies that, for every i, $1 \le i \le n$,

$$h(x_{\wp_1(i),i}\, x_{\wp_2(i),i}\, x_{\wp_3(i),i}\, x_{\wp_4(i),i}) = \mathtt{a}\mathtt{b}^i\mathtt{a}, \text{ and } *_1$$

$$h(x_{i,\wp_1(i)}\, x_{i,\wp_2(i)}\, x_{i,\wp_3(i)}\, x_{i,\wp_4(i)}) \in \{\mathtt{a}\mathtt{b}^{\wp_1(i)}\mathtt{a}\,\mathtt{a}\mathtt{b}^{\wp_2(i)}\mathtt{a}\,\mathtt{a}\mathtt{b}^{\wp_3(i)}\mathtt{a}\,\mathtt{a}\mathtt{b}^{\wp_4(i)}\mathtt{a}, \varepsilon\}. \; *_2$$

Moreover, since $*_1$ holds, for every l, $1 \le l \le 4$, if $h(x_{\wp_l(i),i}) \notin \{\mathtt{a}\mathtt{b}^i\mathtt{a}, \varepsilon\}$, then $h(x_{\wp_l(i),i})$ is non-empty and contains 0 or 1 occurrence of $\mathtt{a}$, which is a contradiction to $*_2$. Thus, for every l, $1 \le l \le 4$, $h(x_{\wp_l(i),i}) \in \{\mathtt{a}\mathtt{b}^i\mathtt{a}, \varepsilon\}$ and, in conjunction with $*_1$ this implies that there exists exactly one such l, $1 \le l \le 4$, with $h(x_{\wp_l(i),i}) = \mathtt{a}\mathtt{b}^i\mathtt{a}$ and $h(x_{\wp_{l'}(i),i}) = \varepsilon$, $1 \le l' \le 4$, $l \ne l'$.

Now, we can conclude just in the same way as in the proof of Lemma 11 that there exists a perfect code for $\mathcal{G}$. □

From Lemma 14 we can conclude the first statement of Theorem 13.

In order to prove statement 2 of Theorem 13, we do not modify $\Psi_{E,2}$, but again $\Psi_{E,1}$ from above. The general idea is that we first use a binary encoding for the different terminal symbols $\{\mathtt{a}_i, \mathfrak{c}_i, \#_j \mid 1 \le i \le n, 1 \le j \le 2n-1\}$, as it is done in the definition of $\Psi_{E,2}$, and then we replace in the pattern all the code words by separator-variables. This is necessary, since using separator-variables for single symbols would conflict with the injectivity. The difficulty is now that we have to construct the pattern and the word in such a way that the separator-variables are necessarily mapped to the corresponding code words, instead of single symbols. This requires a more complicated form of the pattern and the word, compared to the one used in the definition of $\Psi'_{E,1}$. The formal definition of this construction is as follows.

Let $(\widehat{\alpha}, \widehat{w}) := \Psi_{E,1}$. For every i, $1 \le i \le n$, we substitute all occurrences of $\mathtt{a}_i$ in $\widehat{\alpha}$ and $\widehat{w}$ by $\mathtt{a}\mathtt{b}^i\mathtt{a}$. Then, we substitute in $\widehat{\alpha}$ every individual occurrence of a terminal $\#_j$, $1 \le j \le 2n-1$, and $\mathfrak{c}_i$, $1 \le i \le n$, by a new terminal symbol and we do the same with respect to the word $\widehat{w}$. The thus obtained pattern and word are denoted by α' and w', respectively. We assume now that after this transformation, in α' there occur exactly the terminals $\mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_m$. Then, we substitute each $\mathfrak{c}_j$ in α' by a new variable $y_{\mathfrak{c}_j}$ and each $\mathfrak{c}_j$ in w' by $\overline{\mathfrak{c}}_j := \mathfrak{c}\#^j\mathfrak{c}$. We denote these modified versions of α' and w' by α'' and w'', respectively. Next, we define

$$\delta := y_\% y_\% y_{\mathfrak{c}_1} y_\% y_{\mathfrak{c}_2} y_{\mathfrak{c}_1} y_{\mathfrak{c}_3} y_{\mathfrak{c}_2} y_{\mathfrak{c}_4} y_{\mathfrak{c}_3} \cdots y_{\mathfrak{c}_{m-3}} y_{\mathfrak{c}_{m-1}} y_{\mathfrak{c}_{m-2}} y_{\mathfrak{c}_m} y_{\mathfrak{c}_{m-1}},$$

$$w_\delta := \%\%\overline{\mathfrak{c}}_1\%\overline{\mathfrak{c}}_2\overline{\mathfrak{c}}_1\overline{\mathfrak{c}}_3\overline{\mathfrak{c}}_2\overline{\mathfrak{c}}_4\overline{\mathfrak{c}}_3\cdots\overline{\mathfrak{c}}_{m-3}\overline{\mathfrak{c}}_{m-1}\overline{\mathfrak{c}}_{m-2}\overline{\mathfrak{c}}_m\overline{\mathfrak{c}}_{m-1},$$

where % is a new symbol and $y_\%$ is a new variable. It is necessary to illustrate the structure of δ and w_δ. If, for example, $m = 6$, then

$$\delta := y_\% y_\% y_{\mathfrak{c}_1} y_\% y_{\mathfrak{c}_2} y_{\mathfrak{c}_1} y_{\mathfrak{c}_3} y_{\mathfrak{c}_2} y_{\mathfrak{c}_4} y_{\mathfrak{c}_3} y_{\mathfrak{c}_5} y_{\mathfrak{c}_4} y_{\mathfrak{c}_6} y_{\mathfrak{c}_5},$$

$$w_\delta := \%\%\overline{\mathfrak{c}}_1\%\overline{\mathfrak{c}}_2\overline{\mathfrak{c}}_1\overline{\mathfrak{c}}_3\overline{\mathfrak{c}}_2\overline{\mathfrak{c}}_4\overline{\mathfrak{c}}_3\overline{\mathfrak{c}}_5\overline{\mathfrak{c}}_4\overline{\mathfrak{c}}_6\overline{\mathfrak{c}}_5.$$

The idea of δ and w_δ is the following. We first make sure that $y_\%$ is mapped to %, which then forces $y_{\mathfrak{c}_1}$ to be mapped to $\bar{\mathfrak{c}}_1$, since $y_{\mathfrak{c}_1}$ and $\bar{\mathfrak{c}}_1$ are delimited by two occurrences of $y_\%$ and %, respectively. Analogously, $y_{\mathfrak{c}_2}$ is then forced to be mapped to $\bar{\mathfrak{c}}_2$ and by repeating this argument, we can conclude that all the separator variables are mapped to the desired code words. We conclude the definition of the reduction by

$$\alpha := (\delta)^2 (y_{\mathfrak{c}_1} y_{\mathfrak{c}_2} \cdots y_{\mathfrak{c}_m})^3 (\alpha'')^2,$$
$$w := (w_\delta)^2 (\bar{\mathfrak{c}}_1 \bar{\mathfrak{c}}_2 \cdots \bar{\mathfrak{c}}_m)^3 (w'')^2,$$

and $\Psi'_{\mathrm{E},2}(\mathcal{G}) := (\alpha, w)$. We observe that $w \in \{\mathtt{a}, \mathtt{b}, \#, \mathfrak{c}, \%\}^*$, α is terminal-free, for every $x \in (\mathrm{var}(\alpha') \cap \mathrm{var}(\alpha))$, $|\alpha'|_x = 2|\alpha|_x$ and, for every $x \in (\mathrm{var}(\alpha) \setminus \mathrm{var}(\alpha'))$, $|\alpha|_x \leq 9$. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 9$.

Lemma 15. *Let $(\alpha, w) := \Psi'_{\mathrm{E},2}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an* E*-injective substitution h for α such that $h(\alpha) = w$.*

Proof. Let $(\alpha', w') := \Psi_{\mathrm{E},1}(\mathcal{G})$. We prove the statement by using Lemma 11, i.e., we show that there exists an E-injective substitution h for α' such that $h(\alpha') = w'$ if and only if there exists an E-injective substitution g for α such that $g(\alpha) = w$. The *only if* direction follows trivially, i.e., we can extend h such that the separator variables are mapped to the corresponding code words defined above.

It remains to prove the *if* direction. To this end, we assume that there exists an E-injective substitution g for α such that $g(\alpha) = w$. If

$$g((\delta)^2 (y_{\mathfrak{c}_1} \cdots y_{\mathfrak{c}_m})^3) \neq (w_\delta)^2 (\bar{\mathfrak{c}}_1 \cdots \bar{\mathfrak{c}}_m)^3,$$

then either $g((\alpha'')^2)$ is a proper suffix of $(w'')^2$ or $g((\alpha'')^2)$ is of the form $\widehat{w}\,(w'')^2$, where $\widehat{w}$ is a non-empty suffix of $(w_\delta)^2 (\bar{\mathfrak{c}}_1 \bar{\mathfrak{c}}_2 \cdots \bar{\mathfrak{c}}_m)^3$. It can be verified that every proper suffix of $(w'')^2$ is not a square and, for every non-empty suffix $\widehat{w}$ of $(w_\delta)^2 (\bar{\mathfrak{c}}_1 \bar{\mathfrak{c}}_2 \cdots \bar{\mathfrak{c}}_m)^3$, $\widehat{w}\,(w'')^2$ is not a square either. Hence, since $g((\alpha'')^2)$ is a square, we can conclude that

$$g((\delta)^2 (y_{\mathfrak{c}_1} \cdots y_{\mathfrak{c}_m})^3) = (w_\delta)^2 (\bar{\mathfrak{c}}_1 \cdots \bar{\mathfrak{c}}_m)^3.$$

Furthermore, $g(y_\%) = \%\%\cdots$ is not possible, since this implies that there are at least 12 occurrences of % in $g(\alpha)$. If, on the other hand, $g(y_\%) = \varepsilon$, then either,

1. for some i, $1 \leq i \leq m$, $g(y_{\mathfrak{c}_i}) = \cdots\%\cdots$, or,
2. for some i, j, $1 \leq i, j \leq n$, $(x_{i,j}) = \cdots\%\cdots$.

Case 1 is not possible, since there are 8 occurrences of $y_{\mathfrak{c}_i}$ in α, but only 6 occurrences of % in w. If we have case 2, then, for every i, $1 \leq i \leq m$, $g(y_{\mathfrak{c}_i}) = \varepsilon$, which implies that $g(\alpha''\alpha'') = w$. This is a contradiction, since w is not a square. Consequently, $g(y_\%) = \%$, which particularly implies that $g(y_{\mathfrak{c}_1}) = \overline{\mathfrak{c}_1}$. We note that there are exactly as many occurrences of $y_{\mathfrak{c}_1}$ in

$$(\delta)^2 (y_{\mathfrak{c}_1} \cdots y_{\mathfrak{c}_m})^3$$

as there are occurrences of $\bar{\mathfrak{c}}_1$ in

$$(w_\delta)^2 (\bar{\mathfrak{c}}_1 \bar{\mathfrak{c}}_2 \cdots \bar{\mathfrak{c}}_m)^3.$$

This implies that $g(y_{\mathfrak{c}_2}) = \overline{\mathfrak{c}_2}$ and, in a similar way, we can show that, for every i, $2 \leq i \leq m-2$, $g(y_{\mathfrak{c}_i}) = \bar{\mathfrak{c}}_i$ in conjunction with $g(y_{\mathfrak{c}_{i+1}}) = \bar{\mathfrak{c}}_{i+1}$ implies $g(y_{\mathfrak{c}_{i+2}}) = \bar{\mathfrak{c}}_{i+2}$. Therefore, for every i, $1 \leq i \leq m$, $g(y_{\mathfrak{c}_i}) = \bar{\mathfrak{c}}_i$, which implies that $g(\alpha') = w'$. □

From Lemma 15 we can conclude the second statement of Theorem 13, which concludes the proof of Theorem 13.

This solves all the open questions with respect to the injective and erasing case and in the following, we take care of the non-erasing case.

5.2. The non-erasing case

It remains to investigate the question of whether or not the NP-completeness results presented in Theorems 10 and 13 also hold for the non-erasing case. As in the previous section, we start now with the situation that $\rho_{|h(x)|}$ and $\rho_{|\alpha|_x}$ are the parameters that are bounded.

Theorem 16. $[\rho^{36}_{|h(x)|}, \rho^{3}_{|\alpha|_x}]$-[NE, inj, tf]-PMV *is NP-complete.*

We shall prove Theorem 16 by first defining a reduction from 3RPerCode to $[\rho^{36}_{|h(x)|}, \rho^{3}_{|\alpha|_x}]$-[NE, inj, n-tf]-PMV and then we extend this reduction to the terminal-free case.

The task of finding suitable reductions for proving Theorem 16 leads to difficulties that are particular with respect to the injective and non-erasing case. In order to illustrate these problems in a bit more detail, we recall the reduction Φ_{NE} that has been used to reduce 3RPerCode to the non-injective and non-erasing variant of PMV. In this reduction, every variable is mapped to either $\mathtt{a}$ or $\mathtt{aa}$, where the image $\mathtt{aa}$ means that the corresponding vertex is chosen as a member for the perfect code. Here, in order to cater for the injectivity, we map $x_{j,i}$ to $\mathtt{a}_i$ to represent that vertex t_j is chosen from neighbourhood N_i, which is similar to the reduction $\Psi_{\mathrm{E},1}$. However, if t_j is not chosen from neighbourhood N_i, then, since we cannot erase it, we have to map it to a special *false symbol* $\mathtt{b}$. Furthermore, the injectivity condition dictates that all these false symbols must be distinct symbols, which means that $x_{\wp_1(i),i}x_{\wp_2(i),i}x_{\wp_3(i),i}x_{\wp_4(i),i}$, i.e., the pattern representation of N_i, must be mapped to 3 false symbols and $\mathtt{a}_i$. Since we do not know which vertex should be mapped to $\mathtt{a}_i$, we have to allow $x_{\wp_1(i),i}x_{\wp_2(i),i}x_{\wp_3(i),i}x_{\wp_4(i),i}$ to be mapped to either $\mathtt{a}_i\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3}$, $\mathtt{b}_{i,1}\mathtt{a}_i\mathtt{b}_{i,2}\mathtt{b}_{i,3}$, $\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{a}_i\mathtt{b}_{i,3}$ or $\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3}\mathtt{a}_i$, where the $\mathtt{b}_{i,1}$, $\mathtt{b}_{i,2}$ and $\mathtt{b}_{i,3}$ are the individual false symbols and the situation that a variable $x_{\wp_l(i),i}$, $1 \le l \le 4$, is mapped to $\mathtt{b}_{i,r}$, $1 \le r \le 3$, means that $t_{\wp_l(i)}$ is the rth vertex in N_i, that is *not* chosen. This version of the gadget that selects the vertices from the neighbourhoods causes problems for the synchronisation gadget. More precisely, in the synchronisation gadget, if vertex i has been selected, then $x_{i,\wp_1(i)}x_{i,\wp_2(i)}x_{i,\wp_3(i)}x_{i,\wp_4(i)}$ should be mapped to $a_{\wp_1(i)}a_{\wp_2(i)}a_{\wp_3(i)}a_{\wp_4(i)}$. If, on the other hand, vertex i has not been selected, then $x_{i,\wp_1(i)}x_{i,\wp_2(i)}x_{i,\wp_3(i)}x_{i,\wp_4(i)}$ has to be mapped to a factor $y_1y_2y_3y_4$, where $y_r \in \{\mathtt{b}_{\wp_r(i),1}, \mathtt{b}_{\wp_r(i),2}, \mathtt{b}_{\wp_r(i),3}\}$, but the actual values of these y_r, $1 \le r \le 4$, depend on which vertex has been chosen from neighbourhood $N_{\wp_r(i)}$ and, thus, this cannot be explicitly represented in the structure of the pattern and the word. Therefore, in the synchronisation gadget, we have to cater for all these possibilities of mapping the variables $x_{i,\wp_r(i)}$ to symbols from $\{\mathtt{b}_{\wp_r(i),1}, \mathtt{b}_{\wp_r(i),2}, \mathtt{b}_{\wp_r(i),3}\}$, $1 \le r \le 4$.

We are now ready to define a mapping $\Psi_{\mathrm{NE},1}$ that maps $\mathcal{G}$ to a pattern α and a word w over $\Sigma := \{\mathtt{a}_i, \mathtt{b}_{i,j}, \mathtt{e}_{i,j'}, \mathtt{c}_i, \$_i, \# \mid 1 \le i \le n, 1 \le j \le 3, 1 \le j' \le 5\}$, such that, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \le 2$. In the following construction, we shall use many variables that have just one occurrence in the whole pattern α. In order to improve the readability, we denote *all* these single occurrence variables by the symbol z, keeping in mind that any occurrence of symbol z represents an individual variable with just one occurrence. Furthermore, we shall call these variables the *z-variables*.

For every i, $1 \le i \le n$,

$$\beta_i := z\,\mathtt{c}_i\,x_{\wp_1(i),i}\,x_{\wp_2(i),i}\,x_{\wp_3(i),i}\,x_{\wp_4(i),i}\,\mathtt{c}_i\,z$$

$$u_i := \mathtt{c}_i\,\mathtt{c}_i\,\mathtt{a}_i\,\mathtt{b}_{i,1}\,\mathtt{b}_{i,2}\,\mathtt{b}_{i,3}\,\mathtt{c}_i\,\mathtt{b}_{i,1}\,\mathtt{a}_i\,\mathtt{b}_{i,2}\,\mathtt{b}_{i,3}\,\mathtt{c}_i\,\mathtt{b}_{i,1}\,\mathtt{b}_{i,2}\,\mathtt{a}_i\,\mathtt{b}_{i,3}\,\mathtt{c}_i\,\mathtt{b}_{i,1}\,\mathtt{b}_{i,2}\,\mathtt{b}_{i,3}\,\mathtt{a}_i\,\mathtt{c}_i\,\mathtt{c}_i.$$

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

$$\gamma_i := z\,\$_i\,(z\,x_{i,\wp_1(i)}\,z\,x_{i,\wp_2(i)}\,z\,x_{i,\wp_3(i)}\,z\,x_{i,\wp_4(i)}\,z)^2\,\$_i\,z \text{ and}$$

$$\nu_i := \$_i\,\$_i\,(\widehat{\nu}_i)^2\,\$_i\,(\overline{\nu}_i)^2\,\$_i\,\$_i$$

and

$$\widehat{\nu}_i := \mathtt{e}_{i,1}\mathtt{b}_{\wp_1(i),1}\mathtt{b}_{\wp_1(i),2}\mathtt{b}_{\wp_1(i),3}\mathtt{e}_{i,2}\mathtt{b}_{\wp_2(i),1}\mathtt{b}_{\wp_2(i),2}\mathtt{b}_{\wp_2(i),3}$$
$$\mathtt{e}_{i,3}\mathtt{b}_{\wp_3(i),1}\mathtt{b}_{\wp_3(i),2}\mathtt{b}_{\wp_3(i),3}\mathtt{e}_{i,4}\mathtt{b}_{\wp_4(i),1}\mathtt{b}_{\wp_4(i),2}\mathtt{b}_{\wp_4(i),3}\mathtt{e}_{i,5},$$

$$\overline{\nu}_i := \mathtt{e}_{i,1}\,\mathtt{a}_{\wp_1(i)}\,\mathtt{e}_{i,2}\,\mathtt{a}_{\wp_2(i)}\,\mathtt{e}_{i,3}\,\mathtt{a}_{\wp_3(i)}\,\mathtt{e}_{i,4}\,\mathtt{a}_{\wp_4(i)}\,\mathtt{e}_{i,5}.$$

Finally, we define

$$\alpha := \beta_1 \# \beta_2 \# \cdots \# \beta_n \# \gamma_1 \# \gamma_2 \# \cdots \# \gamma_n,$$

$$w := u_1 \# u_2 \# \cdots \# u_n \# \nu_1 \# \nu_2 \# \cdots \# \nu_n,$$

and $\Psi_{\mathrm{NE},1}(\mathcal{G}) := (\alpha, w)$. By definition, each of the z-variables has only one occurrence in α. For every i, $1 \le i \le n$, and every j with $t_j \in N_i$, variable $x_{j,i}$ has exactly one occurrence in β_i and exactly two occurrences in γ_j. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \le 3$.

Lemma 17. *Let $(\alpha, w) := \Psi_{\mathrm{NE},1}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an injective non-erasing substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \le 36$.*

Proof. We begin with the *if* direction and assume that there exists an injective non-erasing substitution h for α, such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(\alpha)| \le 36$. Since in α there are as many occurrences of # as in w, we can conclude that, for every i, $1 \le i \le n$, $h(\beta_i) = u_i$ and $h(\gamma_i) = \nu_i$. Obviously, the factor $x_{\wp_1(i),i}\,x_{\wp_2(i),i}\,x_{\wp_3(i),i}\,x_{\wp_4(i),i}$ of β_i must be mapped to a factor of u_i that is delimited by occurrences of $\mathtt{c}_i$. Moreover, $h(x_{\wp_1(i),i}\,x_{\wp_2(i),i}\,x_{\wp_3(i),i}\,x_{\wp_4(i),i})$ cannot contain occurrences

of $\mathfrak{c}_i$, since this implies that there are occurrences of $\mathfrak{c}_i$ in some $h(\gamma_j)$ as well. Consequently,

$$h(x_{\wp_1(i),i}\, x_{\wp_2(i),i}\, x_{\wp_3(i),i}\, x_{\wp_4(i),i}) \in \{\mathtt{a}_i\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{a}_i\mathtt{b}_{i,2}\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{a}_i\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3}\mathtt{a}_i\}.$$

Thus, for every i, $1 \leq i \leq n$, there exists exactly one l, $1 \leq l \leq 4$, such that $h(x_{\wp_l(i),i}) = \mathtt{a}_i$ and $h(x_{\wp_{l'}(i),i}) \in \{\mathtt{b}_{i,1}, \mathtt{b}_{i,2}, \mathtt{b}_{i,3}\}$, $1 \leq l' \leq 4$, $l \neq l'$. Now let i, $1 \leq i \leq n$, be arbitrarily chosen. We recall that $h(\gamma_i) = v_i$ and we observe that $h(\$_i(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z)^2\$_i) = \$_i(\widehat{v}_i)^2\$_i(\overline{v}_i)^2\$_i$ is not possible, since $(\widehat{v}_i)^2\$_i(\overline{v}_i)^2$ is not a square. Therefore,

$$\$_i(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z)^2\$_i$$

is mapped to either $\$_i(\widehat{v}_i)^2\$_i$ or $\$_i(\overline{v}_i)^2\$_i$, which, in particular, implies that $h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) \in \{\widehat{v}_i, \overline{v}_i\}$. We recall that, as demonstrated above, every variable $x_{i,\wp_l(i)}$, $1 \leq l' \leq 4$, is either mapped to $\mathtt{a}_{\wp_l(i)}$ or to some symbol in $\{\mathtt{b}_{\wp_l(i),1}, \mathtt{b}_{\wp_l(i),2}, \mathtt{b}_{\wp_l(i),3}\}$. Hence, if, for some l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \mathtt{a}_{\wp_l(i)}$, then

$$h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \overline{v}_i$$

and $h(x_{i,\wp_{l'}(i)}) = \mathtt{a}_{\wp_{l'}(i)}$, for every l', $1 \leq l' \leq 4$. Similarly, if $h(x_{i,\wp_l(i)}) \in \{\mathtt{b}_{\wp_l(i),1}, \mathtt{b}_{\wp_l(i),2}, \mathtt{b}_{\wp_l(i),3}\}$, then

$$h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \widehat{v}_i$$

and $h(x_{i,\wp_{l'}(i)}) \in \{\mathtt{b}_{\wp_{l'}(i),1}, \mathtt{b}_{\wp_{l'}(i),2}, \mathtt{b}_{\wp_{l'}(i),3}\}$, for every l', $1 \leq l' \leq 4$. From these observations we can conclude in the usual way that there exists a perfect code for C.

Next, we prove the *only if* direction. Let C be a perfect code for $\mathcal{G}$. We now construct an injective non-erasing substitution h for α with $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 36$. Since C is a perfect code, for every i, $1 \leq i \leq n$, there is exactly one p_i, $1 \leq p_i \leq n$, such that $\{t_{p_i}\} = N_i \cap C$ (i.e., for every i, $1 \leq i \leq n$, t_{p_i} is the vertex that is chosen from N_i as a member of the perfect code). Hence, for every i, $1 \leq i \leq n$, we define $h(x_{p_i,i}) = \mathtt{a}_i$. We define

- $h(x_{\wp_2(i),i}) := \mathtt{b}_{i,1}$, $h(x_{\wp_3(i),i}) = \mathtt{b}_{i,2}$ and $h(x_{\wp_4(i),i}) := \mathtt{b}_{i,3}$, if $p_i = \wp_1(i)$,
- $h(x_{\wp_1(i),i}) := \mathtt{b}_{i,1}$, $h(x_{\wp_3(i),i}) = \mathtt{b}_{i,2}$ and $h(x_{\wp_4(i),i}) := \mathtt{b}_{i,3}$, if $p_i = \wp_2(i)$,
- $h(x_{\wp_1(i),i}) := \mathtt{b}_{i,1}$, $h(x_{\wp_2(i),i}) = \mathtt{b}_{i,2}$ and $h(x_{\wp_4(i),i}) := \mathtt{b}_{i,3}$, if $p_i = \wp_3(i)$,
- $h(x_{\wp_1(i),i}) := \mathtt{b}_{i,1}$, $h(x_{\wp_2(i),i}) = \mathtt{b}_{i,2}$ and $h(x_{\wp_3(i),i}) := \mathtt{b}_{i,3}$, if $p_i = \wp_4(i)$.

Thus,

$$h(x_{\wp_1(i),i}x_{\wp_2(i),i}x_{\wp_3(i),i}x_{\wp_4(i),i}) \in \{\mathtt{a}_i\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{a}_i\mathtt{b}_{i,2}\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{a}_i\mathtt{b}_{i,3},\ \mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3}\mathtt{a}_i\}$$

is satisfied. We note that we can now define the images of the z-variables in β_i in such a way that $h(\beta_i) = u_i$ and these images have a length of at most $|\mathtt{b}_{i,1}\mathtt{a}_i\mathtt{b}_{i,2}\mathtt{b}_{i,3}\mathfrak{c}_i\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{a}_i\mathtt{b}_{i,3}\mathfrak{c}_i\mathtt{b}_{i,1}\mathtt{b}_{i,2}\mathtt{b}_{i,3}\mathtt{a}_i\mathfrak{c}_i\mathfrak{c}_i| = 16$. Hence, for every i, $1 \leq i \leq n$, $h(\beta_i) = u_i$ is satisfied.

Now, for every i, $1 \leq i \leq n$, it only remains to define the z-variables in γ_i in such a way that $h(\gamma_i) = v_i$ is satisfied. To this end, let i, $1 \leq i \leq n$, be arbitrarily chosen. If $t_i \in C$, then this implies that, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) = \mathtt{a}_{\wp_l(i)}$. Thus, we can map the 5 z-variables of $zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z$ to the symbols e_{i_1}, e_{i_2}, e_{i_3}, e_{i_4} and e_{i_5}, respectively, which implies $h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \overline{v}_i$. Furthermore, the substitution words for the remaining z-variables can be defined such that $h(\gamma_i) = v_i$ and all the images of z-variables have a length of at most $|\$_i\,\$_i\,(\widehat{v}_i)^2| = 36$.

If, on the other hand, $t_i \notin C$, then this implies that, for every l, $1 \leq l \leq 4$, $h(x_{i,\wp_l(i)}) \in \{\mathtt{b}_{\wp_l(i),1}, \mathtt{b}_{\wp_l(i),2}, \mathtt{b}_{\wp_l(i),3}\}$. Thus, we can define the substitution words for the z-variables in $zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z$ in such a way that $h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \widehat{v}_i$ is satisfied, which also means that the substitution words for the remaining z-variables can be defined such that $h(\gamma_i) = v_i$ and, furthermore, all the images of z-variables have a length of at most $|(\overline{v}_i)^2\,\$_i\,\$_i| = 20$. We conclude that, for every i, $1 \leq i \leq n$, $h(\gamma_i) = v_i$, which implies $h(\alpha) = w$.

Finally, we have to show that h is injective. To this end, we first note that according to the definitions above, for all i, i', $1 \leq i, i' \leq n$, and for all j, j' with $t_j \in N_i$ and $t_{j'} \in N_{i'}$, if $i \neq i'$ or $j \neq j'$, then $h(x_{j,i}) \neq h(x_{j',i'})$. Furthermore, for every i, $1 \leq i \leq n$, the two z-variables in β_i are mapped to two different words that both contain at least one occurrence of symbol $\mathfrak{c}_i$. For every i, $1 \leq i \leq n$, if $h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \overline{v}_i$, then the z-variables in $zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z$ are mapped to $e_{i,1}, e_{i,2}, \ldots, e_{i,5}$, respectively, and the two remaining z-variables of γ_i are mapped to different words that both

contain $\$_i$. If, on the other hand, $h(zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z) = \widehat{v}_i$, then it can be easily verified that, for every l, $1 \leq l \leq 5$, the lth z-variable of $zx_{i,\wp_1(i)}zx_{i,\wp_2(i)}zx_{i,\wp_3(i)}zx_{i,\wp_4(i)}z$ must be mapped to a word that contains the occurrence of symbol $\mathtt{e}_{i,l}$ and, again, the two remaining z-variables of γ_i are mapped to different words that both contain $\$_i$. These considerations demonstrate that h is an injective substitution, which concludes the proof. □

In order to extend the reduction $\Psi_{\mathrm{NE},1}$ to a reduction $\Psi'_{\mathrm{NE},1}$, which works for the terminal-free case, we apply an idea that is similar to the one used to extend Φ_{NE} to Φ'_{NE}. Let α' and w' be the pattern and word produced by $\Psi_{\mathrm{NE},1}$. In both α' and w', we substitute every individual occurrence of some symbol in $\{\mathtt{c}_i, \$_i, \# \mid 1 \leq i \leq n\}$ by a new symbol. We now assume that after this modification in α' there occur exactly the terminal symbols $\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_m$. Then, we substitute each $\mathtt{c}_j$ by a new variable $y_{\mathtt{c}_j}$ and we denote these modified versions of α' and w' by α'' and w'', respectively. Next, we define

$$\alpha := y_{\%}\, y_{\%}\, y_{\mathtt{c}_1}\, y_{\mathtt{c}_2} \cdots y_{\mathtt{c}_m}\, y_{\%}\, \alpha'',$$
$$w := \%\%\,\mathtt{c}_1\,\mathtt{c}_2 \cdots \mathtt{c}_m\,\%\, w'',$$

and $\Psi'_{\mathrm{NE},1}(\mathcal{G}) := (\alpha, w)$, where $\%$ is a new symbol and $y_{\%}$ is a new variable. We observe that α is terminal-free, for every $x \in (\mathrm{var}(\alpha') \cap \mathrm{var}(\alpha))$, $|\alpha'|_x = |\alpha|_x$ and, for every $x \in (\mathrm{var}(\alpha) \setminus \mathrm{var}(\alpha'))$, $|\alpha|_x \leq 3$. Thus, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x \leq 3$.

Lemma 18. *Let $(\alpha, w) := \Psi'_{\mathrm{NE},1}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an injective non-erasing substitution h for α such that $h(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 36$.*

Since the proof of Lemma 18 is analogous to the one for Lemma 8, we just give a brief sketch. If there is a perfect code in $\mathcal{G}$, then it follows from Lemma 17 that there exists an injective non-erasing substitution h for α' such that $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha')$, $|h(x)| \leq 36$, where $(\alpha', w') := \Psi_{\mathrm{NE},1}(\mathcal{G})$. This implies that there also exists an injective non-erasing substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 36$, since we can extend h by mapping all the separator-variables to the corresponding separator symbols. Moreover, if there exists an injective non-erasing substitution g for α such that $g(\alpha) = w$ and, for every $x \in \mathrm{var}(\alpha)$, $|g(x)| \leq 36$, then, by the same argument used in the proof of Lemma 8, g must map $y_{\%}$ to $\%$ and every $y_{\mathtt{c}_i}$ to $\mathtt{c}_i$. Lemma 18 directly implies Theorem 16.

As in the erasing case, we shall now show that the NP-completeness is preserved if instead of $\rho_{|h(x)|}$, the parameter $\rho_{|\Sigma|}$ is bounded by a constant.

Theorem 19. $[\rho^3_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[NE, inj, tf]-PMV *is NP-complete.*

Analogously to the erasing case, we prove Theorem 19 by encoding different symbols by words over a constant alphabet. It turns out that in the non-erasing case, this can be done in a simple way.

Let α and w be the pattern and the word given by $\Psi'_{\mathrm{NE},1}(\mathcal{G})$. For the sake of convenience, let $\Sigma := \{\mathtt{c}_1, \mathtt{c}_2, \ldots, \mathtt{c}_k\}$ be the symbols in w, let $\alpha = y_1\, y_2 \cdots y_n$, $y_i \in \mathrm{var}(\alpha)$, $1 \leq i \leq n$, and let $w = d_1\, d_2 \cdots d_m$, $d_i \in \Sigma$, $1 \leq i \leq m$. Firstly, we define $\alpha' := y_1\,\mathtt{b}\,y_2\,\mathtt{b}\cdots\mathtt{b}\,y_n$ and $w' := d_1\,\mathtt{b}\,d_2\,\mathtt{b}\cdots\mathtt{b}\,d_m$, where $\mathtt{b} \notin \Sigma$. Then, $\alpha'' := \alpha'$ and $w'' := \pi(w')$, where π is a morphism $(\Sigma \cup \{\mathtt{b}\})^* \to \{\mathtt{a}, \mathtt{b}\}^*$ defined by $\pi(\mathtt{c}_i) := \mathtt{a}^i$, $1 \leq i \leq k$, and $\pi(\mathtt{b}) := \mathtt{b}$. Finally, we define $\Psi_{\mathrm{NE},2}(\mathcal{G}) := (\alpha'', w'')$. We note that, for every $x \in \mathrm{var}(\alpha)$, $|\alpha|_x = |\alpha''|_x$, and $w'' \in \{\mathtt{a}, \mathtt{b}\}^*$; thus, $|\alpha|_x \leq 3$, $x \in \mathrm{var}(\alpha)$.

Lemma 20. *Let $(\alpha, w) := \Psi_{\mathrm{NE},2}(\mathcal{G})$. There is a perfect code in $\mathcal{G}$ if and only if there exists an injective non-erasing substitution h for α such that $h(\alpha) = w$.*

Proof. Let $(\alpha', w') := \Psi'_{\mathrm{NE},1}(\mathcal{G})$. We prove the lemma by applying Lemma 18, i.e., we show that there exists an injective non-erasing substitution h for α' such that $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 36$, if and only if there exists an injective non-erasing substitution g for α such that $g(\alpha) = w$.

We shall first prove the *only if* direction of this statement. To this end, we assume that there exists an injective non-erasing substitution h with $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 36$. We define an injective non-erasing substitution g for α in the following way. For every $x \in \mathrm{var}(\alpha')$ with $h(x) = \mathtt{c}_{j_1}\,\mathtt{c}_{j_2} \cdots \mathtt{c}_{j_l}$, we define $g(x) := \pi(\mathtt{c}_{j_1})\,\mathtt{b}\,\pi(\mathtt{c}_{j_2})\,\mathtt{b}\cdots\mathtt{b}\,\pi(\mathtt{c}_{j_l})$, where π is the morphism defined above. It can be verified with moderate effort that $g(\alpha) = w$. From the fact that h is injective it follows that g is injective as well.

Next, we prove the *if* direction of the statement. Let g be an injective non-erasing substitution for α with $g(\alpha) = w$. We recall that in α every occurrence of a variable is delimited by two occurrences of the symbol $\mathtt{b}$. This implies that if a variable is substituted by a word that starts or ends with $\mathtt{b}$, then in $h(\alpha)$ the factor $\mathtt{bb}$ occurs, which is a contradiction, since this factor does not occur in w. Hence, for every $x \in \mathrm{var}(\alpha)$, $g(x)$ is of form $\mathtt{a}^{j_1}\,\mathtt{b}\,\mathtt{a}^{j_2}\,\mathtt{b}\cdots\mathtt{b}\,\mathtt{a}^{j_l}$ with $l \geq 1$ and $j_1 \neq 0 \neq j_l$. We can now simply translate the factors $\mathtt{ba}^i\mathtt{b}$ back to the symbols $\mathtt{c}_i$ and obtain an injective non-erasing substitution h for α'. More precisely, for every $x \in \mathrm{var}(\alpha)$ with $g(x) = \mathtt{a}^{j_1}\,\mathtt{b}\,\mathtt{a}^{j_2}\,\mathtt{b}\cdots\mathtt{b}\,\mathtt{a}^{j_l}$, we define $h(x) := \mathtt{c}_{j_1}\,\mathtt{c}_{j_2} \cdots \mathtt{c}_{j_l}$. It can be verified with moderate effort that $h(\alpha') = w'$ and, for every $x \in \mathrm{var}(\alpha)$, $|h(x)| \leq 36$. □

Table 3
Summary of all new results.

E/NE	inj/n-inj	tf/n-tf	$\|h(x)\|$	$\|\alpha\|_x$	$\|\Sigma\|$	Compl.	Thm.
E	n-inj	tf, n-tf	**1**	**2**	**2**	**NP-C**	3
NE	n-inj	n-tf	**3**	**2**	**2**	**NP-C**	6
NE	n-inj	tf	**3**	**3**	**4**	**NP-C**	6
E, NE	inj	tf, n-tf	$\mathbb{N}$	–	$\mathbb{N}$	**P**	9
E	inj	n-tf	**5**	**2**	–	**NP-C**	10
E	inj	tf	**5**	**4**	–	**NP-C**	10
E	inj	n-tf	–	**2**	**2**	**NP-C**	13
E	inj	tf	–	**9**	**5**	**NP-C**	13
NE	inj	tf, n-tf	**36**	**3**	–	**NP-C**	16
NE	inj	tf, n-tf	–	**3**	**2**	**NP-C**	19

From Lemma 20 we can directly conclude Theorem 19 and we note that Theorem 19 also implies that $[\rho^3_{|\alpha|_x}, \rho^2_{|\Sigma|}]$-[NE, inj, n-tf]-PMV is NP-complete.

We wish to point out that the simple encoding of different symbols $\mathtt{c}_i$ by unary factors $\mathtt{a}^i$ that are separated by occurrences of symbol $\mathtt{b}$ does not work in the erasing case, where we use more complicated encodings in order to bound the alphabet size (see Theorem 13). This is due to the fact that in the erasing case, variables can be substituted by the empty word, which means that several adjacent occurrences of symbol $\mathtt{b}$ may occur in the word. Therefore, if we encode the terminal word by substituting all symbols by unary factors, separated by occurrences of $\mathtt{b}$, then it is not possible anymore to erase variables from the pattern.

6. Conclusions and future research directions

We shall now summarise all the main results of this work in Table 3 (the entries have the same meaning as for Tables 1 and 2) and then we discuss them in a bit more detail. The results presented in Table 3, together with the observation that if $\rho_{|\operatorname{var}(\alpha)|}$ or $\rho_{|w|}$ is bounded, then PMV can be solved in polynomial time (see Section 3), demonstrate that for every list of parameters P, every $Z \in \{\mathrm{E}, \mathrm{NE}\}$, every $I \in \{\mathrm{inj}, \mathrm{n\text{-}inj}\}$ and every $T \in \{\mathrm{tf}, \mathrm{n\text{-}tf}\}$, we can now answer the question whether or not the parameters in P can be bounded by constants such that P-$[Z, \mathrm{n\text{-}inj}, T]$-PMV is NP-complete.

Most of these questions are answered in the positive and the only polynomial time solvable variants with an unrestricted number of variables and an unrestricted word length are the injective ones, where the maximum length of substitution words as well as the alphabet size is bounded by constants.

We can further note that most of these results that a certain restricted variant of PMV is NP-complete are also true for small constants. In this regard, the strongest results are obtained in the erasing and injective case. More precisely, for $Z = \mathrm{E}$, $I = \mathrm{n\text{-}inj}$ and all $T \in \{\mathrm{tf}, \mathrm{n\text{-}tf}\}$, we know for all *specific* constants whether or not P-$[Z, I, T]$-PMV is NP-complete if the parameters in P are bounded by these constants. On the other hand, in the case $Z = \mathrm{NE}$ and $I = \mathrm{n\text{-}inj}$, even though the NP-completeness results hold for rather small constants, too, we are not able to prove such a strong result, e.g., we do not know whether PMV is NP-complete if $\rho_{|h(x)|}$ is bounded by 2 instead of 3.

In the injective case, the constant bounds for the NP-complete variants are slightly larger, but, for most cases, still quite small. An exception is the non-erasing and injective case, where the number of variable occurrences and the length of the substitution words is bounded. Here, we are only able to establish NP-completeness if the constant bound on the substitution word length is at least 36. We conjecture that it is possible to further lower some of these constants, especially in the injective case, without losing NP-completeness. Furthermore, while the problem 3RPerCode proves itself very useful in order to obtain our hardness results, it may also be possible that by choosing a different NP-complete source problem reductions can be found that improve some of our constant bounds.

References

[1] A. Amir, Y. Aumann, R. Cole, M. Lewenstein, E. Porat, Function matching: algorithms, applications, and a lower bound, in: Proc. 30th International Colloquium on Automata, Languages and Programming, ICALP 2003, in: Lect. Notes Comput. Sci., vol. 2719, 2003, pp. 929–942.
[2] A. Amir, I. Nor, Generalized function matching, J. Discrete Algorithms 5 (2007) 514–523.
[3] D. Angluin, Finding patterns common to a set of strings, in: Proc. 11th Annual ACM Symposium on Theory of Computing, STOC 1979, 1979, pp. 130–141.
[4] D. Angluin, Finding patterns common to a set of strings, J. Comput. Syst. Sci. 21 (1980) 46–62.
[5] B.S. Baker, Parameterized pattern matching: algorithms and applications, J. Comput. Syst. Sci. 52 (1996) 28–42.
[6] P. Barceló, L. Libkin, A.W. Lin, P.T. Wood, Expressive languages for path queries over graph-structured data, ACM Trans. Database Syst. 37 (2012).
[7] J. Bremer, D.D. Freydenberger, Inclusion problems for patterns with a bounded number of variables, Inf. Comput. 220–221 (2012) 15–43.
[8] C. Câmpeanu, K. Salomaa, S. Yu, A formal study of practical regular expressions, Int. J. Found. Comput. Sci. 14 (2003) 1007–1018.
[9] R. Clifford, A.W. Harrow, A. Popa, B. Sach, Generalised matching, in: Proc. 16th International Symposium on String Processing and Information Retrieval, SPIRE 2009, in: Lect. Notes Comput. Sci., vol. 5721, 2009, pp. 295–301.
[10] A. Ehrenfeucht, G. Rozenberg, Finding a homomorphism between two words is NP-complete, Inf. Process. Lett. 9 (1979) 86–88.
[11] H. Fernau, M.L. Schmid, Pattern matching with variables: a multivariate complexity analysis, in: Proc. 24th Annual Symposium on Combinatorial Pattern Matching, CPM 2013, in: Lect. Notes Comput. Sci., vol. 7922, 2013, pp. 83–94.
[12] D.D. Freydenberger, D. Reidenbach, Bad news on decision problems for patterns, Inf. Comput. 208 (2010) 83–96.

[13] D.D. Freydenberger, D. Reidenbach, J.C. Schneider, Unambiguous morphic images of strings, Int. J. Found. Comput. Sci. 17 (2006) 601–628.
[14] J.E.F. Friedl, Mastering Regular Expressions, third edition, O'Reilly, Sebastopol, CA, 2006.
[15] M. Geilke, S. Zilles, Learning relational patterns, in: Proc. 22nd International Conference on Algorithmic Learning Theory, ALT 2011, in: Lect. Notes Comput. Sci., vol. 6925, 2011, pp. 84–98.
[16] T. Harju, J. Karhumäki, Morphisms, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer, 1997, pp. 439–510, chapter 7.
[17] O. Ibarra, T.-C. Pong, S. Sohn, A note on parsing pattern languages, Pattern Recognit. Lett. 16 (1995) 179–182.
[18] T. Jiang, E. Kinber, A. Salomaa, K. Salomaa, S. Yu, Pattern languages with and without erasing, Int. J. Comput. Math. 50 (1994) 147–163.
[19] T. Jiang, A. Salomaa, K. Salomaa, S. Yu, Decision problems for patterns, J. Comput. Syst. Sci. 50 (1995) 53–63.
[20] J. Kratochvíl, M. Křivánek, On the computational complexity of codes in graphs, in: Proc. 13th Symposium on Mathematical Foundations of Computer Science, MFCS 1988, in: Lect. Notes Comput. Sci., vol. 324, 1988, pp. 396–404.
[21] A. Mateescu, A. Salomaa, Finite degrees of ambiguity in pattern languages, RAIRO Theor. Inform. Appl. 28 (1994) 233–253.
[22] Y.K. Ng, T. Shinohara, Developments from enquiries into the learnability of the pattern languages from positive data, Theor. Comput. Sci. 397 (2008) 150–165.
[23] E. Ohlebusch, E. Ukkonen, On the equivalence problem for E-pattern languages, Theor. Comput. Sci. 186 (1997) 231–248.
[24] D. Reidenbach, A non-learnable class of E-pattern languages, Theor. Comput. Sci. 350 (2006) 91–102.
[25] D. Reidenbach, Discontinuities in pattern inference, Theor. Comput. Sci. 397 (2008) 166–193.
[26] D. Reidenbach, M.L. Schmid, 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, in: Lect. Notes Comput. Sci., vol. 6482, 2011, pp. 241–250.
[27] D. Reidenbach, M.L. Schmid, Patterns with bounded treewidth, Inf. Comput. 239 (2014) 87–99.
[28] T.J. Schaefer, The complexity of satisfiability problems, in: Proc. 10th Annual ACM Symposium on Theory of Computing, STOC 1978, ACM, 1978, pp. 216–226.
[29] M.L. Schmid, On the membership problem for pattern languages and related topics, PhD thesis, Department of Computer Science, Loughborough University, 2012.
[30] M.L. Schmid, Inside the class of REGEX languages, Int. J. Found. Comput. Sci. 24 (2013).
[31] M.L. Schmid, A note on the complexity of matching patterns with variables, Inf. Process. Lett. 113 (2013) 729–733.
[32] T. Shinohara, Polynomial time inference of extended regular pattern languages, in: Proc. RIMS Symposium on Software Science and Engineering, in: Lect. Notes Comput. Sci., vol. 147, 1982, pp. 115–127.
[33] T. Shinohara, Polynomial time inference of pattern languages and its application, in: Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, 1982, pp. 191–209.