The Equivalence Problem of E-Pattern Languages with Length Constraints Is Undecidable

Dirk Nowotka
Department of Computer Science, Kiel University, Germany

Max Wiedenhöft
Department of Computer Science, Kiel University, Germany

Abstract

Patterns are words with terminals and variables. The language of a pattern is the set of words obtained by uniformly substituting all variables with words that contain only terminals. Length constraints restrict valid substitutions of variables by associating the variables of a pattern with a system (or disjunction of systems) of linear diophantine inequalities. Pattern languages with length constraints contain only words in which all variables are substituted to words with lengths that fulfill such a given set of length constraints. We consider membership, inclusion, and equivalence problems for erasing and non-erasing pattern languages with length constraints.

Our main result shows that the erasing equivalence problem - one of the most prominent open problems in the realm of patterns - becomes undecidable if length constraints are allowed in addition to variable equality.

Additionally, it is shown that the terminal-free inclusion problem, a prominent problem which has been shown to be undecidable in the binary case for patterns without any constraints, is also generally undecidable for all larger alphabets in this setting.

Finally, we also show that considering regular constraints, i.e., associating variables also with regular languages as additional restrictions together with length constraints for valid substitutions, results in undecidability of the non-erasing equivalence problem. This sets a first upper bound on constraints to obtain undecidability in this case, as this problem is trivially decidable in the case of no constraints and as it has unknown decidability if only regular or only length constraints are considered.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Patterns, Pattern Languages, Length Constraints, Regular Constraints, Decidability, Undecidability, Membership, Inclusion, Equivalence

Digital Object Identifier 10.4230/LIPIcs.CPM.2025.4

Related Version *Full Version*: `https://arxiv.org/abs/2411.06904` [27]

Funding *Max Wiedenhöft*: This work was supported by the DFG project number 437493335.

1 Introduction

A *pattern* is a finite word consisting only of symbols from a finite set of *letters* $\Sigma = \{\mathtt{a}_1, ..., \mathtt{a}_\sigma\}$, also called *terminals*, and from an infinite set of *variables* $X = \{x_1, x_2, ...\}$ such that we have $\Sigma \cap X = \emptyset$. It is a natural and compact device to define formal languages. From patterns, we obtain words consisting only of terminals using a *substitution* h, a terminal preserving morphism that maps all variables in a pattern to words over the terminal alphabet. The *language* of a pattern is the set of all words obtainable using arbitrary substitutions.

We differentiate between two kinds of substitutions. In the original definition of patterns and pattern languages introduced by Angluin [1], only words obtained by *non-erasing substitutions* are considered. Here, all variables are required to be mapped to non-empty words. The resulting languages are called *non-erasing (NE) pattern languages*. Later, so called *erasing-/extended-* or just *E-pattern languages* have been introduced by Shinohara

[35]. Here, variables may also be substituted by the empty word ε. Consider, for example, the pattern $\alpha := x_1\mathtt{a}x_2\mathtt{b}x_1$. Then, if we map x_1 to $\mathtt{ab}$ and x_2 to $\mathtt{bbaa}$ using a substitution h, we obtain the word $h(\alpha) = \mathtt{ababbaabab}$. Considering the E-pattern language of α, we could also map x_1 to the empty word and obtain any word in the language $\{\mathtt{a}\} \cdot \Sigma^* \cdot \{\mathtt{b}\}$.

Due to its practical and simple definition, patterns and their corresponding languages occur in numerous areas in computer science and discrete mathematics. These include, for example, unavoidable patterns [18, 23], algorithmic learning theory [1, 6, 36], word equations [23], theory of extended regular expressions with back references [12], or database theory [10, 34].

There are three main decision problems regarding patterns and pattern languages. Those are the *membership problem* (and its variations [14, 15, 7]), the *inclusion problem*, and the *equivalence problem*. All are considered in the erasing (E) and non-erasing (NE) cases. The membership problem determines if a word belongs to the language of a pattern. This problem has been shown to be NP-complete for both, erasing- and non-erasing, pattern languages [1, 18]. The inclusion problem determines whether the language of one pattern is a subset of the language of another pattern. It has been shown to be generally undecidable by Jiang et al. in [19]. Freydenberger and Reidenbach [11] as well as Bremer and Freydenberger [3] improved that result and showed that it is undecidable for all bounded alphabets of size $|\Sigma| \geq 2$ for both erasing and non-erasing pattern languages. The equivalence problem asks whether the languages of two patterns are equal. For NE-pattern languages, this problem is trivially decidable and characterized by the equality of patterns up to a renaming of their variables [1]. The decidability of the erasing case is one of the major open problems in the field [19, 30, 29, 28, 31]. For terminal-free patterns, however, i.e., patterns without any terminal letters, the inclusion problem as well as the equivalence problem for E-pattern languages have been characterized and shown to be NP-complete [19, 5]. In addition to that, in the case of terminal-free NE-pattern languages, Saarela [32] has shown that the inclusion problem is undecidable in the case of a binary underlying alphabet.

Over time, various extensions to patterns and pattern languages have been introduced, either, to obtain additional expressibility due to some practical context or to get closer to an answer for the remaining open problems. Some examples are the bounded scope coincidence degree, patterns with bounded treewidth, k-local patterns, or strongly-nested patterns (see [4] and references therein). Koshiba [21] introduced so called *typed patterns* that restrict substitutions of variables to types, i.e., arbitrary recursive languages. Geilke and Zilles [16] extended this recently to the notion of *relational patterns* and *relational pattern languages*. In a similar more recent context and with specific relational constraints such as equal length, besides string equality, Freydenberger discusses in [8], building on [2, 13, 9], so called conjunctive regular path queries (CRPQs) with string equality and with length equality. They can be understood as systems of (relational) patterns with regular constraints (restricting variable substitutions to words of given regular languages) and with equal length constraints between variables.

In [26], a special form of typed patterns has been considered, i.e., patterns with regular constraints (also comparable to singleton sets of H-Systems [13]). Here, like mentioned before, variables may be restricted to arbitrary regular languages and the same variable may occur more than once. It has been shown that this notion suffices to obtain undecidability for both main open problems regarding pattern languages, i.e., the equivalence problem of E-pattern languages and the inclusion problem of terminal-free NE-pattern languages with an alphabet greater or equal to 3. Another natural extension other than regular constraints is the notion of length constraints. Here, instead of restricting the choice of words for the substitution of

variables, length constraints just restrict the lengths of substitution of variables in relation to each other. In the field of word equations, length constraints have been considered as a natural extension for a long time and, e.g., answering the decidability of the question whether word equations with length constraints have a solution, is a long outstanding problem (see, e.g., [22] and the references therein).

In this paper, we consider that natural extension of length constraints on patterns, resulting in the class of patterns called *patterns with length constraints*. In general, we say that a *length constraint* ℓ is a disjunction of systems of linear (diophantine) inequalities over the variables of X. We denote the *set of all length constraints* by $\mathcal{C}_{Len}$. A *pattern with length constraints* $(\alpha, \ell_\alpha) \in (\Sigma \cup X)^* \times \mathcal{C}_{Len}$ is a pattern associated with a length constraint. We say that a substitution h is ℓ_α-valid if all variables are substituted according to ℓ_α. Now, the language of (α, ℓ_α) is defined analogously to pattern languages but restricted to ℓ_α-valid substitutions in the erasing- and non-erasing cases.

We examine erasing (E) and non-erasing (NE) pattern languages with length constraints. It can be shown, following existing results for patterns without additional constraints, that the membership problem for both cases in NP-complete. The inclusion problem is shown to be undecidable in both cases, too, notably even for terminal-free pattern languages, which is a difference to the decidability of the inclusion problem in the erasing case for pattern languages without any constraints, and which answers the case of alphabet sizes greater or equal to 3 for non-erasing patterns without constraints. The main result of this paper is the undecidability of the equivalence problem for erasing pattern languages with length constraints in both cases, terminal-free and general, giving an answer to a problem of which the decidability has been an open problem for a long time in the case of no constraints. The final result shows that regular constraints and length constraints combined suffice to show undecidability of the equivalence problem for non-erasing pattern languages, a problem that is trivially decidable in case of no constraints and still open in the cases of just regular constraints or just length constraints.

We begin by introducing the necessary notation in Section 2 and then continue in Section 3 with an examination of patterns with length constraints and their corresponding languages. Here, we will briefly discuss all results regarding the related decision problems (membership, inclusion, equivalence). Due to the extensiveness of some proofs, see [27] for the full formal versions of the shortened ones. In Section 4, we continue with an examination of patterns with regular and length constraints, also giving a full picture of the related decision problems. Finally, in Section 5, we close this paper with a summary and discussion of the obtained results, the methods that were used, and the open problems that remain.

2 Preliminaries

Let $\mathbb{N}$ denote the natural numbers. For $n, m \in \mathbb{N}$ set $[m, n] := \{k \in \mathbb{N} \mid m \leq k \leq n\}$. Denote $[n] := [1, n]$ and $[n]_0 := [0, n]$. The powerset of any set A is denoted by $\mathcal{P}(A)$. An *alphabet* Σ is a non-empty finite set whose elements are called *letters* or *terminals*. A *word* is a finite sequence of letters from Σ. Let Σ^* be the set of all finite words over Σ, thus it is a free monoid with concatenation as operation and the empty word ε as the neutral element. Set $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$. We call the number of letters in a word $w \in \Sigma^*$ *length* of w, denoted by $|w|$. Therefore, we have $|\varepsilon| = 0$. Let Σ^k denote the set of all words of length $k \in \mathbb{N}$ (resp. $\Sigma^{\leq k}$ or $\Sigma^{\geq k}$). If $w = xyz$ for some $x, y, z \in \Sigma^*$, we call x a *prefix* of w, y a *factor* of w, and z a *suffix* of w and denote the sets of all prefixes, factors, and suffixes of w by $\mathrm{Pref}(w)$, $\mathrm{Fact}(w)$, and $\mathrm{Suff}(w)$ respectively. For words $w, u \in \Sigma^*$, let $|w|_u$ denote the number of

distinct occurrences of u in w as a factor. For $w \in \Sigma^*$, let $w[i]$ denote w's i^{th} letter for all $i \in [|w|]$. For reasons of compactness, we denote $w[i] \cdots w[j]$ by $w[i \cdots j]$ for all $i, j \in [|w|]$ with $i < j$. Set $\text{alph}(w) := \{\mathtt{a} \in \Sigma \mid \exists i \in [|w|] : w[i] = \mathtt{a}\}$ as w's alphabet.

Now, we introduce the notion of patterns and pattern languages with and without additional constraints (i.e., length constraints, length constraints, and a combination of both). After that, we briefly introduce the machine models used in the proofs of the main results of this paper.

2.1 Patterns and Pattern Languages with Constraints

Let X be a countable set of variables and Σ be an alphabet such that $\Sigma \cap X = \emptyset$. A *pattern* is then a non-empty, finite word over $\Sigma \cup X$. A pattern is called *terminal-free* if it just consists of variables and is, thus, a non-empty finite word over X. The set of all patterns over $\Sigma \cup X$ is denoted by Pat_Σ. For example, $x_1\mathtt{a}x_2\mathtt{ba}x_2x_3$ is a pattern over $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ with $x_1, x_2, x_3 \in X$. For a pattern $\alpha \in Pat_\Sigma$, let $\text{var}(\alpha) := \{\ x \in X \mid |\alpha|_x \geq 1\ \}$ denote the set of variables occurring in p. A *substitution of* α is a morphism $h : (\Sigma \cup X)^* \to \Sigma^*$ such that $h(\mathtt{a}) = \mathtt{a}$ for all $\mathtt{a} \in \Sigma$ and $h(x) \in \Sigma^*$ for all $x \in X$. If we have $h(x) \neq \varepsilon$ for all $x \in \text{var}(\alpha)$, we call h a *non-erasing substitution* for α. Otherwise, h is an *erasing substitution* for α. The set of all substitutions w.r.t. Σ is denoted by H_Σ. If Σ is clear from the context, we may write just H. Given a pattern $\alpha \in Pat_\Sigma$, its erasing pattern language $L_E(\alpha)$ and its non-erasing pattern language $L_{NE}(\alpha)$ are defined respectively by

$$L_E(\alpha) := \{\ h(\alpha) \mid h \in H, h(x) \in \Sigma^* \text{ for all } x \in \text{var}(\alpha)\}, \text{ and}$$
$$L_{NE}(\alpha) := \{\ h(\alpha) \mid h \in H, h(x) \in \Sigma^+ \text{ for all } x \in \text{var}(\alpha)\}.$$

A *length constraint* ℓ is a disjunction of systems of linear diophantine inequalities over variables of X. We denote the *set of all length constraints* by $\mathcal{C}_{Len}$. A *pattern with length constraints* is a pair $(\alpha, \ell_\alpha) \in \mathtt{Pat}_\Sigma \times \mathcal{C}_{Len}$ where all variables occurring in ℓ_α must occur in α. We denote the *set of all patterns with length constraints* by $\mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$. For some $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ and $h \in H$, we say that h is a ℓ_α*-valid substitution* if ℓ_α is satisfied when associating each variable $x \in \text{var}(\alpha)$ in ℓ_α with the value $|h(x)|$, i.e., the length of the substitution of the variable x. Consider the following example.

▶ **Example 1.** Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and $\alpha = x_1\ \mathtt{a}\ x_2\ \mathtt{a}\ x_1$. Assume we have the length constraint ℓ_α defined by the following system of linear diophantine inequalities:

$$\begin{aligned} 2x_1 + x_2 &\leq 5 \\ x_2 &\geq 1 \end{aligned}$$

Then, we know that any ℓ_α-valid substitution $h \in H$ cannot have $h(x_1) = u$ for some $u \in \Sigma^*$ with $|u| \geq 3$, as this would already imply $2|u| = 2|h(x)| \geq 2 \cdot 3 = 6$ and $6 \geq 5$. Also, we see that $h(x_2) \neq \varepsilon$ as the second constraint demands a substitution of length at least 1. So, for example we could have $h(\alpha) = \mathtt{bbababb}$ or $h(\alpha) = \mathtt{bababab}$ but not $h(\alpha) = \mathtt{aa}$ or $h(\alpha) = \mathtt{bbaaaabb}$.

We denote the set of all ℓ_α-valid substitutions by H_{ℓ_α}. The notion of pattern languages is extended by the following. For any $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ we denote by

$$L_E(\alpha, \ell_\alpha) := \{\ h(\alpha) \mid h \in H_{\ell_\alpha}, h(x) \in \Sigma^* \text{ for all } x \in \text{var}(\alpha)\ \}$$

the *erasing pattern language with length constraints* of (α, ℓ_α) and by

$$L_{NE}(\alpha, \ell_\alpha) := \{\ h(\alpha) \mid h \in H_{\ell_\alpha}, h(x) \in \Sigma^+ \text{ for all } x \in \text{var}(p)\ \}$$

the *non-erasing pattern language with length constraints* of (α, ℓ_α).

Similar to length constraints, we can define *regular constraints* for variables in a pattern. Let $\mathcal{L}_{Reg}$ be the set of all regular languages. We call a mapping $r : X \to \mathcal{L}_{Reg}$ a *regular constraint* on X. If not stated otherwise, we always have $r(x) = \Sigma^*$. We denote the *set of all regular constraints* by $\mathcal{C}_{Reg}$. For some $r \in \mathcal{C}_{Reg}$ we define the *language of a variable* $x \in X$ by $L_r(x) = r(x)$. If r is clear by the context, we omit it and just write $L(x)$. A *pattern with regular constraints* is a pair $(\alpha, r_\alpha) \in Pat_\Sigma \times \mathcal{C}_{Reg}$. We denote the *set of all patterns with regular constraints* by $\texttt{Pat}_{\Sigma,\mathcal{C}_{Reg}}$. For some $(\alpha, r_\alpha) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Reg}}$ and $h \in H$, we say that h is a r_α*-valid substitution* if $h(x) \in L(x)$ for all $x \in \mathrm{var}(\alpha)$. The set of all r_α-valid substitutions is denoted by H_{r_p}. Given some $(\alpha, r_\alpha) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Reg}}$, we analogously define the *erasing- and non-erasing pattern languages with regular constraints* $L_E(\alpha, r_\alpha)$ and $L_{NE}(\alpha, r_\alpha)$ over H_{r_α} as we did for length constraints.

Combining both, we say that a triple $(\alpha, r_\alpha, \ell_\alpha) \in \texttt{Pat} \times \mathcal{C}_{Reg} \times \mathcal{C}_{Len}$ is a *pattern with regular and length constraints* and denote the *set of all patterns with regular and length constraints* by $\texttt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$. Given some $(\alpha, r_\alpha, \ell_\alpha) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$, we say that a substitution $h \in H$ is r_α-ℓ_α*-valid* if it is r_α-valid and ℓ_α-valid and denote the set of all r_α-ℓ_α-valid substitutions by H_{r_α,ℓ_α}. Additionally, we analogously define the *erasing- and non-erasing pattern languages with regular and length constraints* $L_E(\alpha, r_\alpha, \ell_\alpha)$ and $L_{NE}(\alpha, r_\alpha, \ell_\alpha)$ over H_{r_α,ℓ_α} as we did in the previous two cases for length constraints and regular constraints.

In the proofs of our main results Theorem 8, Theorem 12, and Theorem 19, we use two different automata models with undecidable emptiness problems to obtain each result. We will use the notion of *nondeterministic 2-counter automata without input* (see, e.g., [17]), as well as, the notion of a very specific universal Turing machine U as it is used in [3]. As mentioned in the introduction, due to their extensive lengths, the proofs of each of the main theorems is only given as an extended sketch in this paper together with the appendix[1]. In the main body, only a rough explanation, not relying on a formal definition of the machine types, of each proof idea is provided. Hence, the formal definition of the notion of *nondeterministic 2-counter automata without input* and the referenced universal Turing machine U are found in Appendix A and in Appendix B, respectively. We just mention that $\texttt{ValC}(A)$ refers to the set of encodings of valid computations of some nondeterministic 2-counter machine without input A and that $\texttt{ValC}_U(I)$ refers to the set of encodings of valid computations from some initial configuration I over the universal Turing machine U.

We continue with our overview of the results regarding pattern languages with length constraints.

3 Results for Pattern Languages with Length Constraints

To begin developing an understanding of the additional expressiveness gained by allowing for length constraints, notice the following observation for patterns with length or with regular constraints which does not necessarily hold for patterns without any constraints.

▶ **Lemma 2.** *For each pattern with length constraints* $(\alpha, \ell_\alpha) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ *(and for each pattern with regular constraints* $(\beta, r_\beta) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Reg}}$*), there exists some adapted set of length constraints* $\ell'_\alpha \in \mathcal{C}_{Len}$ *(resp., some adapted set of regular constraints* $r'_\beta \in \mathcal{C}_{Reg}$*) such that* $L_{NE}(\alpha, \ell_\alpha) = L_E(\alpha, \ell'_\alpha)$ *(and* $L_{NE}(\beta, r_\beta) = L_E(\beta, r_\beta)$ *resp.).*

Proof. Indeed, given some pattern with length constraints $(\alpha, \ell_\alpha) \in \texttt{Pat}_{\Sigma,\mathcal{C}_{Len}}$, we can define the length constraint ℓ'_α by using all constraints in ℓ_α and additionally, for each $x \in \mathrm{var}(\alpha)$, adding the constraint $x \geq 1$ to ℓ'_α. Then, $L_E(\alpha, \ell'_\alpha) = L_{NE}(\alpha, \ell_\alpha)$.

[1] See [27] for the full formal proofs.

We obtain the same result for pattern languages with regular constraints by intersecting the language of all variables with Σ^+, i.e., given any language $L(x)$ for some variable $x \in X$ that is defined by some regular constraint $r \in \mathcal{C}_{Reg}$, we can define a regular constraint r' that defines the language $L'(x) = L(x) \cap \Sigma^+$. Then, given some pattern $\alpha \in \mathtt{Pat}$, we obtain $L_{NE}(\alpha, r) = L_E(\alpha, r')$. ◀

The following statement then immediately follows by the previous lemma.

▶ **Corollary 3.** *Solving any problem for erasing pattern languages with length (or regular) constraints is at least as hard as solving the same problem for non-erasing pattern languages with length constraints.*

As shown in [26], considering regular constraints, problem cases over all patterns can be reduced to problems involving terminal-free patterns. The same does not hold in general in the case of length constraints, witnessed by the following proposition.

▶ **Proposition 4.** *There exists $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ such that no $(\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ with a terminal-free pattern $\beta \in X^*$ exists, for which we have that $L_X(\alpha, \ell_\alpha) = L_X(\beta, \ell_\beta)$ for $X \in \{E, NE\}$.*

Proof. Assume $|\Sigma| \geq 2$ and assume w.l.o.g. $\mathtt{a}, \mathtt{b} \in \Sigma$ with $\mathtt{a} \neq \mathtt{b}$. Let $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ such that $\alpha = x_1\mathtt{a}$ and ℓ_α is an empty length constraint. So, $L_E(\alpha, \ell_\alpha) = L_E(\alpha) = \Sigma^* \cdot \{\mathtt{a}\}$ ($L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\alpha) = \Sigma^+ \cdot \{\mathtt{a}\}$). Suppose there exists some $\beta \in X^*$ and length constraint ℓ_β such that $L_E(\beta, \ell_\beta) = L_E(\alpha, \ell_\alpha)$ ($L_{NE}(\beta, \ell_\beta) = L_{NE}(\alpha, \ell_\alpha)$). W.l.o.g. we continue with the erasing case. The following also holds for the non-erasing case with small changes. Let $w \in L_E(\alpha, \ell_\alpha)$. Then $w = u\mathtt{a}$ for some $u \in \Sigma^*$. As $L_E(\alpha, \ell_\alpha) = L_E(\beta, \ell_\beta)$, there exists some $h \in H_{\ell_\beta}$ such that $h(\beta) = w = u\mathtt{a}$. Then, $\beta = \beta_1 x \beta_2$ for $x \in X$ and $\beta_1, \beta_2 \in X^*$ as well as $u = u_1u_2$ for some $u_1, u_2 \in \Sigma^*$ such that $h(\beta_1) = u_1$, $h(x) = u_2\mathtt{a}$ and $h(\beta_2) = \varepsilon$. But then, we also find some $h' \in H_{\ell_\beta}$ with $h'(x) = u_1\mathtt{b}$ as $\mathtt{a}$ and $\mathtt{b}$ are obtained at the same position by the same variable. Then, $h'(\beta) = v\mathtt{b}$ for some $v \in \Sigma^*$. As $v\mathtt{b} \notin \Sigma^* \cdot \{\mathtt{a}\}$ but $L_E(\alpha, \ell_\alpha) = \Sigma^* \cdot \{\mathtt{a}\}$, we know that β cannot exist. ◀

With that, we know that we cannot restrict ourselves to only show hardness or undecidability for the general case, as the terminal-free case may result in something else. We begin with the membership problems for both, the erasing- and non-erasing pattern languages. Those have been shown to be NP-complete for patterns without constraints in the terminal-free and general cases (see, e.g.,[1, 18, 33]). Hence, we observe the following for patterns with length constraints.

▶ **Corollary 5.** *Let $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$ and $w \in \Sigma^*$. The decision problem of whether $w \in L_X(\alpha, \ell_\alpha)$ for $X \in \{E, NE\}$ is NP-complete, even if the considered pattern α is terminal-free.*

Indeed, NP-hardness is immediately obtained by the NP-hardness of patterns without any constraints. Given some $(\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Len}}$, NP containment follows by the fact that any valid certificate results in a substitution $h \in H_{\ell_\alpha}$ of length at most $|h(\alpha)| = |w|$ and that we can check in polynomial time whether some given substitution is ℓ_α-valid. This can be done by checking whether the lengths of the variable substitutions $|h(x)|$ for all $x \in \mathrm{var}(\alpha)$ satisfy ℓ_α. This concludes the membership problem for all cases.

Another result that can be immediately obtained is the general undecidability of the inclusion problem for pattern languages with length constraints. For pattern languages without any constraints, this problem has been generally shown to be undecidable, first, for

unbounded alphabets by Jiang et al. [19] and later on for all bounded alphabets of sizes $|\Sigma| \geq 2$ by Freydenberger and Reidenbach in [11] as well as Bremer and Freydenberger in [3]. So, we have the following.

▶ **Theorem 6** ([19, 11, 3]). *Let $\alpha, \beta \in Pat_\Sigma$. In general, for all alphabets Σ with $|\Sigma| \geq 2$, it is undecidable to answer whether $L_X(\alpha) \subseteq L_X(\beta)$ for $X \in \{E, NE\}$.*

Out of that, in the general case, we immediately obtain the following for pattern languages with length constraints.

▶ **Corollary 7.** *Let $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$. For all alphabets Σ with $|\Sigma| \geq 2$ it is undecidable to answer whether $L_X(\alpha, \ell_\alpha) \subseteq L_X(\beta, \ell_\beta)$ for $X \in \{E, NE\}$.*

That leaves us with the terminal-free cases of the inclusion problem as well as the general and terminal-free cases of the equivalence problems for both, erasing and non-erasing pattern languages with length constraints. The first and most significant main result shows that the most prominent open problem for pattern languages, i.e., the equivalence problem for erasing patterns, is undecidable for pattern languages with length constraints, even if we are restricted to terminal-free patterns and use no disjunctions in the length constraints.

▶ **Theorem 8.** *Let $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$. The problem of deciding whether we have $L_E(\alpha, \ell_\alpha) = L_E(\beta, \ell_\beta)$ is undecidable for all alphabets Σ with $|\Sigma| \geq 2$, even if α and β are restricted to be terminal-free and even if ℓ_α and ℓ_β have no disjunctions.*

Proof. Based on the proofs in, e.g., [19] and [11], we reduce the emptiness problem for nondeterministic 2-counter automata without input to the equivalence problem of erasing pattern languages with length constraints. Due to the length of the formal proof, here, we only give a sketch of the idea and provide the formal construction of the reduction. For a full extension of the proof containing a collection of shown properties and a formal proof of the correctness of the reduction itself, see [27].

Given an automaton A, we construct two patterns with length constraints (α, ℓ_α) and (β, ℓ_β) such that $L_E(\alpha, \ell_\alpha) = L_E(\beta, \ell_\beta)$ if and only if $\mathtt{ValC}(A) = \emptyset$. The pattern with length constraints (α, ℓ_α) can be seen as a pattern which allows for all possible words of a specific format to be obtained, especially also encodings of valid computations of A. The pattern with length constraints (β, ℓ_β) can be seen as a *tester* pattern that allows only for words that are not encodings of valid computations of A that have this specific structure. By that, if A has some valid computation and, hence, an encoding of a valid computation of A exists, we have that there exists a word in $L_E(\alpha, \ell_\alpha)$ that is not in $L_E(\beta, \ell_\beta)$, hence $L_E(\alpha, \ell_\alpha) \neq L_E(\beta, \ell_\beta)$.

The previous property is the main argument used in the proofs in [19, 11] to show undecidability of the inclusion problem for erasing pattern languages. In addition to that, using length constraints, (α, ℓ_α) and (β, ℓ_β) can be constructed in such a way, that we always have $L_E(\beta, \ell_\beta) \subseteq L_E(\alpha, \ell_\alpha)$. This is the most significant difference to the aforementioned results. Thus, in this case, we have that the equivalence of $L_E(\alpha, \ell_\alpha)$ and $L_E(\beta, \ell_\beta)$ decides the emptiness problem for A. The latter is undecidable, resulting in the undecidability of the equivalence problem of erasing pattern languages with length constraints. This is similar to the idea of the construction in [26] for patterns with regular constraints, where the same approach but with differently constructed patterns and with another type of constraints has been used. We also mention that length constraints allow for a construction that shows undecidability even in the case of terminal-free patterns. This is interesting, as for terminal-free erasing pattern languages without any constraints, equivalence and inclusion are actually decidable.

Now, we continue with the constructions of (α, ℓ_α) and (β, ℓ_β). First, (α, ℓ_α) is defined by

$$\alpha = y_1\ x_v\alpha_1 x_v\alpha_2 x_v\ y_2\ x_v z z' z' z x_v$$

for independent variables $y_1, y_2, x_v, \alpha_1, \alpha_2, z, z' \in X$, and by the following linear system ℓ_α:

- $x_v = 5$
- $z = 1$, $z' = 1$

Notice that any word in $L_E(\alpha, \ell_\alpha)$ has a suffix of fixed length with the form vuv where $v \in \Sigma^5$ and $u \in \{0000, \#\#\#\#, 0\#\#0, \#00\#\}$. In comparison to the construction in [11] that made use of a specific prefix and a specific suffix to enforce predicate choice, only this suffix suffices for this purpose in this construction.

Next, we construct (β, ℓ_β). We set the pattern β to

$$\beta = y_1'\ \hat{\beta}_1\ \hat{\beta}_2\ \cdots\ \hat{\beta}_{\mu+1}\ y_2'\ \ddot{\beta}_1\ \ddot{\beta}_2\ \cdots\ \ddot{\beta}_{\mu+1}$$

for new and independent variables $y_1', y_2' \in X$ and terminal-free patterns $\hat{\beta}_i, \ddot{\beta}_i \in X^*$. Assume μ to be defined as in [11]. For $i \in [\mu + 1]$, set

- $\hat{\beta}_i = x_i\ \gamma_i\ x_i\ \delta_i\ x_i$,
- $\ddot{\beta}_i = x_i\ \eta_i\ x_i$,

for new and independent variables $x_i \in X$ and terminal-free patterns $\gamma_i, \delta_i, \eta_i \in X^*$. Assume

- $\eta_i = z_i z_i' z_i' z_i$ and $z_i \neq z_i'$ for $i \in [\mu]$,
- $\eta_{\mu+1} = (z_{\mu+1})^4$,
- $\mathrm{var}(\gamma_i\delta_i\eta_i) \cap \mathrm{var}(\gamma_j\delta_j\eta_j) = \emptyset$ if $i \neq j$ for $i, j \in [\mu + 1]$, and
- $x_k, y_1', y_2' \notin \mathrm{var}(\gamma_i\delta_i\eta_i)$ for all $i, k \in [\mu + 1]$

for new and independent variables $z_i, z_i' \in X$. Hence, all variables inside $\mathrm{var}(\gamma_i\delta_i\eta_i)$ do not occur anywhere else but in these factors. The length constraint ℓ_β is defined by the following and only the following system (no additional constraints will be added later on).

1. $\sum_{i=1}^{\mu+1} z_i = 1$
2. $\sum_{i=1}^{\mu} z_i' \leq 1$
3. $\sum_{i=1}^{\mu+1} x_i = 5$
4. $x_i - 5z_i = 0$ for all $i \in [\mu + 1]$
5. $z_i - z_i' = 0$ for all $i \in [\mu]$

For all $i \in [\mu]$, we say that γ_i and δ_i are defined just as in [11]. For the case of $\mu + 1$, we set $\gamma_{\mu+1}$ and $\delta_{\mu+1}$ by

- $\gamma_{\mu+1} = y_{\mu+1}$, and
- $\delta_{\mu+1} = y_{\mu+1}'$

for new and independent variables $y_{\mu+1}$ and $y_{\mu+1}'$.

We now give a really rough sketch of the idea on why this construction works. First, the length constraints ℓ_β allow for only one $\ddot{\beta}_i$, for $i \in [\mu + 1]$, to be substituted to a non-empty word. Any word obtained by $\ddot{\beta}_i$ can be obtained by the suffix $x_v z z' z' z x_v$ of (α, ℓ_α). Additionally, everything obtained in $\hat{\beta}_i$ can then be obtained by $x_v\alpha_1 x_v\alpha_2 x_v$ in (α, ℓ_α). Everything else can be obtained by the variables y_1 and y_2. Hence, we get the most significant property of this proof, i.e., $L_E(\beta, \ell_\beta) \subseteq L_E(\alpha, \ell_\alpha)$.

Now, for the other direction, we consider two cases. These are, given some substitution $h \in H_{\ell_\alpha}$, whether $h(z) = h(z')$ or whether $h(z) \neq h(z')$. By ℓ_α, we know $|h(z)| = |h(z')| = 1$. It can be shown that, if $h(z) = h(z')$, then we can use $\ddot{\beta}_{\mu+1}$ to obtain $h(x_v z z' z' z x_v)$ and $\hat{\beta}_{\mu+1}$ to obtain $h(x_v\alpha_1 x_v\alpha_2 x_v)$. $h(y_1)$ can be obtained in y_1' and the same holds analogously for $h(y_2)$ and y_2'. Hence, if $h(z) = h(z')$, then $h(\alpha) \in L_E(\beta, \ell_\beta)$. If on the other hand we have $h(z) \neq h(z')$ and $h(y_1) = \varepsilon$ as well as $h(y_2) = \varepsilon$, we need to find some $\hat{\beta}_i$ and $\ddot{\beta}_i$, for $i \in [\mu]$, to obtain $h(x_v\alpha_1 x_v\alpha_2 x_v)$ and $h(x_v z z' z' z x_v)$. By the construction in [11], we know that this

only works if $h(\alpha_1)$ is not the encoding of a valid computation of A or if $h(\alpha_2)$ either contains $h(z')$ or $|h(\alpha_2)|$ is a unary word over $h(z)$ that is short enough (If $h(y_1) \neq \varepsilon$ or $h(y_2) \neq \varepsilon$, we can either use y_1' or y_2' in β to obtain parts of $h(\alpha)$, or we also have to rely on some $\ddot{\beta}_i$ and $\hat{\beta}_i$ and the same property as before holds). Hence, if and only if there exists some valid computation for A, we can find a substitution $h \in H_{\ell_\alpha}$ for which $h(\alpha) \notin L_E(\beta, \ell_\beta)$. This concludes the sketch for the binary case of the proof of Theorem 8. See Appendix C for a sketch on how to extend this to arbitrary fixed alphabets. As mentioned before, see [27] for a full formal proof of this result. ◀

Due to the construction in the proof of Theorem 8, the undecidability of the inclusion problem for terminal-free erasing pattern languages immediately follows.

▶ **Corollary 9.** *Let $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$ for terminal-free patterns $\alpha, \beta \in X^*$. The problem of answering whether we have $L_E(\alpha, \ell_\alpha) \subseteq L_E(\beta, \ell_\beta)$ is undecidable for all alphabets Σ with $|\Sigma| \geq 2$.*

Proof. Suppose it was decidable. Then, we could decide whether $L_E(\alpha, \ell_\alpha) \subseteq L_E(\beta, \ell_\beta)$ and whether $L_E(\beta, \ell_\beta) \subseteq L_E(\alpha, \ell_\alpha)$ and by that decide if we have $L_E(\alpha, \ell_\alpha) = L_E(\beta, \ell_\beta)$. That is a contradiction to Theorem 8. ◀

The second main result of the paper, and the last one regarding patterns with only length constraints, shows that the inclusion problem for terminal-free non-erasing pattern languages with length constraints is also undecidable for all alphabets Σ with $|\Sigma| > 2$. In [32], Saarela has shown this problem to be undecidable for patterns without any constraints in the binary case.

▶ **Theorem 10** ([32])**.** *Let $\alpha, \beta \in X^*$ be two terminal-free patterns. Answering whether $L_{NE}(\alpha) \subseteq L_{NE}(\beta)$ is undecidable for alphabets Σ with $|\Sigma| = 2$.*

Hence, the undecidability of the inclusion problem for pattern languages with length constraints follows immediately in this case.

▶ **Corollary 11.** *Let $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$ for terminal-free patterns $\alpha, \beta \in X^*$. Answering whether we have $L_{NE}(\alpha, \ell_\alpha) \subseteq L_{NE}(\beta, \ell_\beta)$ is undecidable for alphabets Σ with $|\Sigma| = 2$.*

In the case of length constraints, a general extension to all alphabet sizes is possible. We obtain the following.

▶ **Theorem 12.** *Let $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$ for terminal-free patterns $\alpha, \beta \in X^*$. Answering whether we have $L_{NE}(\alpha, \ell_\alpha) \subseteq L_{NE}(\beta, \ell_\beta)$ is undecidable for alphabets Σ with $|\Sigma| \geq 2$, even if ℓ_α and ℓ_β have no disjunctions.*

Proof. Again, we only give a sketch of the idea and the formal construction used in this proof. The binary case follows by Corollary 11, i.e., the result by Saarela [32]. The general proof idea is based on the construction done by Freydenberger and Bremer [3] for the binary case and significantly adapted in its extension to arbitrary alphabets. As the extension to larger alphabets is strongly based on the construction for the binary case, we give the construction for the binary case here and sketch out the extension to arbitrary alphabets in Appendix D. As for Theorem 8, due to its extensive length, the full formal proof can be found in [27].

The idea is relatively similar to the idea for the proof of Theorem 8 (or the proof in [3]) in that is utilizes a pattern (α, ℓ_α) in which all words with a specific structure can be obtained, in particular, special encodings of valid computations of the specific universal Turing machine

U over any input I, as defined in Appendix B, and a pattern (β, ℓ_β) from which we can obtain all words that are in $L_{NE}(\alpha, \ell_\alpha)$ but these encodings of valid computations. As this result only shows undecidability of the inclusion problem, we do not have the property $L_{NE}(\beta, \ell_\beta) \subseteq L_{NE}(\alpha, \ell_\alpha)$ and only use the case distinction whether $L_{NE}(\alpha, \ell_\alpha) \subseteq L_{NE}(\beta, \ell_\beta)$ to decide the emptiness of $\texttt{ValC}_U(I)$.

We continue with the formal construction used to reduce the emptiness problem of the universal Turing machine U to the inclusion problem of terminal-free non-erasing pattern languages with length constraints in the binary case. After hat, we give a sketch of the idea on why this construction works. For the binary case, we essentially take slight adaptations of the constructed patterns from [3] but swap out any occurrence of the letter 0 with a new variable x_0 and any occurrence of the letter # with a new variable $x_\#$ (also similar, to some extend, to the idea in [32]). In that sense, the patterns look very similar to the constructed patterns in [3], but do not contain any terminals. With length constraints, we can restrict the obtainable words in such a way that the inclusion property $L_{NE}(\alpha, \ell_\alpha) \subseteq L_{NE}(\beta, \ell_\beta)$ decides the emptiness of $\texttt{ValC}_U(I)$.

First, we construct (β, ℓ_β), as the construction of (α, ℓ_α) is based on it. We define β by

$$\beta = x_0\ x_a x_b\ x_\#^5\ x_a x_1 ... x_{\mu+1} x_b\ x_\#^5\ r_1 \hat{\beta}_1 r_2 \hat{\beta}_2\ ...\ r_{\mu+1} \hat{\beta}_{\mu+1} r_{\mu+2}$$

for new and independent variables $x_0, x_\#, x_a, x_b, x_1, x_2, ..., x_{\mu+1}, r_1, r_2, ..., r_{\mu+2} \in X$ and terminal-free patterns $\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_{\mu+1} \in X^+$. Notice that $\mu \in \mathbb{N}$ is a number given by the construction in [3]. For all $i \in [\mu + 1]$ we say that

$$\hat{\beta}_i = x_0 x_i^4 x_0\ \gamma_i\ x_0 x_i^4 x_0\ \delta_i\ x_0 x_i^4 x_0$$

for terminal-free patterns $\gamma_i, \delta_i \in X^+$ that are defined later. The length constraints ℓ_β are defined by the system

- $x_0 = 1$, $x_\# = 1$, $x_a + x_b = \mu + 2$, and
- $x_i = 1$ for all $i \in [\mu + 1]$

Let $\tau : (\Sigma \times X)^* \to X^*$ be a homomorphism defined by $\tau(0) = x_0$, $\tau(\#) = x_\#$, and $\tau(x) = x$ for all $x \in X$. Then we say that for all $i \in [\mu]$ we have $\gamma_i = \tau(\gamma_i')$ and $\delta_i = \tau(\delta_i')$ for γ_i' and δ_i' being the patterns used in the construction in [3]. The specific construction of each such pattern is not relevant here and for details we refer to [3]. What is important is that we use the same patterns as in [3] but map them to terminal-free patterns that have the variable x_0 instead of each occurrence of a terminal symbol 0 and the variable $x_\#$ instead of each occurrence of #.

For the missing case of $\mu + 1$, we define

$$\gamma_{\mu+1} = x_0^{|I|+4} y_{\mu+1}$$
$$\delta_{\mu+1} = y'_{\mu+1}$$

for new and independent variables $y_{\mu+1}, y'_{\mu+1} \in X$ and I being the encoding of the initial configuration of U as in [3]. This will be used to obtain all substitutions $h(\alpha)$ where we have $h(x_0) = h(x_\#)$ in $L_{NE}(\beta, \ell_\beta)$. Now, we construct (α, ℓ_α). α is defined by

$$\alpha = x_0^{\mu+3}\ x_\#^5\ x_0^{\mu+1} x_\# x_0^{\mu+1}\ x_\#^5\ tv\ x_0 \alpha_1 x_0\ v\ x_0 \alpha_2 x_0\ vt$$

for some patterns α_1 and α_2, x_0 and $x_\#$ defined as in β, $v = x_0 x_\# x_\# x_\# x_0$ and $t = \psi(r_1 \hat{\beta}_1 r_2 ... r_{\mu+1} \hat{\beta}_{\mu+1} r_{\mu+2})$ with $\psi : X^* \to X^*$ defined by $\psi(x_0) = x_0$, $\psi(x_\#) = x_\#$, and $\psi(x) = x_0$ for all other $x \in \text{var}(\beta)$. We define α_1 and α_2 as $\alpha_1 = \tau(\alpha_1')$ and $\alpha_2 = \tau(\alpha_2')$ where α_1' and α_2' are the patterns given in [3, Section 4.4] for the second case.

The most important fact we take from there is that $\alpha_1' = \#\#I\#\#x\#0^60^10\#\#$ for some new and independent variable $x \in X$ and some encoded initial configuration I. By that, we see that $\tau(\alpha_1')$ starts with $x_\#x_\#\tau(I)x_\#x_\#$ which is used later on in this adaptation of the proof. The length constraints ℓ_α are defined by the equations $x_0 = 1$ and $x_\# = 1$. Notice that the x in α_1 has no length constraint. With that, we can now sketch out why this construction works in the binary case.

Notice, if for some $h \in H_{\ell_\alpha}$ we have $h(x_0) \neq h(x_\#)$, then we immediately obtain essentially the same construction as in [3] and can rely on their proof that shows that these patterns can be used to decide the emptiness of $\texttt{ValC}_U(I)$. That leaves us with the case $h(x_0) = h_(x_\#)$. But then, we notice that $h(\alpha_1)$ starts with a unary word of length at least $|x_\#x_\#\tau(I)x_\#x_\#| = |I| + 4$ that has the form $h(\alpha_1) = h(x_\#)^{|I|+4} = h(x_0)^{|I|+4}$. So, we can use $\hat{\beta}_{\mu+1}$ to obtain $h(vx_0\alpha_1x_0vx_0\alpha_2x_0v)$ and use the rest of (β, ℓ_β) to obtain the surrounding factors just as in [3]. So, if $h(x_\#) = h(x_0)$, then we always have $h(\alpha) \in L_{NE}(\beta, \ell_\beta)$. Hence, by the result from [3], there only exists some $h \in H_{\ell_\alpha}$ for which we have $h(\alpha) \notin L_{NE}(\beta, \ell_\beta)$ if and only if $\texttt{ValC}_U(I) \neq \emptyset$, concluding this reduction for the binary case.

To extend this construction to arbitrary alphabets, we can introduce for each additional letter $\mathtt{a} \in \Sigma$ a new variable $x_\mathtt{a}$ in the patterns α and β, resulting in variables $x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}$ for w.l.o.g. $\Sigma = \{0, \#, \mathtt{a}_1, ..., \mathtt{a}_\sigma\}$. With length constraints, we can restrict the length of each substitution on these variables to 1. The construction can be extended in such a way that if any two variables of $x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}$ are substituted by the same letter, then there always exists some $\hat{\beta}_i$ which can be used to obtain anything obtained in $h(vx_0\alpha_1x_0vx_0\alpha_2x_0v)$. In a second extension we can ensure that we can also always find some $\hat{\beta}_i$ if any of the letters obtained by $h(x_{\mathtt{a}_1}...x_{\mathtt{a}_\sigma})$ appears later on somewhere in $h(\alpha_1)$ or $h(\alpha_2)$. By that, we essentially can only obtain words not in $L_{NE}(\beta, \ell_\beta)$, if only the letters obtained in $h(x_\#)$ and $h(x_0)$ are used in $h(\alpha_1)$ and $h(\alpha_2)$. This restricts us to the same cases as in [3] and allows us to, again, rely on their arguments. As mentioned before, the construction of the extension to arbitrary alphabets is outlined in Appendix D. The full formal proof is given in [27]. ◀

▶ Remark 13. Indeed, by Corollary 3, Theorem 12 also implies the undecidability of the inclusion problem of terminal-free erasing pattern languages with length constraints. Hence, this is an additional approach to obtain that result other than Theorem 8.

Whether the equivalence problem of non-erasing pattern languages with length constraints is decidable, is left as an open question with no specific conjecture so far. Interestingly, also in the case of regular constraints, no definite answer for the decidability of this problem could be given, yet (cf. [26]). A summary of the current results can be found in Table 1 at the end of the final discussion.

4 Result for Pattern Languages with Regular and Length Constraints

As there are still open problems for pattern languages with regular (or length) constraints, i.e., the equivalence problem for non-erasing pattern languages with regular (or length constraints), finding an answer to this problem for a larger class of languages using both constraint-types is well motivated. Indeed, considering pattern languages with regular and length constraints suffices to obtain the undecidability of that problem. That is noticeable, since this problem is trivially decidable for patterns without any constraints. We begin with mentioning all results for decision problems on pattern languages with regular and length constraints that we can immediately obtain from pervious results.

For the membership, we immediately obtain NP-completeness for all cases, i.e., erasing- and non-erasing as well as terminal-free and general.

▶ **Corollary 14.** *Let* $(\alpha, r_\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$ *and* $w \in \Sigma^*$*. The decision problem of whether* $w \in L_X(\alpha, r_\alpha, \ell_\alpha)$ *for* $X \in \{E, NE\}$ *is NP-complete, even if the considered pattern* α *is terminal-free.*

Indeed, NP-hardness follows, as before, from the NP-hardness of pattern languages without any constraints. NP-containment follows from the fact NP-containment can be checked in the cases of pattern languages with regular constraints as well as pattern languages with length constraints and that no additional properties have to be checked.

As this class of pattern languages utilizes regular constraints, we immediately obtain the property mentioned before, and shown in [26], that we can always find a terminal-free pattern with regular constraints that expresses the same language as a general pattern with regular constraints.

▶ **Proposition 15** ([26])**.** *Let* $(\alpha, r_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg}}$ *be some pattern with regular constraints. Then, there exists some terminal-free pattern* $\beta \in X^*$ *and a regular constraint* r_β *such that* $L_X(\alpha, r_\alpha) = L_X(\beta, r_\beta)$ *for* $X \in \{E, NE\}$*.*

Using the same logic as used to show that property, we immediately obtain the same for pattern languages with regular and length constraints, as, given some pattern with regular and length constraints $(\alpha, r_\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$, one can introduce for each terminal letter $\mathtt{a}$ a new variable $x_\mathtt{a}$ and set $L(x_\mathtt{a}) = \{\mathtt{a}\}$ while not changing any length constraints.

▶ **Corollary 16.** *Let* $(\alpha, r_\alpha, \ell_\alpha) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$ *be some pattern with regular- and length constraints. Then, there exists some terminal-free pattern* $\beta \in X^*$*, a regular constraint* r_β*, and a length constraint* ℓ_β *such that* $L_X(\alpha, r_\alpha, \ell_\beta) = L_X(\beta, r_\beta, \ell_\beta)$ *for* $X \in \{E, NE\}$*.*

Additionally, as the inclusion problem is undecidable for all cases (E, NE, terminal-free, general) for pattern languages with regular or with length constraints, the same follows immediately for pattern languages with regular and length constraints.

▶ **Corollary 17.** *Let* $(\alpha, r_\alpha, \ell_\alpha), (\beta, r_\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$*. It is undecidable to answer whether* $L_X(\alpha, r_\alpha, \ell_\alpha) \subseteq L_X(\beta, r_\beta, \ell_\beta)$ *for* $X \in \{E, NE\}$ *for all alphabets* Σ *with* $|\Sigma| \geq 2$*, even if* α *and* β *are restricted to be terminal-free.*

The same follows for the equivalence problem of general and terminal-free erasing pattern languages with regular- and length constraints, as also here we have the undecidability for both cases, regular constraints and length constraints, respectively. We obtain the following.

▶ **Corollary 18.** *Let* $(\alpha, r_\alpha, \ell_\alpha), (\beta, r_\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$*. It is undecidable to answer whether* $L_E(\alpha, r_\alpha, \ell_\alpha) = L_E(\beta, r_\beta, \ell_\beta)$ *for all alphabets* Σ *with* $|\Sigma| \geq 2$*, even if* α *and* β *are restricted to be terminal-free.*

For all results obtained so far, no disjunction of systems of linear inequalities have had to be applied. For the equivalence problem of non-erasing pattern languages with regular and length constraints, making use of disjunctions in length constraints in addition to regular constraints allows us to obtain undecidability in this case. The following theorem concludes the third and final main result of this paper.

▶ **Theorem 19.** *Let* $(\alpha, r_\alpha, \ell_\alpha), (\beta, r_\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma,\mathcal{C}_{Reg,Len}}$*. It is undecidable to answer whether* $L_{NE}(\alpha, r_\alpha, \ell_\alpha) = L_{NE}(\beta, r_\beta, \ell_\beta)$ *for all alphabets* Σ *with* $|\Sigma| \geq 2$*, even if* α *and* β *are restricted to be terminal-free.*

Proof. As for the proofs of Theorem 8 and Theorem 12, due to its length, the full version of this proof can be found in [27]. Here, we will only give a rough sketch of the construction and the idea behind it.

The general idea of the proof works similar to the proof of Theorem 8 and the proof of the main result in [26], but needs significant adaptations to work for non-erasing pattern languages. Similar to the proof of Theorem 8, we reduce the emptiness problem for nondeterministic 2-counter automata to the equivalence problem of non-erasing pattern languages with regular and length constraints. So, given some nondeterministic 2-counter automaton without input A, we construct two patterns with regular and length constraints $(\alpha, r_\alpha, \ell_\alpha)$ and $(\beta, r_\beta, \ell_\beta)$ such that $L_{NE}(\alpha, r_\alpha, \ell_\alpha) = L_{NE}(\beta, r_\beta, \ell_\beta)$ if and only if $\texttt{ValC}(A) = \emptyset$. In contrast to the proofs of Theorem 8 and Theorem 12, this time, the length constraints ℓ_β use a disjunction of systems of linear diophantine inequalities while $(\alpha, r_\alpha, \ell_\alpha)$ has no regular or length constraints at all. Again, $(\alpha, r_\alpha, \ell_\alpha)$ serves as a pattern to obtain any word with a specific structure and $(\beta, r_\beta, \ell_\beta)$ serves as a pattern to obtain words with that specific structure that fulfill given properties. These properties are, again, defined by sub-patterns in $(\beta, r_\beta, \ell_\beta)$ that are called predicates. The regular and length constraints r_β and ℓ_β can be set in such a way that always a single predicate has to be chosen, while the variables outside of these predicates have to be substituted in a very fixed way. This results in words that have equal prefixes and suffixes in comparison to all words obtained from $(\alpha, r_\alpha, \ell_\alpha)$, while the middle part is then checked with the predicates. These predicates are constructed similar to the ones in [26], just with slight adaptations, as now we have to use non-erasing substitutions instead of erasing ones. In the end, it is shown, that the predicates cover all words where the middle part is not an encoding of a valid computation in $\texttt{ValC}(A)$. We continue with the general construction.

As we need β to define α, we will begin with the construction of $(\beta, r_\beta, \ell_\beta)$. We define β by

$$\beta = r_1\ \hat{\beta}_1\ r_2\ ...r_\mu\ \hat{\beta}_\mu\ r_{\mu+1}$$

for new and independent variables r_1 to $r_{\mu+1}$ for some $\mu \in \mathbb{N}$ to be specified later. For each $i \in [\mu]$, we define $\hat{\beta}_i = v\ \gamma_i\ v$ for $v = 0\#^3 0$ and γ_i being some terminal-free pattern, also to be defined later. For each $i, j \in [\mu]$ with $i \neq j$ we assume by the following construction that $\mathrm{var}(\gamma_i) \cap \mathrm{var}(\gamma_j) = \varepsilon$, i.e., each variable occurring in one γ_i does not occur anywhere else in the pattern. Also, for any word obtained by γ_i, we assume that it is either $0^{|\gamma_i|}$ or $0u0$ for some $u \in \Sigma^*$ by the construction for this proof. Before giving the constraints r_β and ℓ_β, we continue with the construction of α. We define α by

$$\alpha = t\ v\ 0\ \alpha_1\ 0\ v\ t^2$$

for a new and independent variable α_1, $v = 0\#^3 0$, and a word $t \in \Sigma^*$ that is obtained by the following morphism. Define $\psi : (\Sigma \times X)^* \to \Sigma^*$ by $\psi(0) = 0$, $\psi(\#) = \#$, and $\psi(x) = 0$ for all $x \in X$. Then, we say that $t = \psi(\beta)$. The constraints r_α and ℓ_α are empty, hence $L_{NE}(\alpha, r_\alpha, \ell_\alpha) = L_{NE}(\alpha) = \{tv\} \cdot \Sigma^+ \cdot \{vt^2\}$. We proceed with the definition of the constraints r_β and ℓ_β. For each $i \in [\mu + 1]$, we define the language of the variable r_i by

$$L(r_i) = \{0,\ \psi(r_i\hat{\beta}_i r_{i+1}...r_\mu\hat{\beta}_\mu r_{\mu+1}),\ t\psi(r_1\hat{\beta}_1 r_2...r_{i-1}\hat{\beta}_{i-1}r_i)\}.$$

Thus, notice, that each r_i may only be substituted to one of 3 words: Either 0, a specific suffix of t or a specific prefix of t^2. Additionally, for all variables $x \in \mathrm{var}(\beta)$ in β we add the constraint that $\# \notin L(x)$, i.e., no variable can be substituted to just the letter $\#$. Clearly, any language can be intersected with another regular language to obtain that constraint.

The specific regular and length constraints for each γ_i will be introduced later in the proof. For the length constraints ℓ_β, we construct a disjunction of systems of linear (diophantine) inequalities. Assume w.l.o.g. that for each $i \in [\mu]$ we have $\text{var}(\gamma_i) := \{x_{i,1}, x_{i,2}, ..., x_{i,n_i}\}$ for some $n_i \in \mathbb{N}$. Then, we define for each $i \in [\mu]$, for each $j \in [n_i]$ of the resulting n_i the system $\ell_{\beta,i,j}$ defined by:

- $r_i = |\psi(r_i\hat{\beta}_i r_{i+1}...r_\mu\hat{\beta}_\mu r_{\mu+1})|$
- $r_{i+1} = |t\ \psi(r_1\hat{\beta}_1 r_2...r_i\hat{\beta}_i r_{i+1})|$
- $r_k = 1$ for all $k \in [\mu+1] \setminus \{i, i+1\}$
- $x_{i,j} \geq 2$
- $x_{k,k'} = 1$ for all $k \in [\mu]$ and $x'_k \in \text{var}(\gamma_k)$ for $k \neq i$

Additionally, for some $\ell_{\beta,i,j}$, there are additional length constraints on the specific variables in occurring in γ_i. These are defined properly in the formal proof in [27] and do not interfere with the constraint that γ_i may be substituted to $0^{|\gamma_i|}$ in certain cases. We then define ℓ_β by $\ell_\beta = \ell_{\beta,1,1} \vee ... \vee \ell_{\beta,1,n_1} \vee \ell_{\beta,2,1} \vee ... \vee \ell_{\beta,2,n_2} \vee ... \ ... \vee \ell_{\beta,\mu_1,1} \vee ... \vee \ell_{\beta,\mu,n\mu}$. This constraint results in that exactly two subsequent r_i and r_{i+1} are substituted by words longer than the length of 1 while all others have exactly length 1. The other constraints result in that all variables occurring in some pattern γ_j with $j \neq i$ have to be substituted by length 1 and that at least one variable occurring in γ_i has to be substituted to length 2. These constraints serve selecting one γ_i to obtain substitutions of α_1 as seen in the following parts.

As in the proof of Theorem 8, first, we have $L_E(\beta, \ell_\beta) \subseteq L_{NE}(\alpha, \ell_\alpha)$. Due to the regular and length constraints, it can be shown that each word $w \in L_{NE}(\beta, \ell_\beta)$ has the form $w = tv0u0vtt$ for some $u \in \Sigma^*$. Hence, by setting $h(\alpha_1) = u$, we always find some $h \in H_{\ell_\alpha}$ such that $h(\alpha) = w$. On the other hand, due to the regular and length constraints, it can also be shown that the $v0u0v$ part of w must be obtained by some $\hat{\beta}_i$, $i \in [\mu]$, while the prefix t is obtained by $r_1\hat{\beta}_1...r_i$ and the suffix tt is obtained by $r_{i+1}\hat{\beta}_{i+1}...r_{\mu+1}$. Hence, for some $h \in H_{\ell_\alpha}$, to find some $h' \in H_{\ell_\beta}$ such that $h(\alpha) = h(\beta)$, there must exist some $i \in [\mu]$ such that $h'(\hat{\beta}_i) = h(v0u0v)$, i.e., $h'(\gamma_i) = h(0\alpha_1 0)$. We can construct β in such a way that this is only possible, when $h(\alpha_1)$ is not an encoding of a valid computation of A. The specific construction of all γ_i's is similar to the ones given in [26]. Due to space constraints, see [27] for more details on the construction for this proof.

Following both arguments, we see that there exists some $w \in L_{NE}(\alpha, \ell_\alpha)$ such that $w \notin L_{NE}(\beta, \ell_\beta)$ if and only if there is some valid computation for A, i.e., $\texttt{ValC}(A) \neq \emptyset$, concluding this result. As mentioned before, see [27] for a full formal proof of this result. ◀

The construction of the proof for Theorem 19 uses patterns that consist partly of terminal-symbols. Using Corollary 16, however, we know that we can always find terminal-free patterns that produce the exact same languages. Hence, the result also follows immediately for terminal-free patterns. This concludes all results regarding decision problems for pattern languages with regular- and length constraints. Notice that by these results all main decision problems (i.e., membership, inclusion, and equivalence) for this class of pattern languages have an answer (summarized in Table 1 at the end of the final discussion).

5 Further Discussion

In this paper, we extended the research regarding decision problems on pattern languages. To get closer to an answer for the main open questions, e.g., the equivalence problem for erasing pattern languages or the larger alphabet cases of the terminal-free inclusion problem for non-erasing pattern languages, extending the research from [26], we now introduced length constraints as an alternative approach. With the results provided in this paper, we see that

nearly all problems regarding pattern langauges with length constraints, except membership, are indeed undecidable (in particular the open cases for patterns without constraints). As mentioned at the end of Section 3, similar to [26], the decidability of the equivalence problem for non-erasing pattern languages with length constraints remains an open problem.

By the help of some core ideas from [19, 11, 3, 32], adapted constructions using length (and regular) constraints could be constructed to obtain these results. Regarding our first main result, Theorem 8, similar to [26], the main novelty for the proof is found in the way the constructed checker pattern (β, ℓ_β) does not produce *too much* anymore, while still being able to handle everything from (α, ℓ_α) in a controlled manner. Length constraints helped to reduce the overhead in (β, ℓ_β) to enable the approaches from the inclusion problem proof to be used for the equivalence problem. Now, due to such a construction being shown to actually work for two different constraint types, e.g., regular and length constraints respectively, finding a way to adapt the existing proofs for pattern languages without any constraints in such a way that the same can be achieved here, poses a final challenge. If this can actually be done, the main open problem for pattern languages would be settled. So far, we do not know how likely such a construction is to be found. In addition, the use of length constraints also helped to achieve undecidability for decidable problems regarding patterns without constraints, i.e., the terminal-free inclusion and equivalence problems for erasing pattern languages. Hence, the expressiveness obtained using these constraints and, thus, needed to obtain undecidability of the equivalence problem for erasing pattern languages using the approach posed in this paper, might actually be unfeasible. Regarding the terminal-free inclusion problem in the non-erasing case, due to the undecidability of the binary case having already been shown for patterns without any constraints [32], it is not unlikely that undecidability for larger alphabets can be achieved. Using the approach proposed in the current paper, we would need to somehow be able to restrict substitutions of some variables to single letters and distinct symbols in one case, and enable certain wildcards in any other case, similar to [32], where a property of binary words is used to *flip a switch*, determining where a word obtained from α has to be obtained in β. So far, this remains open.

Regarding the equivalence problem for non-erasing pattern languages for patterns with length constraints, we observe that this problem must be harder than the trivially decidable variant for non-erasing pattern languages without any constraints. This results from the fact that there are several NP-hard problems that are expressible solely by unions of linear diophantine equations. Some of which are, e.g., the subset sum problem, the knapsack problem, or integer programming [20]. Just to give a rough idea, we can easily embed, say, a positive integer subset sum instance (S, T) with $S = \{S_1, ..., S_n\}$, for $S_i \in \mathbb{N}$ with $i \in [n]$, and $T \in \mathbb{N}$ to an instance of the equivalence problem for non-erasing pattern languages with length constraints by setting the pattern $\alpha = x_1 ... x_n$, for variables $x_1, ..., x_n \in X$, defining ℓ_α by setting $(x_i = S_i + 1) \vee (x_i = 1)$, for $i \in [n]$, and setting $\sum_{i \in [n]} x_i = T + n$. Then, we set $\beta = y$, for a variable $y \in X$, with ℓ_β being defined by $y = T + n$. Then, $L_{NE}(\beta, \ell_\beta)$ only has words of length $T + n$ and $L_{NE}(\alpha, \ell_\alpha)$ only contains some words if (S, T) actually has a solution, as otherwise we cannot find a valid substitution for α. If it contains words, they have length $T + n$. So, both languages are equal if and only if there exists a solution for (S, T). Here, disjunctions allowed to embed the choice of setting any x_i to either length 1 or length $S_i + 1$. Hence, we obtain the following.

▶ **Proposition 20.** *Let* $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$ *and let* Σ *be any alphabet with at least* $|\Sigma| \geq 1$*. Deciding whether we have* $L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\beta, \ell_\beta)$ *is NP-hard, even when we are restricted to terminal-free patterns* α *and* β*.*

Notice that the previous result depended on the fact that we allow disjunctions in the given length constraints. The following result shows that at least in the case of unary alphabets, where we basically restrict the problem to a number setting, we can obtain NP-hardness without using disjunctions by a reduction from the the 3SAT problem. Due to length restrictions, the proof of Proposition 21 can be found in Appendix E.

▶ **Proposition 21.** *Let* $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$ *where* ℓ_α *and* ℓ_β *each use no disjunctions. Deciding whether we have* $L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\beta, \ell_\beta)$ *is NP-hard for alphabets* Σ *with* $|\Sigma| = 1$*, even when we are restricted to terminal-free patterns* α *and* β*.*

For all other problems, we obtained an answer and notice that the most prominent open problems, i.e., the equivalence problem for erasing pattern languages and the inclusion problem for terminal-free non-erasing pattern languages for alphabets Σ with $|\Sigma| \geq 3$ are indeed undecidable for patterns with length constraints. For patterns with regular and length constraints, we have even seen the undecidability of the equivalence problem of non-erasing pattern languages, concluding all open problems there. A final overview of the current state of results for decision problems on patterns over various constraints can be found in Table 1. We propose the following open question for which we have no definite conjecture so far.

▶ **Question 1.** *Given* $(\alpha, \ell_\alpha), (\beta, \ell_\beta) \in \mathtt{Pat}_{\Sigma, \mathcal{C}_{Len}}$*, is it generally decidable to answer whether* $L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\beta, \ell_\beta)$*?*

Table 1 Summary of the current state of results regarding the main decision problems for pattern languages with different constraints (NPC = NP-Complete, UD = Undecidable, E = Erasing, NE = Non-Erasing, T.F. = Terminal-Free, Gen. = General, ($\in$) means membership problem, ($\subseteq$) means inclusion problem, and ($=$) means equivalence problem.). The results for No Constraints and Regular Constraints are based on previous research mentioned in the introduction. The results for Length Constraints as well as Regular and Length Constraints summarize the results of this paper.

	No Constraints		Len-Constraints		Reg-Constraints		RegLen-Constraints	
	Gen.	T.F.	Gen.	T.F.	Gen.	T.F.	Gen.	T.F.
E ($\in$)	NPC	NPC	NPC	NPC	NPC	NPC	NPC	NPC
E ($\subseteq$)	UD	NPC	UD	UD	UD	UD	UD	UD
E ($=$)	**Open**	NPC	UD	UD	UD	UD	UD	UD
NE ($\in$)	NPC	NPC	NPC	NPC	NPC	NPC	NPC	NPC
NE ($\subseteq$)	UD	UD[2]	UD	UD	UD	UD	UD	UD
NE ($=$)	P	P	**Open**[3]	**Open**[3]	**Open**[4]	**Open**[4]	UD	UD

References

1 Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980. doi:10.1016/0022-0000(80)90041-0.

2 Pablo Barceló, Leonid Libkin, Anthony W. Lin, and Peter T. Wood. Expressive languages for path queries over graph-structured data. *ACM Trans. Database Syst.*, 37(4), December 2012. doi:10.1145/2389241.2389250.

[2] Undecidable in the binary case by [32], open for larger alphabets.
[3] Gen. and T.F. NP-hard as described above. An upper bound is unknown (may as well be undecidable).
[4] Depending on regular languages representation, at least PSPACE-hard [26].

3 Joachim Bremer and Dominik D. Freydenberger. Inclusion problems for patterns with a bounded number of variables. *Information and Computation*, 220-221:15–43, 2012. doi:10.1016/J.IC.2012.10.003.

4 Joel D. Day, Pamela Fleischmann, Florin Manea, and Dirk Nowotka. Local Patterns. In Satya Lokam and R. Ramanujam, editors, *FSTTCS 2017*, volume 93 of *LIPIcs*, pages 24:1–24:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.FSTTCS.2017.24.

5 Andrzej Ehrenfeucht and Grzegorz Rozenberg. Finding a homomorphism between two words is NP-complete. *Inf. Process. Lett.*, 9(2):86–88, 1979. doi:10.1016/0020-0190(79)90135-2.

6 Henning Fernau, Florin Manea, Robert Mercaş, and Markus L. Schmid. Revisiting Shinohara's algorithm for computing descriptive patterns. *TCS*, 733:44–54, 2018. Special Issue on Learning Theory and Complexity. doi:10.1016/J.TCS.2018.04.035.

7 Pamela Fleischmann, Sungmin Kim, Tore Koß, Florin Manea, Dirk Nowotka, Stefan Siemer, and Max Wiedenhöft. Matching patterns with variables under Simon's congruence. In Olivier Bournez, Enrico Formenti, and Igor Potapov, editors, *Reachability Problems*, pages 155–170, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-45286-4_12.

8 Dominik D. Freydenberger. A logic for document spanners. *Theory of Computing Systems*, 63(7):1679–1754, September 2018. doi:10.1007/S00224-018-9874-1.

9 Dominik D. Freydenberger and Mario Holldack. Document spanners: From expressive power to decision problems. *Theory of Computing Systems*, 62(4):854–898, May 2017. doi:10.1007/S00224-017-9770-0.

10 Dominik D. Freydenberger and Liat Peterfreund. The theory of concatenation over finite models. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *ICALP 2021, Proceedings*, volume 198 of *LIPIcs*, pages 130:1–130:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.130.

11 Dominik D. Freydenberger and D. Reidenbach. Bad news on decision problems for patterns. *Information and Computation*, 208(1):83–96, January 2010. doi:10.1016/J.IC.2009.04.002.

12 Dominik D. Freydenberger and Markus L. Schmid. Deterministic regular expressions with back-references. *Journal of Computer and System Sciences*, 105:1–39, 2019. doi:10.1016/J.JCSS.2019.04.001.

13 Dominik D. Freydenberger and Nicole Schweikardt. Expressiveness and static analysis of extended conjunctive regular path queries. *Journal of Computer and System Sciences*, 79(6):892–909, 2013. JCSS Foundations of Data Management. doi:10.1016/J.JCSS.2013.01.008.

14 Paweł Gawrychowski, Florin Manea, and Stefan Siemer. Matching Patterns with Variables Under Hamming Distance. In Filippo Bonchi and Simon J. Puglisi, editors, *MFCS 2021*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:24, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.MFCS.2021.48.

15 Paweł Gawrychowski, Florin Manea, and Stefan Siemer. Matching patterns with variables under edit distance. In Diego Arroyuelo and Barbara Poblete, editors, *String Processing and Information Retrieval*, pages 275–289, Cham, 2022. Springer International Publishing.

16 Michael Geilke and Sandra Zilles. Learning relational patterns. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 84–98, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-24412-4_10.

17 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, January 1978. doi:10.1145/322047.322058.

18 Tao Jiang, Efim Kinber, Arto Salomaa, Kai Salomaa, and Sheng Yu. Pattern languages with and without erasing. *International Journal of Computer Mathematics*, 50(3-4):147–163, 1994.

19 Tao Jiang, Arto Salomaa, Kai Salomaa, and Sheng Yu. Decision problems for patterns. *Journal of Computer and System Sciences*, 50(1):53–63, February 1995. doi:10.1006/JCSS.1995.1006.

20 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.

21 Takeshi Koshiba. Typed pattern languages and their learnability. In Paul Vitányi, editor, *Computational Learning Theory*, pages 367–379, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. doi:10.1007/3-540-59119-2_192.

22 Anthony Widjaja Lin and Rupak Majumdar. Quadratic word equations with length constraints, counter systems, and presburger arithmetic with divisibility. In *Automated Technology for Verification and Analysis*, 2018.

23 M. Lothaire. *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press, 2 edition, 1997.

24 Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

25 Turlough Neary and Damien Woods. Four small universal turing machines. *Fundam. Inf.*, 91(1):123–144, January 2009. doi:10.3233/FI-2009-0036.

26 Dirk Nowotka and Max Wiedenhöft. The equivalence problem of E-pattern languages with regular constraints is undecidable. In Szilárd Zsolt Fazekas, editor, *Implementation and Application of Automata*, pages 276–288, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-71112-1_20.

27 Dirk Nowotka and Max Wiedenhöft. The equivalence problem of E-pattern languages with length constraints is undecidable, 2025. doi:10.48550/arXiv.2411.06904.

28 Enno Ohlebusch and Esko Ukkonen. On the equivalence problem for E-pattern languages. In *MFCS 1996*, pages 457–468. Springer Berlin Heidelberg, 1996. doi:10.1007/3-540-61550-4_170.

29 Daniel Reidenbach. On the equivalence problem for E-pattern languages over small alphabets. In *DLT*, pages 368–380. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30550-7_31.

30 Daniel Reidenbach. On the learnability of E-pattern languages over small alphabets. In *Learning Theory*, pages 140–154. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-27819-1_10.

31 Daniel Reidenbach. An examination of Ohlebusch and Ukkonen's conjecture on the equivalence problem for E-pattern languages. *J. Autom. Lang. Comb.*, 12(3):407–426, January 2007. doi:10.25596/JALC-2007-407.

32 Aleksi Saarela. Hardness Results for Constant-Free Pattern Languages and Word Equations. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 140:1–140:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.ICALP.2020.140.

33 Markus L. Schmid. *On the membership problem for pattern languages and related topics*. PhD thesis, Loughborough University, January 2012. URL: https://repository.lboro.ac.uk/articles/thesis/On_the_membership_problem_for_pattern_languages_and_related_topics/9407606.

34 Markus L. Schmid and Nicole Schweikardt. Document spanners - A brief overview of concepts, results, and recent developments. In *PODS '22: International Conference on Management of Data*, pages 139–150. ACM, 2022. doi:10.1145/3517804.3526069.

35 Takeshi Shinohara. *Polynomial time inference of extended regular pattern languages*, pages 115–127. Springer Berlin Heidelberg, 1983.

36 Takeshi Shinohara and Setsuo Arikawa. *Pattern inference*, pages 259–291. Springer Berlin Heidelberg, 1995. doi:10.1007/3-540-60217-8_13.

A Definition of Nondeterministic 2-Counter Automata without Input

A *nondeterministic 2-counter automaton without input* (see e.g. [17]) is a 4-tuple $A = (Q, \delta, q_0, F)$ which consists of a set of states Q, a transition function $\delta : Q \times \{0,1\}^2 \to \mathcal{P}(Q \times \{-1, 0, +1\}^2)$, an initial state $q_0 \in Q$, and a set of accepting states $F \subseteq Q$. A *configuration* of

A is defined as a triple $(q, m_1, m_2) \in Q \times \mathbb{N} \times \mathbb{N}$ in which q indicates the current state and m_1 and m_2 indicate the contents of the first and second counter. We define the relation $\vdash_A$ on $Q \times \mathbb{N} \times \mathbb{N}$ by δ as follows. For two configurations (p, m_1, m_2) and (q, n_1, n_2) we say that $(p, m_1, m_2) \vdash_A (q, n_1, n_2)$ if and only if there exist $c_1, c_2 \in \{0, 1\}$ and $r_1, r_2 \in \{-1, 0, +1\}$ such that

1. if $m_i = 0$ then $c_i = 0$, otherwise if $m_i > 0$, then $c_i = 1$, for $i \in \{1, 2\}$,
2. $n_i = m_i + r_i$ for $i \in \{1, 2\}$,
3. $(q, r_1, r_2) \in \delta(p, c_1, c_2)$, and
4. we assume if $c_i = 0$ then $r_i \neq -1$ for $i \in \{1, 2\}$.

Essentially, the machine checks in every state whether the counters equal 0 and then changes the value of each counter by at most one per transition before entering a new state. A *computation* is a sequence of configurations. An *accepting computation* of A is a sequence $C_1, ..., C_n \in (Q \times \mathbb{N} \times \mathbb{N})^n$ with $C_1 = (q_0, 0, 0)$, $C_i \vdash_A C_{i+1}$ for all $i \in \{1, ..., n-1\}$, and $C_n \in F \times \mathbb{N} \times \mathbb{N}$ for some $n \in \mathbb{N}$ with $n > 0$.

We *encode* configurations of A by assuming $Q = \{q_0, ..., q_e\}$ for some $e \in \mathbb{N}$ and defining a function $\text{enc} : Q \times \mathbb{N} \times \mathbb{N} \to \{0, \#\}^*$ by

$$\text{enc}(q_i, m_1, m_2) := 0^{x+i} \# 0^{c_1 + y_2 \cdot m_1} \# 0^{c_2 + y_2 \cdot m_2}$$

for some numbers $x, c_1, y_2, c_1, y_2 \in \mathbb{N}$. The values for these numbers depend on the construction of the respective proofs and are not specified here. Encodings of this kind are used to prove Theorem 8 and Theorem 19. This is extended to encodings of computations by defining for every $n \geq 1$ and every sequence $C_1, ..., C_n \in Q \times \mathbb{N} \times \mathbb{N}$

$$\text{enc}(C_1, ..., C_n) := \#\# \ \text{enc}(C_1) \ \#\# \ ... \ \#\# \ \text{enc}(C_n) \ \#\#.$$

For some nondeterministic 2-counter automaton without input A, define the set of encodings of accepting computations

$$\texttt{ValC}(A) := \{\text{enc}(C_1, ..., C_n) \mid C_1, ..., C_n \text{ is an accepting computation of } A\}$$

and let $\texttt{InvalC}(A) = \{0, \#\}^* \setminus \texttt{ValC}(A)$. The emptiness problem for deterministic 2-counter-automata is undecidable (cf. e.g. [17, 24]), thus it is also undecidable whether a nondeterministic 2-counter automaton without input has an accepting computation [11, 19]. That the emptiness problem for universal Turing machines is undecidable is a known fact.

B Definition of the Universal Turing Machine U

Here, we define the universal Turing machine U used in the referenced proof from [3] and referred to in the proof sketch of Theorem 12. Let $U = (Q, \Gamma, \delta)$ be the universal Turing machine $U_{15,2}$ wih 2 symbols and 15 states as described by Neary and Woods [25]. This machine hat the states $Q = \{q_1, ..., q_{15}\}$ and the tape alphabet $\Gamma = \{0, 1\}$. The transition function $\delta : \Gamma \times Q \to (\Gamma \times \{L, R\} \times Q) \cup \texttt{HALT}$ is given in Table 2.

We follow with the definition of encodings of computations of U as depicted in [3]. The following conventions are needed to discuss configurations of U. The tape content of any configuration of U is characterized by two infinite sequences $t_L = (t_{L,n})_{n \geq 0}$ and $t_R = (r_{R,n})_{n \geq 0}$ over Γ. The sequence t_L describes the *left side of the tape*, the sequence starting at the head position of U (including) and extending to the left. Analogously, t_R describes the *right side of the tape*, the sequence starting directly after the head position and extending to the right. A *configuration* $C = (q_i, t_L, t_R)$ of U is a triple consisting of a state q_i, a left side of the tape t_L and a right side of the tape t_R.

Let $e : \Gamma \to N$ be a function defined by $e(0) := 0$, $e(1) := 1$, and the extension to to infinite sequences $t = (t_n)_{n\geq 0}$ over Γ by $e(t) := \sum_{i=0}^{\infty} e(t_i)$. As in each configuration of U only a finite number of cells consist of no blank symbol (0), $e(t)$ is always finite and well-defined. Notice that we can always obtain the symbol that is closest to the head by $e(t) \mod 2$ (the symbol at the head position in the case of t_L and the symbol right of the head position in the case of t_R). By multiplying or dividing the encoding $e(t)$ by 2, each side can be lengthened or shortened, respectively. The *encoding of configurations of* U indirectly referred to in this paper is defined by

$$\mathrm{enc}_{NE}(q_i, t_L, t_R) = 0^7 0^{e(t_R)} \# 0^7 0^{e(t_L)} \# 0^{i+6}$$

for every configuration (q_i, t_L, t_R). Recall that $i > 0$ as $q_i \in \{q_1, ..., q_{15}\}$. A *computation* $\mathcal{C} = (C_1, ..., C_n)$ on U is a finite sequence of configurations of U. It is *valid* if $C_1 = I$ (I being some initial configuration), C_n is a halting configuration, and C_{i+1} is a valid successor configuration of C_i, for $i \in [n-1]$, as defined by δ. In [3], the notion is adopted that any possible configuration where both tape sides have a finite value under e is a valid successor configuration of a halting configuration. The *encoding of computations* of U is given analogously to the definition in the case of nondeterministic 2-counter automata without input, i.e., for some computation $\mathcal{C} = (C_1, ..., c_n)$, we have

$$\mathrm{enc}_{NE}(\mathcal{C}) = \#\# \,\mathrm{enc}_{NE}(C_1)\#\# \,\mathrm{enc}_{NE}(c_2)\#\# \ ... \ \#\# \,\mathrm{enc}_{NE}(C_n)\#\#.$$

Finally, also analogous to nondeterministic 2-counter automata without input, let

$$\mathtt{ValC}_U(I) = \{\mathrm{enc}_{NE}(\mathcal{C}) \mid \mathcal{C} \text{ is a valid computation from } I \ \}.$$

Table 2 Transition table of U, i.e., definition of δ, as it is given in [3] or [25].

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
0	$(0, R, q_2)$	$(1, R, q_3)$	$(0, L, q_7)$	$(0, L, q_6)$	$(1, R, q_1)$	$(1, L, q_4)$	$(0, L, q_8)$	$(1, L, q_9)$
1	$(1, R, q_1)$	$(1, R, q_1)$	$(0, L, q_5)$	$(1, L, q_5)$	$(1, L, q_4)$	$(1, L, q_4)$	$(1, L, q_7)$	$(1, L, q_7)$
	q_9	q_{10}	q_{11}	q_{12}	q_{13}	q_{14}	q_{15}	
0	$(0, R, q_1)$	$(1, L, q_{11})$	$(0, R, q_{12})$	$(0, R, q_{13})$	$(0, L, q_2)$	$(0, L, q_3)$	$(0, R, q_{14})$	
1	$(1, L, q_{10})$	HALT	$(1, R, q_{14})$	$(1, R, q_{12})$	$(1, R, q_{12})$	$(0, R, q_{15})$	$(1, R, q_{14})$	

C Extension to Larger Alphabets in Theorem 8

To extend the proof idea for Theorem 8 to arbitrary larger alphabets Σ with $|\Sigma| > 2$, we fix w.l.o.g. $\Sigma = \{0, \#, \mathtt{a}_1, \mathtt{a}_2, ..., \mathtt{a}_\sigma\}$ (hence, $\sigma \geq 1$). We can construct adapted patterns with length constraints $(\alpha_\sigma, \ell_{\alpha_\sigma})$ and $(\beta_\sigma, \ell_{\beta_\sigma})$ in the following manner. The idea of this adapted construction is also similar to [11], but adapted to this setting and for terminal-free patterns. First, we define α_σ by

$$\alpha_\sigma = y_1 \ x_v \alpha_1 x_v \alpha_2 x_v \ y_2 \ x_v \ z \ (z')^2 \ (z_{\mathtt{a}_1})^2 \ (z_{\mathtt{a}_2})^2 \ ... \ (z_{\mathtt{a}_\sigma})^2 \ z \ x_v.$$

Notice, that only the parts containing the z's is changed. We construct the adapted length constraints ℓ_{α_σ} in the following manner:

- $x_v = 5$, $z = 1$, $z' = 1$,
- $z_{\mathtt{a}_i} = 1$ for all $i \in [\sigma]$

An immediate observation that can be made, as its done in [11], is the following:

▶ **Observation 22** ([11]). *Let $n \geq 1$, $x_1, ..., x_n \in X$, and $\mathtt{a}_1, \mathtt{a}_2, ..., \mathtt{a}_n \in \Sigma$. Consider the pattern*

$$p = x_1\ x_2x_2\ x_3x_3\ ...\ x_nx_n\ x_1.$$

Then, for each homomorphism h with $h(p) = \mathtt{a}_1\ \mathtt{a}_2\mathtt{a}_2\ \mathtt{a}_3\mathtt{a}_3\ ...\ \mathtt{a}_n\mathtt{a}_n\ \mathtt{a}_1$ we have that $h(x_i) = \mathtt{a}_i$ for all $i \in [n]$.

Now, we explain the changes necessary to create β_σ and ℓ_{β_σ}. First, for each η_i with $i \in [\mu]$, we change its form from $\eta_i = z_i z_i' z_i' z_i$ to

$$\eta_i = z_i\ z_i' z_i'\ z_{i,\mathtt{a}_1} z_{i,\mathtt{a}_1}\ ...\ z_{i,\mathtt{a}_\sigma} z_{i,\mathtt{a}_\sigma}\ z_i.$$

Now, instead of adding $\eta_{\mu+1}$ as before, we add a series of $n \in \mathbb{N}$ many pairs of patterns $\hat{\beta}_j$ and $\ddot{\beta}_j$, for $j \in [\mu + n] \setminus [\mu]$, that cover the cases that two different z variables are substituted equally in α_σ. In each of these newly defined pair of patterns, we would set γ_i and δ_i just to single, new, and independent variables to obtain any substitution $h(\alpha_1)$ and $h(\alpha_2)$. The number of those additional pairs of patterns rises significantly for alphabets of larger size, hence we only give an example here.

Consider the case that $h(z') = h(z_{\mathtt{a}_2})$. To handle this case, we would add a predicate $\pi_{\mu+k}$, for some $k \in [n]$, such that

$$\eta_{\mu+k} = z_{\mu+k}\ (z'_{\mu+k})^2\ (z_{\mu+k,\mathtt{a}_1})^2\ (z'_{\mu+k})^2\ (z_{\mu+k,\mathtt{a}_3})^2\ ...\ (z_{\mu+k,\mathtt{a}_\sigma})^2\ z_{\mu+k}$$

and $\gamma_{\mu+k} = y_{\mu+k}$ as well as $\delta_{\mu+k} = y'_{\mu+k}$, where each new variable is new and independent from other parts of the pattern. In particular, notice, that in this example we have no occurrence of the variable $z_{\mu+k,\mathtt{a}_2}$ and instead four occurrences of $z'_{\mu+k}$. The idea is similar to the previous construction of $\eta_{\mu+1}$ in the binary case to handle the case of $h(z) = h(z')$.

We observe, that if, for some substitution $h \in H_{\ell_{\alpha_\sigma}}$, any two different variables with the base name z are substituted equally, then there exists one pair of patterns $\hat{\beta}_i$ and $\ddot{\beta}_i$ that have the same variables equalled out, hence resulting in the existence of some $h' \in H_{\ell_{\beta_\sigma}}$ for which $h(\alpha_\sigma) = h'(\beta_\sigma)$.

Additionally, we add 2σ many pairs of patterns $\hat{\beta}_j$ and $\ddot{\beta}_j$, for $\mu + n < j \leq \mu + n + 2\sigma$ that handle the occurrence of any of the letters $h(z_{\mathtt{a}_1})$ to $h(z_{\mathtt{a}_\sigma})$ in $h(\alpha_1)$ or $h(\alpha_2)$. These predicates work exactly as in [11], just with the same addition as before that $h(z)$ and $h(z')$ determine the letters that need to be used for the encodings of valid computations and for the substitutions of x_v. All other arguments stay the same. Hence, we conclude the sketch of the extension to larger alphabets for Theorem 8.

D Extension to Larger Alphabets in Theorem 12

To extend the proof idea for Theorem 12 to arbitrary larger alphabets Σ with $|\Sigma| > 2$, we fix w.l.o.g. $\Sigma = \{0, \#, \mathtt{a}_1, \mathtt{a}_2, ..., \mathtt{a}_\sigma\}$ (hence, $\sigma \geq 1$). The basic idea stays the same, but the adaptations to the constructed patterns (α, ℓ_α) and (β, ℓ_β) are more substantial.

For each new letter $\mathtt{a}_i$, we introduce a new variable $x_{\mathtt{a}_i}$ that may be substituted by only a single letter. Similar to the extension to arbitrary alphabets in Theorem 12, the patterns (α, ℓ_α) and (β, ℓ_β) are adapted in such a way that two additional cases are considered for. First, if some substitution $h \in H_{\ell_\alpha}$ replaces two distinct variables in $\{x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$ to the same letter, i.e., $h(x_{\mathtt{a}}) = h(x_{\mathtt{b}})$ for $x_{\mathtt{a}}, x_{\mathtt{b}} \in \{x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$, then we should always find a substitution $h' \in H_{\ell_\beta}$ such that $h(\alpha) = h'(\beta)$. Next, we make sure that only the letters $h(x_\#)$ and $h(x_0)$ may appear in $h(\alpha_1)$ and $h(\alpha_2)$. Otherwise, we should also always find some substitution $h' \in H_{\ell_\beta}$ such that $h(\alpha) = h'(\beta)$.

We continue with the specific adapted constructions. First, each $\hat{\beta}_i$ is redefined to use the structure $\hat{\beta}_i = x_0 x_\#^4 x_0\ \mathbf{x_0 x_\# x_{a_1} x_{a_2} ... x_{a_\sigma}} \gamma_\mathbf{i}\ x_0 x_\#^4 x_0\ \mathbf{x_0 x_\# x_{a_1} x_{a_2} ... x_{a_\sigma}} \delta_i\ x_0 x_\#^4 x_0$. Notice that we just add each variable in $\{x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$ in front of each γ_i and δ_i, for $i \in [\mu + 1]$. Additionally, we redefine β by adding $x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}$ in front of it. The adapted length constraints of ℓ_β look almost identical, but with the addition of constraining the length of all variables in $\{x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$ to 1 and changing the sum of $x_a + x_b$ to correspond to the new number of $\hat{\beta}_i$'s that is explained in the following. In particular, we have $x_0 = 1$, $x_\# = 1$, $x_a + x_b = \mu + 1 + 2\sigma + n + 1$, $x_{\mathtt{a}_i} = 1$ for all $i \in [\sigma]$, and $x_i = 1$ for all $i \in [\mu + 1 + 2\sigma + n]$.

To define the adaptation of (α, ℓ_α), we need to extend the morphism ϕ to take into account the newly defined variables. Simply, for all variables $x \in X$ with $x \notin \{x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$, ϕ is defined just as in the binary case. For all other $x_{\mathtt{a}_i} \in \{x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$, we define $\phi(x) = x_{\mathtt{a}_i}$. Now, we add the prefix $x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}$ to both, α_1 and α_2. Finally, we also add the prefix $x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}$ to the whole of α. We add to the length constraints ℓ_α that we also have $x_{\mathtt{a}_i} = 1$, for all $i \in [\sigma]$. We continue with the construction of all $\hat{\beta}_i$ to cover the additional two emerging cases. Due to space constraints, all formal proofs regarding the correctness of this extension to the reduction can be found in [27].

First, for each $i \in [\sigma]$, we can add two $\hat{\beta}$, i.e., $\hat{\beta}_{\mu+1+i}$ and $\hat{\beta}_{\mu+1+\sigma+i}$, which cover the case that any letter obtained by $\{x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$ occurs somewhere later on in $h(\alpha_1)$ or $h(\alpha_2)$, for $h \in H_{\ell_\alpha}$. For $\hat{\beta}_{\mu+1+i}$, we define $\gamma_{\mu+1+i}$ and $\delta_{\mu+1+i}$ just by

- $\gamma_{\mu+1+i} = x_0\ x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}\ y_{\mu+1+i,1}\ x_{\mathtt{a}_i}\ y_{\mu+1+i,2}$
- $\delta_{\mu+1+i} = x_0\ x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}\ y'_{\mu+1+i}$

for new and independent variables $y_{\mu+1+i,1}, y_{\mu+1+i,2}, y'_{\mu+1+i}$. We can define $\hat{\beta}_{\mu+1+\sigma+i}$ analogously, just by swapping the role of γ and δ.

Finally, we need to cover the case that $h(x_\mathtt{a}) = h(x_\mathtt{b})$, for some $h \in H_{\ell_\alpha}, \mathtt{a}, \mathtt{b} \in \{x_0, x_\#, x_{\mathtt{a}_1}, ..., x_{\mathtt{a}_\sigma}\}$. Assume these variables to be $x_{\mathtt{a}_i}$ and $x_{\mathtt{a}_j}$, for some $i < j$. Then, we just define $\hat{\beta}_{\mu+1+2\sigma+k}$, for $k \in [n]$ (n being the total number of combinations), by setting γ und δ to

- $\gamma_{\mu+1+2\sigma+k} = x_0\ x_0 x_\# x_{\mathtt{a}_1} ... x_{\mathtt{a}_i} ... x_{\mathtt{a}_{j-1}} x_{\mathtt{a}_i} x_{\mathtt{a}_{j+1}} ... x_{\mathtt{a}_\sigma} y_{\mu+1+2\sigma+k}$, and
- $\delta_{\mu+1+2\sigma+k} = x_0\ x_0 x_\# x_{\mathtt{a}_1} ... x_{\mathtt{a}_i} ... x_{\mathtt{a}_{j-1}} x_{\mathtt{a}_i} x_{\mathtt{a}_{j+1}} ... x_{\mathtt{a}_\sigma} y'_{\mu+1+2\sigma+k}$.

So, if and only if the same symbol is used twice in the prefix $x_0 x_\# x_{\mathtt{a}_1} x_{\mathtt{a}_2} ... x_{\mathtt{a}_\sigma}$, then we can use these $\hat{\beta}'s$ to find some $h' \in H_{\ell_\beta}$ such that $h(\alpha) = h'(\beta)$.

By the previous two constructions, all additional cases that emerge due to the larger alphabet are captured by these additional $\hat{\beta}'s$, so they cannot result in substitutions $h \in H_{\ell_\alpha}$ with $h(\alpha) \notin L_{NE}(\beta, \ell_\beta)$. Hence, only analogous cases to the binary construction can result in words that are not in $L_{NE}(\beta, \ell_\beta)$, concluding the sketch of the extension to larger alphabets.

E NP-hardness of the Unary Case for the Equivalence Problem for Non-Erasing Pattern Languages with Length Constraints

This section is dedicated to show Proposition 21. W.l.o.g. assume $\Sigma = \{0\}$. We proceed with a reduction from 3SAT. Assume w.l.o.g. $\mathcal{X} = \{X_1, X_2, ...\}$ to be a set of boolean variables and let $\overline{\mathcal{X}} = \{\overline{X_1}, \overline{X_2}, ...\}$ be the set of their negations. Let $\varphi = (\varphi_1, \varphi_2, ..., \varphi_n)$ for $n \in \mathbb{N}$ with $n \geq 2$ be a 3SAT formula in conjunctive normal form over $\mathcal{X} \cup \overline{\mathcal{X}}$ such that for each clause φ_i with $i \in [n]$ we have w.l.o.g. $\varphi_i = (X_{i,1} \vee X_{i,2} \vee X_{i,3})$ with $X_{i,j} \in \mathcal{X} \cup \overline{\mathcal{X}}$ for $j \in \{1, 2, 3\}$. We define a function $f : \mathcal{X} \cup \overline{\mathcal{X}} \to X$ that maps each boolean variable to a

pattern variable by

$$f(X) = \begin{cases} u_i & \text{, if } x \in \mathcal{X} \text{ and } x = X_i \\ v_i & \text{, if } x \in \overline{\mathcal{X}} \text{ and } x = \overline{X_i} \end{cases}$$

for new and independent pattern variables u_i and v_i. Clearly, one of the two cases is always fulfilled for each boolean variable $X \in \mathcal{X} \cup \overline{\mathcal{X}}$. We proceed by defining the pattern (α, ℓ_α) by

$$\alpha = f(X_{1,1})f(X_{1,2})f(X_{1,3})y_1 \; f(X_{2,1})f(X_{2,2})f(X_{2,3})y_2 \; \ldots \; f(X_{n,1})f(X_{n,2})f(X_{n,3})y_n$$

for new and independent variables $y_i \in X$ with $i \in [n]$. Notice that this pattern is terminal-free. The length constraints ℓ_α are defined by the following system:

- $u_i + v_i = 3$ for all $i \in [n]$
- $y_i \leq 3$ for all $i \in [n]$
- $f(X_{i,1}) + f(X_{i,2}) + f(X_{i,3}) + y_i = 7$ for all $i \in [n]$
- $\sum_{i=1}^{n}(f(X_{i,1}) + f(X_{i,2}) + f(X_{i,3}) + y_i) = 7n$

Notice, that the final constraint results in that $L_{NE}(\alpha, \ell_\alpha)$ may only have the word 0^{7n} in it, as it restricts the length of all symbols in α together. Now, we set the second pattern with length constraints (β, ℓ_β) by

$$\beta = z$$

for a new and independent variable $z \in X$ and define the length constraint ℓ_β by $z = 0^{7n}$. Hence, $L_{NE}(\beta, \ell_\beta) = \{0^{7n}\}$.

To proof the reduction, first, assume there exists a satisfying assignment of variables ϕ for φ. Let h be a substitution such that $h(u_i) = 00$ and $h(v_i) = 0$ if and only if $\phi(X_i) = \mathit{true}$ and $\phi(\overline{X_i}) = \mathit{false}$. Otherwise, set $h(u_i) = 0$ and $h(v_i) = 00$. As ϕ is satisfying φ, we know that $4 \leq |h(f(X_{i,1})f(X_{i,2})f(X_{i,3}))| \leq 6$ for all $i \in [n]$. Hence, we can always set $h(y_i) = 0^{7-|h(f(X_{i,1})f(X_{i,2})f(X_{i,3}))|}$ and obtain $|h(f(X_{i,1})f(X_{i,2})f(X_{i,3})y_i)| = 7$ for all $i \in [n]$. Hence, we h must be ℓ_α-valid and we have $h(\alpha) = 0^7 n$. As this is the only word we may obtain for $L_{NE}(\alpha, \ell_\alpha)$, we have $L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\beta, \ell_\beta)$.

For the other direction, assume $L_{NE}(\alpha, \ell_\alpha) = L_{NE}(\beta, \ell_\beta) = \{0^{7n}\}$. So, there exists some $h \in H_{\ell_\alpha}$ such that $h(\alpha) = 0^{7n}$. By ℓ_α, we know $h(f(X_{i,1})f(X_{i,2})f(X_{i,3})y_i) = 7$ for all $i \in [n]$. As $|h(y_i)| \leq 3$ must be the case, for each $i \in [n]$, there exists some $j \in \{1, 2, 3\}$ such that $|h(f(X_{i,j}))| > 1$, hence by ℓ_α, we must have $|h(f(X_{i,j}))| = 2$, resulting in $h(f(X_{i,j})) = 00$. We set an assignment of variables ϕ by $X_i = \mathit{true}$ and $\overline{X_i} = \mathit{false}$ if and only if $h(u_i) = 00$ and $h(x_i) = 0$, otherwise set $X_i = \mathit{false}$ and $\overline{X_i} = \mathit{true}$. As $|h(u_i)| + |h(v_i)| = 3$, we know such an assignment is a valid assignment for φ. As for each clause ϕ_i for $i \in [n]$ we find a variable that is set to true, we know that φ is satisfied by ϕ, concluding the reduction.