Bad News on Decision Problems for Patterns[☆]

Dominik D. Freydenberger[a,*], Daniel Reidenbach[b]

[a] *Institut für Informatik, Goethe-Universität, Postfach 111932, D-60054 Frankfurt am Main, Germany*
[b] *Department of Computer Science, Loughborough University, Loughborough, Leicestershire, LE11 3TU, United Kingdom*

Abstract

We study the inclusion problem for pattern languages, which – due to Jiang et al. (*Journal of Computer and System Sciences* 50, 1995) – is known to be undecidable. More precisely, Jiang et al. demonstrate that there is no effective procedure deciding the inclusion for the class of *all* pattern languages over *all* alphabets. Most applications of pattern languages, however, consider classes over *fixed* alphabets, and therefore it is practically more relevant to ask for the existence of *alphabet-specific* decision procedures. Our first main result states that, for all but very particular cases, this version of the inclusion problem is also undecidable. The second main part of our paper disproves the prevalent conjecture on the inclusion of so-called similar E-pattern languages, and it explains the devastating consequences of this result for the intensive previous research on the most prominent open decision problem for pattern languages, namely the equivalence problem for general E-pattern languages.

1. Introduction

A *pattern* – a finite string that consists of *variables* and of *terminal symbols* (or: *letters*) – is a compact and natural device to define a formal language. It generates a word by a *substitution* of all variables with arbitrary words of terminal symbols (taken from a fixed alphabet Σ) and, hence, its language is the set of all words under such substitutions. More formally, a pattern language thus is the (typically infinite) set of all images of the pattern under *terminal-preserving* morphisms, i. e. morphisms which map each terminal symbol onto itself. For example, if we consider the pattern

$$\alpha := x_1 \, \mathtt{a} \, x_2 \, \mathtt{b} \, x_1$$

[☆] A preliminary version of this work was presented at DLT 2008.

[*] Corresponding author.
Email addresses: freydenberger@em.uni-frankfurt.de (Dominik D. Freydenberger), D.Reidenbach@lboro.ac.uk (Daniel Reidenbach)

(where the symbols x_1 and x_2 are variables and $\mathtt{a}$ and $\mathtt{b}$ are terminal symbols) then the language generated by α exactly contains those words that consist of an arbitrary prefix u, followed by the letter $\mathtt{a}$, an arbitrary word v, the letter $\mathtt{b}$ and a suffix which equals u again. Consequently, the *pattern language* of α includes, amongst others, the words $w_1 := \mathtt{aabbba}$, $w_2 := \mathtt{abababab}$ and $w_3 := \mathtt{aaabaa}$, and it does not cover the words $w_4 := \mathtt{ba}$, $w_5 := \mathtt{babbba}$ and $w_6 := \mathtt{abba}$. It is a well-known fact that pattern languages in general are not context-free.

Basically, two types of pattern languages are considered in literature: *NE*-pattern languages and *E*-pattern languages. The definition of the former was introduced by Angluin [1], and it disallows the variables to be substituted with the empty word (hence, "NE" is short for "nonerasing"). The latter kind of pattern languages additionally consider those substitutions which map one or more variables onto the empty word (so "E" stands for "erasing" or "extended"); this definition goes back to Shinohara [26]. Thus, in our above example, the word w_3 is contained in the E-pattern language of α, but not in its NE-pattern language. Surprisingly, this small difference in the definitions leads to significant differences in the characteristics of the resulting (classes of) languages.

As a consequence of their simple definition, which comprises nothing but finite strings and (a particular type of) morphisms, pattern languages show numerous connections to other fundamental topics in computer science and discrete mathematics, including classical ones such as (un-)avoidable patterns (cf. Jiang et al. [9]), word equations (cf. Mateescu and Salomaa [13], Karhumäki et al. [11]) and equality sets (and, thus, the Post Correspondence Problem, cf. Reidenbach [19]) as well as emerging ones such as extended regular expressions (cf. Câmpeanu et al. [3]) and the ambiguity of morphisms (cf. Freydenberger et al. [7], Reidenbach [19]). In terms of the basic *decision problems*, pattern languages show a wide range of behaviors: trivial (linear time) decidability (e. g., the equivalence of NE-pattern languages), NP-completeness (e. g., the membership in NE- and E-pattern languages) and undecidability (e. g., the inclusion of NE- and E-pattern languages); furthermore, the decidability of quite a number of these problems is still open (e. g., the equivalence problem for E-pattern languages). Surveys on these topics are provided by, e. g., Mateescu and Salomaa [14] and Salomaa [24].

Among the established properties (and even among all results on pattern languages), the proof for the undecidability of the inclusion problem by Jiang, Salomaa, Salomaa and Yu [10] is considered to be one of the most notable achievements, and this is mainly due to the very hard proof, which answered a longstanding open question, and the fact that the result remarkably contrasts with the trivial decidability of the equivalence problem for NE-pattern languages. Furthermore, the inclusion problem is of vital importance for the main field of application of pattern languages, namely *inductive inference*. Inductive inference of pattern languages – which deals with an approach to the important problem of computing a pattern that is common to a given set of strings – is a both classical and active area of research in learning theory; a survey is provided by Ng and Shinohara [17]. It is closely connected to the inclusion problem for

pattern languages since, according to the celebrated characterization by Angluin [2], the inferrability of any indexable class of languages largely depends on the inclusion relation between the languages in the class. Consequently, many (both early and recent) papers on inductive inference of classes of pattern languages nearly exclusively deal with questions related to the inclusion problem for these classes (see, e. g., Mukouchi [16], Reidenbach [20, 22], Luo [12]).

Unfortunately, from this rather practical point of view, the inclusion problem for E- and for NE-pattern languages as understood and successfully tackled by Jiang et al. [10] is not very significant, since they prove that there is no *single* procedure which, *for every terminal alphabet* Σ and for every pair of patterns, decides on the inclusion between the languages over Σ generated by these patterns. Hence, and slightly more formally, Jiang et al. [10] demonstrate that the inclusion problem is undecidable for (a technical subclass of) the class of all pattern languages over all alphabets, and the requirement for any decision procedure to handle pattern languages over various alphabets is extensively utilized in the proof. Contrary to this, in inductive inference of pattern languages – and virtually every other field of application of pattern languages known to the authors – one always considers a class of pattern languages over a *fixed* alphabet. Consequently, it seems practically more relevant to investigate the problem of whether, for any alphabet Σ, there exists a procedure deciding the inclusion problem for the class of (E/NE-)pattern languages *over this alphabet* Σ.

In the present paper we study and answer this question (or rather: these infinitely many questions). Our considerations reveal that, for every finite alphabet Σ with at least two letters, the inclusion problem is undecidable for the full classes of E-pattern languages over Σ. Furthermore, with regard to the class of NE-pattern languages over any Σ, we prove the equivalent result, but our reasoning does not cover binary and ternary alphabets. Although we thus have the same outcome as Jiang et al. [10] for their variant of the inclusion problem, the proof for our much stronger statement considerably differs from their argumentation; consequently, it suggests that there is no straightforward way from the well-established result to ours. Moreover, we feel that our insights (and our uniform reasoning for all alphabet sizes) are a little surprising, since the inferrability of classes of pattern languages is known to be discontinuous depending on the alphabet size and the question of whether NE- or E-pattern languages are considered (cf. Reidenbach [22]).

The second main part of our paper addresses the other major topic in [10]: we discuss the extensibility of a positive decidability result given in [10] on the inclusion problem for the class of *terminal-free* E-pattern languages (generated by those patterns that consist of variables only) to classes of so-called *similar* E-pattern languages. This question is intensively discussed in literature (e. g. by Ohlebusch and Ukkonen [18]) as it is linked to the still unresolved equivalence problem for the full class of E-pattern languages. We demonstrate that, in contrast to the prevalent conjecture, the inclusion of similar E-pattern languages does *not* show an analogous behavior to that of terminal-free E-pattern languages, and we explain the fatal impact of this insight on the previous research

dealing with the equivalence problem.

2. Preliminaries

This paper is largely self-contained. For notations not explicitly defined, Rozenberg and Salomaa [23] can be consulted.

Let $\mathbb{N} := \{1, 2, 3, \ldots\}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. The symbol ∞ stands for infinity. For an arbitrary alphabet A, a *string* (over A) is a finite sequence of symbols from A, and λ stands for the *empty string*. The symbol A^+ denotes the set of all nonempty strings over A, and $A^* := A^+ \cup \{\lambda\}$. For the *concatenation* of two strings w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. We say that a string $v \in A^*$ is a *factor* of a string $w \in A^*$ if there are $u_1, u_2 \in A^*$ such that $w = u_1 v u_2$. If $u_1 = \lambda$ (or $u_2 = \lambda$), then v is a *prefix* of w (or a *suffix*, respectively). The notation $|x|$ stands for the size of a set x or the length of a string x. For any $w \in \Sigma^*$ and any $n \in \mathbb{N}_0$, w^n denotes the *n-fold concatenation of* w, with $w^0 := \lambda$. Furthermore, we use $\cdot$ and the regular operations $*$ and $+$ on sets and strings in the usual way.

For any alphabets A, B, a *morphism* is a function $h : A^* \to B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$. Given morphisms $f : A^* \to B^*$ and $g : B^* \to C^*$ (for alphabets A, B, C), their *composition* $g \circ f$ is defined as $g \circ f(w) := g(f(w))$ for all $w \in A^*$. A morphism $h : A^* \to B^*$ is *nonerasing* if $h(a) \neq \lambda$ for all $a \in A$.

Let Σ be a (finite or infinite) alphabet of so-called *terminal symbols* (or: *letters*) and X an infinite set of *variables* with $\Sigma \cap X = \emptyset$. Unless specified differently, we assume $X = \{x_i \mid i \in \mathbb{N}\}$, with $x_i \neq x_j$ for all $i \neq j$. A *pattern* is a string over $\Sigma \cup X$, a *terminal-free pattern* is a string over X and a *word* is a string over Σ. The set of all patterns over $\Sigma \cup X$ is denoted by Pat_Σ, the set of terminal-free patterns by $\mathrm{Pat}_{\mathrm{tf}}$. For any pattern α, we refer to the set of variables in α as $\mathrm{var}(\alpha)$ and to the set of terminal symbols as $\mathrm{term}(\alpha)$. In accordance with Ohlebusch and Ukkonen [18], we say that two patterns $\alpha, \beta \in \mathrm{Pat}_\Sigma$ are *similar* if their factors over Σ are identical and occur in the same order in the patterns. More formally, α, β are similar if $\alpha = \alpha_0 u_1 \alpha_1 u_2 \ldots \alpha_{n-1} u_n \alpha_n$ and $\beta = \beta_0 u_1 \beta_1 u_2 \ldots \beta_{n-1} u_n \beta_n$ for some $n \in \mathbb{N}_0$, $\alpha_i, \beta_i \in X^+$ for each $i \in \{1, \ldots, n-1\}$, $\alpha_0, \beta_0, \alpha_n, \beta_n \in X^*$ and $u_j \in \Sigma^+$ for each $j \in \{1, \ldots, n\}$.

A morphism $\sigma : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ is called *terminal-preserving* if $\sigma(a) = a$ for every $a \in \Sigma$. A terminal-preserving morphism $\sigma : (\Sigma \cup X)^* \to \Sigma^*$ is called a *substitution*. The *E-pattern language* $L_{\mathrm{E},\Sigma}(\alpha)$ of a pattern $\alpha \in \mathrm{Pat}_\Sigma$ is given by

$$L_{\mathrm{E},\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \to \Sigma^* \text{ is a substitution}\};$$

accordingly, the *NE-pattern language* $L_{\mathrm{NE},\Sigma}(\alpha)$ of α is given by

$$L_{\mathrm{NE},\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \to \Sigma^* \text{ is a nonerasing substitution}\}.$$

The term *pattern language* refers to any of the definitions introduced above. Two pattern languages are called *similar* if they have generating patterns that are similar. Accordingly, we call a pattern language *terminal-free* if it is generated

by a terminal-free pattern. We denote the class of all E-pattern languages over Σ with ePAT_Σ and the class of all NE-pattern languages over Σ with nePAT_Σ.

For any alphabet Σ and any class $\mathrm{PAT}_{\star,\Sigma} \subseteq \mathrm{ePAT}_\Sigma$ (or $\mathrm{PAT}_{\star,\Sigma} \subseteq \mathrm{nePAT}_\Sigma$) of pattern languages over Σ, the *inclusion* problem is said to be *decidable* if and only if there exists a total computable function χ such that, for every pair of a patterns $\alpha, \beta \in \mathrm{Pat}_\Sigma$ with $L_{\mathrm{E},\Sigma}(\alpha), L_{\mathrm{E},\Sigma}(\beta) \in \mathrm{PAT}_{\star,\Sigma}$ (or, alternatively, $L_{\mathrm{NE},\Sigma}(\alpha), L_{\mathrm{NE},\Sigma}(\beta) \in \mathrm{PAT}_{\star,\Sigma}$), χ decides on whether or not $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ (or, alternatively, $L_{\mathrm{NE},\Sigma}(\alpha) \subseteq L_{\mathrm{NE},\Sigma}(\beta)$). The inclusion problem for $\mathrm{PAT}_{\star,\Sigma}$ is *undecidable* if it is not decidable, i. e. the said function χ does not exist. Decidability of the *equivalence problem* is defined analogously.

A vital part of our considerations relies on the following concepts: A *nondeterministic 2-counter automaton without input* (cf. Ibarra [8]) is a 4-tuple $\mathcal{A} = (Q, \delta, q_0, F)$, consisting of a state set Q, a transition relation $\delta : Q \times \{0,1\}^2 \to Q \times \{-1,0,+1\}^2$, the initial state $q_0 \in Q$ and a set of accepting states $F \subseteq Q$. A *configuration* of $\mathcal{A}$ is a triple $(q, m_1, m_2) \in Q \times \mathbb{N}_0 \times \mathbb{N}_0$, where q indicates the state of $\mathcal{A}$ and m_1 (or m_2) denotes the content of the first (or second, respectively) counter. The relation $\vdash_{\mathcal{A}}$ on $Q \times \mathbb{N}_0 \times \mathbb{N}_0$ is defined by δ as follows: Let $p, q \in Q$, $m_1, m_2, n_1, n_2 \in \mathbb{N}_0$. Then $(p, m_1, m_2) \vdash_{\mathcal{A}} (q, n_1, n_2)$ if and only if there exist $c_1, c_2 \in \{0,1\}$ and $r_1, r_2 \in \{-1, 0, +1\}$ such that

(i) $c_i = 0$ if $m_i = 0$ and $c_i = 1$ if $m_i \geq 1$ for $i \in \{1,2\}$,
(ii) $n_i = m_i + r_i$ for $i \in \{1,2\}$, and
(iii) $(q, r_1, r_2) \in \delta(p, c_1, c_2)$.

Furthermore, for $i \in \{1,2\}$, we assume that $r_i \neq -1$ if $c_i = 0$. Intuitively, in every state $\mathcal{A}$ is only able to check whether the counters equal zero, change each counter by at most one and switch into a new state.

A *computation* is a sequence of configurations, and an *accepting computation* of $\mathcal{A}$ is a sequence $C_1, \ldots, C_n \in Q \times \mathbb{N}_0 \times \mathbb{N}_0$ (for some $n \in \mathbb{N}_0$) with $C_1 = (q_0, 0, 0)$, $C_n \in F \times \mathbb{N}_0 \times \mathbb{N}_0$ and $C_i \vdash_{\mathcal{A}} C_{i+1}$ for all $i \in \{1, \ldots, n-1\}$. In order to encode configurations of $\mathcal{A}$, we assume that $Q = \{q_0, \ldots, q_s\}$ for some $s \in \mathbb{N}_0$ and define a function $\mathrm{enc} : Q \times \mathbb{N}_0 \times \mathbb{N}_0 \to \{0, \#\}^*$ by $\mathrm{enc}(q_i, m_1, m_2) := 0^{i+1}\#0^{m_1+1}\#0^{m_2+1}$ and extend this to an encoding of computations by defining

$$\mathrm{enc}(C_1, C_2, \ldots, C_n) := \#\# \, \mathrm{enc}(C_1) \, \#\# \, \mathrm{enc}(C_2) \, \#\# \ldots \#\# \, \mathrm{enc}(C_n) \#\#$$

for every $n \geq 1$ and every sequence $C_1, \ldots, C_n \in Q \times \mathbb{N}_0 \times \mathbb{N}_0$. Furthermore, let $\mathrm{VALC}(\mathcal{A}) := \{\mathrm{enc}(C_1, \ldots, C_n) \mid C_1, \ldots, C_n \text{ is an accepting computation of } \mathcal{A}\}$, and $\mathrm{INVALC}(\mathcal{A}) := \{0, \#\}^* \setminus \mathrm{VALC}(\mathcal{A})$. As the emptiness problem for 2-counter automata with input is undecidable (cf. Minsky [15], Ibarra [8]), it is also undecidable whether a nondeterministic 2-counter automaton without input has an accepting computation.

3. The inclusion of pattern languages over fixed alphabets

In this section, we discuss the decidability of the inclusion problem for ePAT_Σ and nePAT_Σ. We begin with all non-unary finite alphabets Σ; the special case $|\Sigma| \in \{1, \infty\}$ is studied separately.

As demonstrated by Jiang, Salomaa, Salomaa and Yu [10], the *general inclusion problem for E-pattern languages* is undecidable:

Theorem 1 (Jiang et al. [10])**.** *There is no total computable function χ_{E} that, for every alphabet Σ and for every pair of patterns $\alpha, \beta \in \mathrm{Pat}_\Sigma$, decides on whether or not $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$.*

Technically, Jiang et al. show that, given a nondeterministic 2-counter automaton without input $\mathcal{A}$, one can effectively construct an alphabet Σ and patterns $\alpha_{\mathcal{A}}, \beta_{\mathcal{A}} \in \mathrm{Pat}_\Sigma$ such that $L_{\mathrm{E},\Sigma}(\alpha_{\mathcal{A}}) \subseteq L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$ if and only if $\mathcal{A}$ has an accepting computation. As this problem is known to be undecidable, the general inclusion problem for E-pattern languages must also be undecidable.

In their construction, Σ contains one letter for every state of $\mathcal{A}$, and six further symbols that are used for technical reasons. Quite obviously, we cannot use such an approach to prove undecidability of the inclusion problem for ePAT_Σ with some fixed alphabet Σ, since we had to limit the number of states of the automata under consideration. This step, in turn, would lead to a finite class of possible automata, and, hence, we had a trivially decidable emptiness problem for that class. Consequently, as mentioned by Reidenbach [19] and Salomaa [25], there seems to be no straightforward way from the undecidability result by Jiang et al. [10] to the undecidability of the inclusion problem for ePAT_Σ, especially when Σ is comparatively small. Nevertheless, our first main theorem states:

Theorem 2. *Let Σ be a finite alphabet with $|\Sigma| \geq 2$. Then the inclusion problem for ePAT_Σ is undecidable.*

The proof of this theorem is rather complex and can be found in Section 3.1. It is in principle based on the construction by Jiang et al. [10], with two key differences. First, the problem of an unbounded number of states (and therefore the number of letters necessary to encode these states) is handled by using a unary encoding instead of special letters to designate the states in configurations; second, the special control symbols are encoded over a binary alphabet or removed. These modifications enforce considerable changes to the patterns and the underlying reasoning. But before we go into these details, we first discuss the immediate consequences of Theorem 2. In fact, the proof demonstrates a stronger result:

Corollary 1. *Let Σ be a finite alphabet with $|\Sigma| \geq 2$, and let $a \in \Sigma$. There is no total computable function χ that, for every pair of patterns $\alpha \in \mathrm{Pat}_\Sigma$ and $\beta \in (\{a\} \cup X)^*$, decides on whether or not $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$.*

This corollary is the alphabet specific version of Jiang et al.'s Corollary 5.1 in [10] that is used to obtain the following result on the *general inclusion problem for NE-pattern languages*:

Theorem 3 (Jiang et al. [10])**.** *There is no total computable function χ_{NE} that, for every alphabet Σ and for every pair of patterns $\alpha, \beta \in \mathrm{Pat}_\Sigma$, decides on whether or not $L_{\mathrm{NE},\Sigma}(\alpha) \subseteq L_{\mathrm{NE},\Sigma}(\beta)$.*

In our terminology, the proof of Theorem 3 in [10] reduces the inclusion problem for ePAT_Σ (for patterns of a restricted form as in Corollary 1) to the inclusion problem for $\mathrm{nePAT}_{\Sigma\cup\{\star,\$\}}$, where $\star$ and \$ are two extra letters that are not contained in Σ. If we consider the reasoning on Theorem 3 as given by Jiang et al. and substitute our Corollary 1 for their Corollary 5.1, we immediately obtain the following result:

Theorem 4. *Let Σ be a finite alphabet with $|\Sigma| \geq 4$. Then the inclusion problem for nePAT_Σ is undecidable.*

As the construction used in the reduction heavily depends on the two extra letters, we do not see a straightforward way to adapt it to binary or ternary alphabets. Therefore, the decidability of the inclusion problem for NE-pattern languages over these alphabets remains open:

Open Problem 1. *Let Σ be an alphabet with $|\Sigma| = 2$ or $|\Sigma| = 3$. Is the inclusion problem for nePAT_Σ decidable?*

We conclude this section with a brief look at the special cases of unary and infinite alphabets. Here we can state that the inclusion of pattern languages is less complex than in the standard case:

Proposition 1. *Let Σ be an alphabet, $|\Sigma| \in \{1, \infty\}$. Then the inclusion problem is decidable for ePAT_Σ and for nePAT_Σ.*

Proof. It is a well-known fact that every pattern language over a *unary* alphabet is regular (cf. Salomaa [24] or Reidenbach [19]) and that, e. g., an NFA accepting such a pattern language can be effectively constructed. Therefore the decidability of the inclusion problem for the full classes of these languages follows from the decidability of the inclusion problem for NFA (see, e. g., Rozenberg and Salomaa [23]).

With regard to an *infinite* alphabet, we initially discuss the problem for NE-pattern languages. In this regard, it is sufficient to prove the following statement: For every $\alpha, \beta \in \mathrm{Pat}_\Sigma$, $L_{\mathrm{NE},\Sigma}(\alpha) \subseteq L_{\mathrm{NE},\Sigma}(\beta)$ if and only if there exists a terminal-preserving morphism $\phi : (\Sigma \cup X)^+ \to (\Sigma \cup X)^+$ such that $\phi(\beta) = \alpha$. From this insight we then can directly conclude that our claim on nePAT_Σ is correct, since the existence of ϕ is decidable (by just checking the finitely many candidates for such a morphism). Moreover, it can be easily verified that the existence of a morphism $\phi : (\Sigma \cup X)^+ \to (\Sigma \cup X)^+$ with $\phi(\beta) = \alpha$ is a *sufficient* condition for $L_{\mathrm{NE},\Sigma}(\alpha) \subseteq L_{\mathrm{NE},\Sigma}(\beta)$ (cf. Lemma 3.1 by Angluin [1]). Consequently, the verification of the *if* direction of the above statement is straightforward, and therefore we only have to prove the *only if* direction.

Hence, let $\alpha, \beta \in \mathrm{Pat}_\Sigma$ with $L_{\mathrm{NE},\Sigma}(\alpha) \subseteq L_{\mathrm{NE},\Sigma}(\beta)$. Let $\tau_{\mathrm{iso}} : (\Sigma\cup X)^+ \to \Sigma^+$ be any substitution such that, for every $x \in \mathrm{var}(\alpha)$, $|\,\tau_{\mathrm{iso}}(x)| = 1$ and $\tau_{\mathrm{iso}}(x) \notin \mathrm{term}(\alpha) \cup \mathrm{term}(\beta)$ and, for every $y \in \mathrm{var}(\alpha)$ with $x \neq y$, $\tau_{\mathrm{iso}}(x) \neq \tau_{\mathrm{iso}}(y)$. The existence of τ_{iso} is granted by the condition $|\Sigma| = \infty$. Since $w := \tau_{\mathrm{iso}}(\alpha)$

is actually nothing but a "renaming" of α, it is obvious that there exists an inverse morphism $\tau_{\text{iso}}^{-1} : \Sigma^+ \to (\Sigma \cup X)^+$ such that $\tau_{\text{iso}}^{-1}(w) = \alpha$. Furthermore, the definition of τ_{iso} implies that, for every $\mathtt{A} \in \text{term}(w)$, we have $\tau_{\text{iso}}^{-1}(\mathtt{A}) \in X$ if and only if $\mathtt{A} \notin \text{term}(\alpha) \cup \text{term}(\beta)$.

As $L_{\text{NE},\Sigma}(\alpha) \subseteq L_{\text{NE},\Sigma}(\beta)$, there is a substitution $\tau' : (\Sigma \cup X)^+ \to \Sigma^+$ such that $w = \tau'(\beta)$. We now define $\phi := \tau_{\text{iso}}^{-1} \circ \tau'$. Then ϕ is terminal-preserving because τ' is a substitution (which implies that it is terminal-preserving) and, for every letter $\mathtt{A}$ in w that is mapped by τ_{iso}^{-1} onto a variable, $\mathtt{A} \notin \text{term}(\beta)$. Furthermore, ϕ is a morphism mapping a string in $(\Sigma \cup X)^+$ onto a string in $(\Sigma \cup X)^+$ and, evidently, $\phi(\beta) = \alpha$. This proves the claim on nePAT_Σ.

Using a morphism $\phi : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$, an analogous reasoning proves the statement on E-pattern languages. □

Obviously, Proposition 1 implies that the *equivalence* problem is decidable, too, for ePAT_Σ and nePAT_Σ over unary or infinite alphabets Σ. Furthermore, with regard to $2 \leq |\Sigma| < \infty$, it is shown by Angluin [1] that two patterns generate the same *NE*-pattern language if and only if they are the same (apart from a renaming of variables). Thus, the equivalence problem for nePAT_Σ is trivially decidable for every Σ, a result which nicely contrasts with the undecidability of the inclusion problem established above. The equivalence problem for ePAT_Σ, however, is still an open problem in case of $2 \leq |\Sigma| < \infty$. In Section 4 we present and discuss a result that has a significant impact on this widely-discussed topic.

3.1. Proof of Theorem 2

We begin with the case $|\Sigma| = 2$, so let $\Sigma := \{0, \#\}$. Let $\mathcal{A} := (Q, \delta, q_0, F)$ be a nondeterministic 2-counter automaton; w. l. o. g. let $Q := \{q_0, \ldots, q_s\}$ for some $s \in \mathbb{N}_0$. Our goal is to construct patterns $\alpha_\mathcal{A}, \beta_\mathcal{A} \in \text{Pat}_\Sigma$ such that $L_{\text{E},\Sigma}(\alpha_\mathcal{A}) \subseteq L_{\text{E},\Sigma}(\beta_\mathcal{A})$ if and only if $\text{VALC}(\mathcal{A}) = \emptyset$. We define

$$\alpha_\mathcal{A} := v\, v\, \#^4\, v\, x\, v\, y\, v\, \#^4\, v\, u\, v,$$

where x, y are distinct variables, $v = 0\#^3 0$ and $u = 0\#\#0$. Furthermore, for a yet unspecified $\mu \in \mathbb{N}$ that shall be defined later, let

$$\beta_\mathcal{A} := (x_1)^2 \ldots (x_\mu)^2 \#^4 \hat{\beta}_1 \ldots \hat{\beta}_\mu \#^4 \ddot{\beta}_1 \ldots \ddot{\beta}_\mu,$$

with, for all $i \in \{1, \ldots, \mu\}$, $\hat{\beta}_i := x_i\, \gamma_i\, x_i\, \delta_i\, x_i$ and $\ddot{\beta}_i := x_i\, \eta_i\, x_i$, where $x_1, \ldots, x_\mu$ are distinct variables and all $\gamma_i, \delta_i, \eta_i \in X^*$ are terminal-free patterns. The patterns γ_i and δ_i shall be defined later; for now, we only mention:

1. $\eta_i := z_i(\hat{z}_i)^2 z_i$ and $z_i \neq \hat{z}_i$ for all $i \in \{1, \ldots, \mu\}$,
2. $\text{var}(\gamma_i \delta_i \eta_i) \cap \text{var}(\gamma_j \delta_j \eta_j) = \emptyset$ for all $i, j \in \{1, \ldots, \mu\}$ with $i \neq j$,
3. $x_k \notin \text{var}(\gamma_i \delta_i \eta_i)$ for all $i, k \in \{1, \ldots, \mu\}$.

Thus, for every i, the elements of $\text{var}(\gamma_i \delta_i \eta_i)$ appear nowhere but in these three factors. Let H be the set of all substitutions $\sigma : (\Sigma \cup \{x, y\})^* \to \Sigma^*$. We interpret each triple $(\gamma_i, \delta_i, \eta_i)$ as a predicate $\pi_i : H \to \{0, 1\}$ in such a way

that $\sigma \in H$ *satisfies* π_i if there exists a morphism $\tau : \mathrm{var}(\gamma_i\delta_i\eta_i)^* \to \Sigma^*$ with $\tau(\gamma_i) = \sigma(x)$, $\tau(\delta_i) = \sigma(y)$ and $\tau(\eta_i) = u$ – in the terminology of word equations (cf. Karhumäki et al. [11]), this means that σ satisfies π_i if and only if the system consisting of the three equations $\gamma_i = \sigma(x)$, $\delta_i = \sigma(y)$ and $\eta_i = u$ has a solution τ. Later, we shall see that $L_{\mathrm{E},\Sigma}(\alpha_\mathcal{A}) \setminus L_{\mathrm{E},\Sigma}(\beta_\mathcal{A})$ exactly contains those $\sigma(\alpha_\mathcal{A})$ for which σ does not satisfy any of π_1 to π_μ, and choose these predicates to describe INVALC($\mathcal{A}$). The encoding of INVALC($\mathcal{A}$) shall be handled by π_4 to π_μ, as each of these predicates describes a sufficient criterium for membership in INVALC($\mathcal{A}$). But at first we need a considerable amount of technical preparations. A substitution σ is of *good form* if $\sigma(x) \in \{0,\#\}^*$, $\sigma(x)$ does not contain $\#^3$ as a factor, and $\sigma(y) \in 0^*$. Otherwise, σ is of *bad form*. The predicates π_1 and π_2 handle all cases where σ is of bad form and are defined through

$$\gamma_1 := y_{1,1}(\hat{z}_1)^3 y_{1,2}, \qquad \gamma_2 := y_2,$$
$$\delta_1 := \hat{y}_1, \qquad \delta_2 := \hat{y}_{2,1}\,\hat{z}_2\,\hat{y}_{2,2},$$

where $y_{1,1}$, $y_{1,2}$, y_2, $\hat{y}_1$, $\hat{y}_{2,1}$, $\hat{y}_{2,2}$, $\hat{z}_1$ and $\hat{z}_2$ are pairwise distinct variables. Recall that $\eta_i = z_i(\hat{z}_i)^2 z_i$ for all i. It is not very difficult to see that π_1 and π_2 characterize the morphisms that are of bad form:

Lemma 1. *A substitution $\sigma \in H$ is of bad form if and only if σ satisfies π_1 or π_2.*

Proof. We begin with the *only if* direction. If $\sigma(x) = w_1\#^3 w_2$ for some $w_1, w_2 \in \Sigma^*$, choose $\tau(y_{1,1}) := w_1$, $\tau(y_{1,2}) := w_2$, $\tau(\hat{z}_1) := \#$, $\tau(\hat{y}_1) := \sigma(y)$ and $\tau(z_1) := 0$. Then $\tau(\gamma_1) = \sigma(x)$, $\tau(\delta_1) = \sigma(y)$ and $\tau(\eta_1) = u$; thus, σ satisfies π_1.

If $\sigma(y) = w_1\#w_2$ for some $w_1, w_2 \in \Sigma^*$, let $\tau(y_2) := \sigma(x)$, $\tau(\hat{y}_{2,1}) := w_1$, $\tau(\hat{y}_{2,2}) := w_2$ and $\tau(\hat{z}_2) := \#$, and $\tau(z_2) := 0$. It is easy to see that σ satisfies π_2.

For the *if* direction, if σ satisfies π_1, then there exists a morphism τ with $\tau(\gamma_1) = \sigma(x)$ and $\tau(\eta_1) = 0\#^2 0$. Thus, $\tau(\hat{z}_1) = \#$ and $\tau(z_1) = 0$ must hold. Consequently, $\sigma(x)$ contains $\#^3$, and σ is of bad form.

Analogously, if σ satisfies π_2, then $\sigma(y)$ contains the letter $\#$, and σ is of bad form. □

This allows us to make the following observation, which serves as the central part of the construction and is independent from the exact shape of π_3 to π_μ:

Lemma 2. *For every substitution $\sigma \in H$, $\sigma(\alpha_\mathcal{A}) \in L_{\mathrm{E},\Sigma}(\beta_\mathcal{A})$ if and only if σ satisfies one of the predicates π_1 to π_μ.*

Proof. We begin with the *if* direction. Assume $\sigma \in H$ satisfies some predicate π_i. Then there exists a morphism $\tau : \mathrm{var}(\gamma_i\delta_i\eta_i) \to \Sigma^*$ such that $\tau(\gamma_i) = \sigma(x)$, $\tau(\delta_i) = \sigma(y)$ and $\tau(\eta_i) = u$. We extend τ to a substitution τ' defined by

1. $\tau'(x) := \tau(x)$ for all $x \in \mathrm{var}(\gamma_i\delta_i\eta_i)$
2. $\tau'(x_i) := 0\#^3 0 = v$,

3. $\tau'(0) := 0$ and $\tau'(\#) := \#$,
4. $\tau'(x) := \lambda$ in all other cases.

By definition, none of the variables in $\mathrm{var}(\gamma_i\delta_i\eta_i)$ appears outside of these factors. Thus, τ' can always be defined in this way. We obtain

$$\begin{aligned}\tau'(\hat{\beta}_i) &= \tau'(x_i\,\gamma_i\,x_i\,\delta_i\,x_i)\\ &= v\,\tau(\gamma_i)\,v\,\tau(\delta_i)\,v\\ &= v\,\sigma(x)\,v\,\sigma(y)\,v,\\ \tau'(\ddot{\beta}_i) &= \tau'(x_i\,\eta_i\,x_i)\\ &= v\,\tau(\eta)\,v\\ &= v\,u\,v.\end{aligned}$$

As $\tau'(\gamma_j) = \tau'(\delta_j) = \tau'(\eta_j) = \tau'(\hat{\beta}_j) = \tau'(\ddot{\beta}_j) = \lambda$ for all $j \neq i$, this leads to

$$\begin{aligned}\tau'(\beta_{\mathcal{A}}) &= \tau'\left((x_1)^2\ldots(x_\mu)^2\#^4\hat{\beta}_1\ldots\hat{\beta}_\mu\#^4\ddot{\beta}_1\ldots\ddot{\beta}_\mu\right)\\ &= \tau'\left((x_i)^2\right)\#^4\tau'(\hat{\beta}_i)\#^4\tau'(\ddot{\beta}_i)\\ &= v\,v\,\#^4\,v\,\sigma(x)\,v\,\sigma(y)\,v\,\#^4\,v\,u\,v\\ &= \sigma(\alpha_{\mathcal{A}}).\end{aligned}$$

This proves $\sigma(\alpha_{\mathcal{A}}) \in L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$.

For the other direction, assume that $\sigma(\alpha_{\mathcal{A}}) \in L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$. If σ is of bad form, then by Lemma 1, σ satisfies π_1 or π_2. Thus, assume $\sigma(x)$ does not contain $\#^3$ as a factor, and $\sigma(y) \in 0^*$. Let τ be a substitution with $\tau(\beta_{\mathcal{A}}) = \sigma(\alpha_{\mathcal{A}})$.

Now, as σ is of good form, $\sigma(\alpha_{\mathcal{A}})$ contains exactly two occurrences of $\#^4$, and these are non-overlapping. As $\sigma(\alpha_{\mathcal{A}}) = \tau(\beta_{\mathcal{A}})$, the same holds for $\tau(\beta_{\mathcal{A}})$. Thus, the equation $\sigma(\alpha_{\mathcal{A}}) = \tau(\beta_{\mathcal{A}})$ can be decomposed into the system consisting of the following three equations:

$$0\#^3 0\,0\#^3 0 = \tau\left((x_1)^2\ldots(x_\mu)^2\right), \tag{1}$$

$$0\#^3 0\,\sigma(x)\,0\#^3 0\,\sigma(y)\,0\#^3 0 = \tau(\hat{\beta}_1\ldots\hat{\beta}_\mu), \tag{2}$$

$$0\#^3 0\,u\,0\#^3 0 = \tau(\ddot{\beta}_1\ldots\ddot{\beta}_\mu). \tag{3}$$

First, consider equation (1) and choose the smallest i for which $\tau(x_i) \neq \lambda$. Then $\tau(x_i)$ has to start with 0, and as

$$\tau\left((x_i)^2\ldots(x_\mu)^2\right) = 0\#^3 0\,0\#^3 0,$$

it is easy to see that $\tau(x_i) = 0\#^3 0 = v$ and $\tau(x_j) = \lambda$ for all $j \neq i$ must hold.

Note that u does not contain $0\#^3 0$ as a factor, and does neither begin with $\#^3 0$, nor end on $0\#^3$. But as $\tau(\ddot{\beta}_i)$ begins with and ends on $0\#^3 0$, we can use equation (3) to obtain $0\#^3 0\,u\,0\#^3 0 = \tau(\ddot{\beta}_i)$ and $\tau(\ddot{\beta}_j) = \lambda$ for all $j \neq i$. As $\ddot{\beta}_i = x_i\eta_i x_i$ and $\tau(x_i) = 0\#^3 0$, $\tau(\eta_i) = u$ must hold.

As σ is of good form, $\sigma(0\#^30\ x\ 0\#^30\ y\ 0\#^30)$ contains exactly three occurrences of $\#^3$. But there are already three occurrences of $\#^3$ in $\tau(\hat{\beta}_i) = 0\#^30\ \tau(\gamma_i)\ 0\#^30\ \tau(\delta_i)\ 0\#^30$. This, and equation (2), lead to $\tau(\hat{\beta}_j) = \lambda$ for all $j \neq i$ and, more importantly, $\tau(\gamma_i) = \sigma(x)$ and $\tau(\delta_i) = \sigma(y)$. Therefore, σ satisfies the predicate π_i. □

Thus, we can select predicates π_1 to π_μ in such a way that $L_{\mathrm{E},\Sigma}(\alpha_\mathcal{A}) \setminus L_{\mathrm{E},\Sigma}(\beta_\mathcal{A}) = \emptyset$ if and only if $\mathrm{VALC}(\mathcal{A}) = \emptyset$ by describing $\mathrm{INVALC}(\mathcal{A})$ through a disjunction of predicates on H. The proof of Lemma 2 shows that if $\sigma(\alpha_\mathcal{A}) = \tau(\beta_\mathcal{A})$ for substitutions σ, τ, where σ is of good form, there exists exactly one i $(3 \leq i \leq \mu)$ such that $\tau(x_i) = 0\#^30$.

Due to technical reasons, we need a predicate π_3 that, if unsatisfied, sets a lower bound on the length of $\sigma(y)$, defined by

$$\gamma_3 := y_{3,1}\ \hat{y}_{3,1}\ y_{3,2}\ \hat{y}_{3,2}\ y_{3,3}\ \hat{y}_{3,3}\ y_{3,4},$$
$$\delta_3 := \hat{y}_{3,1}\ \hat{y}_{3,2}\ \hat{y}_{3,3},$$

where all of $y_{3,1}$ to $y_{3,4}$ and $\hat{y}_{3,1}$ to $\hat{y}_{3,3}$ are pairwise distinct variables. Clearly, if some $\sigma \in H$ satisfies π_3, $\sigma(y)$ is a concatenation of three (possibly empty) factors of $\sigma(x)$. Thus, if σ satisfies none of π_1 to π_3, $\sigma(y)$ must be longer than the three longest non-overlapping sequences of 0s in $\sigma(x)$. This allows us to identify a class of predicates definable by a rather simple kind of expression, which we use to define π_4 to π_μ in a less technical way.

Let $X' := \{\hat{x}_1, \hat{x}_2, \hat{x}_3\} \subset X$, let G denote the set of those substitutions in H that are of good form and let R be the set of all substitutions $\rho : (\Sigma \cup X')^* \to \Sigma^*$ for which $\rho(0) = 0$, $\rho(\#) = \#$ and $\rho(\hat{x}_i) \in 0^*$ for all $i \in \{1, 2, 3\}$. For patterns $\alpha \in (\Sigma \cup X')^*$, we define $R(\alpha) := \{\rho(\alpha) \mid \rho \in R\}$.

Definition 1. A predicate $\pi : G \to \{0, 1\}$ is called a *simple predicate* if there exist a pattern $\alpha \in (\Sigma \cup X')^*$ and languages $L_1, L_2 \in \{\Sigma^*, \{\lambda\}\}$ such that σ satisfies π if and only if $\sigma(x) \in L_1\ R(\alpha)\ L_2$.

From a slightly different point of view, the elements of X' can be understood as numerical parameters describing (concatenational) powers of 0, with substitutions $\rho \in R$ acting as assignments. For example, if $\sigma \in G$ satisfies a simple predicate π if and only if $\sigma(x) \in \Sigma^* R(\#\hat{x}_1\#\hat{x}_2 0\#\hat{x}_1)$, we can also write that σ satisfies π if and only if $\sigma(x)$ has a suffix of the form $\#0^m\#0^n0\#0^m$ (with $m, n \in \mathbb{N}_0$), which could also be written as $\#0^m\#0^*0\#0^m$, as n occurs only once in this expression. Using π_3, our construction is able to express all simple predicates:

Lemma 3. *For every simple predicate π_S over n variables with $n \leq 3$, there exists a predicate π defined by terminal-free patterns γ, δ, η such that for all substitutions $\sigma \in G$:*

1. *if σ satisfies π_S, then σ also satisfies π or π_3,*
2. *if σ satisfies π, then σ also satisfies π_S.*

Proof. We first consider the case of $L_1 = L_2 = \Sigma^*$. Assume that π_S is a simple predicate, and $\alpha \in (\Sigma \cup X')^*$ is a pattern such that $\sigma \in G$ satisfies π_S if and only if $\sigma(x) \in L_1 R(\alpha) L_2$. Then define $\gamma := y_1 \alpha' y_2$, where α' is obtained from α by replacing all occurrences of 0 with a new variable z and all occurrences of # with a different variable $\hat{z}$, while leaving all present elements of X' unchanged. Furthermore, let $\delta := \hat{x}_1\hat{x}_2\hat{x}_3\hat{y}$ and (in order to stay consistent with the η_i appearing in $\beta_{\mathcal{A}}$) $\eta := z(\hat{z})^2 z$. Note that $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$, y_1, y_2, z and $\hat{z}$ are pairwise distinct variables.

Now, assume that $\sigma \in G$ satisfies π_S. Then there exist words $w_1, w_2 \in \Sigma^*$ and a substitution $\rho \in R$ such that $\sigma(x) = w_1\, \rho(\alpha)\, w_2$. If $\sigma(y)$ is not longer than any three non-overlapping factors of the form 0^* of $\sigma(x)$ combined, π_3 is satisfied. Otherwise, we can define τ by setting $\tau(y_1) := w_1$, $\tau(y_2) := w_2$, $\tau(z) := 0$, $\tau(\hat{z}) := \#$, $\tau(\hat{x}_i) := \rho(\hat{x}_i)$ for all $i \in \{1, \ldots, 3\}$ where $\hat{x}_i$ appears in α and $\tau(\hat{x}_i) := \lambda$ where $\hat{x}_i$ does not appear in α. Finally, let $\tau(\hat{y}) := 0^m$, where

$$m := |\sigma(y)| - \sum_{\hat{x} \in \mathrm{var}(\alpha)} |\sigma(\hat{x})|$$

($m > 0$ must hold, as σ does not satisfy π_3). Then $\tau(\alpha') = \rho(\alpha)$, and

$$\begin{aligned}
\tau(\gamma) &= \tau(y_1)\, \tau(\alpha')\, \tau(y_2) \\
&= w_1\, \rho(\alpha)\, w_2 = \sigma(x), \\
\tau(\delta) &= \tau(\hat{x}_1\hat{x}_2\hat{x}_3\, \hat{y}) \\
&= 0^{|\sigma(y)|} = \sigma(y), \\
\tau(\eta) &= \tau(z\, (\hat{z})^2\, z) \\
&= 0\#\#0 = u.
\end{aligned}$$

Therefore, σ satisfies π, which concludes this direction.

For the other direction, assume that $\sigma \in G$ satisfies π. Then there is a morphism τ such that $\sigma(x) = \tau(\gamma)$, $\sigma(y) = \tau(\delta)$ and $\tau(\eta) = u$. As $\eta = z\, (\hat{z})^2\, z$ and $u = 0\#\#0$, $\tau(z) = 0$ and $\tau(\hat{z}) = \#$ must hold. By definition $\tau(y_1), \tau(y_2) \in \Sigma^*$. If we define $\rho(\hat{x}_i) := \tau(\hat{x}_i)$ for all $i \in \{1, 2, 3\}$, we see that $\sigma(x) \in L_1 R(\alpha) L_2$ holds. Thus, σ satisfies π_S as well.

The other three cases for choices of L_1 and L_2 can be handled analogously by omitting y_1 or y_2 as needed. Note that this proof also works in the case $\alpha = \lambda$. □

Roughly speaking, if σ does not satisfy π_3, then $\sigma(y)$ (which is in 0^*, due to $\sigma \in G$) is long enough to provide building blocks for simple predicates using variables from X'.

Our next goal is a set of predicates that (if unsatisfied) forces $\sigma(x)$ into a basic shape common to all elements of $\mathrm{VALC}(\mathcal{A})$. We say that a word $w \in \{0, \#\}^*$ is of *good structure* if $w \in (\#\#0^+\#0^+\#0^+)^+\,\#\#$. Otherwise, w is of *bad structure*. Recall that due to the definition of enc, all elements of $\mathrm{VALC}(\mathcal{A})$ are of good structure, thus being of bad structure is a sufficient criterion for belonging to

INVALC($\mathcal{A}$). In order to cover the morphisms σ where $\sigma(x)$ is of bad structure, we define predicates π_4 to π_{13} through simple predicates as follows:

$\pi_4 : \sigma(x) = \lambda$,	$\pi_9 : \sigma(x)$ ends on 0,
$\pi_5 : \sigma(x) = \#$,	$\pi_{10} : \sigma(x)$ ends on 0#,
$\pi_6 : \sigma(x) = \#\#$,	$\pi_{11} : \sigma(x)$ contains a factor $\#\#0^*\#\#$,
$\pi_7 : \sigma(x)$ begins with 0,	$\pi_{12} : \sigma(x)$ contains a factor $\#\#0^*\#0^*\#\#$,
$\pi_8 : \sigma(x)$ begins with #0,	$\pi_{13} : \sigma(x)$ contains a factor $\#\#0^*\#0^*\#0^*\#0$.

Due to Lemma 3, the predicates π_1 to π_{13} do not strictly give rise to a characterization of substitutions with images that are of bad structure, as there are $\sigma \in G$ where $\sigma(x)$ is of good structure, but π_3 is satisfied due to $\sigma(y)$ being too short. But this problem can be avoided by choosing $\sigma(y)$ long enough to leave π_3 unsatisfied, and the following holds:

Lemma 4. *A word $w \in \Sigma^*$ is of good structure if and only if there exists a substitution $\sigma \in H$ with $\sigma(x) = w$ such that σ satisfies none of the predicates π_1 to π_{13}.*

Proof. We begin with the *if* direction. Assume $\sigma \in H$ such that there is no $i \in \{1, \ldots, 13\}$ for which σ satisfies π_i. Due to Lemma 1, σ is of good form and, thus, $\sigma(y) \in 0^*$. As π_3 does not hold, $\sigma(y)$ is also longer than any three non-overlapping factors 0^* of $\sigma(x)$. Thus, the structure of $\sigma(x)$ can be inferred by intersecting the complements of the simple predicates given in the definitions of π_4 to π_{13}.

As σ does not satisfy π_4, $\sigma(x) \neq \lambda$. Due to π_7 and π_9, the first and the last letter of $\sigma(x)$ is #, and neither is #0 a prefix, nor 0# a suffix of $\sigma(x)$, as otherwise π_8 or π_{10} would be satisfied. Therefore, $\sigma(x)$ has ## as prefix and suffix, but, as π_6 is not satisfied, $\sigma(x) \neq \#\#$. As σ is of good form, $\sigma(x)$ does not contain ### as a factor, and

$$\sigma(x) \in \#\#0^+\Sigma^*\#\#$$

must hold. But as π_{11} is not satisfied, it is possible to refine this observation to

$$\sigma(x) \in \#\#0^+\#0^+\Sigma^*\#\#,$$

which in turn leads to

$$\sigma(x) \in \#\#0^+\#0^+\#0^+\Sigma^*\#\#$$

by considering π_{12} as well. Now, there are two possibilities. If

$$\sigma(x) \in \#\#0^+\#0^+\#0^+\#\#,$$

then $\sigma(x)$ is of good structure, but if

$$\sigma(x) \in \#\#0^+\#0^+\#0^+\#\Sigma^*\#\#,$$

then π_{13} and $\sigma \in G$ lead to

$$\sigma(x) \in \#\#0^+\#0^+\#0^+\#\#0^+\Sigma^*\#\#.$$

In this case, we can continue referring subsequently to one of π_{11}, π_{12} and π_{13} together with $\sigma \in G$, and conclude

$$\sigma(x) \in \left(\#\#0^+\#0^+\#0^+\right)^+\#\#.$$

Therefore, if σ satisfies none of π_1 to π_{13}, then $\sigma(x)$ has to be of good structure.

Regarding the *only if* direction, assume some $w \in \Sigma^*$ is of good structure. Define σ by $\sigma(x) = w$ and $\sigma(y) = 0^{w+1}$. As σ is of good form, Lemma 1 demonstrates that σ satisfies neither π_1 nor π_2; and as $\sigma(y)$ is longer than any word which results from concatenating any number of non-overlapping factors of the form 0^* of w, π_3 cannot be satisfied either. By looking at the cases used above to define π_4 to π_{13}, we see that none of these predicates can be satisfied. □

For every w of good structure, there exist uniquely determined n, i_1, j_1, k_1, $\ldots$, i_n, j_n, $k_n \in \mathbb{N}_1$ such that $w = \#\#0^{i_1}\#0^{j_1}\#0^{k_1}\#\#\ldots\#\#0^{i_n}\#0^{j_n}\#0^{k_n}\#\#$. Thus, if $\sigma \in H$ does not satisfy any of π_1 to π_{13}, $\sigma(x)$ can be understood as an encoding of a sequence $T_1, \ldots, T_n$ of triples $T_i \in (\mathbb{N}_1)^3$, and for every sequence of that form, there is a $\sigma \in H$ such that $\sigma(x)$ encodes a sequence of triples of positive integers, and σ does not satisfy any of π_1 to π_{13}.

In the encoding of computations that is defined by enc, ## is always a border between the encodings of configurations, whereas single # separate the elements of configurations. As we encode every state q_i with 0^{i+1}, the predicate π_{14}, which is to be satisfied whenever $\sigma(x)$ contains a factor $\#\#00^{s+1}$, handles all encoded triples (i, j, k) with $i > s + 1$. If σ does not satisfy this simple predicate (in addition to the previous ones), there is a computation $C_1, \ldots, C_n$ of $\mathcal{A}$ with $\mathrm{enc}(C_1, \ldots, C_n) = \sigma(x)$.

All that remains is to choose an appropriate set of predicates that describe all cases where C_1 is not the initial configuration, C_n is not an accepting configuration, or there are configurations C_i, C_{i+1} such that $C_i \vdash_{\mathcal{A}} C_{i+1}$ does not hold (thus, the exact value of μ depends on the number of invalid transitions in $\mathcal{A}$).

To ensure $C_1 = (q_0, 0, 0)$, we define a predicate

1. $\sigma(x)$ has a prefix of the form $\#\#00$,

that is satisfied if C_1 has a state q_i with $i > 0$, and the two predicates

2. $\sigma(x)$ has a prefix of the form $\#\#0^*\#00$,
3. $\sigma(x)$ has a prefix of the form $\#\#0^*\#0^*\#00$,

to cover all cases where one of the counters is set to a value other than 0. Next, we handle the cases where the last state is not an accepting state. For every i with $q_i \in Q \setminus F$, define a predicate that is satisfied if

4. $\sigma(x)$ has a suffix of the form $\#\#\, 00^i \,\#\, 0^* \,\#\, 0^* \,\#\#$.

Thus, if $\sigma \in H$ satisfies none of the predicates defined up to this point, $\sigma(x) = \mathrm{enc}(C_1, \ldots, C_n)$ for some computation of $\mathcal{A}$ where C_1 is the initial configuration $(q_0, 0, 0)$, and C_n is an accepting configuration. Likewise, for every computation $C_1, \ldots, C_n$ with $C_1 = (q_0, 0, 0)$ and $C_n \in F \times \mathbb{N}_0 \times \mathbb{N}_0$, there is a $\sigma \in H$ with $\sigma(x) = \mathrm{enc}(C_1, \ldots, C_n)$, and σ satisfies none of these predicates.

All that remains is to define a set of predicates that describe those C_i, C_{i+1} for which $C_i \vdash_{\mathcal{A}} C_{i+1}$ does not hold. To simplify this task, we define the following four predicates that are satisfied if one of the counters is changed by more than 1:

5. $\sigma(x)$ contains a factor of the form $\#0^m\#0^*\#\#0^*\#00\,0^m$ for some $m \in \mathbb{N}_0$,
6. $\sigma(x)$ contains a factor of the form $0^m\,00\#0^*\#\#0^*\#0^m\#$ for some $m \in \mathbb{N}_0$,
7. $\sigma(x)$ contains a factor of the form $\#0^m\#\#0^*\#0^*\#00\,0^m$ for some $m \in \mathbb{N}_0$,
8. $\sigma(x)$ contains a factor of the form $0^m\,00\#\#0^*\#0^*\#0^m\#$ for some $m \in \mathbb{N}_0$.

Here, the first two predicates cover incrementing (or decrementing) the first counter by 2 or more; the other two do the same for the second counter. Then, for all $i, j \in \{1, \ldots, s\}$, all $c_1, c_2 \in \{0, 1\}$ and all $r_1, r_2 \in \{-1, 0, +1\}$ for which $(q_j, r_1, r_2) \notin \delta(q_i, c_1, c_2)$, we define a predicate that is satisfied if $\sigma(x)$ contains such a transition. We demonstrate this only for the exemplary case $c_1 = 0$, $c_2 = 1$, $r_1 = +1$, $r_2 = 0$ without naming i or j explicitly. The predicate covering non-existing transitions of this form is

9. $\sigma(x)$ contains a factor of the form $\#\#0^{i+1}\#0\#000^m\#\#0^{j+1}\#00\#000^m\#\#$ for some $m \in \mathbb{N}_0$.

All other predicates for illegal transitions are defined analogously. Note that we can safely assume that none of the counters is changed by more than 1, as these errors are covered by the predicates we defined under points 5 to 8. The number of predicates required for these points and point 9 determine the exact value of μ.

Now, if there is a substitution σ that does not satisfy any of π_1 to π_μ, then $\sigma(x) = \mathrm{enc}(C_1, \ldots, C_n)$ for a computation $C_1, \ldots, C_n$, where C_1 is the initial and C_n a final configuration, and for all $i \in \{1, \ldots, n-1\}$, $C_i \vdash_{\mathcal{A}} C_{i+1}$. Thus, if $\sigma(\alpha_{\mathcal{A}}) \notin L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$, then $\sigma(x) \in \mathrm{VALC}(\mathcal{A})$, which means that $\mathcal{A}$ has an accepting computation.

Conversely, if there is some accepting computation $C_1, \ldots, C_n$ of $\mathcal{A}$, we can define σ through $\sigma(x) := \mathrm{enc}(C_1, \ldots, C_n)$, and choose $\sigma(y)$ to be an appropriately long sequence from 0^*. Then σ does not satisfy any of the predicates π_1 to π_μ defined above, thus $\sigma(\alpha_{\mathcal{A}}) \notin L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$, and $L_{\mathrm{E},\Sigma}(\alpha_{\mathcal{A}}) \not\subseteq L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$.

We conclude that $\mathcal{A}$ has an accepting computation iff $L_{\mathrm{E},\Sigma}(\alpha_{\mathcal{A}})$ is not a subset of $L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$. Therefore, any algorithm deciding the inclusion problem for ePAT_Σ can be used to decide whether a nondeterministic 2-counter automaton without input has an accepting computation. As this problem is known to be undecidable, the inclusion problem for ePAT_Σ is also undecidable.

This proof can be extended to larger (finite) alphabets. Assume that $\Sigma = \{0, \#, \mathtt{a}_1, \ldots, \mathtt{a}_n\}$ for some $n \geq 1$. We extend H to the set of all substitutions

$\sigma : (\Sigma \cup \{x, y\})^* \to \Sigma^*$, but do not extend the definition of substitutions of good form to our new and larger alphabet. Thus, $\sigma \in H$ is of good form if $\sigma(x) \in \{0, \#\}^*$, $\sigma(y) \in 0^*$ and $\sigma(x)$ does not contain $\#^3$ as a factor. In addition to the predicates π_1 to π_μ, for each new letter $\mathtt{a}_i$, we define a predicate $\pi_{\mu+2i-1}$ which implies that $\sigma(x)$ contains an occurrence of $\mathtt{a}_i$, and a predicate $\pi_{\mu+2i}$ which implies that $\sigma(y)$ contains an occurrence of $\mathtt{a}_i$. To this end, we define

$$\alpha_{\mathcal{A}} := v\, v\, \#^4\, v\, x\, v\, y\, v\, \#^4\, v\, u\, v,$$

with $v = 0\#^3 0$ and $u = 0\#\#\, \mathtt{a}_1\, \mathtt{a}_1 \ldots \mathtt{a}_n\, \mathtt{a}_n\, 0$ (instead of $u = 0\#\#0$), add the new predicates $\pi_{\mu+1}$ to $\pi_{\mu+2n}$ (which we still leave unspecified for a moment) to $\beta_{\mathcal{A}}$ and use

$$\eta_i := z_i(\hat{z}_i)^2(\ddot{z}_{i,1})^2 \ldots (\ddot{z}_{i,n})^2 z_i$$

instead of $\eta_i = z_i(\hat{z}_i)^2 z_i$, where all z_i, $\hat{z}_i$, $\ddot{z}_{i,j}$ are pairwise different variables. Referring to the new shape of u, we can make the following observation:

Lemma 5. *Let $n \geq 1$, $\{x_1 \ldots, x_n\} \subset X$ and $\{\mathtt{a}_1, \ldots, \mathtt{a}_n\} \subseteq \Sigma$. If*

$$\alpha = x_1\, (x_2)^2 \ldots (x_n)^2\, x_1$$

and there is a morphism $\sigma : X^* \to \Sigma^*$ with $\sigma(\alpha) = \mathtt{a}_1\, (\mathtt{a}_2)^2 \ldots (\mathtt{a}_n)^2\ \mathtt{a}_1$, then $\sigma(x_i) = \mathtt{a}_i$ for each $i \in \{1, \ldots, n\}$.

Proof. Assume $\sigma\left(x_1\, (x_2)^2 \ldots (x_n)^2\, x_1\right) = \mathtt{a}_1\, (\mathtt{a}_2)^2 \ldots (\mathtt{a}_n)^2\ \mathtt{a}_1$. If $\sigma(x_1) = \lambda$, then

$$\sigma\left((x_2)^2 \ldots (x_n)^2\right) = \mathtt{a}_1\, (\mathtt{a}_2)^2 \ldots (\mathtt{a}_n)^2\ \mathtt{a}_1$$

leads to an immediate contradiction. But $\sigma(x_1) \neq \lambda$ implies $\sigma(x_1) = \mathtt{a}_1$. Therefore,

$$\sigma\left((x_2)^2 \ldots (x_n)^2\right) = (\mathtt{a}_2)^2 \ldots (\mathtt{a}_n)^2$$

must hold. Now, for every $i \in \{2, \ldots, n\}$ with $\sigma(x_i) \neq \lambda$, $|\sigma(x_i)| = 1$ must hold, as $\sigma(\alpha)$ does not contain squares that are longer than two letters. Thus, every $(x_i)^2$ generates at most one factor $(\mathtt{a}_j)^2$, and every factor $(\mathtt{a}_j)^2$ has to be generated by some $(x_i)^2$. We conclude that for every x_i there is a j with $\sigma(x_i) = \mathtt{a}_j$. Of course, this is only possible if $i = j$ in all cases; therefore, the claim holds. □

Lemma 5 allows $\pi_{\mu+1}$ to $\pi_{\mu+2n}$ to be analogously constructed to π_2. To this end, we define

$$\gamma_{\mu+2i-1} := y_{\mu+2i-1,1}\, \ddot{z}_{\mu+2i-1,i}\, y_{\mu+2i-1,2}, \quad \gamma_{\mu+2i} := y_{\mu+2i},$$
$$\delta_{\mu+2i-1} := \hat{y}_{\mu+2i-1}, \quad \delta_{\mu+2i} := \hat{y}_{\mu+2i,1}\, \ddot{z}_{\mu+2i,i}\, \hat{y}_{\mu+2i,2}.$$

for each $i \in \{1, \ldots, n\}$. Again, all $y_{j,k}$, $\hat{y}_{j,k}$, z_j, $\hat{z}_j$ and $\ddot{z}_{j,k}$ are pairwise different variables. Now Lemma 1 applies (mutatis mutandis) as for binary alphabets, and since all substitutions of good form behave for Σ as for the binary alphabet, we can use the very same predicates and the same reasoning as before to prove undecidability of the inclusion problem for ePAT_Σ.

This concludes the proof of Theorem 2.

4. The inclusion of similar E-pattern languages

It can be easily observed that the patterns $\alpha_{\mathcal{A}}$ and $\beta_{\mathcal{A}}$ used in Section 3.1 for establishing the undecidability of the inclusion problem for E-pattern languages are not *similar* (see the definition of this term in Section 2). Hence, our reasoning in Section 3.1 does not provide any insights into the inclusion of similar E-pattern languages. A promising and substantial first statement on this natural subproblem is given by Jiang et al. [10]; they show that for the full class of the *simplest* similar E-pattern languages, namely those generated by *terminal-free* patterns, inclusion is decidable. This directly results from the following characterization:

Theorem 5 (Jiang et al. [10])**.** *Let Σ be an alphabet, $|\Sigma| \geq 2$, and let $\alpha, \beta \in \mathrm{Pat}_{\mathrm{tf}}$ be terminal-free patterns. Then $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ if and only if there exists a morphism $\phi : X^* \to X^*$ satisfying $\phi(\beta) = \alpha$.*

Note that the decidability of the inclusion problem for terminal-free *NE*-pattern languages is still open.

The problem of the extensibility of Theorem 5 to general similar patterns (replacing $\phi : X^* \to X^*$ by a terminal-preserving morphism $\phi : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$) is not only of intrinsic interest, but it has a major impact on the so far unresolved *equivalence* problem for E-pattern languages (see our explanations below). Therefore it has attracted a lot of attention, and it is largely conjectured in literature (e. g., Dányi and Fülöp [4], Ohlebusch and Ukkonen [18]) that the inclusion of arbitrary similar E-pattern languages shows the same property as that of terminal-free E-pattern languages. Our main result of the present section, however, demonstrates that this conjecture is not correct:

Theorem 6. *For every finite alphabet Σ, there exist similar patterns $\alpha, \beta \in \mathrm{Pat}_{\Sigma}$ such that*

- *$L_{\mathrm{E},\Sigma}(\alpha) \subset L_{\mathrm{E},\Sigma}(\beta)$ and*
- *there is no terminal-preserving morphism $\phi : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ satisfying $\phi(\beta) = \alpha$.*

Proof. If Σ is unary, e. g. $\Sigma := \{\mathtt{a}\}$, then Theorem 6 is proved by the patterns $\alpha := x_1\,\mathtt{a}\,x_1$ and $\beta := x_1\,\mathtt{a}\,x_2x_2$. We leave the verification of this claim to the reader.

For the remainder of this proof, we now assume that $|\Sigma| \geq 2$. Hence, let $\Sigma := \{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n\}$ with $n \geq 2$ and $\mathtt{a}_i \neq \mathtt{a}_j$ for all i, j satisfying $i \neq j$. If n is odd then we define

$$\alpha := \alpha_0\,\mathtt{a}_1\,\alpha_1\,\mathtt{a}_1\,\alpha_2\,\mathtt{a}_2\,\alpha_3\,\mathtt{a}_2\,\alpha_4\ \ldots\ \alpha_{2n-4}\,\mathtt{a}_{n-1}\,\alpha_{2n-3}\,\mathtt{a}_{n-1}\,\alpha_{2n-2}\,\mathtt{a}_n\,\alpha_{2n-1}\,\mathtt{a}_n\,\alpha_{2n}$$

and

$$\beta := \beta_0\,\mathtt{a}_1\,\beta_1\,\mathtt{a}_1\,\beta_2\,\mathtt{a}_2\,\beta_3\,\mathtt{a}_2\,\beta_4\ \ldots\ \beta_{2n-4}\,\mathtt{a}_{n-1}\,\beta_{2n-3}\,\mathtt{a}_{n-1}\,\beta_{2n-2}\,\mathtt{a}_n\,\beta_{2n-1}\,\mathtt{a}_n\,\beta_{2n}$$

with

$$\alpha_i := \begin{cases} x_{i+1} & , \quad i = 0 \text{ or } i \text{ is even}\,, \\ x_2 & , \quad i \text{ is odd}\,, \end{cases}$$

and

$$\beta_i := \begin{cases} x_{i+1} & , \quad i = 0 \text{ or } i \text{ is even}\,, \\ x_{i+1}x_{i+3} & , \quad i \text{ is odd and } i \neq 2n-1\,, \\ x_{2n}x_2 & , \quad i = 2n-1\,. \end{cases}$$

If n is even then

$$\begin{aligned} \alpha \;:=\; & \alpha_0\,\mathtt{a}_1\,\alpha_1\,\mathtt{a}_1\,\alpha_2\,\mathtt{a}_2\,\alpha_3\,\mathtt{a}_2\,\alpha_4\;\ldots\;\alpha_{2n-4}\,\mathtt{a}_{n-1}\,\alpha_{2n-3}\,\mathtt{a}_{n-1}\,\alpha_{2n-2}\,\mathtt{a}_n\,\alpha_{2n-1}\,\mathtt{a}_n\,\alpha_{2n} \\ & \mathtt{a}_n\,\alpha_{2n+1}\,\mathtt{a}_n\,\alpha_{2n+2} \end{aligned}$$

and

$$\begin{aligned} \beta \;:=\; & \beta_0\,\mathtt{a}_1\,\beta_1\,\mathtt{a}_1\,\beta_2\,\mathtt{a}_2\,\beta_3\,\mathtt{a}_2\,\beta_4\;\ldots\;\beta_{2n-4}\,\mathtt{a}_{n-1}\,\beta_{2n-3}\,\mathtt{a}_{n-1}\,\beta_{2n-2}\,\mathtt{a}_n\,\beta_{2n-1}\,\mathtt{a}_n\,\beta_{2n} \\ & \mathtt{a}_n\,\beta_{2n+1}\,\mathtt{a}_n\,\beta_{2n+2} \end{aligned}$$

with

$$\alpha_i := \begin{cases} x_{i+1} & , \quad i = 0 \text{ or } i \text{ is even}\,, \\ x_2 & , \quad i \text{ is odd}\,, \end{cases}$$

and

$$\beta_i := \begin{cases} x_{i+1} & , \quad i = 0 \text{ or } i \text{ is even}\,, \\ x_{i+1}x_{i+3} & , \quad i \text{ is odd and } i \neq 2n+1\,, \\ x_{2n+2}x_2 & , \quad i = 2n+1\,. \end{cases}$$

This means that, e. g., for $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$ the patterns read

$$\begin{aligned} \alpha &= x_1\,\mathtt{a}\,x_2\,\mathtt{a}\,x_3\,\mathtt{b}\,x_2\,\mathtt{b}\,x_5\,\mathtt{c}\,x_2\,\mathtt{c}\,x_7\,\mathtt{d}\,x_2\,\mathtt{d}\,x_9\,\mathtt{e}\,x_2\,\mathtt{e}\,x_{11}, \\ \beta &= x_1\,\mathtt{a}\,x_2x_4\,\mathtt{a}\,x_3\,\mathtt{b}\,x_4x_6\,\mathtt{b}\,x_5\,\mathtt{c}\,x_6x_8\,\mathtt{c}\,x_7\,\mathtt{d}\,x_8x_{10}\,\mathtt{d}\,x_9\,\mathtt{e}\,x_{10}x_2\,\mathtt{e}\,x_{11}, \end{aligned}$$

and, for $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$, our definition leads to

$$\begin{aligned} \alpha &= x_1\,\mathtt{a}\,x_2\,\mathtt{a}\,x_3\,\mathtt{b}\,x_2\,\mathtt{b}\,x_5\,\mathtt{c}\,x_2\,\mathtt{c}\,x_7\,\mathtt{d}\,x_2\,\mathtt{d}\,x_9\,\mathtt{e}\,x_2\,\mathtt{e}\,x_{11}\,\mathtt{f}\,x_2\,\mathtt{f}\,x_{13}\,\mathtt{f}\,x_2\,\mathtt{f}\,x_{15}, \\ \beta &= x_1\,\mathtt{a}\,x_2x_4\,\mathtt{a}\,x_3\,\mathtt{b}\,x_4x_6\,\mathtt{b}\,x_5\,\mathtt{c}\,x_6x_8\,\mathtt{c}\,x_7\,\mathtt{d}\,x_8x_{10}\,\mathtt{d}\,x_9\,\mathtt{e}\,x_{10}x_{12}\,\mathtt{e}\,x_{11}\,\mathtt{f}\,x_{12}x_{14}\,\mathtt{f}\,x_{13} \\ &\quad \mathtt{f}\,x_{14}x_2\,\mathtt{f}\,x_{15}. \end{aligned}$$

It follows by definition that, for every $n \geq 2$, the corresponding patterns α and β are similar.

We first prove that, for every $n \geq 2$, there is no terminal-preserving morphism ϕ mapping β onto α. Assume to the contrary that there is such a ϕ. If n is odd then the variable x_2 has n occurrences in α. This means that ϕ maps the pattern $\beta_1\beta_3\beta_5\ldots\beta_{2n-1}$ onto the pattern x_2^n. Since each variable in $\beta_1\beta_3\beta_5\ldots\beta_{2n-1}$ has an even number of occurrences, this is a contradiction. If n is even then the variable x_2 has $n+1$ occurrences in α. Consequently, ϕ maps the pattern $\beta_1\beta_3\beta_5\ldots\beta_{2n+1}$ onto the pattern x_2^{n+1}. Since each variable in $\beta_1\beta_3\beta_5\ldots\beta_{2n+1}$ again has an even number of occurrences, this leads to the same contradiction.

We now demonstrate that, for every finite alphabet Σ, the corresponding patterns α and β satisfy $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$. To this end, let v be an arbitrary word in $L_{\mathrm{E},\Sigma}(\alpha)$; hence, there is a substitution $\tau : (\Sigma \cup X)^* \to \Sigma^*$ with $\tau(\alpha) = v$. We now define a substitution τ' with $\tau'(\beta) = v$.

Case 1: If $\tau(x_2) = \lambda$ then, for every $x_k \in \mathrm{var}(\beta)$,

$$\tau'(x_k) := \begin{cases} \lambda & , \quad k \text{ is even}\,, \\ \tau(x_k) & , \quad k \text{ is odd}\,. \end{cases}$$

Case 2: If, for $\mathtt{a}_j \in \Sigma$ and $u \in \Sigma^*$, $\tau(x_2) = \mathtt{a}_j\, u$ then we introduce a set ERASE $\subset \mathrm{var}(\beta)$ by

$$\mathrm{ERASE} := \mathrm{var}(\beta) \cap \{x_s \mid s = 2j - 4t \text{ or } s = 2j + 2 + 4t \text{ for a } t \in \mathbb{N}_0\},$$

and, for every $x_k \in \mathrm{var}(\beta)$, we define τ' by

$$\tau'(x_k) := \begin{cases} \mathtt{a}_j\ u & , \quad k \text{ is even and } x_k \notin \mathrm{ERASE}\,, \\ \lambda & , \quad x_k \in \mathrm{ERASE}\,, \\ \tau(x_k) & , \quad k \text{ is odd and } k \neq 2j+1\,, \\ u\ \mathtt{a}_j\, \tau(x_k) & , \quad k = 2j+1\,. \end{cases}$$

Note that, due to our definition of the set ERASE, it is guaranteed that in Case 2 both variables in β_{2j-1} (which is the factor that is surrounded by the two occurrences of $\mathtt{a}_j$ in β) are mapped by τ' onto the empty word, whereas, for each i with $i \neq j$, the factor β_{2i-1} contains exactly one such variable (and the same holds for the factor β_{2n+1} which is contained in β for an even n only). Referring to this observation, in both Case 1 and Case 2 the verification of $\tau'(\beta) = v$ is straightforward. Therefore, we merely illustrate the correctness of our claim for the case $\tau(x_2) \neq \lambda$ and an odd alphabet size n with $n \geq 3$ and $1 < j < n$ by the following diagram:

$$\begin{aligned} v \;=\; & \underbrace{\tau(x_1)}_{\tau'(x_1)}\, \mathtt{a}_1\, \underbrace{\mathtt{a}_j\, u}_{\tau'(x_2x_4)}\, \mathtt{a}_1\, \underbrace{\tau(x_3)}_{\tau'(x_3)} \ldots \underbrace{\tau(x_{2j-1})}_{\tau'(x_{2j-1})}\, \mathtt{a}_j\, \underbrace{\lambda}_{\tau'(x_{2j}x_{2j+2})}\, \mathtt{a}_j\, \underbrace{u\, \mathtt{a}_j\, \tau(x_{2j+1})}_{\tau'(x_{2j+1})} \ldots \\ & \underbrace{\tau(x_{2n-1})}_{\tau'(x_{2n-1})}\, \mathtt{a}_n\, \underbrace{\mathtt{a}_j\, u}_{\tau'(x_{2n}x_2)}\, \mathtt{a}_n\, \underbrace{\tau(x_{2n+1})}_{\tau'(x_{2n+1})}\,. \end{aligned}$$

Consequently, $v \in L_{\mathrm{E},\Sigma}(\beta)$ and, thus, $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$.

Finally, we show that $L_{\mathrm{E},\Sigma}(\alpha) \neq L_{\mathrm{E},\Sigma}(\beta)$. To this end, we consider the substitution $\sigma : (\Sigma \cup X)^* \to \Sigma^*$ given by, for every $x_k \in \mathrm{var}(\beta)$,

$$\sigma(x_k) := \begin{cases} \mathtt{a}_2 & , \quad k = 2\,, \\ \lambda & , \quad \text{else}\,. \end{cases}$$

We assume to the contrary that there is a substitution σ' satisfying $\sigma'(\alpha) = \sigma(\beta)$. By definition, the word $w := \sigma(\beta)$ is of length $2n + 2$ if n is odd and

$2(n+1)+2$ if n is even. Moreover, w contains the factor $\mathtt{a}_1\,\mathtt{a}_2\,\mathtt{a}_1$, and these are the only occurrences of letter $\mathtt{a}_1$ in w. Therefore, σ' has to satisfy $\sigma'(x_2) = \mathtt{a}_2$. However, since the variable x_2 has at least 3 occurrences in α this implies that $|\sigma'(\alpha)| \geq 2n+3$ if n is odd and $|\sigma'(\alpha)| \geq 2(n+1)+3$ if n is even. This is a contradiction. Consequently, $w \in L_{\mathrm{E},\Sigma}(\beta) \setminus L_{\mathrm{E},\Sigma}(\alpha)$, and therefore $L_{\mathrm{E},\Sigma}(\alpha) \neq L_{\mathrm{E},\Sigma}(\beta)$. □

We expect that the patterns α and β introduced in the proof are the shortest examples showing such a property. Alternatively, for any alphabet $\Sigma := \{\mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n\}$ with $|\Sigma| \geq 3$, the proof of Theorem 6 could use the following pattern β' instead of β:

$$\begin{aligned}\beta' \;:=\; & x_1\,\mathtt{a}_1\,x_2x_4\;\ldots\;x_{2n-2}\,\mathtt{a}_1\\ & x_3\,\mathtt{a}_2\,x_2x_4\;\ldots\;x_{2n-4}x_{2n}\,\mathtt{a}_2\\ & x_5\,\mathtt{a}_3\,x_2x_4\;\ldots\;x_{2n-6}x_{2n-2}x_{2n}\,\mathtt{a}_3\\ & \ldots\\ & x_{2n-1}\,\mathtt{a}_n\,x_4x_6\;\ldots\;x_{2n}\,\mathtt{a}_n\,x_{2n+1}\,.\end{aligned}$$

E. g., for $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, this means that

$$\begin{aligned}\beta' \;=\; & x_1\,\mathtt{a}\,x_2x_4x_6x_8\,\mathtt{a}\,x_3\,\mathtt{b}\,x_2x_4x_6x_{10}\,\mathtt{b}\,x_5\,\mathtt{c}\,x_2x_4x_8x_{10}\,\mathtt{c}\,x_7\,\mathtt{d}\,x_2x_6x_8x_{10}\,\mathtt{d}\,x_9\\ & \mathtt{e}\,x_4x_6x_8x_{10}\,\mathtt{e}\,x_{11}\end{aligned}$$

and, for $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$,

$$\begin{aligned}\beta' \;=\; & x_1\,\mathtt{a}\,x_2x_4x_6x_8x_{10}\,\mathtt{a}\,x_3\,\mathtt{b}\,x_2x_4x_6x_8x_{12}\,\mathtt{b}\,x_5\,\mathtt{c}\,x_2x_4x_6x_{10}x_{12}\,\mathtt{c}\,x_7\\ & \mathtt{d}\,x_2x_4x_8x_{10}x_{12}\,\mathtt{d}\,x_9\,\mathtt{e}\,x_2x_6x_8x_{10}x_{12}\,\mathtt{e}\,x_{11}\,\mathtt{f}\,x_4x_6x_8x_{10}x_{12}\,\mathtt{f}\,x_{13}\,.\end{aligned}$$

Note that if β' is chosen instead of β for the proof of Theorem 6 then this decision implies that, for both odd and even alphabet sizes, the corresponding pattern α needs to have the shape as introduced for odd alphabet sizes.

The relevance of Theorem 6 for the research on the equivalence problem for E-pattern languages follows from a result by Jiang et al. [9], which says that, for alphabets with at least three letters, two patterns need to be similar if they generate the same E-pattern language:

Theorem 7 (Jiang et al. [9])**.** *Let Σ be an alphabet, $|\Sigma| \geq 3$, and let $\alpha, \beta \in \mathrm{Pat}_\Sigma$. If $L_{\mathrm{E},\Sigma}(\alpha) = L_{\mathrm{E},\Sigma}(\beta)$ then α and β are similar.*

Consequently, in literature the inclusion problem for similar E-pattern languages is mainly understood as a tool for gaining a deeper understanding of the equivalence problem, and the main conjecture by Ohlebusch and Ukkonen [18] expresses the expectation that the relation between inclusion problem for similar E-pattern languages and equivalence problem might be equivalent to the relation between these problems for terminal-free patterns (cf. Theorem 5):

Conjecture 1 (Ohlebusch and Ukkonen [18]). *Let Σ be an alphabet, $|\Sigma| \geq 3$, and let $\alpha, \beta \in \mathrm{Pat}_\Sigma$. Then $L_{\mathrm{E},\Sigma}(\alpha) = L_{\mathrm{E},\Sigma}(\beta)$ if and only if there exist terminal-preserving morphisms $\phi, \psi : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ satisfying $\phi(\beta) = \alpha$ and $\psi(\alpha) = \beta$.*

Note that the existence of ϕ and ψ necessarily implies that α and β are similar.

Ohlebusch and Ukkonen [18] demonstrate that Conjecture 1 holds true for a variety of rich classes of E-pattern languages. In general, however, the conjecture is disproved by Reidenbach [21] using complex counter example patterns. These patterns are valid for alphabet sizes 3 and 4 only, and their particular construction seems not to be extendable to larger alphabets. Concerning finite alphabets Σ with $|\Sigma| \geq 5$, our result in Theorem 6 does not directly contradict Conjecture 1, since our patterns α, β do not generate identical languages. Thus, there is still a chance that the conjecture is correct for alphabet sizes greater than or equal to 5. Nevertheless, as the considerations by Ohlebusch and Ukkonen [18] are based on a specific expectation concerning the inclusion of similar E-pattern languages which Theorem 6 demonstrates to be incorrect, it seems that the insights given in the present section disprove the very foundations of their approach to the equivalence problem for the full class of E-pattern languages. Therefore we feel that the only remaining evidence that still supports Conjecture 1 for $|\Sigma| \geq 5$ is the lack of known counter-examples.

Furthermore, our result definitely affects the use of the sophisticated proof technique introduced by Filè [5] and Jiang et al. [10] for the proof of Theorem 5. For *terminal-free* patterns α, β and any alphabet Σ with $|\Sigma| \geq 2$, this technique constructs a particular substitution τ_β such that $\tau_\beta(\alpha) \in L_{\mathrm{E},\Sigma}(\beta)$ if *and only if* there is a morphism mapping β onto α. After considerable effort made by Dányi and Fülöp [4], Ohlebusch and Ukkonen [18] and Reidenbach [21] to extend this approach to *general* similar patterns, Theorem 6 demonstrates that, for certain pairs of such patterns, such a substitution τ_β does not exist, since, for every finite alphabet Σ, there are similar patterns α, β such that $L_{\mathrm{E},\Sigma}(\beta)$ contains *all* words in $L_{\mathrm{E},\Sigma}(\alpha)$, although there is no terminal-preserving morphism mapping β onto α. Consequently, Theorem 6 shows that the main tool for tackling the inclusion problem for terminal-free E-pattern languages – namely our profound knowledge on the properties of the abovementioned substitution τ_β – necessarily fails if we want to extend it to arbitrary similar patterns, and therefore it seems that the research on the inclusion problem for similar E-pattern languages (and, thus, the equivalence problem for general E-pattern languages) needs to start virtually from scratch again.

Acknowledgements

The authors wish to thank the referees of the conference version [6] of this paper for their valuable comments.

References

[1] Angluin, D., 1980. Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62.

[2] Angluin, D., 1980. Inductive inference of formal languages from positive data. Information and Control 45, 117–135.

[3] Câmpeanu, C., Salomaa, K., Yu, S., 2003. A formal study of practical regular expressions. International Journal of Foundations of Computer Science 14, 1007–1018.

[4] Dányi, G., Fülöp, Z., 1996. A note on the equivalence problem of E-patterns. Information Processing Letters 57, 125–128.

[5] Filè, G., 1988. The relation of two patterns with comparable language. In: Proc. 5th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1988. Vol. 294 of Lecture Notes in Computer Science. pp. 184–192.

[6] Freydenberger, D., Reidenbach, D., 2008. Bad news on decision problems for patterns. In: Proc. 12th International Conference on Developments in Language Theory, DLT 2008. Vol. 5257 of Lecture Notes in Computer Science.

[7] Freydenberger, D., Reidenbach, D., Schneider, J., 2006. Unambiguous morphic images of strings. International Journal of Foundations of Computer Science 17, 601–628.

[8] Ibarra, O., 1978. Reversal-bounded multicounter machines and their decision problems. Journal of the ACM 25, 116–133.

[9] Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., Yu, S., 1994. Pattern languages with and without erasing. International Journal of Computer Mathematics 50, 147–163.

[10] Jiang, T., Salomaa, A., Salomaa, K., Yu, S., 1995. Decision problems for patterns. Journal of Computer and System Sciences 50, 53–63.

[11] Karhumäki, J., Mignosi, F., Plandowski, W., 2000. The expressibility of languages and relations by word equations. Journal of the ACM 47, 483–505.

[12] Luo, W., 2005. Compute inclusion depth of a pattern. In: Proc. 18th Annual Conference on Learning Theory, COLT 2005. Vol. 3559 of Lecture Notes in Artificial Intelligence. pp. 689–690.

[13] Mateescu, A., Salomaa, A., 1994. Finite degrees of ambiguity in pattern languages. RAIRO Informatique théoretique et Applications 28, 233–253.

[14] Mateescu, A., Salomaa, A., 1997. Patterns. In: Rozenberg, G., Salomaa, A. (Eds.), Handbook of Formal Languages. Vol. 1. Springer, Ch. 4.6, pp. 230–242.

[15] Minsky, M., 1961. Recursive unsolvability of Post's problem of "Tag" and other topics in theory of Turing machines. Annals of Mathematics 74, 437–455.

[16] Mukouchi, Y., 1991. Characterization of pattern languages. In: Proc. 2nd International Workshop on Algorithmic Learning Theory, ALT 1991. pp. 93–104.

[17] Ng, Y., Shinohara, T., 2008. Developments from enquiries into the learnability of the pattern languages from positive data. Theoretical Computer Science 397, 150–165.

[18] Ohlebusch, E., Ukkonen, E., 1997. On the equivalence problem for E-pattern languages. Theoretical Computer Science 186, 231–248.

[19] Reidenbach, D., 2006. The Ambiguity of Morphisms in Free Monoids and its Impact on Algorithmic Properties of Pattern Languages. Logos Verlag, Berlin.

[20] Reidenbach, D., 2006. A non-learnable class of E-pattern languages. Theoretical Computer Science 350, 91–102.

[21] Reidenbach, D., 2007. An examination of Ohlebusch and Ukkonen's conjecture on the equivalence problem for E-pattern languages. Journal of Automata, Languages and Combinatorics 12, 407–426.

[22] Reidenbach, D., 2008. Discontinuities in pattern inference. Theoretical Computer Science 397, 166–193.

[23] Rozenberg, G., Salomaa, A., 1997. Handbook of Formal Languages. Vol. 1. Springer, Berlin.

[24] Salomaa, K., 2004. Patterns. In: Martin-Vide, C., Mitrana, V., Păun, G. (Eds.), Formal Languages and Applications. No. 148 in Studies in Fuzziness and Soft Computing. Springer, pp. 367–379.

[25] Salomaa, K., 2006. Patterns. Lecture, 5th PhD School in Formal Languages and Applications, URV Tarragona.

[26] Shinohara, T., 1982. Polynomial time inference of extended regular pattern languages. In: Proc. RIMS Symposia on Software Science and Engineering. Vol. 147 of Lecture Notes in Computer Science. pp. 115–127.