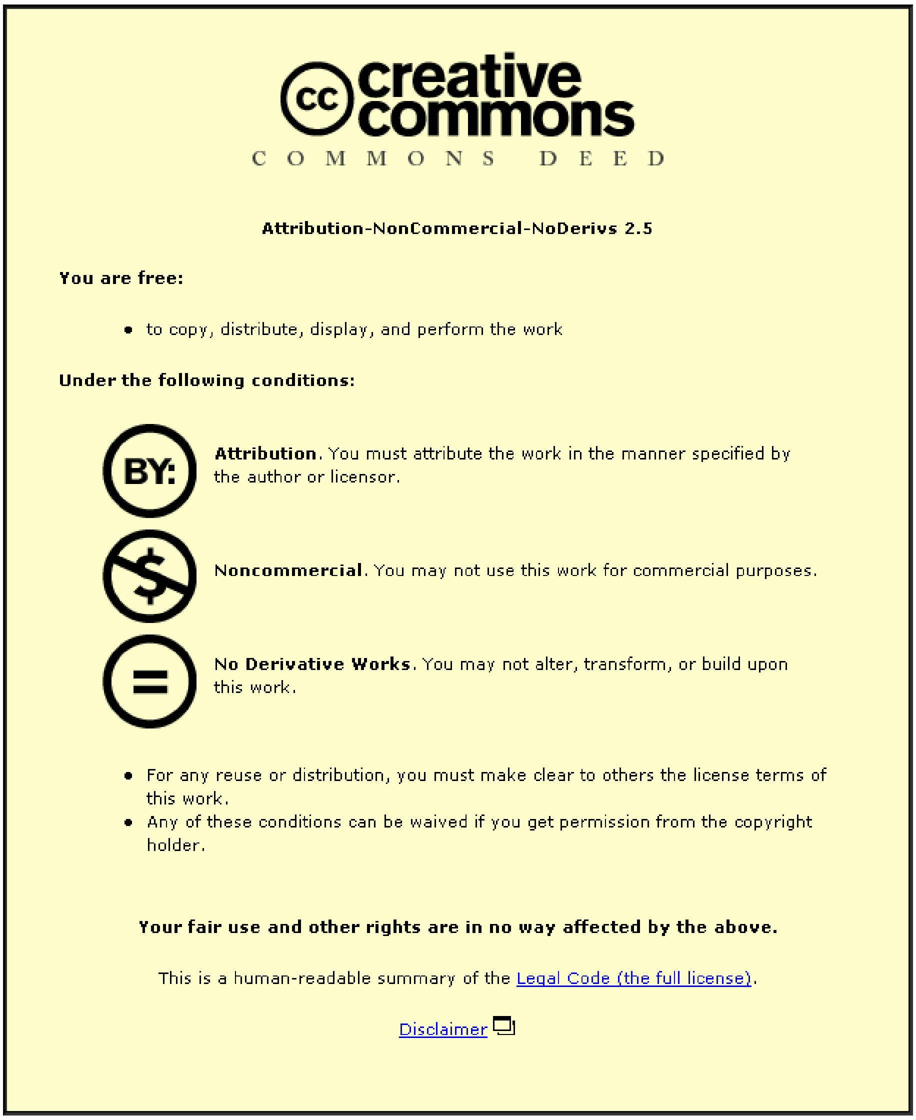

Bad News on Decision Problems for Patterns

Dominik D. Freydenberger*[1] and Daniel Reidenbach[2]

[1] Institut für Informatik, Goethe-Universität, Postfach 111932,
D-60054 Frankfurt am Main, Germany
freydenberger@em.uni-frankfurt.de

[2] Department of Computer Science, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, United Kingdom
D.Reidenbach@lboro.ac.uk

Abstract. We study the inclusion problem for pattern languages, which is shown to be undecidable by Jiang et al. (J. Comput. System Sci. 50, 1995). 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 $\alpha := x_1\,\mathtt{a}\,x_2\,\mathtt{b}\,x_1$ (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 which 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 $w_1 := \mathtt{a\,a\,b\,b\,b\,a}$, $w_2 := \mathtt{a\,b\,a\,b\,a\,b\,a\,b}$ and $w_3 := \mathtt{a\,a\,a\,b\,a\,a}$, and it does not cover the words $w_4 := \mathtt{b\,a}$, $w_5 := \mathtt{b\,a\,b\,b\,b\,a}$ and $w_6 := \mathtt{a\,b\,b\,a}$. It is a well-known fact that pattern languages in general are not context-free.

* Corresponding author.

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 that variables are substituted with the empty word (hence, "NE" is short for "nonerasing"). The latter kind of pattern languages additionally consider those substitutions which map one ore more variables onto the empty word (so "E" stands for "erasing" or "extended"); this definition goes back to Shinohara [25]. 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. [8]), word equations (cf. Mateescu, Salomaa [12], Karhumäki et al. [10]) and equality sets (and, thus, the Post Correspondence Problem, cf. Reidenbach [18]) as well as emerging ones such as extended regular expressions (cf. Câmpeanu et al. [3]) and the ambiguity of morphisms (cf. Freydenberger et al. [6], Reidenbach [18]). 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 [13] and Salomaa [23].

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 [9] is considered to be one of the most notable achievements, and this is mainly due to the very hard proof, which answers a long-standing 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 [16]. 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 classical 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 [15], Reidenbach [19, 21], Luo [11]).

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. [9] 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, slightly more formally, Jiang et al. [9] 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. [9] 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 [21]). The second main part of our paper addresses the other major topic in [9]: we discuss the extensibility of a positive decidability result given in [9] 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, Ukkonen [17]) as it is of major importance for 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

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 $h : A^* \to B^*$ with $h(vw) = h(v)h(w)$ 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 $\mathrm{term}(\alpha)$. 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 all $a \in \Sigma$. 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 the set of all $w \in \Sigma^*$ such that $\sigma(\alpha) = w$ for some substitution σ; the *NE-pattern language* $L_{\mathrm{NE},\Sigma}(\alpha)$ is defined in the same way, but restricted to nonerasing substitutions. 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.

A *nondeterministic 2-counter automaton without input* (cf. Ibarra [7]) 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 $(p, m_1, m_2) \vdash_{\mathcal{A}} (q, n_1, n_2)$ iff 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 $Q = \{q_0, \dots, q_s\}$ for some $s \in \mathbb{N}_0$ and define a function $\mathrm{cod} : Q \times \mathbb{N}_0 \times \mathbb{N}_0 \to \{0, \#\}^*$ by $\mathrm{cod}(q_i, m_1, m_2) := 0^{i+1}\#0^{m_1+1}\#0^{m_2+1}$ and extend this to an encoding of computations by defining $\mathrm{cod}(C_1, C_2, \dots, C_n) := \#\# \,\mathrm{cod}(C_1)\, \#\# \,\mathrm{cod}(C_2)\, \#\# \dots \#\# \,\mathrm{cod}(C_n) \#\#$ for every $n \geq 1$ and every sequence $C_1, \dots, C_n \in Q \times \mathbb{N}_0 \times \mathbb{N}_0$. Furthermore, let $\mathrm{VALC}(\mathcal{A}) := \{\mathrm{cod}(C_1, \dots, C_n) \mid C_1, \dots, C_n$ 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 [14], Ibarra [7]), 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. Jiang, Salomaa, Salomaa and Yu [9] prove the undecidability of the *general inclusion problem for E-pattern languages*:

Theorem 1 (Jiang et al. [9]). *There is no total computable function χ_E which, 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 $L_{\mathrm{E},\Sigma}(\alpha_\mathcal{A}) \subseteq L_{\mathrm{E},\Sigma}(\beta_\mathcal{A})$ iff $\mathcal{A}$ 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. As limiting the number of states would lead to a finite number of possible automata (and thus trivial and inapplicable decidability), one cannot simply fix the number of states in order to adapt this result to the inclusion problem for ePAT_Σ with some fixed alphabet Σ. Thus, as mentioned by Reidenbach [18] and Salomaa [24], there seems to be no straightforward way from this undecidability result 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 lengthy and can be found in Section 3.1. It is in principle based on the construction by Jiang et al.[9], 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 $|\Sigma| \geq 2$. Given two patterns $\alpha \in \mathrm{Pat}_\Sigma$ and $\beta \in (\{a\} \cup X)^*$ for some terminal $a \in \Sigma$, it is in general undecidable whether $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$.*

This corollary is the alphabet specific version of Jiang et al.'s Corollary 5.1 in [9] that is used to obtain the following result on the *general inclusion problem for NE-pattern languages*:

Theorem 3 (Jiang et al. [9]). *There is no total computable function χ_{NE} which, 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 the terminology used in the present paper, the proof of Theorem 3 in [9] 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. Using the same reasoning as Jiang et al. in their proof of Theorem 3, but when substituting their Corollary 5.1 with Corollary 1 above, one immediately achieves 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, the authors 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 now take 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.*

The proof for Proposition 1 is omitted due to space constraints.

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 iff 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

Due to space constraints, the proofs of all lemmas in this section are omitted. 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 $Q := \{q_0, \dots, q_s\}$ for some $s \in \mathbb{N}_0$. Our goal is to construct patterns $\alpha_{\mathcal{A}}, \beta_{\mathcal{A}} \in \mathrm{Pat}_\Sigma$ such that $L_{\mathrm{E},\Sigma}(\alpha_{\mathcal{A}}) \subseteq L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$ iff $\mathrm{VALC}(\mathcal{A}) = \emptyset$. We define $\alpha_{\mathcal{A}} := vv\#^4 vxvyv\#^4 vuv$, 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 $\beta_{\mathcal{A}} := (x_1)^2 \dots (x_\mu)^2 \#^4 \hat{\beta}_1 \dots \hat{\beta}_\mu \#^4 \ddot{\beta}_1 \dots \ddot{\beta}_\mu$, with, for all $i \in \{1, \dots, \mu\}$, $\hat{\beta}_i := x_i\, \gamma_i\, x_i\, \delta_i\, x_i$ and $\ddot{\beta}_i := x_i\, \eta_i\, x_i$, where $x_1, \dots, x_\mu$ 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 $i \in \{1, \dots, \mu\}$,
2. $\mathrm{var}(\gamma_i\delta_i\eta_i) \cap \mathrm{var}(\gamma_j\delta_j\eta_j) = \emptyset$ for all $i, j \in \{1, \dots, \mu\}$ with $i \neq j$,
3. $x_k \notin \mathrm{var}(\gamma_i\delta_i\eta_i)$ for all $i, k \in \{1, \dots, \mu\}$.

Thus, for every i, the elements of $\mathrm{var}(\gamma_i\delta_i\eta_i)$ 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. [10]), this means that σ satisfies π_i iff 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 $\mathrm{INVALC}(\mathcal{A})$. The encoding of $\mathrm{INVALC}(\mathcal{A})$ shall be handled by π_4 to π_μ, as each of these predicates describes a sufficient criterium for membership in $\mathrm{INVALC}(\mathcal{A})$. 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 $\gamma_1 := y_{1,1}(\hat{z}_1)^3 y_{1,2}$, $\delta_1 := \hat{y}_1$, $\gamma_2 := y_2$, and $\delta_2 := \hat{y}_{2,1}\, \hat{z}_2\, \hat{y}_{2,2}$, 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 iff σ satisfies π_1 or π_2.*

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 $\sigma \in H$, $\sigma(\alpha_{\mathcal{A}}) \in L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}})$ iff σ satisfies one of the predicates π_1 to π_μ.*

Thus, we can select predicates π_1 to π_μ in such a way that $L_{\mathrm{E},\Sigma}(\alpha_{\mathcal{A}}) \setminus L_{\mathrm{E},\Sigma}(\beta_{\mathcal{A}}) = \emptyset$ iff $\mathrm{VALC}(\mathcal{A}) = \emptyset$ 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 $(3 \leq i \leq \mu)$ s.t. $\tau(x_i) = 0\#^3 0$. 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 $\gamma_3 := y_{3,1}\,\hat{y}_{3,1}\,y_{3,2}\,\hat{y}_{3,2}\,y_{3,3}\,\hat{y}_{3,3}\,y_{3,4}$, and $\delta_3 := \hat{y}_{3,1}\,\hat{y}_{3,2}\,\hat{y}_{3,3}$, 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 iff $\sigma(x) \in L_1\, R(\alpha)\, L_2$.*

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 iff $\sigma(x) \in \Sigma^* R(\#\hat{x}_1\#\hat{x}_2 0\#\hat{x}_1)$, we can also write that σ satisfies π iff $\sigma(x)$ 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.*

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 cod, all elements of $\mathrm{VALC}(\mathcal{A})$ are of good structure, thus being of bad structure is a sufficient criterion for belonging to $\mathrm{INVALC}(\mathcal{A})$. 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:

$\pi_4 : \sigma(x) = \lambda$,

$\pi_5 : \sigma(x) = \#$,

$\pi_6 : \sigma(x) = \#\#$,

$\pi_7 : \sigma(x)$ begins with 0,

$\pi_8 : \sigma(x)$ begins with $\#0$,

$\pi_9 : \sigma(x)$ ends on 0,

$\pi_{10} : \sigma(x)$ ends on $0\#$,

$\pi_{11} : \sigma(x)$ contains a factor $\#\#0^*\#\#$,

$\pi_{12} : \sigma(x)$ contains a factor $\#\#0^*\#0^*\#\#$,

$\pi_{13} : \sigma(x)$ 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 iff there exists a substitution $\sigma \in H$ with $\sigma(x) = w$ such that σ satisfies none of the predicates π_1 to π_{13}.*

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 cod, ## 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 $\mathrm{cod}(C_1, \ldots, C_n) = \sigma(x)$.

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 $\mathcal{A}$). As this construction is rather lengthy, but similar to the approach of Jiang et al. [9], we abstain from giving a detailed description of the predicates π_{15} to π_μ.

Now, if there is a substitution σ that does not satisfy any of π_1 to π_μ, then $\sigma(x) = \mathrm{cod}(C_1, \ldots, C_n)$ 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 $\sigma(x) := \mathrm{cod}(C_1, \ldots, C_n)$, 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 automata without input has an accepting computation. As this problem is known to be undecidable, the inclusion problem for ePAT_Σ is also undecidable.

The proof for larger (finite) alphabets requires only little changes to the way the patterns $\alpha_{\mathcal{A}}$ and $\beta_{\mathcal{A}}$ are derived from a given automaton $\mathcal{A}$. Thus, we omit this part of the proof.

This concludes the proof of Theorem 2.

4 The inclusion of similar E-pattern languages

It can be easily observed that the patterns used for establishing the undecidability of the inclusion problem for E-pattern languages are not *similar* (cf. Section 2). Hence, our reasoning in Section 3.1 does not answer the question of whether the inclusion problem is undecidable for these natural subclasses. In this regard, Jiang et al. [9] demonstrate that for the full class of the *simplest* similar E-pattern languages, namely those generated by *terminal-free* patterns, inclusion is decidable. This insight directly results from the following characterization:

Theorem 5 (Jiang et al. [9]). *Let Σ be an alphabet, $|\Sigma| \geq 2$, and let $\alpha, \beta \in \mathrm{Pat}_{\mathrm{tf}}$ be terminal-free patterns. Then $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ iff 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, Fülöp [4], Ohlebusch, Ukkonen [17]) that the inclusion of 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, surprisingly, 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 $\phi(\beta) = \alpha$.*

Due to space constraints, we do not present a proof for Theorem 6, but we merely give appropriate example patterns for $\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, i. e. $|\Sigma| = 5$:

$$\alpha = x_1\,\mathtt{a}\,x_2\,\mathtt{a}\,x_3\,\mathtt{b}\,x_2\,\mathtt{b}\,x_5\,\mathtt{c}\,x_2\,\mathtt{c}\,x_7\,\mathtt{d}\,x_2\,\mathtt{d}\,x_9\,\mathtt{e}\,x_2\,\mathtt{e}\,x_{11},$$
$$\beta = x_1\,\mathtt{a}\,x_2 x_4\,\mathtt{a}\,x_3\,\mathtt{b}\,x_4 x_6\,\mathtt{b}\,x_5\,\mathtt{c}\,x_6 x_8\,\mathtt{c}\,x_7\,\mathtt{d}\,x_8 x_{10}\,\mathtt{d}\,x_9\,\mathtt{e}\,x_{10} x_2\,\mathtt{e}\,x_{11}.$$

The relevance of Theorem 6 for the research on the equivalence problem for E-pattern languages follows from a result by Jiang et al. [8] 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. [8]). *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 [17] 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, Ukkonen [17]). Let Σ be an alphabet, $|\Sigma| \geq 3$, and let $\alpha, \beta \in \mathrm{Pat}_\Sigma$. Then $L_{\mathrm{E},\Sigma}(\alpha) = L_{\mathrm{E},\Sigma}(\beta)$ iff there exist terminal-preserving morphisms $\phi, \psi : (\Sigma \cup X)^* \rightarrow (\Sigma \cup X)^*$ 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 [17] 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 [20] using very 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 [17] 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. [9] 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 [17] and Reidenbach [20] to extend this approach to *general* similar patterns, Theorem 6 demonstrates that such a substitution τ_β does not exist for every pair of such patterns, 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.

References

1. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
2. D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
3. C. Câmpeanu, K. Salomaa, and S. Yu. A formal study of practical regular expressions. *Int. J. Found. Comput. Sci.*, 14:1007–1018, 2003.

4. G. Dányi and Z. Fülöp. A note on the equivalence problem of E-patterns. *Information Processing Letters*, 57:125–128, 1996.
5. G. Filè. The relation of two patterns with comparable language. In *Proc. STACS 1988*, LNCS 294, pages 184–192, 1988.
6. D.D. Freydenberger, D. Reidenbach, and J.C. Schneider. Unambiguous morphic images of strings. *Int. J. Found. Comput. Sci.*, 17:601–628, 2006.
7. O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25:116–133, 1978.
8. T. Jiang, E. Kinber, A. Salomaa, K. Salomaa, and S. Yu. Pattern languages with and without erasing. *Int. J. Comput. Math.*, 50:147–163, 1994.
9. T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Decision problems for patterns. *Journal of Computer and System Sciences*, 50:53–63, 1995.
10. J. Karhumäki, F. Mignosi, and W. Plandowski. The expressibility of languages and relations by word equations. *Journal of the ACM*, 47:483–505, 2000.
11. W. Luo. Compute inclusion depth of a pattern. In *Proc. COLT 2005*, LNAI 3559, pages 689–690, 2005.
12. A. Mateescu and A. Salomaa. Finite degrees of ambiguity in pattern languages. *RAIRO Informatique théoretique et Applications*, 28:233–253, 1994.
13. A. Mateescu and A. Salomaa. Patterns. In [22], pages 230–242. 1997.
14. Marvin Minsky. Recursive unsolvability of Post's problem of "Tag" and other topics in the theory of turing machines. *Ann. of Math.*, 74:437–455, 1961.
15. Y. Mukouchi. Characterization of pattern languages. In *Proc. 2nd International Workshop on Algorithmic Learning Theory, ALT 1991*, pages 93–104, 1991.
16. Y.K. Ng and T. Shinohara. Developments from enquiries into the learnability of the pattern languages from positive data. *Theor. Comp. Sci.*, 397:150–165, 2008.
17. E. Ohlebusch and E. Ukkonen. On the equivalence problem for E-pattern languages. *Theor. Comput. Sci.*, 186:231–248, 1997.
18. D. Reidenbach. *The Ambiguity of Morphisms in Free Monoids and its Impact on Algorithmic Properties of Pattern Languages.* Logos Verlag, Berlin, 2006.
19. D. Reidenbach. A non-learnable class of E-pattern languages. *Theor. Comput. Sci.*, 350:91–102, 2006.
20. D. Reidenbach. 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, 2007.
21. D. Reidenbach. Discontinuities in pattern inference. *Theor. Comput. Sci.*, 397:166–193, 2008.
22. G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*, volume 1. Springer, Berlin, 1997.
23. K. Salomaa. Patterns. In C. Martin-Vide, V. Mitrana, and G. Păun, editors, *Formal Languages and Applications*, number 148 in Studies in Fuzziness and Soft Computing, pages 367–379. Springer, 2004.
24. K. Salomaa. Patterns. Lecture, 5th PhD School in Formal Languages and Applications, URV Tarragona, 2006.
25. T. Shinohara. Polynomial time inference of extended regular pattern languages. In *Proc. RIMS Symposia on Software Sci. Eng.*, LNCS 147, pages 115–127, 1982.