Inclusion Problems for Patterns With a Bounded Number of Variables

Joachim Bremer and Dominik D. Freydenberger*

Institut für Informatik, Goethe Universität, Frankfurt am Main, Germany,
bremer@cs.uni-frankfurt.de, freydenberger@em.uni-frankfurt.de

Abstract. We study the inclusion problems for pattern languages that are generated by patterns with a bounded number of variables. This continues the work by Freydenberger and Reidenbach (Information and Computation 208 (2010)) by showing that restricting the inclusion problem to significantly more restricted classes of patterns preserves undecidability, at least for comparatively large bounds. For smaller bounds, we prove the existence of classes of patterns with complicated inclusion relations, and an open inclusion problem, that are related to the Collatz Conjecture. In addition to this, we give the first proof of the undecidability of the inclusion problem for NE-pattern languages that, in contrast to previous proofs, does not rely on the inclusion problem for E-pattern languages, and proves the undecidability of the inclusion problem for NE-pattern languages over binary and ternary alphabets.

1 Introduction

Patterns – finite strings that consist of *variables* and *terminals* – are compact and natural devices for the definition of formal languages. A pattern generates a word by a *substitution* of the variables with arbitrary strings of terminals from a fixed alphabet Σ (where all occurrences of a variable in the pattern must be replaced with the same word), and its language is the set of all words that can be obtained under substitutions. In a more formal manner, the language of a pattern can be understood as the set of all images under *terminal-preserving* morphisms; i.e., morphisms that map variables to terminal strings, and each terminal to itself. For example, the pattern $\alpha = x_1 x_1 \mathbf{a} \mathbf{b} x_2$ (where x_1 and x_2 are variables, and $\mathbf{a}$ and $\mathbf{b}$ are terminals) generates the language of all words that have a prefix that consists of a square, followed by the word $\mathbf{a} \mathbf{b}$.

The study of patterns in strings goes back to Thue [20] and is a central topic of combinatorics on words (cf. the survey by Choffrut and Karhumäki [3]), while the investigation of pattern languages was initiated by Angluin [1]. Angluin’s definition of pattern languages permits only the use of *nonerasing* substitutions (hence, this class of pattern languages is called *NE-pattern languages*). Later, Shinohara [19] introduced *E-pattern languages* (E for ‘erasing’ or ‘extended’), where erasing substitutions are permitted.

* Corresponding author.

This small difference in the definitions leads to immense differences in the properties of these two classes. For example, while the equivalence problem for NE-pattern languages is trivially decidable, the equivalence problem for E-pattern languages is a hard open problem. Although both classes were first introduced in the context of *inductive inference* (which deals with the problem of learning patterns for given sets of strings, for a survey see Ng and Shinohara [15]), they have been widely studied in Formal Language Theory (cf. the surveys by Mitrana [13], Salomaa [18]). Due to their compact definition, patterns or their languages occur in numerous prominent areas of computer science and discrete mathematics, including *unavoidable patterns* (cf. Jiang et al. [8]), *practical regular expressions* (cf. Cămpeanu et al. [2]), or *word equations* and the *positive theory of concatenation* (cf. Choffrut and Karhumäki [3]).

One of the most notable results on pattern languages is the proof of the undecidability of the inclusion problem by Jiang et al. [9], a problem that was open for a long time and is of vital importance for the inductive inference of pattern languages. Unfortunately, this proof heavily depends on the availability of an unbounded number of terminals, which might be considered impractical, as pattern languages are mostly used in settings with fixed (or at least bounded) alphabets. But as shown by Freydenberger and Reidenbach [6], undecidability holds even if the terminal alphabet is bounded. As the proof by Jiang et al. and its modification by Freydenberger and Reidenbach require the number of variables of the involved patterns to be unbounded, we consider it a natural question whether the inclusion problems remain undecidable even if bounds are imposed on the number of variables in the pattern; especially as bounding the number of variables changes the complexity of the membership problem from NP-complete to P (cf. Ibarra et al. [7]). Similar restrictions have been studied in the theory of concatenation (cf. Durnev [4]).

Apart from potential uses in inductive inference or other areas, and the search for an approach that could provide the leverage needed to solve the equivalence problem for E-pattern languages, our main motivation for deeper research into the inclusion problems is the question how strongly patterns and their languages are connected. All known cases of (non-trivial) decidability of the inclusion problem for various classes of patterns rely on the fact that for these classes, inclusion is characterized by the existence of a terminal-preserving morphism mapping one pattern to the other. This is a purely syntactical condition that, although NP-complete (cf. Ehrenfeucht and Rozenberg [5]), can be straightforwardly verified. Finding cases of inclusion that are not covered by this condition, but still decidable, could uncover (or rule out) previously unknown phenomena, and be of immediate use for related areas of research.

Our results can be summarized as follows: We show that the inclusion problems for E- and NE-patterns with a bounded (but large) number of variables are indeed undecidable. For smaller bounds, we prove the existence of classes of patterns with complicated inclusion relations, and an open inclusion problem. Some of these inclusions can simulate iterations of the Collatz function, while others could (in principle) be used to settle an important part of the famous

Collatz Conjecture. In contrast to the aforementioned previous proofs, our proof of the undecidability of the inclusion problem for NE-pattern languages is not obtained through a reduction of the inclusion problem for E-pattern languages. Apart from the technical innovation, this allows to prove the undecidability of the inclusion problem for NE-pattern languages over binary and ternary alphabets, which was left open by Freydenberger and Reidenbach.

2 Preliminaries

2.1 Basic Definitions and Pattern Languages

Let $\mathbb{N}_1 := \{1, 2, 3, \dots\}$ and $\mathbb{N}_0 := \mathbb{N}_1 \cup \{0\}$. The function div denotes the integer division, and mod its remainder. The symbols $\subseteq$, $\subset$, $\supseteq$ and $\supset$ refer to subset, proper subset, superset and proper superset relation, respectively. The symbol $\setminus$ denotes the set difference, and $\emptyset$ the empty set.

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 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).

For any alphabet A , a *language* L (over A) is a set of strings over A , i.e. $L \subseteq A^*$. A language L is *empty* if $L = \emptyset$; otherwise, it is *nonempty*.

The notation $|K|$ stands for the size of a set K or the length of a string K ; the term $|w|_a$ refers to the number of occurrences of the symbol a in the string w . 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 $h : A^* \rightarrow B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$. A morphism $h : A^* \rightarrow B^*$ is said to be *nonerasing* if $h(a) \neq \lambda$ for all $a \in A$. For any string $w \in C^*$, where $C \subseteq A$ and $|w|_a \geq 1$ for every $a \in C$, the morphism $h : A^* \rightarrow B^*$ is called a *renaming (of w)* if $h : C^* \rightarrow B^*$ is injective and $|h(a)| = 1$ for every $a \in C$.

Let Σ be a (finite or infinite) alphabet of so-called *terminals* and X an infinite set of *variables* with $\Sigma \cap X = \emptyset$. We normally assume $\{\mathbf{a}, \mathbf{b}, \dots\} \subseteq \Sigma$ and $\{x_1, x_2, x_3, \dots\} \subseteq X$. A *pattern* is a string over $\Sigma \cup X$, a *terminal-free pattern* is a string over X and a *terminal-string* is a string over Σ . For any pattern α , we refer to the set of variables in α as $\text{var}(\alpha)$. The set of all patterns over $\Sigma \cup X$ is denoted by Pat_Σ ; the set of all terminal-free patterns is denoted by Pat_{tf} . For every $n \geq 0$, let $\text{Pat}_{n, \Sigma}$ denote the set of all patterns over Σ that contain at most n variables; that is, $\text{Pat}_{n, \Sigma} := \{\alpha \in \text{Pat}_\Sigma \mid |\text{var}(\alpha)| \leq n\}$.

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

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

and the *NE-pattern language* $L_{NE,\Sigma}(\alpha)$ of a pattern $\alpha \in \text{Pat}_\Sigma$ is given by

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

If the intended meaning is clear, we write $L(\alpha)$ instead of $L_{E,\Sigma}(\alpha)$ or $L_{NE,\Sigma}(\alpha)$ for any $\alpha \in \text{Pat}_\Sigma$. Furthermore, let ePAT_Σ denote the class of all E-pattern languages over Σ , and nePAT_Σ the class of all NE-pattern languages over Σ . Likewise, we define $\text{ePAT}_{\text{tf},\Sigma}$ as the class of all $L_{E,\Sigma}(\alpha)$ with $\alpha \in \text{Pat}_{\text{tf}}$, and, for any $n \geq 0$, $\text{ePAT}_{n,\Sigma}$ as the class of all $L_{E,\Sigma}(\alpha)$ with $\alpha \in \text{Pat}_{n,\Sigma}$. The classes $\text{nePAT}_{\text{tf},\Sigma}$ and $\text{nePAT}_{n,\Sigma}$ are defined accordingly. Let P_1, P_2 be two classes of patterns, and $\text{PAT}_1, \text{PAT}_2$ be the corresponding classes of pattern languages (either the class of all E-pattern languages or the class of all NE-pattern languages over some alphabet Σ that are generated by patterns from P_1 or P_2). We say that the *inclusion problem for PAT_1 in PAT_2 is decidable* if there exists a total computable function χ such that, for every pair of patterns $\alpha \in P_1$ and $\beta \in P_2$, χ decides on whether or not $L(\alpha) \subseteq L(\beta)$. If no such function exists, this inclusion problem is *undecidable*. If both classes of pattern languages are the same class $\text{PAT}_{*,\Sigma}$, we simply refer to the *inclusion problem of $\text{PAT}_{*,\Sigma}$* .

The concepts contained in the following two Sections are a vital part of our considerations.

2.2 A Universal Turing Machine

Let U be the universal Turing machine $U_{15,2}$ with 2 symbols and 15 states described by Neary and Woods [14]. This machine has the state set $Q = \{q_1, \dots, q_{15}\}$ and operates on the tape alphabet $\Gamma = \{0, 1\}$ (where 0 is the blank symbol). In order to discuss configurations of U , we adopt the following conventions. The tape content of any configuration of U is characterized by the two infinite sequences $t_L = (t_{L,n})_{n \geq 0}$ and $t_R = (t_{R,n})_{n \geq 0}$ over Γ . Here, t_L describes the content of what we shall call the *left side of the tape*, the infinite word that starts at the position of the machine's head and extends to the left. Likewise, t_R describes the *right side of the tape*, the infinite word that starts immediately to the right of the head and extends to the right.

Next, we define the function $e : \Gamma \rightarrow \mathbb{N}_0$ as $e(0) := 0$ and $e(1) := 1$, and extend this to an encoding of infinite sequences $t = (t_n)_{n \geq 0}$ over Γ by $e(t) := \sum_{i=0}^{\infty} 2^i e(t_i)$. As we consider only configurations where all but finitely many cells of the tape consist of the blank symbol 0 (which is encoded as 0), $e(t)$ is always finite and well-defined. Note that for every side t of the tape, $e(t) \bmod 2$ returns the encoding of the symbol that is closest to the head (the symbol under the head for t_L , and the symbol to the right of the head for t_R). Furthermore, each side can be lengthened or shortened by multiplying or dividing (respectively) its encoding $e(t)$ by 2. The encodings enc_E and enc_{NE} of configurations of U are defined by

$$\begin{aligned} \text{enc}_E(q_i, t_L, t_R) &:= 0^{e(t_R)} \# 0^{e(t_L)} \# 0^i, \\ \text{enc}_{NE}(q_i, t_L, t_R) &:= 0^7 0^{e(t_R)} \# 0^7 0^{e(t_L)} \# 0^{i+6}, \end{aligned}$$

for every configuration (q_i, t_L, t_R) . Note that both functions are almost identical; the only difference is that enc_{NE} adds six additional occurrences of 0 to each of the three continuous blocks of 0.

We extend each of these encodings to an encoding of finite sequences of configurations $C = (C_i)_{i=1}^n$ by $\text{enc}(C) := \#\#\text{enc}(C_1)\#\#\dots\#\#\text{enc}(C_n)\#\#$ for $\text{enc} = \text{enc}_{\text{E}}$ or $\text{enc} = \text{enc}_{\text{NE}}$. Let I be any configuration of U . A *valid computation from I* is a finite sequence $C = (C_i)_{i=1}^n$ (with $n \geq 2$) of configurations of U such that $C_1 = I$, C_n is a halting configuration, and C_{i+1} is a valid successor configuration of C_i for every i with $1 \leq i < n$. We adopt the convention that any possible configuration where both tape sides have a finite value under e is a valid successor configuration of a halting configuration. This extended definition of succession does not change the acceptance behavior of U . Finally, let

$$\begin{aligned}\text{VALC}_{\text{E}}(I) &:= \{\text{enc}_{\text{E}}(C) \mid C \text{ is a valid computation from } I\}, \\ \text{VALC}_{\text{NE}}(I) &:= \{\text{enc}_{\text{NE}}(C) \mid C \text{ is a valid computation from } I\}.\end{aligned}$$

Each of the two sets is nonempty if and only if U accepts the input of the initial configuration I , and can thus be used to decide the halting problem of U . As U is universal, there can be no recursive function that, on input I , decides whether $\text{VALC}_{\text{E}}(I)$ is empty or not (the same holds for $\text{VALC}_{\text{NE}}(I)$).

2.3 Collatz Iterations

The Collatz function $\mathcal{C} : \mathbb{N}_1 \rightarrow \mathbb{N}_1$ is defined by $\mathcal{C}(n) := \frac{1}{2}n$ if n is even, and $\mathcal{C}(n) := 3n + 1$ if n is odd. For any $i \geq 0$ and any $n \geq 1$, let $\mathcal{C}^0(n) := n$ and $\mathcal{C}^{i+1}(n) := \mathcal{C}(\mathcal{C}^i(n))$. A number n *leads $\mathcal{C}$ into a cycle* if there are i, j with $1 \leq i < j$ and $\mathcal{C}^i(n) = \mathcal{C}^j(n)$. The cycle is *non-trivial* if $\mathcal{C}^k(n) \neq 1$ for every $k \geq 0$; otherwise, it is the *trivial cycle*.

The Collatz Conjecture states that every natural number leads $\mathcal{C}$ into the trivial cycle 4, 2, 1. Regardless of the considerable effort spent on this problem (see the bibliographies by Lagarias [10, 11]), the conjecture remains unsolved, as the iterated function often behaves rather unpredictably. For this reason, iterations of the Collatz function have been studied in the research of small Turing machines. Margenstern [12] conjectures that every class of Turing machines (as characterized by the number of states and symbols) that contains a machine that is able to simulate the iteration of the Collatz function, also contains a machine that has an undecidable halting problem.

Similar to the definition of $\text{VALC}_{\text{E}}(I)$ and $\text{VALC}_{\text{NE}}(I)$, we encode those iterations of the Collatz function that lead to the number 1 (and thus, to the trivial cycle) in languages over the alphabet $\{0, \#\}$. For every $N \in \mathbb{N}_1$, let

$$\begin{aligned}\text{TRIV}_{\text{E}}(N) &:= \{\#0^{\mathcal{C}^0(N)}\#0^{\mathcal{C}^1(N)}\#\dots\#0^{\mathcal{C}^n(N)}\# \mid n \geq 1, \mathcal{C}^n(N) = 1\}, \\ \text{TRIV}_{\text{NE}}(N) &:= \{\#0^{6+\mathcal{C}^0(N)}\#0^{6+\mathcal{C}^1(N)}\#\dots\#0^{6+\mathcal{C}^n(N)}\# \mid n \geq 1, \mathcal{C}^n(N) = 1\}.\end{aligned}$$

By definition, $\text{TRIV}_{\text{E}}(N)$ (and $\text{TRIV}_{\text{NE}}(N)$) are empty if and only if N does not lead $\mathcal{C}$ into the trivial cycle. As we shall see, our constructions are able to

express an even stronger problem, the question whether there are any numbers that lead $\mathcal{C}$ to a non-trivial cycle. We define NTCC_E as the set of all strings

$$\#0^{\mathcal{C}^0(N)}\#0^{\mathcal{C}^1(N)}\#\dots\#0^{\mathcal{C}^n(N)}\#,$$

where $n, N \geq 1$, $\mathcal{C}^i(N) \neq 1$ for all $i \in \{0, \dots, n\}$, and $\mathcal{C}^j(N) = \mathcal{C}^n(N)$ for some $j < n$. Analogously, NTCC_{NE} is defined to be the set of all strings

$$\#0^{6+\mathcal{C}^0(N)}\#0^{6+\mathcal{C}^1(N)}\#\dots\#0^{6+\mathcal{C}^n(N)}\#,$$

with the same restrictions on n and N . Obviously, both sets are nonempty if and only if there exist non-trivial cycles in the iteration of $\mathcal{C}$. This is one of the two possible cases that would disprove the Collatz Conjecture, the other being the existence of a number N with $\mathcal{C}^i(N) \neq \mathcal{C}^j(N)$ for all $i \neq j$.

3 Main Results

In this section, we study the inclusion problems of various classes of pattern languages generated by patterns with a bounded number of variables.

As shown by Jiang et al. [9], the *general inclusion problem for pattern languages* is undecidable, both in the case of E- and NE-patterns:

Theorem 1 (Jiang et al. [9]). *Let $Z \in \{E, NE\}$. There is no total computable function χ_Z which, for every alphabet Σ and for every pair of patterns $\alpha, \beta \in \text{Pat}_\Sigma$, decides on whether or not $L_{Z,\Sigma}(\alpha) \subseteq L_{Z,\Sigma}(\beta)$.*

The proof for the E-case uses an involved construction that relies heavily on the unboundedness of the terminal alphabet Σ . For the NE-case, Jiang et al. give a complicated reduction of the inclusion problem for ePAT_Σ to the inclusion problem for nePAT_{Σ_2} , where Σ_2 is an alphabet with two additional terminals. As shown by Freydenberger and Reidenbach [6], the inclusion problem remains undecidable for most cases of a fixed terminal alphabet:

Theorem 2 (Freydenberger and Reidenbach [6]). *Let Σ be a finite alphabet. If $|\Sigma| \geq 2$, the inclusion problem of ePAT_Σ is undecidable. If $|\Sigma| \geq 4$, the inclusion problem of nePAT_Σ is undecidable.*

The proof for the E-case consists of a major modification of the construction for the general inclusion problem for E-pattern languages, and relies on the presence of an unbounded number of variables in one of the patterns. The NE-case of the result follows from the same reduction as in the proof of Theorem 1 (thus, the difference in $|\Sigma|$), and also relies on an unbounded number of variables.

As patterns with an arbitrarily large number of variables might seem somewhat artificial for many applications, we consider it natural to bound this number in order to gain decidability of (or at least further insights on) the inclusion of pattern languages. We begin our considerations with an observation from two classical papers on pattern languages:

Theorem 3 (Angluin [1], Jiang et al. [8]). *The inclusion problem for nePAT_Σ in $\text{nePAT}_{1,\Sigma}$ and the inclusion problem for ePAT_Σ in $\text{ePAT}_{1,\Sigma}$ are decidable.*

The proofs for both cases of this theorem rely on the following sufficient condition for inclusion of pattern languages:

Theorem 4 (Jiang et al. [8], Angluin [1]). *Let Σ be an alphabet and $\alpha, \beta \in \text{Pat}_\Sigma$. If there is a terminal-preserving morphism $\phi : (\Sigma \cup X)^* \rightarrow (\Sigma \cup X)^*$ with $\phi(\beta) = \alpha$, then $L_{\text{E},\Sigma}(\alpha) \subseteq L_{\text{E},\Sigma}(\beta)$. If ϕ is also nonerasing, then $L_{\text{NE},\Sigma}(\alpha) \subseteq L_{\text{NE},\Sigma}(\beta)$.*

In fact, the proofs of both parts of Theorem 3 show that, for every alphabet Σ and all patterns $\alpha \in \text{Pat}_\Sigma$, $\beta \in \text{Pat}_{1,\Sigma}$, $L(\alpha) \subseteq L(\beta)$ holds *if and only if* there is a terminal-preserving (and, in the NE-case, nonerasing) morphism ϕ with $\phi(\beta) = \alpha$. As the existence of such a morphism is a decidable property (although in general NP-complete, cf. Ehrenfeucht and Rozenberg [5]), the respective inclusion problems for these classes are decidable.

There are numerous other classes of pattern languages where this condition is not only sufficient, but characteristic; e. g. the terminal-free E-pattern languages (cf. Jiang et al. [9]), some of their generalizations (cf. Ohlebusch and Ukkonen [16]), and pattern languages over infinite alphabets (cf. Freydenberger and Reidenbach [6]). As far as we know, all non-trivial decidability results for pattern languages over non-unary alphabets rely on this property¹. Contrariwise, the existence of patterns where inclusion is not characterized by the existence of an appropriate morphism between them is a necessary condition for an undecidable inclusion problem for this class.

The same phenomenon as in Theorem 3 does not occur if we swap the bounds. For the nonerasing case, this is illustrated by the following example:

Example 1 (Reidenbach [17], Example 3.2). Let $\Sigma = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ with $n \geq 2$, and consider the pattern $\alpha_n := x \mathbf{a}_1 x \mathbf{a}_2 x \dots x \mathbf{a}_n x$, $\beta := xy y z$. Then there is no terminal-preserving morphism ϕ with $\phi(\beta) = \alpha_n$, but every word from $L_{\text{NE},\Sigma}(\alpha_n)$ contains an inner square. Thus, $L_{\text{NE},\Sigma}(\alpha_n) \subseteq L_{\text{NE},\Sigma}(\beta)$. $\diamond$

Thus, regardless of the size of $|\Sigma|$, even the inclusion problem of $\text{nePAT}_{1,\Sigma}$ in $\text{nePAT}_{3,\Sigma}$ is too complex to be characterized by the existence of a nonerasing terminal-preserving morphism between the patterns.

A similar phenomenon can be observed for E-pattern languages:

Proposition 1. *For every finite alphabet Σ with $|\Sigma| \geq 2$, there are patterns $\alpha \in \text{Pat}_{1,\Sigma}$ and $\beta \in \text{Pat}_{2|\Sigma|+2,\Sigma}$ such that $L_{\text{E},\Sigma}(\alpha) \subseteq L_{\text{E},\Sigma}(\beta)$, but there is no terminal-preserving morphism $\phi : (\Sigma \cup X)^* \rightarrow (\Sigma \cup X)^*$ with $\phi(\beta) = \alpha$.*

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

The proof also shows that, if Σ has an odd number of letters, the bound on the number of variables in the second class of patterns can be lowered to

¹ Non-trivial meaning that the involved classes are neither finite, nor restricted in some artificial way that leads to trivial decidability.

$2|\Sigma|$. We do not know whether this lower bound is strict, or if there are patterns $\alpha \in \text{Pat}_{1,\Sigma}$, $\beta \in \text{Pat}_{n,\Sigma}$ with $n < 2|\Sigma|$ such that $L_{E,\Sigma}(\alpha) \subseteq L_{E,\Sigma}(\beta)$, but there is no terminal-preserving morphism mapping β to α .

For $|\Sigma| = 2$, according to Proposition 1, the inclusion of $\text{ePAT}_{1,\Sigma}$ in $\text{ePAT}_{6,\Sigma}$ is not characterized by the existence of such a morphism. As this bound (and the bound on NE-patterns from Example 1) are the lowest known bounds for ‘morphism-free’ inclusion, we want to emphasize the following problem:

Open Problem 1 *Let $|\Sigma| = 2$. Is the inclusion problem of $\text{ePAT}_{1,\Sigma}$ in $\text{ePAT}_{6,\Sigma}$ decidable? Is the inclusion problem of $\text{nePAT}_{1,\Sigma}$ in $\text{nePAT}_{3,\Sigma}$ decidable?*

In principle, both inclusion problems might be undecidable; but comparing these bounds to the ones in the following results, this seems somewhat improbable, and suggests that if these problems are undecidable, the proof would need to be far more complicated than the proofs in the present paper. On the other hand, these classes are promising candidates for classes of pattern languages where the inclusion is decidable, but not characterized by the existence of an appropriate morphism.

As evidenced by our first two main theorems, bounding the number of variables preserves the undecidability of the inclusion problem:

Theorem 5. *Let $|\Sigma| = 2$. The following problems are undecidable:*

1. *The inclusion problem of $\text{ePAT}_{3,\Sigma}$ in $\text{ePAT}_{2854,\Sigma}$,*
2. *the inclusion problem of $\text{ePAT}_{2,\Sigma}$ in $\text{ePAT}_{2860,\Sigma}$.*

Theorem 6. *Let $|\Sigma| = 2$. The following problems are undecidable:*

1. *The inclusion problem of $\text{nePAT}_{3,\Sigma}$ in $\text{nePAT}_{3541,\Sigma}$,*
2. *the inclusion problem of $\text{nePAT}_{2,\Sigma}$ in $\text{nePAT}_{3549,\Sigma}$.*

Note that the cases of all larger (finite) alphabets are handled in Section 4.1. The bounds presented in these two theorems are not optimal. Through additional effort and some encoding tricks, it is possible to reduce each bound on the number of variables in the second pattern by a few hundred variables. As the resulting number would still be far away from the bounds presented in the theorems further down in this section, we felt that these optimizations would only add additional complexity to the proofs, without providing deeper insight, and decided to give only the less optimal bounds present above.

The proofs for both theorems use the same basic approach as the proofs of the E-case in Theorems 1 and 2. We show that, for a given configuration I of U , one can effectively construct patterns α, β in the appropriate classes of patterns such that $L(\alpha) \subseteq L(\beta)$ if and only if U halts after starting in I . As this would decide the halting problem of the universal Turing machine U , the inclusion problems must be undecidable.

For the E-case, we show this using a nontrivial but comparatively straightforward modification of the proof for the E-case of Theorem 2. As this construction is still very complicated, a brief sketch can be found in Section 3.1, while the full construction is omitted due to space constraints.

For the NE-case, we show that a comparable construction can be realized with NE-patterns. This observation is less obvious than it might appear and requires extensive modifications to the E-construction. As previous results on the non-decidability of the inclusion problem for NE-patterns rely on an involved construction from [9], we consider the construction used for our proof of Theorem 6 a significant technical breakthrough; especially as this result (together with its extension following from the modification in Section 4.1) allows us to solve Open Problem 1 in [6], concluding that the inclusion problem for NE-patterns over binary and ternary alphabets is undecidable. Some remarks on the construction are sketched in Section 3.2, while the full construction is omitted.

Although encoding the correct operation of a Turing machine (or any similar device) in patterns requires a considerable amount of variables, the simple structure of iterating the Collatz function $\mathcal{C}$ can be expressed in a more compact form. With far smaller bounds, we are able to obtain the following two results using the same constructions as for the proof of Theorems 5 and 6:

Theorem 7. *Let Σ be a binary alphabet. Every algorithm that decides the inclusion problem of $\text{ePAT}_{2,\Sigma}$ in $\text{ePAT}_{74,\Sigma}$ can be converted into an algorithm that, for every $N \in \mathbb{N}_1$, decides whether N leads $\mathcal{C}$ into the trivial cycle.*

Theorem 8. *Let Σ be a binary alphabet. Every algorithm that decides the inclusion problem of $\text{nePAT}_{2,\Sigma}$ in $\text{nePAT}_{145,\Sigma}$ can be converted into an algorithm that, for every $N \in \mathbb{N}_1$, decides whether N leads $\mathcal{C}$ into the trivial cycle.*

The proofs are sketched in Sections 3.1 and 3.2. As mentioned in Section 2.3, this demonstrates that, even for these far tighter bounds, the inclusion problems are able to express comparatively complicated sets. Moreover, a slight modification of the result allows us to state the following far stronger results:

Theorem 9. *Let Σ be a binary alphabet. Every algorithm that decides the inclusion problem for $\text{ePAT}_{4,\Sigma}$ in $\text{ePAT}_{80,\Sigma}$ can be used to decide whether any number $N \geq 1$ leads $\mathcal{C}$ into a non-trivial cycle.*

Theorem 10. *Let Σ be a binary alphabet. Every algorithm that decides the inclusion problem for $\text{nePAT}_{4,\Sigma}$ in $\text{nePAT}_{153,\Sigma}$ can be used to decide whether any number $N \geq 1$ leads $\mathcal{C}$ into a non-trivial cycle.*

The proofs are sketched in Sections 3.1 and 3.2. These two results need to be interpreted very carefully. Of course, the existence of non-trivial cycles is trivially decidable (by a constant predicate); but these results are stronger than mere decidability, as the patterns are constructed effectively. Thus, deciding the inclusion of any of the two pairs of patterns defined in the proofs would allow us to prove the existence of a counterexample to the Collatz Conjecture, or to rule out the existence of one important class of counterexamples, and thus solve ‘one half’ of the Collatz Conjecture. More pragmatically, we think that these results give reason to suspect that the inclusion problems of these classes of pattern languages are probably not solvable (even if effectively, then not efficiently), and definitely very complicated.

3.1 Sketch of the Construction for E-Patterns

As the construction is rather involved, we only give a basic sketch, and omit the full technical details. In each of the proofs, our goal is to decide the emptiness of a set V , which is one of $\text{TRIV}_E(N)$ (for some $N \geq 1$), NTCC_E , or $\text{VALC}_E(I)$ (for some configuration I). For this, we construct two patterns α and β such that $L_{E,\Sigma}(\alpha) \setminus L_{E,\Sigma}(\beta) \neq \emptyset$ if and only if $V \neq \emptyset$. The pattern α contains two subpatterns α_1 and α_2 , where α_2 is a terminal-free pattern with $\text{var}(\alpha_2) \subseteq \text{var}(\alpha_1) \cup \{y\}$, and y is a variable that occurs exactly once in α_2 , but does not occur in α_1 .

Glossing over details (and ignoring the technical role of α_2), the main goal is to define β in such a way that, for every substitution σ , $\sigma(\alpha) \in L_{E,\Sigma}(\beta)$ if and only if $\sigma(\alpha_1) \in V$. More explicitly, the subpattern α_1 generates a set of possible strings, and β encodes a disjunction of predicates on strings that describe the complement of V through all possible errors. If one of these errors occurs in $\sigma(\alpha_1)$, we can construct a substitution τ with $\tau(\beta) = \sigma(\alpha)$. If $V = \emptyset$, every $\sigma(\alpha)$ belongs to $L_{E,\Sigma}(\beta)$. Otherwise, any element of V can be used to construct a word $\sigma(\alpha) \notin L_{E,\Sigma}(\beta)$. The proof of Theorem 2 in [6] can be interpreted as a special case of this construction, using $\alpha_1 := x$ and $\alpha_2 := y$. Through our modification, we are able to exert more control on the elements of $L_{E,\Sigma}(\alpha_1)$, and use this to define required repetitions, prefixes or suffixes for all $\sigma(\alpha_1)$ with $\sigma(\alpha) \notin L_{E,\Sigma}(\beta)$. The variables in $\text{var}(\alpha_2) \setminus \{y\}$ are even further restricted, and can only be mapped to 0^* .

3.2 Sketch of the Construction for NE-Patterns

The detailed definition of the construction, and the associated proofs, are omitted due to space constraints. Describing the NE-construction on the same level of detail as the E-construction, both appear to be identical, including the presence and the role of subpatterns α_1 and α_2 in α . But as evidenced in the full proof, the peculiarities of NE-patterns require considerable additional technical effort. For example, the E-construction heavily depends on being able to map most variables in β to the empty word; dealing with these ‘superfluous’ variables is the largest difficulty for the modification. In order to overcome this problem, the pattern α contains long terminal-strings, which makes it possible to map every variable in β to at least one terminal. These terminal-strings complicate one of the main proofs for the E-construction, as we have to ensure that these terminal-strings do not prevent a necessary mapping, while not allowing any unintended mappings. The E-construction uses a set of variables x_i of which, under some preconditions, all but one have to be mapped to the empty word. That variable is then used to enforce certain decompositions of β in a way that allows us to encode the predicates in a system of word equations. In the NE-construction, we use a more complicated prefix-construction to obtain a set of variables, which (again under some preconditions) all but one have to be mapped to the terminal 0 , while the single remaining variable has to be mapped to the terminal $\#$. Some minor changes make sure that the number of different

variables in β does not increase too much in comparison to the E-construction – this is one reason for the different definitions of the encoding sets for the erasing and the nonerasing case in Sections 2.2 and 2.3. On the other hand, the modifications of the construction and the use of nonerasing substitutions make the implementation of the extensions in Section 4 simpler than for the erasing case.

4 Extensions of the Main Theorems

In this section, we extend the main theorems of the previous section to larger alphabets (Section 4.1), and show that all patterns from the second class can be replaced with terminal-free patterns (Section 4.2).

4.1 Larger Alphabets

As mentioned in Lemma 5 in [6], the construction for E-patterns can be adapted to all finite alphabets $|\Sigma|$ with $|\Sigma| \geq 3$. This modification is comparatively straightforward, but would require $2(|\Sigma| - 2)$ additional predicates, and increase the number of required variables in β by $|\Sigma| - 2$ for each predicate. With additional effort, both constructions can be adapted to arbitrarily large alphabets:

Theorem 11. *Let Σ be a finite alphabet with $|\Sigma| \geq 3$. The following problems are undecidable:*

1. *The inclusion problem of $\text{ePAT}_{2,\Sigma}$ in $\text{ePAT}_{2882,\Sigma}$,*
2. *the inclusion problem of $\text{nePAT}_{2,\Sigma}$ in $\text{nePAT}_{3563,\Sigma}$.*

The required modifications and the proof of their correctness for the E- and the NE-construction are omitted. Using the same modifications to the constructions, the remaining cases from Theorems 5 and 6 and Theorems 7 to 10 can also be adapted to ternary (or larger) alphabets, using only 22 additional variables.

4.2 Inclusion in $\text{ePAT}_{\text{tf},\Sigma}$ or $\text{nePAT}_{\text{tf},\Sigma}$

Both constructions can also be adapted to use terminal-free patterns β :

Theorem 12. *Let $|\Sigma| = 2$. The following problems are undecidable:*

1. *The inclusion problem of $\text{ePAT}_{2,\Sigma}$ in $\text{ePAT}_{\text{tf},\Sigma}$,*
2. *the inclusion problem of $\text{nePAT}_{2,\Sigma}$ in $\text{nePAT}_{\text{tf},\Sigma}$.*

Again, the technical details are omitted for space reasons. Note that the number of variables in the patterns from Pat_{tf} remains bounded. Although one might expect that this result could be modified to show that the open inclusion problem for $\text{nePAT}_{\text{tf},\Sigma}$ is undecidable, we consider this doubtful, as the modified NE-construction relies heavily on the terminal symbols in α . Furthermore, although it is considerably easier to modify the NE-construction, the fact that the inclusion problem for $\text{ePAT}_{\text{tf},\Sigma}$ is decidable casts further doubt on that expectation. As in Section 4.1, all other results that are based on one of the two constructions can be adapted as well.

References

1. D. Angluin. Finding patterns common to a set of strings. In *Proc. 11th Annual ACM Symposium on Theory of Computing*, pages 130–141, 1979.
2. C. Câmpeanu, K. Salomaa, and S. Yu. A formal study of practical regular expressions. *Int. J. Found. Comput. Sci.*, 14:1007–1018, 2003.
3. C. Choffrut and J. Karhumäki. Combinatorics of words. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 6, pages 329–438. Springer, 1997.
4. V.G. Durnev. Undecidability of the positive $\forall\exists^3$ -theory of a free semigroup. *Siberian Math. J.*, 36(5):917–929, 1995.
5. A. Ehrenfeucht and G. Rozenberg. Finding a homomorphism between two words is NP-complete. *Inform. Process. Lett.*, 9:86–88, 1979.
6. D.D. Freydenberger and D. Reidenbach. Bad news on decision problems for patterns. *Inform. and Comput.*, 208(1):83–96, 2010.
7. O.H. Ibarra, T.C. Pong, and S.M. Sohn. A note on parsing pattern languages. *Pattern Recognit. Lett.*, 16(2):179–182, 1995.
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. *J. Comput. Syst. Sci.*, 50:53–63, 1995.
10. J.C. Lagarias. The $3x+1$ problem: An annotated bibliography (1963–1999), Aug 2009. <http://arxiv.org/abs/math/0309224>.
11. J.C. Lagarias. The $3x+1$ problem: An annotated bibliography, II (2000–2009), Aug 2009. <http://arxiv.org/abs/math/0608208>.
12. M. Margenstern. Frontier between decidability and undecidability: a survey. *Theor. Comput. Sci.*, 231(2):217–251, 2000.
13. V. Mitrana. Patterns and languages: An overview. *Grammars*, 2(2):149–173, 1999.
14. T. Neary and D. Woods. Four small universal Turing machines. *Fundam. Inform.*, 91(1):123–144, 2009.
15. Y.K. Ng and T. Shinohara. Developments from enquiries into the learnability of the pattern languages from positive data. *Theor. Comput. Sci.*, 397:150–165, 2008.
16. E. Ohlebusch and E. Ukkonen. On the equivalence problem for E-pattern languages. *Theor. Comput. Sci.*, 186:231–248, 1997.
17. D. Reidenbach. *The Ambiguity of Morphisms in Free Monoids and its Impact on Algorithmic Properties of Pattern Languages*. PhD thesis, Fachbereich Informatik, Technische Universität Kaiserslautern, 2006. Logos Verlag, Berlin.
18. 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.
19. T. Shinohara. Polynomial time inference of pattern languages and its application. In *Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science*, pages 191–209, 1982.
20. A. Thue. Über unendliche Zeichenreihen. *Kra. Vidensk. Selsk. Skrifter. I Mat. Nat. Kl.*, 7, 1906.