A Note on the Complexity of Matching Patterns with Variables

Markus L. Schmid

Fachbereich 4 – Abteilung Informatik, Universität Trier, D-54296 Trier, Germany

Abstract

A word matches a pattern with variables (i. e., a string that contains terminal symbols and variable symbols) if and only if it can be obtained from the pattern by substituting the variables by terminal words. To decide for a given word whether or not it matches a pattern with variables is an NP-complete problem, which has been independently discovered and investigated in different areas of theoretical computer science and which has applications in various contexts. In this work, we show that the problem of matching patterns with variables remains NP-complete even if every variable has at most two occurrences. In addition to this, we show that if patterns can be represented as special kinds of planar graphs, then they can be matched in polynomial time.

Keywords: Pattern matching with variables, Parameterised pattern matching, NP-complete problems, Membership problem for pattern languages

1. Matching Patterns with Variables

In the context of this paper, a *pattern* (*with variables*) is a string containing *variables* from $\{x_1, x_2, x_3, \ldots\}$ and *terminal symbols* from a finite alphabet Σ, e. g., $\alpha := x_1\,\mathtt{a}\,x_1\,\mathtt{b}\,x_2\,x_2$ is a pattern, where $\mathtt{a}, \mathtt{b} \in \Sigma$. A word w over Σ *matches* a pattern α if and only if w can be derived from α by uniformly substituting the variables in α by terminal words. The problem of matching patterns with variables is then to decide for a given pattern and a given word, whether or not the word matches the pattern. For example, the pattern α from above is matched by the word $u :=$ `bacaabacabbaba`, since substituting x_1 and x_2 in α by `baca` and `ba`, respectively, yields u. On the other hand, α is not matched by the word $v :=$ `cbcabbcbbccbc`, since v cannot be obtained by substituting the variables of α by some words.

Matching patterns with variables is an NP-complete problem, which, to the knowledge of the author, has first been introduced and investigated in 1979 by Angluin [1, 2] and, independently, by Ehrenfeucht and Rozenberg [3]. In Angluin's work, it arises in form of the membership problem for so-called *nonerasing pattern languages*, i. e., the set of all words that can be derived from a specific pattern by substituting the variables by *non-empty* words only. Ehrenfeucht and Rozenberg investigate the more general problem of deciding on the existence of a (possibly erasing) morphism between two given words, which is equivalent to matching patterns without terminal symbols, where variables can also be substituted by the empty word. In both [2] and [3] the NP-completeness of these slightly different versions of the pattern matching problem described above is shown by a reduction from 3SAT.

Since their introduction, Angluin's pattern languages have been intensely studied in the context of inductive inference (see, e. g., Angluin [2], Shinohara [4], Reidenbach [7, 8] and, for a survey, Ng and Shinohara [9]). Furthermore, several language theoretical aspects of pattern languages – such as their expressive power and the decidability of their inclusion and equivalence problems – have been studied (see, e. g., Angluin [2], Jiang et al. [10], Ohlebusch and Ukkonen[11], Freydenberger and Reidenbach [12], Bremer and Freydenberger [13]). However, a detailed investigation of the complexity of their membership problem has been somewhat neglected. Some of the early work that is worth

Email address: `M.Schmid@lboro.ac.uk` (Markus L. Schmid)

mentioning in this regard is by Ibarra et al. [14], who provide a more thorough worst case complexity analysis, and by Shinohara [15], who shows that matching patterns with variables can be done in polynomial time provided that every variable occurs only once in the pattern or the pattern is *non-cross*[1]. Recently, Reidenbach and Schmid [16, 17] presented more complicated parameters of patterns that, if bounded by a constant, allow the matching of patterns to be performed in polynomial time (see also Schmid [18]).

In the pattern matching community, independent from Angluin's work, the problem of matching patterns with variables has been rediscovered by a series of papers (see Baker [19], Amir et al.[20], Amir and Nor [21], Clifford et al. [22]). In [22], the NP-completeness of the problem is shown in a similar way as done by Angluin [2] and Ehrenfeucht and Rozenberg [3], i. e., by a reduction from 3SAT. Furthermore, in [22] it is shown that matching patterns with variables remains NP-complete even if injectivity is required, i. e., different variables must be substituted by different words. In [21], motivations for matching patterns with variables can be found from such diverse areas as software engineering, image searching, DNA analysis, poetry and music analysis, or author validation.

Another motivation for investigating patterns with variables are so-called *regular expressions with backreferences* or *REGEX*, for short (see, e. g., Câmpeanu et al. [23]), since the problem of matching patterns with variables is a special case of the matchtest for these REGEX, which is of considerable practical importance.

In order to improve the practicability of patterns with variables it might be helpful to identify preferably large classes of patterns that can be matched in polynomial time. For example, every class of patterns with bounded scope coincidence degree[2] is such a class (see [17]). On the other hand, in order to narrow down the search space where we have to look for such classes, it is useful to know preferably small classes of patterns for which the matching problem remains NP-complete. For example, from the NP-completeness proof of Ehrenfeucht and Rozenberg [3] it follows that matching patterns with variables is NP-complete even if the terminal alphabet has a cardinality of at most 2.

In this paper we present one result of each of the two above described types. More precisely, we show that the problem of matching patterns with variables remains NP-complete even if every variable has at most two occurrences. Furthermore, we identify classes of patterns that can be represented as a special kind of planar graphs and show that these can be matched in polynomial time.

2. Definitions

Let $\mathbb{N} := \{1, 2, 3, \ldots\}$. For an arbitrary alphabet A, a *word* (*over* A) is a finite sequence of symbols from A, and ε is the *empty word*. The notation A^+ denotes the set of all nonempty words over A, and $A^* := A^+ \cup \{\varepsilon\}$. For the *concatenation* of two words w_1, w_2 we write $w_1 w_2$. We say that a word $v \in A^*$ is a *factor* of a word $w \in A^*$ if there are $u_1, u_2 \in A^*$ such that $w = u_1\, v\, u_2$. The notation $|K|$ stands for the size of a set K or the length of a word K.

Let $X := \{x_1, x_2, x_3, \ldots\}$ and every $x \in X$ is a *variable*. Let Σ be a finite alphabet of *terminals*. Every $\alpha \in (X \cup \Sigma)^+$ is a *pattern* and every $w \in \Sigma^*$ is a (*terminal*) *word*. For any pattern α, we refer to the set of variables in α as $\mathrm{var}(\alpha)$ and, for any variable $x \in \mathrm{var}(\alpha)$, $|\alpha|_x$ denotes the number of occurrences of x in α.

Let α be a pattern. A *substitution* (*for* α) is a mapping $h : \mathrm{var}(\alpha) \to \Sigma^*$. For every $x \in \mathrm{var}(\alpha)$, we say that *$x$ is substituted by* $h(x)$ and $h(\alpha)$ denotes the word obtained by substituting every occurrence of a variable x in α by $h(x)$ and leaving the terminals unchanged. Let $w \in \Sigma^*$. A substitution for α is a *match for α and w* (or simply *match*, if α and w are clear from the context) if $h(\alpha) = w$ is satisfied. For example, $h(x_1) := \mathtt{aba}$ and $h(x_2) := \mathtt{ba}$ is a match for $x_1\, \mathtt{a}\, x_2\, \mathtt{b}\, x_2\, x_1$ and $\mathtt{abaababbaaba}$. We say that a word w *matches* a pattern α if and only if there exists a match for α and w. Next, we define the problem of deciding whether a given word over some alphabet Σ matches a given pattern from some set $P \subseteq (X \cup \Sigma)^+$ of patterns:

VPATMATCH$_\Sigma(P)$

Instance: A pattern $\alpha \in P$ and a word $w \in \Sigma^*$.

Question: Does w match α?

[1]A pattern is non-cross if and only if between any two occurrences of the same variable x no other variable different from x occurs, e. g., the pattern $\mathtt{a}\, x_1\, \mathtt{ba}\, x_1\, x_2\, \mathtt{a}\, x_2\, x_2\, x_3\, x_3\, \mathtt{b}\, x_4$ is non-cross, whereas $x_1\, \mathtt{b}\, x_1\, x_2\, \mathtt{ba}\, x_3\, x_3\, x_4\, x_4\, \mathtt{bc}\, x_2$ is not.

[2]The scope coincidence degree of a pattern is the maximum number of intervals that cover a common position in the pattern, where each interval is given by the leftmost and rightmost occurrence of a variable in the pattern.

The above problem can also be defined without the dependency on the underlying terminal alphabet Σ, which is implicitly given by the pattern and the input word. We refer to this problem by VPATMATCH(P).

As already mentioned before, $\text{VPATMATCH}_\Sigma((X \cup \Sigma)^+)$ is NP-complete even for the special case where $|\Sigma| = 2$. If $|\Sigma| = 1$, on the other hand, then the problem can be easily solved in polynomial time. In Section 3, we show that $\text{VPATMATCH}_\Sigma(P)$ is NP-complete, if P is the set of all patterns that have at most two occurrences per variable and $|\Sigma| = 2$. In Section 4, large classes P of patterns are identified that satisfy that $\text{VPATMATCH}_\Sigma(P)$ is solvable in polynomial time regardless of the cardinality of Σ.

Next, we recall some basic concepts of propositional logic. Every v_i, $i \in \mathbb{N}$, is a *Boolean variable*, for any Boolean variable v, the expression $\overline{v}$ denotes the negation of v and any Boolean variable or its negation is a *literal.* A *clause* is a set of literals and a set of clauses is a *Boolean formula in conjunctive normal form* (*in CNF*, for short). If every clause has a cardinality of 3, then the Boolean formula is in 3*CNF.* A *truth assignment* for a Boolean formula C in CNF is a function that maps the variables of C to either *true* or *false.* If, for every clause c in C, at least one literal in c is assigned *true*, then the assignment is *satisfying.* If, for every clause c in C, *exactly one* literal in c is assigned *true*, then the assignment is *one-satisfying.*

The problems to determine for a given Boolean formula in 3CNF whether it has a satisfying (one-satisfying) truth assignment is denoted by 3SAT (1-IN-3 3SAT, respectively). Moreover, by NF 1-IN-3 3SAT we denote the problem 1-IN-3 3SAT with the additional condition that there is no negated variable in the input formula. We assume the reader to be familiar with the concept of a polynomial reduction from one problem to another.

3. Matching Patterns with at Most Two Occurrences per Variable

Before we are able to state the main result of this section, i. e., matching patterns with variables is NP-complete even for the restricted case where every variable has at most two occurrences, we first have to take a closer look at the problems 1-IN-3 3SAT and NF 1-IN-3 3SAT. Schaefer [26] shows that the former problem is NP-complete and Garey and Johnson [27] mention that this also holds for NF 1-IN-3 3SAT, but a formal proof for this claim is omitted. However, a polynomial reduction from 1-IN-3 3SAT to NF 1-IN-3 3SAT is straightforward: we replace every negated variable $\overline{v}$ in the Boolean formula by a new variable v' and add the clauses $\{v, v', x\}$ and $\{x, x, y\}$, where x and y are new variables with $x \neq y$. Hence, NF 1-IN-3 3SAT is NP-complete, since it is obviously in NP.

In the following, let P_2 be the set of all patterns α with $|\alpha|_x \leq 2$, $x \in \text{var}(\alpha)$. We are now ready to prove the main result of this section:

Lemma 1. *Let* $|\Sigma| = 2$. *There exists a polynomial reduction from* NF 1-IN-3 3SAT *to* $\text{VPATMATCH}_\Sigma(\text{P}_2)$.

Proof. First, we note that the existing reductions from 3SAT to the problem of matching patterns with variables given by Angluin [2], Ehrenfeucht and Rozenberg [3] and Clifford et al. [22] rely on translating every Boolean variable into a variable of the pattern. So the main difficulty that we are facing in the following reduction is that every variable in the pattern cannot occur more than twice and, thus, occurrences of the same Boolean variables must be represented by different variables in the pattern, which then need to be synchronised.

We now define a transformation from instances of NF 1-IN-3 3SAT into patterns in P_2 and words over Σ. Let $C := \{c_1, c_2, \dots, c_n\}$ be an instance of the problem NF 1-IN-3 3SAT that contains exactly the variables $\{v_1, v_2, \dots, v_m\}$ and, for the sake of convenience, we assume the c_i to be tuples, which contain the variables in ascending order with respect to their indices, i. e., a clause $\{v_5, v_1, v_3\}$ is represented as (v_1, v_3, v_5). Throughout the following definitions, for illustrating our constructions, we shall use the example formula

$$\{(v_2, v_3, v_4), (v_1, v_2, v_3), (v_1, v_3, v_4)\}\,.$$

First, we transform C into $C' := \{c'_1, c'_2, \dots, c'_n\}$ by relabelling the variables in C in such a way that the j^{th} occurrence of variable v_i is replaced by $v_{i,j}$. For our example above, this leads to

$$\{(v_{2,1}, v_{3,1}, v_{4,1}), (v_{1,1}, v_{2,2}, v_{3,2}), (v_{1,2}, v_{3,3}, v_{4,2})\}\,.$$

For every i, $1 \leq i \leq m$, let o_i be the number of occurrences of variable v_i in C.

For the sake of convenience, in this proof we assume that $\{x_{i,j} \mid i, j \in \mathbb{N}\} \subseteq X$ and, without loss of generality, we define $\Sigma := \{\mathtt{a}, \mathtt{b}\}$. Next, we transform every c'_i, $1 \leq i \leq n$, into a pattern β_i in the following way. Let $c'_i = (v_{i_1,j_1}, v_{i_2,j_2}, v_{i_3,j_3})$. Then we define $\beta_i := x_{i_1,j_1}\, x_{i_2,j_2}\, x_{i_3,j_3}$. Furthermore, let $\beta := \beta_1\, \mathtt{b}\, \beta_2\, \mathtt{b} \cdots \mathtt{b}\, \beta_n$ and let $u := a_1\, \mathtt{b}\, a_2\, \mathtt{b} \cdots \mathtt{b}\, a_n$, where, for every i, $1 \leq i \leq n$, $a_i = \mathtt{a}$. With respect to our example, this means

$$\beta = x_{2,1}\, x_{3,1}\, x_{4,1}\, \mathtt{b}\, x_{1,1}\, x_{2,2}\, x_{3,2}\, \mathtt{b}\, x_{1,2}\, x_{3,3}\, x_{4,2}\,,$$
$$u = \mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,.$$

Obviously, if there exists a match for u and β, then it substitutes every variable in β by either $\mathtt{a}$ or ε. The general idea of the reduction is that the substitution of a variable by $\mathtt{a}$ means that the corresponding Boolean variable is assigned *true*, whereas the substitution by ε means that it is assigned *false*. Since two variables $x_{i,j}$, $x_{i,j'}$, $j \neq j'$, correspond to the same Boolean variable v_i in C, we have to make sure that they cannot be substituted by different strings. This is the purpose of the remaining part of the construction. For every i, $1 \leq i \leq m$, we define

$$\gamma_i := z_i\, \mathtt{b}\, x_{i,1}\, x_{i,2}\, \cdots\, x_{i,o_i}\, \mathtt{b}\, z'_i\,,$$
$$w_i := \mathtt{b}\mathtt{a}^{o_i}\mathtt{b}\mathtt{b}\,,$$

where, for every i, $1 \leq i \leq m$, the z_i, z'_i are distinct variables different from the variables $x_{i,j}$, $1 \leq i \leq m$,$1 \leq j \leq o_i$. Finally, we define the pattern α and the word w by

$$\alpha := \beta\, \mathtt{a}\mathtt{b}^3\mathtt{a}\, \gamma_1\, \mathtt{a}\mathtt{b}^4\mathtt{a}\, \gamma_2\, \mathtt{a}\mathtt{b}^5\mathtt{a} \cdots \mathtt{a}\mathtt{b}^{m+2}\mathtt{a}\, \gamma_m\,,$$
$$w := u\, \mathtt{a}\mathtt{b}^3\mathtt{a}\, w_1\, \mathtt{a}\mathtt{b}^4\mathtt{a}\, w_2\, \mathtt{a}\mathtt{b}^5\mathtt{a} \cdots\ \mathtt{a}\mathtt{b}^{m+2}\mathtt{a}\, w_m\,.$$

Referring to our example from above, this means that

$$\begin{aligned}\alpha = {} & x_{2,1}\, x_{3,1}\, x_{4,1}\, \mathtt{b}\, x_{1,1}\, x_{2,2}\, x_{3,2}\, \mathtt{b}\, x_{1,2}\, x_{3,3}\, x_{4,2}\\ & \mathtt{a}\mathtt{b}^3\mathtt{a}\, z_1\, \mathtt{b}\, x_{1,1}\, x_{1,2}\, \mathtt{b}\, z'_1\, \mathtt{a}\mathtt{b}^4\mathtt{a}\, z_2\, \mathtt{b}\, x_{2,1}\, x_{2,2}\, \mathtt{b}\, z'_2\\ & \mathtt{a}\mathtt{b}^5\mathtt{a}\, z_3\, \mathtt{b}\, x_{3,1}\, x_{3,2}\, x_{3,3}\, \mathtt{b}\, z'_3\, \mathtt{a}\mathtt{b}^6\mathtt{a}\, z_4\, \mathtt{b}\, x_{4,1}\, x_{4,2}\, \mathtt{b}\, z'_4\,,\\ w = {} & \mathtt{ababa}\, \mathtt{a}\mathtt{b}^3\mathtt{a}\, \mathtt{b}\mathtt{a}^2\mathtt{bb}\, \mathtt{a}\mathtt{b}^4\mathtt{a}\, \mathtt{b}\mathtt{a}^2\mathtt{bb}\, \mathtt{a}\mathtt{b}^5\mathtt{a}\, \mathtt{b}\mathtt{a}^3\mathtt{bb}\\ & \mathtt{a}\mathtt{b}^6\mathtt{a}\, \mathtt{b}\mathtt{a}^2\mathtt{bb}\,,\end{aligned}$$

This concludes the definition of the transformation and the following claim establishes that it is a reduction.

Claim. The word w matches α if and only if there exists a one-satisfying truth assignment for C.

Proof (Claim). We begin with the *only if* direction and assume that there exists a match h for α and w. For every i, $1 \leq i \leq m$, there exists exactly one occurrence of the factor $\mathtt{a}\mathtt{b}^{i+2}\mathtt{a}$ in α and in w, which directly implies that $h(\beta) = u$ and, for every i, $1 \leq i \leq m$, $h(\gamma_i) = w_i$. From $h(\beta) = u$, we can conclude that, for every i, $1 \leq i \leq m$, $h(\beta_i) = \mathtt{a}$ holds, which directly implies that, for every j, $1 \leq j \leq o_i$, $h(x_{i,j}) \in \{\mathtt{a}, \varepsilon\}$.

Now we assume that, for some i, $1 \leq i \leq m$, the word $h(x_{i,1}\, x_{i,2} \cdots x_{i,o_i})$ is not of form $\mathtt{a}^{o_i}$ or ε. This implies that there is a factor $\mathtt{b}\mathtt{a}^k\mathtt{b}$ in $h(\gamma_i)$, where $0 < k < o_i$. This is a contradiction, since $h(\gamma_i) = w_i$ and there is no such factor in $w_i = \mathtt{b}\mathtt{a}^{o_i}\mathtt{bb}$. Consequently, for every i, $1 \leq i \leq m$, either $h(x_{i,j}) = \mathtt{a}$ for every j, $1 \leq j \leq o_i$, or $h(x_{i,j}) = \varepsilon$ for every j, $1 \leq j \leq o_i$.

We can now construct a truth assignment for C in the following way. For every i, $1 \leq i \leq m$, value *true* is assigned to variable v_i in C if $h(x_{i,1}) = \mathtt{a}$ and *false* is assigned if $h(x_{i,1}) = \varepsilon$. Now let c_i be an arbitrary clause of C. If k, $0 \leq k \leq 3$, of the variables in c_i are assigned *true*, then this implies that $h(\beta_i) = \mathtt{a}^k$. Since $h(\beta_i) = \mathtt{a}$ holds, we can conclude that exactly one of the variables in c_i must be assigned *true*, which implies that the assignment for C is one-satisfying. This concludes the proof of the *only if* direction.

Next, we prove the *if* direction of the claim. To this end, we assume that there exists a one-satisfying truth assignment for C. We construct now a match h for α and w. For every i, $1 \leq i \leq m$, if variable v_i is assigned *true*, then, for every j, $1 \leq j \leq o_i$, we define $h(x_{i,j}) := \mathtt{a}$ and if, on the other hand, variable v_i is assigned *false*, then, for every j, $1 \leq j \leq o_i$, we define $h(x_{i,j}) := \varepsilon$. Since in every clause exactly one variable is assigned value *true*, we can conclude that $h(\beta) = u$. It only remains to define, for every i, $1 \leq i \leq m$, $h(z_i)$ and $h(z'_i)$ in such a way that $h(\gamma_i) = w_i$ is satisfied. This is achieved by the following definitions. For every i, $1 \leq i \leq m$, if $h(x_{i,j}) = \mathtt{a}$, $1 \leq j \leq o_i$, then we define $h(z_i) := \varepsilon$ and $h(z'_i) := \mathtt{b}$, and if $h(x_{i,j}) = \varepsilon$, $1 \leq j \leq o_i$, then we define $h(z_i) := \mathtt{b}\mathtt{a}^{o_i}$ and $h(z'_i) := \varepsilon$. These definitions imply that if $h(x_{i,j}) = \mathtt{a}$, $1 \leq j \leq o_i$, then

$$h(\gamma_i) = h(z_i)\, \mathtt{b}\, \mathtt{a}^{o_i}\, \mathtt{b}\, h(z'_i) = \mathtt{b}\, \mathtt{a}^{o_i}\, \mathtt{b}\, \mathtt{b} = w_i$$

and if $h(x_{i,j}) = \varepsilon$, $1 \leq j \leq o_i$, then

$$h(\gamma_i) = h(z_i)\, \mathtt{b}\, \mathtt{b}\, h(z'_i) = \mathtt{b}\, \mathtt{a}^{o_i}\, \mathtt{b}\, \mathtt{b} = w_i\,.$$

Consequently, for every i, $1 \leq i \leq m$, $h(\gamma_i) = w_i$ is satisfied, which implies that $h(\alpha) = w$.

q.e.d. (Claim)

We note that, for every i, $1 \leq i \leq m$, and j, $1 \leq j \leq o_i$, variable $x_{i,j}$ occurs exactly once in β and once in γ_i. All the other variables, i. e., variables z_i, z'_i, $1 \leq i \leq m$, have only one occurrence in α. Thus, $\alpha \in \mathrm{P}_2$. The factors β and u can be constructed in time linear in the size of C and, for every i, $1 \leq i \leq m$, the construction of γ_i and w_i can be carried out in time that is linear in o_i. Since $\sum_{i=1}^{m} o_i = \mathrm{O}(|C|)$, α and w can be constructed in time linear in the size of C. Consequently, the defined transformation is in fact a polynomial reduction from the problems NF 1-IN-3 3SAT to $\mathrm{VPATMATCH}_\Sigma(\mathrm{P}_2)$, which concludes the proof of the lemma. □

Since, for every Σ, $\mathrm{VPATMATCH}_\Sigma(\mathrm{P}_2)$ is in NP, the NP-completeness of NF 1-IN-3 3SAT and Lemma 1 directly imply the following Theorem.

Theorem 2. *Let* $|\Sigma| \geq 2$. $\mathrm{VPATMATCH}_\Sigma(\mathrm{P}_2)$ *is NP-complete.*

It should be pointed out that proving the NP-completeness of $\mathrm{VPATMATCH}(\mathrm{P}_2)$, i. e., the problem of matching patterns with at most two occurrences per variable where the terminal alphabet is *not* fixed, is easier than proving Theorem 2, since in the polynomial reduction of Lemma 1 we can then use an unbounded number of individual terminal symbols as separators between the γ_i and w_i factors. However, the reduction of the proof of Lemma 1 is worth the effort, since it allows us to conclude the NP-completeness of a much more restricted version of the problem of matching patterns with variables; thus, we obtain a stronger result.

Regarding the complexity of matching patterns with variables, we have now identified a borderline with respect to the number of occurrences per variable and the cardinality of the terminal alphabet: the problem is solvable in polynomial time if $|\Sigma| = 1$ or if every variable occurs at most once, but if both, the cardinality of Σ and the maximum number of occurrences per variable is at least 2, then it is already NP-complete.

4. Matching Planar Patterns

We now summarise a specific encoding of patterns by graphs, which has been introduced in [17]:

Definition 3. *Let* $\alpha := y_1\, y_2 \dots y_n$, $y_i \in (\Sigma \cup X)$, $1 \leq i \leq n$, *be a pattern. The* α-graph *is the graph* $\mathcal{G}_\alpha := (V, E)$, *where* $V := \{1, 2, \dots, |\alpha|\}$ *and* $E := S \cup M$, *where* $S := \{\{i, i+1\} \mid 1 \leq i \leq |\alpha| - 1\}$ *and* $M := \{\{p, q\} \mid 1 \leq p < q \leq |\alpha|, y_p \in \mathrm{var}(\alpha), y_p = y_q$ *and* $y_k \neq y_p, p < k < q\}$.

Intuitively speaking, an α-graph is obtained by treating the positions of α as a path and connecting each vertex that corresponds to a variable with the next vertex that corresponds to the same variable. For example, for $x_1\, x_2\, x_1\, x_1\, x_2\, x_2\, x_1$ the α-graph is given by $\mathcal{G}_\alpha := (V, E)$, where $V := \{1, 2, \dots, 7\}$ and $E := S \cup M$, where $S := \{\{1, 2\}, \{2, 3\}, \dots, \{6, 7\}\}$ and $M := \{\{1, 3\}, \{2, 5\}, \{3, 4\}, \{4, 7\}, \{5, 6\}\}$.

In [17], it has been shown that if patterns have α-graphs with bounded treewidth (for the concept of the treewidth see, e. g., Bodlaender [28]), then they can be matched efficiently:

Lemma 4 (Reidenbach and Schmid [17])**.** *Let P be a set of patterns and, for every $\alpha \in P$, let $\mathcal{G}_\alpha$ be the α-graph. If $\{\mathcal{G}_\alpha \mid \alpha \in P\}$ has bounded treewidth, then the problem* $\mathrm{VPATMATCH}_\Sigma(P)$ *can be solved in polynomial time.*

Next, we recall the concept of outerplanar graphs (see Harary [29]). A graph is called *outerplanar* if and only if it can be drawn on the plane in such a way that no two edges cross each other and all vertices lie on the exterior face. The concept of outerplanarity has been generalised by Baker [30] to k-outerplanarity in the following way. The 1-outerplanar graphs are exactly the outerplanar graphs and, for every $k \geq 2$, a graph is k-outerplanar if and only if it can be drawn on the plane and, furthermore, if we remove all vertices on the exterior face and all their adjacent edges, then all remaining components are $(k-1)$-outerplanar.

In the following, for every pattern α, we say that α is k-outerplanar if and only if the α-graph is k-outerplanar. For an arbitrary $k \in \mathbb{N}$, it can be decided in polynomial time whether or not a given pattern is k-outerplanar. This follows from results by Hopcroft and Tarjan [31] and Bienstock and Monma [32].

Proposition 5. *For a given pattern α and a given $k \in \mathbb{N}$, it can be decided in polynomial time whether α is k-outerplanar.*

Proof. For an arbitrary pattern α, the α-graph can be constructed in polynomial time. Then, by applying the algorithm given in Hopcroft and Tarjan [31], we can check in polynomial time whether or not the α-graph is planar. Bienstock and Monma [32] show that for planar graphs, we can compute in polynomial time the smallest k such that the graph is k-outerplanar. □

In Bodlaender [33] it is shown that the treewidth of k-outerplanar graphs is bounded by a function of k:

Theorem 6 (Bodlaender [33]). *If $\mathcal{G}$ is a k-outerplanar graph, then* $\text{tw}(\mathcal{G}) \leq 3^k - 1$.

For every $k \in \mathbb{N}$, let $P_{\text{k-op}}$ be the set of k-outerplanar patterns. By the above results, we can conclude the following:

Corollary 7. *For every constant $k \in \mathbb{N}$, the problem* $\text{VPATMATCH}_\Sigma(P_{\text{k-op}})$ *is decidable in polynomial time.*

Acknowledgements

The author wishes to thank Daniel Reidenbach for valuable remarks and suggestions, which strengthened the results and improved the readability of this paper.

References

[1] D. Angluin, Finding patterns common to a set of strings, in: Proc. 11th Annual ACM Symposium on Theory of Computing, 1979, pp. 130–141.

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

[3] A. Ehrenfeucht, G. Rozenberg, Finding a homomorphism between two words is NP-complete, Information Processing Letters 9 (1979) 86–88.

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

[5] S. Lange, R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, New Generation Computing 8 (1991) 361–370.

[6] P. Rossmanith, T. Zeugmann, Stochastic finite learning of the pattern languages, Machine Learning 44 (2001) 67–91.

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

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

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

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

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

[12] D. Freydenberger, D. Reidenbach, Bad news on decision problems for patterns, Information and Computation 208 (2010) 83–96.

[13] J. Bremer, D. D. Freydenberger, Inclusion problems for patterns with a bounded number of variables, in: Proc. 14th International Conference on Developments in Language Theory, DLT 2010, Vol. 6224 of Lecture Notes in Computer Science, 2010, pp. 100–111.

[14] O. Ibarra, T.-C. Pong, S. Sohn, A note on parsing pattern languages, Pattern Recognition Letters 16 (1995) 179–182.

[15] T. Shinohara, Polynomial time inference of pattern languages and its application, in: Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science, 1982, pp. 191–209.

[16] D. Reidenbach, M. L. Schmid, A polynomial time match test for large classes of extended regular expressions, in: Proc. 15th International Conference on Implementation and Application of Automata, CIAA 2010, Vol. 6482 of Lecture Notes in Computer Science, 2011, pp. 241–250.

[17] D. Reidenbach, M. L. Schmid, Patterns with bounded treewidth, in: Proc. 6th International Conference on Language and Automata Theory and Applications, LATA 2012, Vol. 7183 of Lecture Notes in Computer Science, 2012, pp. 468–479.

[18] M. L. Schmid, On the membership problem for pattern languages and related topics, Ph.D. thesis, Department of Computer Science, Loughborough University (2012).

[19] B. S. Baker, Parameterized pattern matching: Algorithms and applications, Journal of Computer and System Sciences 52 (1996) 28–42.

[20] A. Amir, Y. Aumann, R. Cole, M. Lewenstein, E. Porat, Function matching: Algorithms, applications, and a lower bound, in: Proc. 30th International Colloquium on Automata, Languages and Programming, ICALP 2003, 2003, pp. 929–942.

[21] A. Amir, I. Nor, Generalized function matching, Journal of Discrete Algorithms 5 (2007) 514–523.

[22] R. Clifford, A. W. Harrow, A. Popa, B. Sach, Generalised matching, in: Proc. 16th International Symposium on String Processing and Information Retrieval, SPIRE 2009, Vol. 5721 of Lecture Notes in Computer Science, 2009, pp. 295–301.

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

[24] A. Aho, Algorithms for finding patterns in strings, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity, MIT Press, 1990, pp. 255–300.

[25] J. E. F. Friedl, Mastering Regular Expressions, 3rd Edition, O'Reilly, Sebastopol, CA, 2006.

[26] T. J. Schaefer, The complexity of satisfiability problems, in: Proc. 10th Annual ACM Symposium on Theory of Computing, STOC 1978, ACM, 1978, pp. 216–226.

[27] M. R. Garey, D. S. Johnson, Computers And Intractability, W. H. Freeman and Company, 1979.

[28] H. L. Bodlaender, Treewidth: Characterizations, appli-

cations, and computations, in: Graph-Theoretic Concepts in Computer Science, Vol. 4271 of Lecture Notes in Computer Science, 2006, pp. 1–14.

[29] F. Harary, Graph Theory, Addison-Wesley Publishing Company, Reading, Massachusetts, 1969.

[30] B. S. Baker, Approximation algorithms for np-complete problems on planar graphs, Journal of the ACM 41 (1994) 153–180.

[31] J. Hopcroft, R. Tarjan, Efficient planarity testing, Journal of the ACM 21 (1974) 549–568.

[32] D. Bienstock, C. L. Monma, On the complexity of embedding planar graphs to minimize certain distance measures, Algorithmica 5 (1990) 93–109.

[33] H. Bodlaender, Classes of graphs with bounded treewidth, Tech. Rep. RUU-CS-86-22, Department of Information and Computing Sciences, Utrecht University (1986).