Technical Reports Mathematics/Computer Science
FB IV – Department of Computer Science
University of Trier
54296 Trier, Germany

# Revisiting Shinohara's Algorithm for Computing Descriptive Patterns

Henning Fernau, Florin Manea, Robert Mercaş, Markus L. Schmid

Technical Report No. 14-3 September 2014

# Revisiting Shinohara's Algorithm for Computing Descriptive Patterns

Henning Fernau[1] Florin Manea[2] Robert Mercaş[2]
Markus L. Schmid[1]

[1]Fachbereich IV – Abteilung Informatikwissenschaften, Universität Trier,
D-54286 Trier, Germany, {Fernau,MSchmid}@uni-trier.de

[2]Kiel University, Department of Computer Science,
D-24098 Kiel, Germany, {flm,rgm}@informatik.uni-kiel.de

**Abstract**

A pattern is a string consisting of variables and terminal symbols, and its language is the set of all words that can be obtained by substituting arbitrary words for the variables. The membership problem for pattern languages, i. e., deciding on whether or not a given word is in the pattern language of a given pattern is NP-complete. We show that any parameter of patterns that is an upper bound for the treewidth of appropriate encodings of patterns as relational structures, if restricted, allows the membership problem for pattern languages to be solved in polynomial time. Furthermore, we identify new such parameters.

## 1 Introduction

The class of pattern languages was introduced by Angluin [1] as a formalism to describe similarities of words with respect to their repeating factors. For example, the words $w_1 = \mathtt{abbaabaa}$, $w_2 = \mathtt{baabbabaabba}$ and $w_3 = \mathtt{abaaaba}$ share the common feature of having a prefix that contains an occurrences of $\mathtt{ba}$ that is surrounded by exactly the same factor. These commonalities can be described by the pattern $\alpha = x_1\mathtt{ba}x_1x_2$, where $x_1$ and $x_2$ are variables that stand for arbitrary factors. The pattern language defined by $\alpha$, denoted by $L(\alpha)$, is the set of all words that can be obtained from $\alpha$ by uniformly substituting the occurrences of variables $x_1$ and $x_2$ by some non-empty words. For example, the words $w_1$, $w_2$ and $w_3$ can be obtained from $\alpha$ by the substitutions $(x_1 \mapsto \mathtt{ab}, x_2 \mapsto \mathtt{aa})$, $(x_1 \mapsto \mathtt{baab}, x_2 \mapsto \mathtt{ba})$ and $(x_1 \mapsto \mathtt{a}, x_2 \mapsto \mathtt{aba})$, respectively, which shows that $w_1, w_2, w_3 \in L(\alpha)$. In [19], Shinohara introduces *extended* or *erasing* pattern languages, where variables can be substituted by the empty word. In this work, we are only concerned with classical pattern languages; for more informations about erasing pattern languages, the reader is referred to the survey [13].

Pattern languages are important in the context of learning theory since they constitute a prominent example of a language class that is inferable from positive data. In fact, their introduction – along with Angluin's characterisation of those language classes that are inferable from positive data (see [2]) – brought new life to the field of inductive inference. Nowadays there exists an active research field devoted to specific aspects of the learnability of pattern languages (see, e. g., [12, 14, 18, 20, 22] and, for a survey, [21]).

A useful tool for the inductive inference of pattern languages are so-called *descriptive patterns*, introduced in [1]. A pattern $\alpha$ is descriptive of a finite set $S$ of words if $S \subseteq L(\alpha)$ and there is no pattern $\beta$ that describes $S$ more accurately, i. e., $S \subseteq L(\beta) \subset L(\alpha)$. For example, the pattern $x_1\mathtt{baa}x_2x_3\mathtt{a}$ is descriptive of $S = \{w_1, w_2, w_3\}$, where the $w_i$'s are as defined above, while the pattern $\alpha$ introduced at the beginning of the work is not, since $S \subseteq L(x_1\mathtt{ba}x_1x_2\mathtt{a}) \subset L(\alpha)$.

Independent of its application for inductive inference, the task of computing descriptive patterns constitutes an interesting problem in its own right. For example, a descriptive pattern $\alpha$ may serve as a representation of the structural commonalities of some set $S$ of textual data (e. g., employee files, entries of a bibliographical database, etc.) and in order to check whether a new data element meets this common structure, it is sufficient to check whether it can be generated by $\alpha$. The main obstacle of this application of descriptive patterns is that deciding on whether a given word can be generated by a given pattern, i. e., the membership problem for pattern languages, is $\mathcal{NP}$-complete [1]. Furthermore, it has been shown in [1] that an algorithm computing a descriptive pattern of *maximal size* necessarily solves the membership problem and therefore, assuming $\mathcal{P} \neq \mathcal{NP}$, it cannot have a polynomial running time.

In [20], Shinohara introduces an exponential time algorithm that computes descriptive patterns by performing membership queries, and he also provides subclasses of patterns for which the membership problem can be solved efficiently and for these his algorithm computes descriptive patterns in polynomial time. Note that the concept of descriptiveness can be easily restricted to an arbitrary subclass $\Pi$ of patterns; more precisely, a pattern $\alpha$ is $\Pi$-descriptive if $\alpha \in \Pi$, $S \subseteq L(\alpha)$ and there is no other pattern $\beta \in \Pi$ with $S \subseteq L(\beta) \subset L(\alpha)$.

We unify and further extend both Angluin's insights with respect to the hardness of computing descriptive patterns as well as Shinohara's work on their efficient computation as follows. We show that for every $S$ a $\Pi$-descriptive patterns exist if and only if $x \in \Pi$ and, furthermore, that a modified version of Shinohara's algorithm can be used to compute $\Pi$-descriptive patterns if and only if $\Pi$ is a *Shinohara-class*, i. e., it contains the set $\{x_1x_2 \cdots x_k \mid k \in \mathbb{N}\}$ and, for every $\alpha \in \Pi$, the pattern $\alpha'$ obtained by substituting some length $i$ suffix of $\alpha$ by a sequence of new variables $y_1y_2 \cdots y_i$ is also in $\Pi$. Within the set of Shinohara-classes of patterns, we prove that $\Pi$-descriptive patterns can be computed in polynomial time (by any algorithm) if and only if the question whether $\alpha \in \Pi$ and the membership problem for $\Pi$ can be decided in polynomial time.

We also investigate the consistency problem for classes $\Pi$ of patterns, which is the problem to decide, for two given finite sets $P$ and $N$ of words, whether there exists a pattern $\alpha \in \Pi$, such that $P \subseteq L(\alpha)$ and $L(\alpha) \cap N = \emptyset$. As shall be demonstrated, this problem is much more difficult than the membership problem for pattern languages or the problem of computing descriptive patterns.

## 2 Preliminaries

Let $\mathbb{N} = \{1, 2, \ldots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and $A$ be a finite alphabet of *symbols*. A *word* (over $A$) is any sequence of symbols from $A$. For any word $w$ over $A$, $|w|$ denotes its length and $\varepsilon$ denotes the *empty word*, i.e., $|\varepsilon| = 0$. By $A^+$ we denote the set of all non-empty words over $A$ and $A^* = A^+ \cup \{\varepsilon\}$. For the *concatenation* of two words $w_1, w_2$ we write $w_1 \cdot w_2$ or simply $w_1 w_2$. Let $w \in A^*$ be a word. We say that $v \in A^*$ is a *factor* of $w$ if $w = u_1 v u_2$ for some $u_1, u_2 \in A^*$. If $u_1 = \varepsilon$, or $u_2 = \varepsilon$, then $v$ is a *prefix*, or a *suffix*, respectively, of $w$. For any $b \in A$, by $|w|_b$ we denote the number of occurrences of $b$ in $w$. For each $1 \leq i \leq j \leq |w|$, let $w[i..j] = w[i] \cdots w[j]$, where $w[k]$ is the letter on position $k$ in $w$, for $1 \leq k \leq |w|$.

For any alphabets $A, B$, a *morphism* is a function $h : A^* \to B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$; $h$ is *nonerasing* if and only if, for every $a \in A$, $h(a) \neq \varepsilon$. Let $\Sigma$ be a finite alphabet of so-called *terminal symbols* and $X$ a countably infinite set of *variables* with $\Sigma \cap X = \emptyset$. We normally assume $X = \{x_1, x_2, \ldots\}$. A *pattern* (*over* $\Sigma$) is a non-empty word over $\Sigma \cup X$ and a (*terminal*) *word* is a string over $\Sigma$. We define $\Sigma$-$\textsc{Pat} = (\Sigma \cup X)^+$ and $\textsc{Pat} = \bigcup_{\Sigma} \Sigma$-$\textsc{Pat}$. For any pattern $\alpha$, we refer to the set of variables as $\mathrm{var}(\alpha)$ and to the set of terminal symbols as $\mathrm{term}(\alpha)$. If $\mathrm{term}(\alpha) = \emptyset$, then $\alpha$ is *terminal-free*. We also use $\mathrm{term}(w)$ and $\mathrm{term}(S)$ in order to denote the set of symbols that occur in a word $w \in \Sigma^*$ or in a set $S$ of words. A pattern $\alpha$ is in *canonical form* if and only if, for some $k \in \mathbb{N}$, $\mathrm{var}(\alpha) = \{x_1, \ldots, x_k\}$ and, for every $i$, $1 \leq i \leq k-1$, the leftmost occurrence of $x_i$ is to the left of the leftmost occurrence of $x_{i+1}$. For a pattern $\alpha$, by $\mathrm{cf}(\alpha)$, we denote its canonical form, i.e., $\mathrm{cf}(\alpha)$ is obtained from $\alpha$ by renaming the variables in such a way that a pattern in canonical form is constructed. A morphism $h : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ is called a *substitution* if $h(a) = a$ for every $a \in \Sigma$ and a substitution of form $(\Sigma \cup X)^* \to \Sigma^*$ is a *terminal substitution*. For a pattern $\alpha \in \Sigma$-$\textsc{Pat}$, the *pattern language of* $\alpha$ (*over* $\Sigma$) is defined by $L_\Sigma(\alpha) = \{h(\alpha) \mid h : (\Sigma \cup X)^* \to \Sigma^*$ is a nonerasing terminal substitution$\}$ (we also write $L(\alpha)$ if the alphabet is clear from the context). This particularly means that $L_\Sigma(\alpha)$ is only defined if $\mathrm{term}(\alpha) \subseteq \Sigma$. For any $\Pi \subseteq \textsc{Pat}$, $\{L_\Sigma(\alpha) \mid \alpha \in \Pi, \mathrm{term}(\alpha) \subseteq \Sigma\}$ is the set of $\Pi$-*pattern languages*.

Let $\Pi \subseteq \textsc{Pat}$. The class $\Pi$ is *natural* if, for a given $\alpha$, the question $\mathrm{cf}(\alpha) \in \Pi$ is decidable and the pattern $x_1$ is in $\Pi$. The class $\Pi$ is called *tractable* if the question whether $\mathrm{cf}(\alpha)$ is in $\Pi$ can be decided in polynomial time and the membership problem for $\Pi$-pattern languages can be decided in polynomial time. For any pattern $\alpha$ and for every $i$, $0 \leq i \leq |\alpha|$, we define the *$i$-tail-generalisation of* $\alpha$ by $\mathrm{tg}(\alpha, i) = \mathrm{cf}(\alpha[1..i] \cdot y_1 y_2 \cdots y_{|\alpha|-i})$, where $\{y_1, y_2, \ldots, y_{|\alpha|-i}\} \subseteq (X \setminus$

$\mathrm{var}(\alpha)$) with $|\{y_1, y_2, \ldots, y_{|\alpha|-i}\}| = |\alpha| - i$. The *tail-generalisation of* $\alpha$ is the set $\mathrm{tg}(\alpha) = \{\mathrm{tg}(\alpha, i) \mid 1 \leq i \leq |\alpha|\}$ and this definition is lifted to sets $\Pi$ of patterns by $\mathrm{tg}(\Pi) = \bigcup_{\alpha \in \Pi} \mathrm{tg}(\alpha)$. A natural class $\Pi$ of patterns is a *Shinohara-class* if, for every $k \in \mathbb{N}$, $\Pi$ contains a pattern of length $k$ and $\mathrm{tg}(\Pi) = \Pi$. The following proposition follows immediately from the definition.

**Proposition 1.** *Let $\Pi$ be a Shinohara-class of patterns. Then $\{x_1 x_2 \cdots x_n \mid n \in \mathbb{N}\} \subseteq \Pi$ and, for every $\alpha \in \Pi$, $\mathrm{tg}(\alpha) \subseteq \Pi$.*

As a convention, we shall always assume that classes $\Pi$ of patterns satisfy $\Pi = \{\mathrm{cf}(\alpha) \mid \alpha \in \Pi\}$, i. e., they only contain patterns in canonical form.

The binary relations $\sqsubseteq$ and $\equiv$ on $\textsc{Pat}$, introduced in [1], are defined as follows. For every $\alpha, \beta \in \textsc{Pat}$, $\alpha \sqsubseteq \beta$ if there exists a non-erasing substitution $h$ with $h(\beta) = \alpha$ and $\alpha \equiv \beta$ if there exists a non-erasing *renaming of variables* $h$ with $h(\beta) = \alpha$, i. e., $h$ is a non-erasing substitution with $|h(x)| = 1$, for all $x \in \mathrm{var}(\beta)$, and $h(x) = h(y)$ if and only if $x = y$, for all $x, y \in \mathrm{var}(\beta)$. Alternatively, as can be easily verified, $\alpha \equiv \beta$ if and only if $\mathrm{cf}(\alpha) = \mathrm{cf}(\beta)$.

**Lemma 2** (Angluin [1])**.** *Let $\alpha, \beta \in \textsc{Pat}$. The relation $\sqsubseteq$ is transitive. If $\alpha \sqsubseteq \beta$, then, for every alphabet $\Sigma$ with $\mathrm{term}(\alpha) \subseteq \Sigma$, $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$. We have that $\alpha \equiv \beta$ if and only if $\alpha \sqsubseteq \beta$ and $\beta \sqsubseteq \alpha$. If $|\alpha| = |\beta|$ and $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$ for some alphabet $\Sigma$ with $|\Sigma| \geq 2$, then $\alpha \sqsubseteq \beta$.*

Lemma 2 shows that, for every alphabet $\Sigma$, $\alpha \sqsubseteq \beta$ is a sufficient condition for $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$, but not a necessary one (see [1]). However, for the special case that $|\alpha| = |\beta|$, $\alpha \sqsubseteq \beta$ is characteristic for $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$ if $\Sigma$ contains at least 2 symbols. In particular, since patterns describing the same pattern language must have the same length, this implies that $\alpha \equiv \beta$ if and only if $L_\Sigma(\alpha) = L_\Sigma(\beta)$.

Let $\Pi \subseteq \textsc{Pat}$. The *membership problem for $\Pi$-pattern languages* asks to decide for a given pattern $\alpha \in \Pi$ and a word $w$, whether $w \in L_{\mathrm{symb}(w) \cup \mathrm{term}(\alpha)}(\alpha)$.

**Theorem 3** (Angluin [1])**.** *The membership problem for $\textsc{Pat}$-pattern languages is $\mathcal{NP}$-complete.*

In [1] a stronger result than Theorem 3 is shown, i. e., the membership problem is $\mathcal{NP}$-complete even if the terminal alphabet $\Sigma$ is a fixed binary alphabet. Let $\Sigma$ be an alphabet, $S$ be a finite set of words and $\Pi \subseteq \textsc{Pat}$. A pattern $\alpha$ is $\Sigma$-$\Pi$-*descriptive of* $S$ if $\alpha \in \Sigma$-$\textsc{Pat}$, $\mathrm{cf}(\alpha) \in \Pi$, $S \subseteq L_\Sigma(\alpha)$ and there does not exist a $\beta \in \Pi$ with $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$.[1] In the following, we call a finite set $S$ of words a *sample* (*over* $\Sigma$).

**Example 4** (Angluin [1], Freydenberger and Reidenbach [6])**.** *Let $\Sigma_1 = \{\mathtt{a}, \mathtt{b}\}$, $\Sigma_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, $S_1 = \{\mathtt{aabaa}, \mathtt{babab}, \mathtt{aabab}, \mathtt{babaa}\}$, $S_2 = \{\mathtt{ababa}, \mathtt{ababbababbab}, \mathtt{babab}\}$ and $S_3 = \{\mathtt{aabcaaa}, \mathtt{caacbca}, \mathtt{bbcccbbbc}\}$. The patterns $x\mathtt{aba}y$ and $xxy$ are both $\Sigma_1$-$\textsc{Pat}$-descriptive of $S_1$. The patterns $xyxyx$ and $x\mathtt{ab}y$ are both $\Sigma_1$-$\textsc{Pat}$-descriptive of $S_2$. For every $i \in \mathbb{N}$, let $\textsc{Pat}_{\mathrm{var} \leq i} = \{\alpha \in \textsc{Pat} \mid |\mathrm{var}(\alpha)| \leq i\}$*

[1]Descriptive patterns for *erasing* pattern languages as well as for *infinite* sets have also been investigated (see Jiang et al. [7] and Freydenberger and Reidenbach [6]).

*denote the set of patterns with at most $i$ variables. The pattern $xy\mathtt{c}zx$ is $\Sigma_2$-$\mathrm{PAT}_{\mathrm{var}\leq 3}$-descriptive of $S_3$ and $x$ is $\Sigma_2$-$\mathrm{PAT}_{\mathrm{var}\leq 1}$-descriptive of $S_3$.*

Let $\Sigma$ be an alphabet, $S$ be a sample and $\Gamma$ be the smallest alphabet with $S \subseteq \Gamma^*$. If a pattern $\alpha$ is $\Sigma$-$\Pi$-descriptive of $S$, then $S \subseteq L_\Sigma(\alpha)$ is satisfied, which implies $\Gamma \subseteq \Sigma$ and $\alpha \in \Gamma$-$\mathrm{PAT}$. Hence, if a pattern is $\Sigma$-$\Pi$-descriptive of $S$, then $\Sigma$ is necessarily a superset of $\Gamma$. However, for two different supersets $\Sigma$ and $\Sigma'$ of $\Gamma$, it seems that it makes a difference whether a pattern is $\Sigma$-$\Pi$-descriptive or $\Sigma'$-$\Pi$-descriptive of $S$. Proposition 5 shows that a $\Gamma$-$\Pi$-descriptive pattern is also $\Sigma$-$\Pi$-descriptive for every superset $\Sigma$ of $\Gamma$.

**Proposition 5.** *Let $\Sigma$ be an alphabet, $S \subseteq \Sigma^*$ be a sample, $\Gamma$ be the smallest alphabet with $S \subseteq \Gamma^*$ and $\Pi \subseteq \mathrm{PAT}$. If a pattern is $\Gamma$-$\Pi$-descriptive of $S$, then it is also $\Sigma$-$\Pi$-descriptive of $S$.*

*Proof.* Let $\alpha$ be $\Gamma$-$\Pi$-descriptive of $S$. Then $\mathrm{cf}(\alpha) \in \Pi$ and, since $\Gamma \subseteq \Sigma$, $L_\Gamma(\alpha) \subseteq L_\Sigma(\alpha)$ is implied and therefore $S \subseteq L_\Sigma(\alpha)$ holds. If $\alpha$ is not $\Sigma$-$\Pi$-descriptive of $S$, then there is a $\beta \in \Pi$ with $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$. This implies $\alpha \not\cong \beta$ and $L_\Gamma(\beta) \neq L_\Gamma(\alpha)$. From $\Gamma \subseteq \Sigma$ and $L_\Sigma(\beta) \subset L_\Sigma(\alpha)$, we can conclude $L_\Gamma(\beta) \subseteq L_\Sigma(\alpha)$. Since all words in $L_\Gamma(\beta)$ are defined over $\Gamma$ and are images of $\alpha$, it follows that $L_\Gamma(\beta) \subseteq L_\Gamma(\alpha)$. Thus, $S \subseteq L_\Gamma(\beta) \subset L_\Gamma(\alpha)$, which is a contradiction to the assumption that $\alpha$ is $\Gamma$-$\Pi$-descriptive of $S$. □

As justified by Proposition 5 and the preceding discussion, in the following we are only concerned with $\Sigma$-$\Pi$-descriptive patterns, where $\Sigma$ is the set of symbols that occur in the sample. For the sake of convenience, we say that a pattern is $\Pi$-descriptive of $S$ if it is $\Sigma$-$\Pi$-descriptive for the smallest $\Sigma$ with $S \subseteq \Sigma^*$.

### A Brief Discussion of The Role of the Terminal Alphabet

By definition, the terminal alphabet over which a pattern is defined is implicitly given by the pattern itself. On the other hand, it is important to explicitly state the underlying terminal alphabet for the pattern language of a pattern (i. e., $L_\Sigma(\alpha) \neq L_{\Sigma'}(\alpha)$ if $\Sigma \neq \Sigma'$). Furthermore, in accordance with the existing literature, we always assume that in a pattern language all terminal symbols that occur in the pattern are also available as symbols in the terminal alphabet (e. g., $L_{\{\mathtt{a},\mathtt{b}\}}(x\mathtt{a}x\mathtt{c}yy\mathtt{a})$ is undefined). For descriptive patterns, as pointed out by Proposition 5, the terminal alphabet under consideration is determined by the sample. For the membership problem, we ask for a given word $w$ and a pattern $\alpha$ whether $w \in L_{\mathrm{term}(w)\cup\mathrm{term}(\alpha)}(\alpha)$. This makes sense, since $\mathrm{term}(w) \not\subseteq \Sigma'$ implies $w \notin L_{\Sigma'}(\alpha)$ and if $\mathrm{term}(\alpha) \not\subseteq \Sigma'$, then $L_\Sigma(\alpha)$ is not defined. As mentioned before, in the literature, a stronger version of the membership problem is also considered, where the input pattern and the input word are required to be defined over a fixed alphabet $\Sigma$.

## 3 The Hardness of Computing $\Pi$-Descriptive Patterns

We now investigate the problem of computing a $\Pi$-descriptive pattern for a sample $S$. Since, by definition, $S \subseteq L_\Sigma(\alpha)$ implies $|\alpha| \leq m = \min\{|w| \mid w \in S\}$, an obvious approach to find a descriptive pattern is to search all patterns that describe $S$ for one that is minimal with respect to the subset relation of the corresponding pattern languages. Unfortunately, this approach cannot be carried out by an algorithm, since the inclusion problem for pattern languages is undecidable (see [5]). However, as shown in [1], in order to compute PAT-descriptive patterns, it is sufficient to only search the patterns of maximal size, for which the inclusion is characterised by the relation $\sqsubseteq$ (Lemma 2). This idea can be extended to natural classes of patterns and, furthermore, if it is possible to compute a $\Pi$-descriptive pattern for any sample, then $\Pi$ must be natural.

**Theorem 6.** *Let $\Pi \subseteq$ PAT. There is an effective procedure that, for a sample $S$, computes a $\Pi$-descriptive pattern of $S$ if and only if $\Pi$ is natural.*

*Proof.* In order to prove the *if* direction, we assume that $\Pi$ is natural. Let $\Sigma$ be the smallest alphabet with $S \subseteq \Sigma^*$. We compute the set $Q$ of all patterns $\alpha$ in canonical form that satisfy $\alpha \in \Pi$ and $S \subseteq L_\Sigma(\alpha)$ by enumerating all patterns $\alpha \in (\Sigma \cup X)^*$ in canonical form up to length $m = \min\{|w| \mid w \in S\}$ and checking whether $\alpha \in \Pi$ (this is possible since $\Pi$ is natural) and $S \subseteq L_\Sigma(\alpha)$ (this is possible since $S$ is finite and the membership problem for $\Pi$-pattern languages is decidable). For any pattern $\alpha$, if $|\alpha| > m$, then $S \nsubseteq L_\Sigma(\alpha)$; thus, the construction of $Q$ from above is correct. By definition, $Q$ is finite and, since $x_1 \in \Pi$ and $S \subseteq L_\Sigma(x_1)$, $Q$ is non-empty. Now let $Q_{\max} \subseteq Q$ contain all elements of $Q$ with maximum length. Next, we compute a pattern $\beta \in Q_{\max}$ that is minimal for the set $Q_{\max}$ with respect to $\sqsubseteq$, which can be done by computing the relation $\sqsubseteq$ for the whole set $Q_{\max}$. We note that if $\beta$ is not $\Pi$-descriptive of $S$, then there exists an $\alpha \in Q$ with $S \subseteq L_\Sigma(\alpha) \subset L_\Sigma(\beta)$. If $\alpha \in Q_{\max}$, then, since $|\alpha| = |\beta|$, $\alpha \sqsubseteq \beta$ is implied, which contradicts the fact that $\beta$ is minimal with respect to $Q_{\max}$ and $\sqsubseteq$. On the other hand, if $\alpha \notin Q_{\max}$, then $|\alpha| < |\beta|$, which contradicts to $L_\Sigma(\alpha) \subset L_\Sigma(\beta)$. Thus, $\beta$ is $\Pi$-descriptive of $S$.

In order to prove the *only if* direction, we assume that there is an effective procedure $\chi$ that, for a given sample $S$, computes a pattern that is $\Pi$-descriptive of $S$. If $x_1 \notin \Pi$, then there is no $\alpha \in \Pi$ with $\{\mathtt{a}, \mathtt{b}\} \subseteq L_{\{\mathtt{a},\mathtt{b}\}}(\alpha)$; thus $\chi$ does not compute a $\Pi$-descriptive pattern on input $\{\mathtt{a}, \mathtt{b}\}$. It remains to show that, for every $\alpha \in$ PAT, the question $\mathrm{cf}(\alpha) \in \Pi$ is decidable. Let $\alpha'$ be obtained from $\alpha$ by substituting each occurrence of a variable $x \in \mathrm{var}(\alpha)$ by a distinct terminal symbol $a_x \notin \mathrm{term}(\alpha)$ and let $\gamma$ be computed by $\chi$ on input $\{\alpha'\}$. Obviously, since $\gamma \in \Pi$, $\gamma \equiv \alpha$ implies $\mathrm{cf}(\alpha) \in \Pi$. Next, we assume that $\gamma \not\equiv \alpha$. Since $\gamma$ is $\Pi$-descriptive of $\{\alpha'\}$, $\alpha' \in L_{\mathrm{term}(\alpha')}(\gamma)$ holds, which implies $\alpha' \sqsubseteq \gamma$. Furthermore, since $\alpha'$ is a renaming of $\alpha$, $\alpha \sqsubseteq \gamma$ is satisfied as well. Consequently, $\{\alpha'\} \subseteq L_{\mathrm{term}(\alpha')}(\alpha) \subset L_{\mathrm{term}(\alpha')}(\gamma)$, which means that if $\mathrm{cf}(\alpha) \in \Pi$, then $\gamma$ is not $\Pi$-descriptive; thus, $\mathrm{cf}(\alpha) \notin \Pi$. □

The procedure of the proof of Theorem 6 is obviously not efficient. Furthermore, we note that it computes a descriptive pattern of *maximal* length. In [1], it is shown that, if $\mathcal{P} \neq \mathcal{NP}$ and $\Pi = \textsc{Pat}$, then, even if the terminal alphabet is a fixed binary alphabet, computing a $\Pi$-descriptive pattern of maximal length cannot be done in polynomial time. We can prove a similar result that is stronger in the sense that it does not need the maximality condition and the size of $S$ can be restricted to 2, but weaker in the sense that the alphabet is considered a part of the input and is not fixed.

**Lemma 7.** *Let $\Pi$ be a natural class of patterns. If there exists a polynomial time algorithm that, for a given sample $S$ of size 2, computes a pattern that is $\Pi$-descriptive of $S$, then the membership problem for $\Pi$-pattern languages is decidable in polynomial time.*

*Proof.* Let $w$ be a word over some alphabet $\Sigma$ and $\alpha \in \Pi$. Without loss of generality, we can also assume that $\alpha \in \Sigma\text{-}\textsc{Pat}$, since otherwise $\mathrm{term}(\alpha) \nsubseteq \Sigma$ and therefore, by definition, $w \notin L_\Sigma(\alpha)$. For every $x \in \mathrm{var}(\alpha)$, let $a_x$ be a distinct terminal symbol with $a_x \notin \Sigma$ and let $\alpha'$ be obtained from $\alpha$ by substituting every occurrence of every variable $x$ by $a_x$. We define $\Sigma' = \Sigma \cup \{a_x \mid x \in \mathrm{var}(\alpha)\}$.

*Claim*: Let $\gamma$ be $\Pi$-descriptive of $\{\alpha', w\}$. Then $\gamma \equiv \alpha$ if and only if $w \in L_\Sigma(\alpha)$.
*Proof of Claim*: The *only if* direction follows trivially, since if $\gamma \equiv \alpha$ and $\gamma$ is $\Pi$-descriptive of $\{w, \alpha'\}$, then $w \in L_\Sigma(\gamma) = L_\Sigma(\alpha)$. We prove the *if* direction by contraposition. To this end, we assume that $\gamma \not\equiv \alpha$. Since $\gamma$ is $\Pi$-descriptive of $\{\alpha', w\}$, $\alpha' \in L_{\Sigma'}(\gamma)$ and therefore $\alpha' \sqsubseteq \gamma$. Moreover, since $\alpha'$ is a renaming of $\alpha$, we can conclude that $\alpha \sqsubseteq \gamma$, which implies $L_{\Sigma'}(\alpha) \subseteq L_{\Sigma'}(\gamma)$. Furthermore, since $\alpha \not\equiv \gamma$, $L_{\Sigma'}(\alpha) \neq L_{\Sigma'}(\gamma)$ and, thus, it follows that $L_{\Sigma'}(\alpha) \subset L_{\Sigma'}(\gamma)$. Now if $w \in L_\Sigma(\alpha)$, then $w \in L_{\Sigma'}(\alpha)$, which implies $\{\alpha', w\} \subseteq L_{\Sigma'}(\alpha)$. Consequently, $\{\alpha', w\} \subseteq L_{\Sigma'}(\alpha) \subset L_{\Sigma'}(\gamma)$, which leads to the contradiction that $\gamma$ is not $\Pi$-descriptive of $\{\alpha', w\}$ and therefore $w \notin L_{\Sigma'}(\alpha)$. (Claim) □

We conclude the proof by observing that if there is a polynomial time algorithm $\chi$ that, for a given sample $S$ of size 2, computes a pattern that is $\Pi$-descriptive of $S$, then we can decide, in polynomial time, whether $w \in L_\Sigma(\alpha)$ by computing a pattern $\gamma$ that is $\Pi$-descriptive of $\{\alpha', w\}$ and checking whether or not $\gamma \equiv \alpha$ holds, which can obviously be done in polynomial time. □

The next lemma shows a similar result, but with respect to the question whether a pattern $\alpha$ is a member of a class $\Pi$ of patterns.

**Lemma 8.** *Let $\Pi$ be a natural class of patterns. If there exists a polynomial time algorithm that, for a given sample $S$, computes a pattern that is $\Pi$-descriptive of $S$, then there exists a polynomial time algorithm that, for a given pattern $\alpha$, decides whether $\mathrm{cf}(\alpha) \in \Pi$.*

*Proof.* Let $\chi$ be a polynomial time algorithm that, for a given sample $S$, computes a pattern that is $\Pi$-descriptive of $S$ and let $\alpha$ be a pattern. For every $x \in \mathrm{var}(\alpha)$, let $a_x$ be a distinct symbol with $a_x \notin \mathrm{term}(\alpha)$ and let $\alpha'$ be obtained from $\alpha$ by substituting every occurrence of every variable $x$ by $a_x$. Let

$\gamma$ be the pattern that is $\Pi$-descriptive of $\{\alpha'\}$ computed by $\chi$. If $\alpha \equiv \gamma$, then, since $\gamma \in \Pi$, $\mathrm{cf}(\alpha) \in \Pi$ follows. Hence, we now assume that $\alpha \not\equiv \gamma$ and observe that this implies $L_{\mathrm{term}(\alpha')}(\alpha) \neq L_{\mathrm{term}(\alpha')}(\gamma)$. Since $\alpha' \in L_{\mathrm{term}(\alpha')}(\gamma)$, $\alpha' \sqsubseteq \gamma$ and, since $\alpha$ is a renaming of $\alpha'$, $\alpha \sqsubseteq \gamma$ follows, which implies $L_{\mathrm{term}(\alpha')}(\alpha) \subseteq L_{\mathrm{term}(\alpha')}(\gamma)$. Furthermore, $\alpha' \in L_{\mathrm{term}(\alpha')}(\alpha)$ obviously holds, which means that $\{\alpha'\} \subseteq L_{\mathrm{term}(\alpha')}(\alpha) \subset L_{\mathrm{term}(\alpha')}(\gamma)$. Consequently, $\mathrm{cf}(\alpha) \in \Pi$ directly implies that $\gamma$ is not $\Pi$-descriptive of $\{\alpha'\}$, which is a contradiction. Thus, $\alpha \equiv \gamma$ if and only if $\mathrm{cf}(\alpha) \in \Pi$ and therefore we can decide in polynomial time whether $\mathrm{cf}(\alpha) \in \Pi$ by computing $\gamma$ with $\chi$ and then checking whether $\alpha \equiv \gamma$. □

Hence, under the assumption $\mathcal{P} \neq \mathcal{NP}$, for the polynomial time computation of descriptive patterns for natural classes $\Pi$ of patterns (i. e., for classes for which descriptive patterns exist) it is necessary that $\Pi$ is tractable. In the following we show that for Shinohara-classes of patterns this condition is also sufficient.

## 4 Computing Descriptive Patterns for Shinohara-Classes

The procedure of the proof of Theorem 6 is inefficient in two regards: there might be an exponential number of patterns to enumerate and for each such pattern we need to solve the membership problem, which, at least in the general case, is $\mathcal{NP}$-complete. In [20], Shinohara presents an algorithm for computing Pat-descriptive patterns in which the only non-efficient element are membership queries. We generalise this algorithm such that it computes $\Pi$-descriptive patterns for an arbitrary Shinohara-class $\Pi$ of patterns (see Algorithm 1). We denote by $\alpha[x \mapsto \pi]$ the pattern obtained from $\alpha$ by substituting $x$ with $\pi$.

$\Pi$-DescPat works as follows. We start with a pattern $\alpha = x_1 x_2 \cdots x_m$, where $m$ is the length of a shortest word $w$ in the sample $S$. Then we move over $\alpha$ from left to right and at every position $i$, we try to refine $\alpha$ by first replacing $x_i$ by the ith symbol of $w$ (Line 4) and then consecutively by all the variables that occur in the prefix $\alpha[1..i-1]$ (Line 7). As soon as one of these refinements yields a pattern that describes the sample $S$ (and that is still in $\Pi$), we move on to the next position and if all refinements fail, then we keep variable $x_i$ at position $i$ (which means that $x_i$ occurs in the final pattern that is computed).

**Lemma 9.** *Let $\Pi$ be a Shinohara-class with shortest word $w$. On input $(S, w)$,* $\Pi$-DescPat *computes a $\Pi$-descriptive pattern of $S$. If $\Pi$ is tractable, then* $\Pi$-DescPat *can be implemented such that it has polynomial running time.*

*Proof.* Let $\alpha$ be the output of $\Pi$-DescPat on input $(S, w)$. We first prove that $\alpha$ is $\Pi$-descriptive of $S$. To this end, let $m = |w|$ and, for every $i$, $1 \leq i \leq m+1$, let $\alpha_i$ be the pattern at the beginning of the ith iteration of the main loop of $\Pi$-DescPat, i. e., $\alpha_i \equiv \mathrm{tg}(\alpha, i-1)$, $1 \leq i \leq m+1$; in particular, $\alpha_1 = x_1 x_2 \cdots x_m$ and $\alpha_{m+1} = \alpha$. We note that if both $S \subseteq L_\Sigma(\alpha_i)$ and $\mathrm{cf}(\alpha_i) \in \Pi$ are satisfied at the beginning of the ith iteration of the main loop, then $\alpha_i$ is changed into $\alpha_{i+1}$ with $S \subseteq L_\Sigma(\alpha_{i+1})$ and $\mathrm{cf}(\alpha_{i+1}) \in \Pi$. This is due to the fact that if $\alpha_i$ is changed

**Algorithm 1:** Π-DescPat

```
Input  : A sample $S \in \Sigma^*$, a shortest word $w$ of $S$.
Output: A Π-descriptive pattern
1  $m := |w|$, $\alpha_1 := x_1 x_2 \cdots x_m$;
2  for $i := 1$ to $m$ do
3      $q :=$ true, $j := 1$;
4      if $\mathrm{cf}(\alpha_i[x_i \mapsto w[i]]) \in \Pi$ and $S \subseteq L_\Sigma(\alpha_i[x_i \mapsto w[i]])$ then
5          $\alpha_{i+1} := \alpha_i[x_i \mapsto w[i]]$ and $q :=$ false;
6      while $j < i$ and $q$ do
7          if $x_j \in \mathrm{var}(\alpha[1, i-1])$, $\mathrm{cf}(\alpha_i[x_i \mapsto x_j]) \in \Pi$, $S \subseteq L_\Sigma(\alpha_i[x_i \mapsto x_j])$
           then
8              $\alpha_{i+1} := \alpha_i[x_i \mapsto x_j]$ and $q :=$ false;
9          else
10             $j := j + 1$;
11     if $q =$ true then
12         $\alpha_{i+1} := \alpha_i$;
13 return $\mathrm{cf}(\alpha_{m+1})$
```

into $\alpha_{i+1}$ by Lines 5 or 8, then the conditions of Lines 4 or 7, respectively, are satisfied, and if Lines 5 or 8 are never executed, then Line 12 is executed, which sets $\alpha_{i+1}$ to $\alpha_i$ and, by assumption, $S \subseteq L_\Sigma(\alpha_i)$ and $\mathrm{cf}(\alpha_i) \in \Pi$. Hence, since $S \subseteq L_\Sigma(\alpha_1)$ and $\mathrm{cf}(\alpha_1) \in \Pi$ is satisfied (see Proposition 1), $S \subseteq L_\Sigma(\alpha)$ and $\alpha \in \Pi$ follows. It remains to show that there is no pattern $\beta \in \Pi$ with $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$. For the sake of contradiction, we assume that there exists such a pattern $\beta \in \Pi$ with $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$.

We first note that $|\alpha| = |\beta|$, which follows from the observation that $|\alpha| < |\beta|$ implies $w \notin L_\Sigma(\beta)$ and $|\beta| < |\alpha|$ implies $L_\Sigma(\beta) \not\subseteq L_\Sigma(\alpha)$. Hence, we have $L_\Sigma(\beta) \subset L_\Sigma(\alpha)$ and $|\alpha| = |\beta|$, which, by Lemma 2, implies $\beta \sqsubseteq \alpha$. Without loss of generality, we can assume that, for every $i$, $1 \leq i \leq m$, if $\beta[i] = x_j$ and $|\beta[1..i-1]|_{x_j} = 0$, then $i = j$ (note that all $\alpha_i$ have this property, too). Since $L_\Sigma(\beta) \subset L_\Sigma(\alpha)$, $\alpha \not\equiv \beta$ and therefore $\alpha \neq \beta$ is implied; thus, there exists a $p$, $1 \leq p \leq |\alpha|$, with $\alpha[p] \neq \beta[p]$ and $\alpha[1..p-1] = \beta[1..p-1]$. As shown above, $\beta \sqsubseteq \alpha$, which implies that $\alpha[p] = x_q$ (for some $x_q \in \mathrm{var}(\alpha)$) and $\beta[p] = z$ (for some $z \in \mathrm{var}(\beta) \cup \Sigma$), i. e., $\beta \sqsubseteq \alpha[x_q \mapsto z]$. Since $x_q \in \mathrm{var}(\alpha)$, position $q$ is the first occurrence of $x_q$ in $\alpha$ and since $\beta[q] = z \neq x_q$ and $\alpha[1..p-1] = \beta[1..p-1]$, it follows that $p = q$. In particular, since $\alpha_q \equiv \mathrm{tg}(\alpha, q-1)$, this also means that $\alpha[x_q \mapsto z] \sqsubseteq \alpha_q[x_q \mapsto z]$ and, since $\sqsubseteq$ is transitive (see Lemma 2), $\beta \sqsubseteq \alpha_q[x_q \mapsto z]$ follows, which implies $S \subseteq L_\Sigma(\alpha_q[x_q \mapsto z])$. Moreover, $\mathrm{cf}(\alpha_q[x_q \mapsto z]) = \mathrm{tg}(\beta, q)$ and $\beta \in \Pi$; thus, with Proposition 1, $\mathrm{cf}(\alpha_q[x_q \mapsto z]) \in \Pi$ is implied. If $z \in \mathrm{var}(\beta)$, then $z \in \{x_1, x_2, \ldots, x_{q-1}\}$. This is due to the fact that, by our assumption from above, the first occurrence of any variable $x_j$, $j \geq q+1$, is to the right of position $q$. If, on the other hand, $z \in \Sigma$, then clearly $z = w[q]$. Consequently, in iteration $q$ of the main loop, either $\mathrm{cf}(\alpha_q[x_q \mapsto w[q]]) \in \Pi$ and $S \subseteq L_\Sigma(\alpha_q[x_q \mapsto w[q]])$ is satisfied or $\mathrm{cf}(\alpha_q[x_q \mapsto x_j]) \in \Pi$ and $S \subseteq L_\Sigma(\alpha_q[x_q \mapsto$

$x_j]$) with $1 \leq j \leq q-1$ is satisfied. This implies that Line 5 or 8 is executed, which means that in $\alpha$ there is no occurrence of variable $x_q$. Since this is clearly a contradiction, we conclude that $\alpha$ is in fact $\Pi$-descriptive of $S$.

It remains to prove that if, for any pattern $\beta$, the question $\mathrm{cf}(\beta) \in \Pi$ and the membership problem for $\Pi$-pattern languages are decidable in polynomial time, then $\Pi$-DescPat is a polynomial time algorithm. To this end, note that the for-loop has $m$ iterations and the while-loop has at most $m$ iterations. Therefore Lines 4 and 7 are executed $\mathcal{O}(m^2)$ times, and for each execution we have to check, for a pattern $\alpha_i$, whether $\mathrm{cf}(\alpha_i) \in \Pi$ and $S \subseteq L_\Sigma(\alpha_i)$. Hence, by first checking $\mathrm{cf}(\alpha_i) \in \Pi$ in polynomial time and then checking $S \subseteq L_\Sigma(\alpha_i)$ only in the case that $\mathrm{cf}(\alpha_i) \in \Pi$, Lines 4 and 7 can be executed in polynomial time. $\square$

From Lemmas 7, 8 and 9 we can conclude the following meta-theorem:

**Theorem 10.** *Let $\Pi$ be a Shinohara-class of patterns. There exists a polynomial time algorithm that, for a given sample $S$, computes a pattern that is $\Pi$-descriptive of $S$ if and only if $\Pi$ is tractable.*

## 4.1 Applications of the Meta-Theorem

The significance of Theorem 10 is brought out by the observation that many classes of patterns that are known to be tractable are in fact Shinohara-classes. We shall now give a brief overview of such classes of patterns.

The class $\textsc{Pat}_{\mathrm{reg}}$ of *regular patterns*, where every variable has only one occurrence (e. g., $x_1\mathtt{ab}x_2x_3\mathtt{a}x_4$), and the class $\textsc{Pat}_{\mathrm{nc}}$ of *non-cross* patterns, where the occurrences of variables are sorted by their index (e. g., $x_1\mathtt{a}x_1x_1x_2\mathtt{b}x_2x_3\mathtt{ab}x_3x_3$), are the classes for which Shinohara originally formulated his algorithm in [20]. However, these classes have the disadvantage of being rather strongly restricted, which means that descriptive regular or non-cross patterns do not very accurately represent the common structure of the words in a sample $S$.

In [17], an infinite hierarchy of classes of patterns has been introduced, where every level of the hierarchy is a tractable Shinohara-class. We recall the definition of this hierarchy. For every $y \in \mathrm{var}(\alpha)$, the *scope of $y$ in $\alpha$* is defined by $\mathrm{sc}_\alpha(y) = \{i, i+1, \ldots, j\}$, where $i$ is the leftmost and $j$ the rightmost position of $y$ in $\alpha$. The scopes of some variables $y_1, y_2, \ldots, y_k \in \mathrm{var}(\alpha)$ *coincide in $\alpha$* if $\bigcap_{1 \leq i \leq k} \mathrm{sc}_\alpha(y_i) \neq \emptyset$. The *scope coincidence degree* of $\alpha$ ($\mathrm{scd}(\alpha)$ for short) is the maximum number of variables in $\alpha$ such that their scopes coincide. For every $k \in \mathbb{N}$, let $\textsc{Pat}_{\mathrm{scd} \leq k} = \{\alpha \in \textsc{Pat} \mid \mathrm{scd}(\alpha) \leq k\}$. Since, for every $k \in \mathbb{N}$, the membership problem for $\textsc{Pat}_{\mathrm{scd} \leq k}$-pattern languages is solvable in polynomial time [17] and $\textsc{Pat}_{\mathrm{scd} \leq k}$ is a Shinohara-class, we can compute $\textsc{Pat}_{\mathrm{scd} \leq k}$-descriptive patterns in polynomial time. Furthermore, by increasing the bound on the scope coincidence degree, we can boost the accuracy of the computed descriptive patterns at the expense of a slower running time and, conversely, by decreasing this bound, we improve on the running time, but lose accuracy of the computed descriptive patterns.

The algorithm $\Pi$-DescPat seems to be of no use, if $\Pi$ is not a Shinohara-class, e. g., the well-known classes $\mathrm{Pat}_{\mathrm{var} \leq k} = \{\alpha \mid |\mathrm{var}(\alpha)| \leq k\}$, $k \in \mathbb{N}$, of

*$k$-variable patterns* (briefly mentioned in Example 4). The membership problem for these classes can obviously be solved in polynomial time and Angluin shows in [1] that it is possible to compute $\mathrm{Pat}_{\mathrm{var}\leq 1}$-descriptive patterns in polynomial time. However, to the knowledge of the authors, it is still an open question whether or not $\mathrm{Pat}_{\mathrm{var}\leq k}$-descriptive patterns can be computed in polynomial time, for $k \geq 2$ (see also [4,8,18]). In contrast to the classes $\mathrm{Pat}_{\mathrm{var}\leq k}$, the classes $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k} = \{\alpha \mid |\{x \in \mathrm{var}(\alpha) \mid |\alpha|_x \geq 2\}| \leq k\}$, $k \in \mathbb{N}$, of patterns with at most *$k$ repeated* variables are Shinohara-classes. The algorithm $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-DescPat can therefore be used in order to compute $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-descriptive patterns and this even in polynomial time since the conditions of Theorem 10 are satisfied. Of course, a $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-descriptive pattern $\alpha$ is not necessarily $\mathrm{Pat}_{\mathrm{var}\leq k}$-descriptive, but, since $\mathrm{Pat}_{\mathrm{var}\leq k} \subseteq \mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$, $\alpha$ covers $S$ at least as closely as a $\mathrm{Pat}_{\mathrm{var}\leq k}$-descriptive one, i. e., it is impossible that a $\mathrm{Pat}_{\mathrm{var}\leq k}$-descriptive pattern $\beta$ exists with $S \subseteq L(\beta) \subset L(\alpha)$. So if we are interested in $\mathrm{Pat}_{\mathrm{var}\leq k}$-descriptive patterns it seems that computing $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-descriptive patterns instead is a good alternative.

We shall now exhibit in more detail the connections between descriptive patterns and inductive inference of pattern languages (for unexplained concepts and definitions the reader is referred to [21]). It has been shown in [2] that, for a class $\Pi$ of patterns, the following procedure describes an inference machine for $\Pi$-pattern languages: for every new word $w$ that is not described by the current hypothesis, we output as new hypothesis a pattern that is $\Pi$-descriptive of all formerly received words and $w$. If $\Pi$-descriptive patterns can be computed in polynomial time, then this inference machine infers the $\Pi$-pattern languages in polynomial time. Thus, we obtain the following corollary of Theorem 10:

**Corollary 11.** *Let $\Pi$ be a tractable Shinohara-class of patterns. Then the class of $\Pi$-pattern languages is polynomial time inferable from positive data.*

Consequently, Shinohara's algorithm is very useful for the polynomial time inference of $\Pi$-pattern languages if $\Pi$ is a Shinohara-class.

The classes of $k$-variable patterns were also the first for which polynomial time inference was investigated. Since these classes are no Shinohara-classes, we can not use the approach from above to obtain a polynomial time inference machine for $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern languages. If instead we use the algorithm $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-DescPat, we get an inference machine for $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$-pattern languages rather than for $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern language. Since $\mathrm{Pat}_{\mathrm{var}\leq k} \subset \mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$, this inference machine could also be used for inferring $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern languages, but then it produces hypotheses that are not in $\mathrm{Pat}_{\mathrm{var}\leq k}$ and therefore, in the strict sense, it is not an inference machine for $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern languages. It can nevertheless be transformed into an *inconsistent* inference machine (i. e., one that may output hypotheses that do not describe all the received words) for $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern languages by simply keeping the current hypothesis if the new hypothesis would be in $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k} \setminus \mathrm{Pat}_{\mathrm{var}\leq k}$ or, alternatively, even into a consistent one by replacing every hypothesis from $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k} \setminus \mathrm{Pat}_{\mathrm{var}\leq k}$ by the hypothesis $x_1$. This provides an alternative proof for the result first shown by

Lange [11] that $\text{Pat}_{\text{var}\leq k}$-pattern languages are consistently polynomial time inferable (in fact, the inference machine of [11] also produces the overly general hypothesis $x_1$ from time to time).

## 5 The Consistency Problem for Patterns

The *consistency problem* (for patterns) is to decide for given finite sets $P, N \subseteq \Sigma^*$, whether there exists a pattern $\alpha$ that is *consistent* with $P$ and $N$, i. e., $P \subseteq L_\Sigma(\alpha)$ and $N \cap L_\Sigma(\alpha) = \emptyset$. For any class $\Pi$ of patterns, the $\Pi$-*consistency problem* is to find a pattern of $\Pi$ that is consistent with $P$ and $N$.

The consistency problem (sometimes also called *separation problem*) is a formalisation of the natural task to find a rule that separates one set of examples from another and it arises in various contexts, e. g., learning theory, artificial intelligence and model checking. It is also crucial for *probably approximately correct (PAC) learning*, introduced by Valiant [23], since its polynomial time solvability is necessary for polynomial time PAC learning (see, Blumer et al. [3]).

For arbitrary classes $\Pi$ of patterns, the $\Pi$-consistency problem is in $\Sigma_2^P$, the second level of the polynomial-time hierarchy (see, e. g., [16]), since it can be solved by first guessing a pattern $\alpha$ and then checking by membership queries whether $\alpha$ is consistent (note that this directly implies containment in $\mathcal{NP}$ if the membership problem for $\Pi$-pattern languages can be solved in polynomial time) and Ko and Tzeng show in [10] that the PAT-consistency problem is even $\Sigma_2^P$-complete (an $\mathcal{NP}$-hardness result is given in [9]). This result suggests that, unlike for the problem of computing descriptive patterns, the hardness of the membership problem is not solely responsible for the hardness of the PAT-consistency problem and this intuition is supported by the fact that the consistency problem is even $\mathcal{NP}$-hard for regular patterns (see Miyano et al. [15]), for which the membership problem can be easily solved in polynomial time. It turns out that, by modifying the construction and the proof given in [15], this result can be strengthened in the following way:

**Theorem 12.** *Let* $\Pi \subseteq \text{PAT}$*,* $\sigma$ *be a symbol and* $\Gamma = \{\beta_1 \cdots \beta_n \mid n \in \mathbb{N}, \beta_i \in \{x_i\sigma, \sigma x_i\}, 1 \leq i \leq n\}$*. If* $\Gamma \subseteq \Pi$*, then the* $\Pi$*-consistency problem is* $\mathcal{NP}$*-hard.*

*Proof.* In [15, Theorem 3.1], Miyano et al. show, by a reduction from 3SAT, that the consistency problem of regular patterns is $\mathcal{NP}$-complete. By a minor modification of the construction and the argument, this result also holds for all classes $\Pi$ that contain the class $\Gamma$.

We shall now recall the construction from [15]. Let $F = \{C_1, C_2, \dots, C_m\}$ be a 3-CNF formula with clauses $C_i$, $1 \leq i \leq m$, and Boolean variables $y_1, y_2, \dots, y_n$. We assume that no clause contains both $y_i$ and $\overline{y_i}$, the variable $y_i$ in negated form. The sets $P$ and $N$ of words over $\{a, b\}$ are defined in the following way:

$$
\begin{aligned}
s_0 &= (aa)^n,\\
\widehat{t}_j &= a^j, 1 \le j \le 2n-1.\\
s_i &= (aa)^{i-1}\, aba\, (aa)^{n-i}, 1 \le i \le n,\\
t_i &= (aa)^{i-1}\, bb\, (aa)^{n-i}, 1 \le i \le n,\\
d_k &= r_1 r_2 \cdots r_n, 1 \le k \le m, \text{ with } r_\ell = \begin{cases} ab & \text{if } y_\ell \in C_k,\\ ba & \text{if } \overline{y_\ell} \in C_k,\ 1 \le \ell \le n,\\ aa & \text{else.}\end{cases}\\
P &= \{s_i \mid 0 \le i \le n\},\\
N &= \{\widehat{t}_j, t_i, d_k \mid 1 \le j \le 2n-1, 1 \le i \le n, 1 \le k \le m\}.
\end{aligned}
$$

The only difference compared to the reduction from [15] is that we add the words $\widehat{t}_j$, $1 \le j \le 2n-1$, to $N$. It can be verified in the same way as done in [15] that there exists $\alpha \in \Gamma$ that is consistent with $P$ and $N$ if and only if $F$ is satisfiable (more precisely, let $\sigma : \{y_i \mid 1 \le i \le n\} \to \{\text{true}, \text{false}\}$ and $\alpha = \beta_1\beta_2 \cdots \beta_n \in \Gamma$ such that $\sigma(y_i) = \text{true}$ if and only if $\beta_i = x_i a$ and $\sigma(y_i) = \text{false}$ if and only if $\beta_i = a x_i$, then $\sigma$ satisfies $F$ if and only if $\alpha$ is consistent with $P$ and $N$).

It remains to show that there exists a pattern that is consistent with $P$ and $N$ if and only if there exists a pattern in $\Gamma$ that is consistent with $P$ and $N$. Since the *if* direction is obviously true, we shall now assume that there exists a pattern $\alpha$ that is consistent with $P$ and $N$. Since $s_0 \in P$, we can conclude that $\alpha$ is a pattern over $(X \cup \{a\})^*$ of length at most $|s_0|$. If $|\alpha| = j < |s_0| = 2n$, then $\alpha$ can generate $\widehat{t}_j$; thus, $|\alpha| = 2n$ follows. The words $s_i$ have length $2n+1$ with $s_i[2i] = b$. This means that it must be possible to map $\alpha$ to $s_i$ in such a way that $s_i[2i]$ is generated by either $\alpha[2i-1]$ or $\alpha[2i]$, which implies that $\alpha[2i-1]$ or $\alpha[2i]$ is a variable. Furthermore, since $s_i[2i]$ is the only occurrence of $b$ in $s_i$, this variable has only one occurrence in $\alpha$. Hence, $\alpha = \beta_1\beta_2 \cdots \beta_n$ with $\beta_i = z_i x_i$ or $\beta_i = x_i z_i$, where $|\alpha|_{x_i} = 1$ and $z_i \in (X \cup \{a\})$. If, for some $i$, $1 \le i \le n$, $z_i \in X$ and $|\alpha|_{z_i} = 1$, then $t_i$ can be generated by $\alpha$, which is a contradiction; thus, for every $i$, $1 \le i \le n$, either $z_i = a$ or $z_i \in X$ with $|\alpha|_{z_i} \ge 2$. We assume now that, for some $i'$ with $1 \le i' \le n$, $z_{i'} \in X$ with $|\alpha|_{z_{i'}} \ge 2$. Since $\alpha$ is consistent, it can generate every $s_i$, $0 \le i \le n$, and since $|\alpha| = 2n$, $|s_i| \le 2n+1$, $|s_i|_b \le 1$, $1 \le i \le n$, this can only be done by substituting $z_{i'}$ with the single letter $a$. This implies that we can substitute every occurrence of $z_{i'}$ in $\alpha$ by $a$ and obtain a pattern that is still consistent with $P$ and $N$ (note that the impossibility of generating words from $N$ is not affected). Consequently, by replacing all $z_i$ with $|\alpha|_{z_i} \ge 2$ in $\alpha$ by $a$, we can transform $\alpha$ into a pattern $\alpha' \in \Gamma$ that is consistent with $P$ and $N$. □

Theorem 12 is a strong negative result, since it implies the $\mathcal{NP}$-hardness of the consistency problem for all the classes $\text{PAT}_{\text{reg}}$-, $\text{PAT}_{\text{nc}}$-, $\text{PAT}^{\mathsf{r}}_{\text{var}\le k}$- and $\text{PAT}_{\text{scd}\le k}$, $k \in \mathbb{N}$, for which the membership problem is known to be solvable in polynomial time. As $\Gamma \nsubseteq \textsc{Pat}_{\mathrm{var}\leq k}$, the question arises whether the $\textsc{Pat}_{\mathrm{var}\leq k}$-consistency problem can be solved in polynomial time, for some $k \in \mathbb{N}$.

These observations point out that the $\Pi$-consistency problem can be intractable even though the membership problem for $\Pi$-pattern languages can be solved efficiently. As reported in the previous sections, this contrasts with the problem of computing descriptive patterns. Nevertheless, we are able to prove a result about the consistency problem that is similar to Lemma 7, i. e., if the membership problem for $\Pi$-pattern languages is $\mathcal{NP}$-hard, then the $\Pi$-consistency problem is $\mathcal{NP}$-hard as well (at least for classes $\Pi$ of patterns with a bounded number of occurrences of terminal symbols). Before we state this result, we first cite the following two lemmas.

**Lemma 13** (Angluin [1])**.** *Let $\Sigma$ be an alphabet with $|\Sigma| \geq 2$ and let $\alpha \in \Sigma$-*$\textsc{Pat}$*. There exists a set $S_\alpha \subseteq L_\Sigma(\alpha)$ such that, for every pattern $\beta$ with $|\alpha| = |\beta|$, $S_\alpha \subseteq L_\Sigma(\beta)$ implies $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$. Furthermore, $S_\alpha$ can be computed in time linear in $|\alpha|$.*

**Lemma 14** (Ko and Tzeng [10])**.** *Let $\Sigma$ be an alphabet, let $\alpha \in \Sigma$-*$\textsc{Pat}$* be a pattern. There exist finite sets $P_\alpha, N_\alpha \subseteq \Sigma^*$ with $P_\alpha \subseteq L_\Sigma(\alpha)$ and $N_\alpha \cap L_\Sigma(\alpha) = \emptyset$ with the following properties. For every $P, N \subseteq \Sigma^*$ with $P_\alpha \subseteq P$ and $N_\alpha \subseteq N$, if $\beta$ is consistent with $P$ and $N$, then $|\beta| = |\alpha|$. The sets $P_\alpha$ and $N_\alpha$ can be constructed in time $\mathcal{O}(2^k(|\alpha| + 1)^{k+1})$, where $k = \sum_{a\in\mathrm{term}(\alpha)} |\alpha|_a$.*

We are now ready to state and prove the result mentioned above.

**Theorem 15.** *Let $\Pi \subseteq \textsc{Pat}$ with $\sum_{a\in\mathrm{term}(\alpha)} |\alpha|_a \leq k$ for all $\alpha \in \Pi$ and a constant $k$. If the $\Pi$-consistency problem is solvable in polynomial time, then the membership problem for $\Pi$-pattern languages is solvable in polynomial time.*

*Proof.* Let $\alpha$ be a pattern, let $w$ be a word and let $\Sigma = \mathrm{term}(\alpha) \cup \mathrm{term}(w)$ with $|\Sigma| \geq 2$. Furthermore, let $P = S_\alpha \cup P_\alpha \subseteq \Sigma^*$ and $N = N_\alpha \cup \{w\} \subseteq \Sigma^*$ (where $S_\alpha$, $P_\alpha$ and $N_\alpha$ are the sets given by Lemmas 13 and 14).

*Claim*: There exists a pattern that is consistent with $P$ and $N$ if and only if $w \notin L_\Sigma(\alpha)$.

*Proof of Claim*: We first prove the *only if* direction and assume that $\beta$ is a pattern that is consistent with $P$ and $N$. By Lemma 14, $P_\alpha \subseteq P$ and $N_\alpha \cap L_\Sigma(\beta) = \emptyset$ implies $|\alpha| = |\beta|$. Furthermore, since $S_\alpha \subseteq L_\Sigma(\beta)$, we conclude with Lemma 13 that $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$. Now if $w \in L_\Sigma(\alpha)$, then $w \in L_\Sigma(\beta)$, which is a contradiction to the assumption that $\beta$ is consistent with $P$ and $N$; thus, $w \notin L_\Sigma(\alpha)$ follows. In order to prove the *if* direction, we assume that there does not exist a pattern that is consistent with $P$ and $N$. In particular, this means that $\alpha$ is not consistent with $P$ and $N$. By Lemmas 13 and 14, we know that $P \subseteq L_\Sigma(\alpha)$ and $N_\alpha \cap L_\Sigma(\alpha) = \emptyset$, which implies that $w \in L_\Sigma(\alpha)$, since otherwise $\alpha$ would be consistent with $P$ and $N$. (Claim) □

Now let $\Pi \subseteq \textsc{Pat}$ and $\sum_{a\in\mathrm{term}(\alpha)} |\alpha|_a \leq k$ for all $\alpha \in \Pi$ and some constant $k$. We assume that there exists a polynomial time algorithm $\chi$ that solves the $\Pi$-consistency problem. We can now solve the membership problem for $\Pi$-pattern languages for an instance $\alpha$ and $w$ in the following way. We first construct the sets $P = S_\alpha \cup P_\alpha$ and $N = N_\alpha \cup \{w\}$. Since $\sum_{a \in \mathrm{term}(\alpha)} |\alpha|_a \leq k$ for all $\alpha \in \Pi$, this can be done in polynomial time (see Lemma 14). Then we use $\chi$ in order to decide whether or not there exists a pattern in $\Pi$ that is consistent with $P$ and $S$ in polynomial time, which, as stated by the Claim, answers whether or not $w \in L_\Sigma(\alpha)$. $\square$

The requirement in Theorem 15 that the patterns have a bounded number of constants seems to be a strong restriction. However, the set $\text{Pat}_{\text{tf}}$ of terminal-free patterns is a prominent class of patterns that has been studied in the context of learning theory and language theory; moreover, terminal-free patterns are generally used in order to describe combinatorial properties in words.

Next, we try to answer the question whether there are non-trivial classes $\Pi$ of patterns for which the consistency problem can be solved in polynomial time. In this regard, we can state the following simple sufficient condition.

**Proposition 16.** *Let $\Pi \subseteq \text{Pat}$. If, for every word $w$ over an alphabet $\Sigma$, all $\alpha \in \Pi$ with $w \in L_\Sigma(\alpha)$ and all $\beta \in \Pi$ with $w \notin L_\Sigma(\beta)$ can be enumerated in polynomial time, then the $\Pi$-consistency problem is solvable in polynomial time.*

Note that the condition of Proposition 16 is equivalent to the requirement that the membership problem for $\Pi$-pattern languages can be solved efficiently and, for every word $w$, all $\alpha \in \Pi$ with $w \in L_\Sigma(\alpha)$ *or* all $\beta \in \Pi$ with $w \notin L_\Sigma(\beta)$ can be enumerated in polynomial time.

It turns out that there are structurally simple, yet interesting examples of classes of patterns for which the condition of Proposition 16 is satisfied, e. g., the class $\{x_1 x_2^k x_3 \mid k \geq 2\}$ of patterns that describe words that contain repetitions of exponent at least 2 and the class $\{x_1 x_2 x_3 x_2 (x_3 x_2)^k x_4 \mid k \geq 1\}$ describing words that contain overlaps. While these are fairly special classes of patterns that have no applications in learning or language theory, for them the consistency problem can be solved in linear time and they are relevant in other parts of theory, e. g., combinatorics and algorithimcs on words. Another example of a larger class of patterns satisfying the condition of Proposition 16 is the class $(\text{Pat}_{\text{nc}} \cap \text{Pat}_{\text{var}\leq k} \cap \text{Pat}_{\text{tf}})$.

## References

[1] D. Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21:46–62, 1980.

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

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[4] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, and T. Zeugmann. Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. *Theor. Comput. Sci.*, 261:119–156, 2001.

[5] D.D. Freydenberger and D. Reidenbach. Bad news on decision problems for patterns. *Inform. and Comput.*, 208:83–96, 2010.

[6] D.D. Freydenberger and D. Reidenbach. Existence and nonexistence of descriptive patterns. *Theor. Comput. Sci.*, 411:3274–3286, 2010.

[7] T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Decision problems for patterns. *J. Comput. Syst. Sci.*, 50:53–63, 1995.

[8] K.-I. Ko and C.-M. Hua. A note on the two-variable pattern-finding problem. *J. Comput. Syst. Sci.*, 34:75–86, 1987.

[9] K.-I. Ko, A. Marron, and W.-G. Tzeng. Learning string patterns and tree patterns from examples. In *Proc. of the seventh international conference on Machine learning*, pages 384–391, 1990.

[10] K.-I. Ko and W.-G. Tzeng. Three $\Sigma_2^P$-complete problems in computational learning theory. *Computational Complexity*, 1:269–310, 1991.

[11] S. Lange. A note on polynominal-time inference of $k$-variable pattern languages. In *Proc. 1st International Workshop on Nonmonotonic and Inductive Logic*, volume 543 of *LNCS*, pages 178–183, 1991.

[12] S. Lange and R. Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.

[13] A. Mateescu and A. Salomaa. Patterns. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 230–242. Springer, 1997.

[14] Z. Mazadi, Z. Gao, and S. Zilles. Distinguishing pattern languages with membership examples. In *Proc. 8th LATA*, volume 8370 of *LNCS*, pages 528–540, 2014.

[15] S. Miyano, A. Shinohara, and T. Shinohara. Polynomial-time learning of elementary formal systems. *New Generation Comput.*, 18(3):217–242, 2000.

[16] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[17] D. Reidenbach and M.L. Schmid. Patterns with bounded treewidth. *Inform. and Comput.* To appear.

[18] R. Reischuk and T. Zeugmann. Learning one-variable pattern languages in linear average time. In *Proc. 11th COLT*, pages 198–208, 1998.

[19] T. Shinohara. Polynomial time inference of extended regular pattern languages. In *Proc. RIMS Symposium on Software Science and Engineering*, volume 147 of *LNCS*, pages 115–127, 1982.

[20] T. Shinohara. Polynomial time inference of pattern languages and its application. In *Proc. 7th IBM Symp. Math. Found. Comp. Sci.*, pages 191–209, 1982.

[21] T. Shinohara and S. Arikawa. Pattern inference. In K.P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems, GOSLER Final Report*, volume 961 of *LNAI*, pages 259–291. 1995.

[22] T. Shinohara and H. Arimura. Inductive inference of unbounded unions of pattern languages from positive data. *Theor. Comput. Sci.*, 241:191–209, 2000.

[23] L.G. Valiant. A theory of the learnable. *Commun. ACM*, 27:1134–1142, 1984.