Revisiting Shinohara's Algorithm for Computing Descriptive Patterns

Henning Fernau[a], Florin Manea[b], Robert Mercaş[b], Markus L. Schmid[a]

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

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

Abstract

A pattern α is a word consisting of constants and variables and it describes the pattern language $L(\alpha)$ of all words that can be obtained by uniformly replacing the variables with constant words. In 1982, Shinohara presents an algorithm that computes a pattern that is descriptive for a finite set S of words, i. e., its pattern language contains S in the closest possible way among all pattern languages. We generalise Shinohara's algorithm to subclasses of patterns and characterise those subclasses for which it is applicable. Furthermore, within this set of pattern classes, we characterise those for which Shinohara's algorithm has a polynomial running time (under the assumption $\mathcal{P} \neq \mathcal{NP}$). Moreover, we also investigate the complexity of the consistency problem of patterns, i. e., finding a pattern that separates two given finite sets of words.

Keywords: Pattern Languages, Inductive Inference, Descriptive Patterns, $\mathcal{NP}$-Hard Problems
2010 MSC: 68Q25, 68Q17, 68Q32

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 α, denoted by $L(\alpha)$, is the set of all words that can be obtained from α 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 α 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 [2], 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 [3].

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 [4]) – 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., [5, 6, 7, 8, 9] and, for a survey, [10]).

A useful tool for the inductive inference of pattern languages are so-called *descriptive patterns*, introduced in [1]. A pattern α is descriptive of a finite set S of words if $S \subseteq L(\alpha)$ and there is no pattern β 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 α 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 α 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 α. 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 length* necessarily solves the membership problem and therefore, assuming $\mathcal{P} \neq \mathcal{NP}$, it cannot have a polynomial running time.

Descriptive patterns have also been applied in approaches of learning upper-best approximations of other types of formal languages (see Kobayashi and Yokomori [11], and Freydenberger and Reidenbach [12]).

In [7], 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 Π of patterns; more precisely, a pattern α is Π-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 Π-descriptive pattern exists if and only if the pattern x_1 is in Π and, furthermore, that a modified version of Shinohara's algorithm can be used to compute Π-descriptive patterns (of maximal length) if and only if Π is a *Shinohara-class*, i. e., it contains the set $\{x_1 x_2 \cdots x_k \mid k \in \mathbb{N}\}$ and, for every $\alpha \in \Pi$, the pattern α' obtained by substituting some length i suffix of α by a sequence of new variables $y_1 y_2 \cdots y_i$ is also in Π. Within the set of Shinohara-classes of patterns, we prove that Π-descriptive patterns can be computed in polynomial time if and only if the question whether $\alpha \in \Pi$ and the membership problem for Π can be decided in polynomial time.

We also investigate the consistency problem for classes Π 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 (under the assumption $\mathcal{P} \neq \mathcal{NP}$).

The outline of the paper is as follows. In Section 2, we define the central concepts and present some preliminary observations. Then, we demonstrate in Section 3 that the hardness of solving the membership problem for a class of patterns entails the hardness of computing descriptive patterns for this class. Shinohara's algorithm is extended to Shinohara-classes of patterns in Section 4 and some possible applications are discussed. In Section 5, we investigate the complexity of the consistency problem.

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 ε 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 $1 \leq k \leq |w|$, and, for each $1 \leq i < j \leq |w|$, let $w[j..i] = \varepsilon$.

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 Σ 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* Σ) is a non-empty word over $\Sigma \cup X$ and a (*terminal*) *word* is a string over Σ. We define Σ-$\textsc{Pat} = (\Sigma \cup X)^+$ and $\textsc{Pat} = \bigcup_{\Sigma} \Sigma$-$\textsc{Pat}$. For any pattern α, 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 α 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 α is in *canonical form* if and only if, for some $k \in \mathbb{N}$,

$\mathrm{var}(\alpha) = \{x_1, \dots, x_k\}$ 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 α, by $\mathrm{cf}(\alpha)$, we denote its canonical form, i. e., $\mathrm{cf}(\alpha)$ is obtained from α 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 $\alpha \in \Sigma\text{-}\textsc{Pat}$, the *pattern language of* α (*over* Σ) 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 Π-*pattern languages.*

Let $\Pi \subseteq \textsc{Pat}$. The class Π is *natural* if, for a given α, the question $\mathrm{cf}(\alpha) \in \Pi$ is decidable and the pattern x_1 is in Π. The class Π is called *tractable* if the question whether $\mathrm{cf}(\alpha)$ is in Π can be decided in polynomial time and the membership problem for Π-pattern languages can be decided in polynomial time. For any pattern α and for every i, $0 \leq i \leq |\alpha|$, we define the *i-tail-generalisation of* α by $\mathrm{tg}(\alpha, i) = \mathrm{cf}(\alpha[1..i] \cdot y_1 y_2 \cdots y_{|\alpha|-i})$, where $\{y_1, y_2, \dots, y_{|\alpha|-i}\} \subseteq (X \setminus \mathrm{var}(\alpha))$ with $|\{y_1, y_2, \dots, y_{|\alpha|-i}\}| = |\alpha| - i$. The *tail-generalisation of* α is the set $\mathrm{tg}(\alpha) = \{\mathrm{tg}(\alpha, i) \mid 0 \leq i \leq |\alpha|\}$ and this definition is lifted to sets Π of patterns by $\mathrm{tg}(\Pi) = \bigcup_{\alpha \in \Pi} \mathrm{tg}(\alpha)$. A natural class Π of patterns is a *Shinohara-class* if, for every $k \in \mathbb{N}$, Π contains a pattern of length k and $\mathrm{tg}(\Pi) = \Pi$. The following proposition follows immediately from the definition.

Proposition 1. *Let Π 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 Π 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 Σ 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 Σ with $|\Sigma| \geq 2$, then $\alpha \sqsubseteq \beta$.*

Lemma 2 only shows that, for every alphabet Σ, $\alpha \sqsubseteq \beta$ is a sufficient condition for $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$ (see [1] for an example that shows that $\alpha \sqsubseteq \beta$ is in fact not a necessary condition). However, for the special case that $|\alpha| = |\beta|$, $\alpha \sqsubseteq \beta$ is characteristic for $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$ if Σ contains at least 2 terminal 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 Π-pattern languages* asks to decide for a given pattern $\alpha \in \Pi$ and a word w, whether $w \in L_{\mathrm{term}(w) \cup \mathrm{term}(\alpha)}(\alpha)$.

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 Σ is a fixed binary alphabet. Let Σ be an alphabet, S be a finite set of words and $\Pi \subseteq \textsc{Pat}$. A pattern α is Σ-Π-*descriptive of* S if $\alpha \in \Sigma\text{-}\textsc{Pat}$, $\mathrm{cf}(\alpha) \in \Pi$, $S \subseteq L_\Sigma(\alpha)$ 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* Σ).

Example 1 (Angluin [1], Freydenberger and Reidenbach [14]). We define the alphabets $\Sigma_1 = \{\mathtt{a}, \mathtt{b}\}$ and $\Sigma_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, and the samples $S_1 = \{\mathtt{aabaa}, \mathtt{babab}, \mathtt{aabab}, \mathtt{babaa}\}$, $S_2 = \{\mathtt{ababa}, \mathtt{ababbababbab},$

[1]Descriptive patterns for *erasing* pattern languages as well as for *infinite* sets have also been investigated (see Jiang et al. [13] and Freydenberger and Reidenbach [14]).

$\texttt{babab}\}$ and $S_3 = \{\texttt{aabcaaa}, \texttt{caacbca}, \texttt{bbcccbbbc}\}$. The patterns $x_1\texttt{aba}x_2$ and $x_1x_1x_2$ are both Σ_1-PAT-descriptive of S_1. The patterns $x_1x_2x_1x_2x_1$ and $x_1\texttt{ab}x_2$ are both Σ_1-PAT-descriptive of S_2. For every $i \in \mathbb{N}$, let $\text{PAT}_{\text{var}\leq i} = \{\alpha \in \text{PAT} \mid |\text{var}(\alpha)| \leq i\}$ denote the set of patterns with at most i variables. The pattern $x_1x_2\texttt{c}x_3x_1$ is Σ_2-$\text{PAT}_{\text{var}\leq 3}$-descriptive of S_3 and x_1 is Σ_2-$\text{PAT}_{\text{var}\leq 1}$-descriptive of S_3.

By definition, it is possible that a pattern is Σ-Π-descriptive of a sample $S \subseteq \Gamma^*$, where $\Gamma \subset \Sigma$. However, as pointed out by the following proposition, this aspect is negligible as long as the sample contains at least one non-unary word.

Proposition 4. *Let Σ be an alphabet, $S \subseteq \Sigma^*$ a sample that contains a non-unary word, $\Gamma = \text{term}(S)$ and $\Pi \subseteq \text{PAT}$. A pattern is Γ-Π-descriptive of S if and only if it is Σ-Π-descriptive of S.*

PROOF. We first note that $\Gamma \subseteq \Sigma$. If α is not Γ-Π-descriptive of S, then $\alpha \notin \Gamma$-PAT, $\text{cf}(\alpha) \notin \Pi$ or $S \nsubseteq L_\Gamma(\alpha)$, which directly implies that α is not Σ-Π-descriptive of S; or these three conditions are satisfied, but there exists a $\beta \in \Pi$ with $S \subseteq L_\Gamma(\beta) \subset L_\Gamma(\alpha)$, which also means that $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$ and therefore α is not Σ-Π-descriptive of S.

On the other hand, if α is Γ-Π-descriptive of S, then $\text{cf}(\alpha) \in \Pi$ and, since $\Gamma \subseteq \Sigma$, $L_\Gamma(\alpha) \subseteq L_\Sigma(\alpha)$ is implied and therefore $S \subseteq L_\Sigma(\alpha)$ holds. If there is a $\beta \in \Pi$ with $S \subseteq L_\Sigma(\beta) \subset L_\Sigma(\alpha)$, then $\alpha \not\cong \beta$ and $L_\Gamma(\beta) \neq L_\Gamma(\alpha)$ (since $|\Gamma| \geq 2$). 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 Γ and are images of α, 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 α is Γ-Π-descriptive of S. □

As justified by Proposition 4, in the following we are only concerned with Σ-Π-descriptive patterns, where $\Sigma = \text{term}(S)$. For the sake of convenience, we say that a pattern is Π-descriptive of S if it is Σ-Π-descriptive for $\Sigma = \text{term}(S)$.

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

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., $L_{\{\texttt{a},\texttt{b}\}}(x\texttt{a}x\texttt{c}yy\texttt{a})$ is undefined). For descriptive patterns, as pointed out by Proposition 4, the terminal alphabet under consideration is determined by the sample. For the membership problem, we ask for a given word w and a pattern α whether $w \in L_{\text{term}(w)\cup\text{term}(\alpha)}(\alpha)$. This makes sense, since $\text{term}(w) \nsubseteq \Sigma$ implies $w \notin L_\Sigma(\alpha)$ and if $\text{term}(\alpha) \nsubseteq \Sigma$, 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 word and the pattern language (and therefore also the pattern) are required to be defined over a fixed alphabet Σ. For the consistency problem, the underlying terminal alphabet is given by the sets P and N, i. e., $\Sigma = \text{term}(P) \cup \text{term}(N)$.

Consequently, for all our complexity results regarding these computational problems, the terminal alphabet is not fixed, but rather a part of the input (implicitly given by the terminal words). We emphasise that analogous statements with respect to fixed alphabets would be stronger and must not be confused with the results of this work.

3. The Hardness of Computing Π-Descriptive Patterns

We now investigate the problem of computing a Π-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 [15]). However, as shown in [1], in order to compute PAT-descriptive patterns, it is sufficient to only search the patterns of maximal length, for which the inclusion is characterised by the relation $\sqsubseteq$ (Lemma 2). In the following, we extend this idea to natural classes of patterns and show that if it is possible to compute a Π-descriptive pattern for any sample, then Π must be natural.

Theorem 5. *Let $\Pi \subseteq \text{PAT}$. There is an effective procedure that, for a sample S, computes a Π-descriptive pattern of S if and only if Π is natural.*

PROOF. In order to prove the *if* direction, we assume that Π is natural. Let $\Sigma = \text{term}(S)$. We compute the set Q of all patterns α 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 Π is natural) and $S \subseteq L_\Sigma(\alpha)$ (this is possible since S is finite and the membership problem for Π-pattern languages is decidable). For any pattern α, 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 β is not Π-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 β 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, β is Π-descriptive of S.

In order to prove the *only if* direction, we assume that there is an effective procedure χ that, for a given sample S, computes a pattern that is Π-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 χ does not compute a Π-descriptive pattern on input $\{\mathtt{a}, \mathtt{b}\}$. It remains to show that, for every $\alpha \in \text{PAT}$, the question $\text{cf}(\alpha) \in \Pi$ is decidable. Let $\overline{\alpha}$ be obtained from α by substituting each occurrence of a variable $x \in \text{var}(\alpha)$ by a distinct terminal symbol $\overline{a}_x \notin \text{term}(\alpha)$ and, likewise, let $\widehat{\alpha}$ be obtained by substituting each occurrence of an $x \in \text{var}(\alpha)$ by a distinct terminal symbol $\widehat{a}_x \notin (\text{term}(\alpha) \cup \text{term}(\overline{\alpha}))$. Furthermore, let γ be the pattern computed by χ on input $\{\overline{\alpha}, \widehat{\alpha}\}$. In the following, we shall show that $\text{cf}(\alpha) \in \Pi$ if and only if $\alpha \equiv \gamma$, which concludes the proof.

We first observe that since $\gamma \in \Pi$, $\gamma \equiv \alpha$ implies $\text{cf}(\alpha) \in \Pi$. Next, we assume that $\gamma \not\equiv \alpha$. Since γ is Π-descriptive of $\{\overline{\alpha}, \widehat{\alpha}\}$, $\{\overline{\alpha}, \widehat{\alpha}\} \subseteq L_{\text{term}(\overline{\alpha}) \cup \text{term}(\widehat{\alpha})}(\gamma)$ holds, which implies $\overline{\alpha} \sqsubseteq \gamma$ and $\widehat{\alpha} \sqsubseteq \gamma$. This means that there is a non-erasing substitution h with $h(\gamma) = \overline{\alpha}$ and, furthermore, all occurrences of terminal symbols $\overline{a}_x$ with $x \in \text{var}(\alpha)$ in $\overline{\alpha}$ are generated by some variable of γ, since otherwise some of these terminal symbols must occur in γ and this is a contradiction to $\widehat{\alpha} \sqsubseteq \gamma$. Hence, the substitution h' that is obtained from h by replacing, in the images of h, each occurrence of a terminal symbol $\overline{a}_x$ by the variable x satisfies $h'(\gamma) = \alpha$ and therefore $\alpha \sqsubseteq \gamma$. We conclude that $\{\overline{\alpha}, \widehat{\alpha}\} \subseteq L_{\text{term}(\overline{\alpha}) \cup \text{term}(\widehat{\alpha})}(\alpha) \subset L_{\text{term}(\overline{\alpha}) \cup \text{term}(\widehat{\alpha})}(\gamma)$, where the second inclusion follows from $\alpha \sqsubseteq \gamma$ and it is proper due to the assumption $\gamma \not\equiv \alpha$. In particular, this implies $\text{cf}(\alpha) \notin \Pi$, as otherwise γ would not be Π-descriptive. □

The procedure of the proof of Theorem 5 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 $\Pi = \text{PAT}$, then computing a Π-descriptive pattern of maximal length cannot be done in polynomial time. By modifying the proof technique of [1], we can show a similar result with respect to natural classes Π of patterns without the maximality condition. Furthermore, while for the result from [1] the terminal alphabet is fixed and binary, but the size of the sample is unbounded, in our result, we need an unbounded alphabet, but samples of size only 2.

Lemma 6. *Let Π 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 Π-descriptive of S, then the membership problem for Π-pattern languages is decidable in polynomial time.*

PROOF. Let w be a word over some alphabet Σ and $\alpha \in \Pi$. Without loss of generality, we can also assume that $\alpha \in \Sigma\text{-PAT}$, since otherwise $\text{term}(\alpha) \nsubseteq \Sigma$ 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 α' be obtained from α 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 γ be Π-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 γ is Π-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 γ is Π-descriptive of $\{\alpha', w\}$, $\alpha' \in L_{\Sigma'}(\gamma)$ and therefore $\alpha' \sqsubseteq \gamma$. Moreover, since α' is obtained from α by injectively replacing variables by terminal symbols, 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 γ is not Π-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 χ that, for a given sample S of size 2, computes a pattern that is Π-descriptive of S, then we can decide, in polynomial time, whether $w \in L_\Sigma(\alpha)$ by computing a pattern γ that is Π-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 α is a member of a class Π of patterns.

Lemma 7. *Let Π be a natural class of patterns. If there exists a polynomial time algorithm that, for a given sample S, computes a pattern that is Π-descriptive of S, then there exists a polynomial time algorithm that, for a given pattern α, decides whether* $\mathrm{cf}(\alpha) \in \Pi$.

Proof. Let χ be a polynomial time algorithm that, for a given sample S, computes a pattern that is Π-descriptive of S and let α be a pattern. We construct patterns $\overline{\alpha}$ and $\widehat{\alpha}$ just like in the proof of Theorem 5 and let γ be the pattern that is Π-descriptive of $\{\overline{\alpha}, \widehat{\alpha}\}$ computed by χ. In the same way as in the proof of Theorem 5, we can now show that $\mathrm{cf}(\alpha) \in \Pi$ if and only if $\alpha \equiv \gamma$. Since, by assumption, γ can be computed in polynomial time and $\alpha \equiv \gamma$ can be checked in polynomial time, the observations from above yield a polynomial time algorithm for deciding whether or not $\mathrm{cf}(\alpha) \in \Pi$. □

Hence, under the assumption $\mathcal{P} \neq \mathcal{NP}$, for the polynomial time computation of descriptive patterns for natural classes Π of patterns (i. e., for classes for which descriptive patterns exist) it is necessary that Π 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 5 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 [7], Shinohara presents an algorithm for computing Pat-descriptive patterns in which the only non-efficient element is the necessity of membership queries. We generalise this algorithm so that it computes Π-descriptive patterns for an arbitrary Shinohara-class Π of patterns (see Algorithm 1). We denote by $\alpha[x \mapsto \pi]$ the pattern obtained from α by substituting x with π.

Π-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 α from left to right and at every position i, we try to refine α by first replacing x_i by the i^{th} 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 Π), 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). Since this algorithm always computes a descriptive pattern of the length of a shortest word in the sample, it will always produce a descriptive pattern of maximal length.

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 $q$ and $j < i$ do
7          if $x_j \in \mathrm{var}(\alpha_i[1, i-1])$, $\mathrm{cf}(\alpha_i[x_i \mapsto x_j]) \in \Pi$, $S \subseteq L_\Sigma(\alpha_i[x_i \mapsto x_j])$ 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 $\mathrm{cf}(\alpha_{m+1})$
```

Lemma 8. *Let Π be a Shinohara-class and let S be a sample with shortest word w. On input (S, w), Π-*DescPat *computes a Π-descriptive pattern of S. If Π is tractable, then* Π-DescPat *can be implemented so that it has polynomial running time.*

Proof. Let α be the output of Π-DescPat on input (S, w). We first prove that α is Π-descriptive of S. To this end, let $m = |w|$ and, for every i, $1 \leq i \leq m+1$, let α_i be the pattern at the beginning of the i^{th} iteration of the main loop of Π-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 i^{th} iteration of the main loop, then α_i is changed into α_{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 α_i is changed into α_{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 α_{i+1} to α_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 $L_\Sigma(\beta) \nsubseteq L_\Sigma(\alpha)$. 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 α_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 α 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 α there is no occurrence of variable x_q. Since this is clearly a contradiction, we conclude that α is in fact Π-descriptive of S.

It remains to prove that if, for any pattern β, the question $\mathrm{cf}(\beta) \in \Pi$ and the membership problem for

Π-pattern languages is decidable in polynomial time, then Π-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 α_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. □

From Lemmas 6, 7 and 8 we can conclude the following meta-theorem:

Theorem 9. *Let Π be a Shinohara-class of patterns. There exists a polynomial time algorithm that, for a given sample S, computes a pattern that is Π-descriptive of S if and only if Π is tractable.*

4.1. Applications of the Meta-Theorem

The significance of Theorem 9 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 $\mathrm{PAT}_{\mathrm{reg}}$ 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 $\mathrm{PAT}_{\mathrm{nc}}$ 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 [7]. 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 [16], 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 α* 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 α. The scopes of some variables $y_1, y_2, \ldots, y_k \in \mathrm{var}(\alpha)$ *coincide in α* if $\bigcap_{1 \leq i \leq k} \mathrm{sc}_\alpha(y_i) \neq \emptyset$. The *scope coincidence degree* of α ($\mathrm{scd}(\alpha)$ for short) is the maximum number of variables in α such that their scopes coincide. For every $k \in \mathbb{N}$, let $\mathrm{PAT}_{\mathrm{scd} \leq k} = \{\alpha \in \mathrm{PAT} \mid \mathrm{scd}(\alpha) \leq k\}$. Since, for every $k \in \mathbb{N}$, the membership problem for $\mathrm{PAT}_{\mathrm{scd} \leq k}$-pattern languages is solvable in polynomial time [16] and $\mathrm{PAT}_{\mathrm{scd} \leq k}$ is a Shinohara-class, we can compute $\mathrm{PAT}_{\mathrm{scd} \leq k}$-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 Π-DescPat seems to be of no use, if Π 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 1). 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 [17, 6, 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 9 are satisfied.[2] Of course, a $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var} \leq k}$-descriptive pattern α is not necessarily $\mathrm{Pat}_{\mathrm{var} \leq k}$-descriptive, but, since $\mathrm{Pat}_{\mathrm{var} \leq k} \subseteq \mathrm{Pat}^{\mathsf{r}}_{\mathrm{var} \leq k}$, α 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 β 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.

For the classes of pattern languages mentioned above, fast algorithms for the membership problem have been presented recently in [19], e. g., the membership problem for patterns with at most one repeated variable (note that this is a proper superclass of regular patterns) can be solved in time $\mathcal{O}(|w|(|w| + |\alpha|))$, whereas for non-cross patterns it can be solved in time $\mathcal{O}(|w| m \log(|w|))$ (here, m is the number of factors $(x_i)^k$ of maximum length in the pattern).

[2]Note that $\mathrm{Pat}^{\mathsf{r}}_{\mathrm{var} \leq k}$ has a scope coincidence degree bounded by $k + 1$ and, thus, is tractable.

We shall now exhibit in more detail the connections between descriptive patterns and inductive inference of pattern languages. To this end, we first recall the definition of inductive inference from positive data (for the case of pattern languages, see [10] for a more general treatment). A *positive representation* of a pattern language L is an infinite sequence $w_1, w_2, w_3, \ldots$, such that $L = \{w_i \mid 1 \leq i\}$. An *inference machine* M is an effective procedure that, given a positive representation of a pattern language, produces a sequence $\alpha_1, \alpha_2, \alpha_3, \ldots$ of *hypothesis patterns.* We say that M *converges* to α if the sequence of hypothesis patterns is finite and ends with α or if there exists an integer k, such that $\alpha_i = \alpha$, for every i, $i \geq k$. If, for a class Π of patterns, M always converges to an α' with $L(\alpha') = L(\alpha)$ on any given positive representation of $L(\alpha)$, then M *infers* Π*-pattern languages from positive data.*

It has been shown in [4] that, for a class Π of patterns, the following procedure describes an inference machine for Π-pattern languages: for every new word w that is not described by the current hypothesis, we output as new hypothesis a pattern that is Π-descriptive of all formerly received words and w. If Π-descriptive patterns can be computed in polynomial time, then this inference machine infers the Π-pattern languages in polynomial time.[3] Thus, we obtain the following corollary of Theorem 9:

Corollary 10. *Let* Π *be a tractable Shinohara-class of patterns. Then the class of* Π*-pattern languages is polynomial time inferable from positive data.*

Consequently, Shinohara's algorithm is very useful for the polynomial time inference of Π-pattern languages if Π 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 [21] that $\mathrm{Pat}_{\mathrm{var}\leq k}$-pattern languages are consistently polynomial time inferable (in fact, the inference machine of [21] also produces the overly general hypothesis x_1 from time to time).

5. The Consistency Problem for Patterns

The consistency problem (sometimes also called *separation problem*), which is formally defined in Section 2, 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. [24]). Furthermore, the computational hardness of the consistency problem for pattern languages has been used in [22], in order to demonstrate that polynomial time learnability of pattern languages (with informant) can only be achieved by ignoring data, i. e., by inconsistent learning strategies (as defined at the end of Section 4.1).

For arbitrary classes Π of patterns, the Π-consistency problem is in Σ_2^P, the second level of the polynomial-time hierarchy (see, e. g., [25]), since it can be solved by first guessing a pattern α and then checking by membership queries whether α is consistent (note that this directly implies containment in $\mathcal{NP}$ if the membership problem for Π-pattern languages can be solved in polynomial time) and Ko and Tzeng show

[3]Here, polynomial time inference means that the time to compute a new hypothesis can be bounded by a polynomial in the total length of the words received so far. It should be pointed out that this notion of polynomial time inference is controversial (see [20] for a discussion)

in [26] that the Pat-consistency problem is even Σ_2^P-complete (an $\mathcal{NP}$-hardness result is given in [27]). 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. [28]), 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 [28], this result can be strengthened in the following way:

Theorem 11. *Let $\Pi \subseteq$ Pat, c be a terminal symbol and $\Gamma = \{\beta_1 \cdots \beta_n \mid n \in \mathbb{N}, \beta_i \in \{x_i c, c x_i\}, 1 \leq i \leq n\}$. If $\Gamma \subseteq \Pi$, then the Π-consistency problem is $\mathcal{NP}$-hard.*

Proof. In [28, 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 Π that contain the class Γ.

We shall now recall the construction from [28]. 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:

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

The only difference compared to the reduction from [28] is that we add the words $\widehat{t}_j$, $1 \leq j < 2n-1$, to N. It can be verified in the same way as done in [28] that there exists $\alpha \in \Gamma$ that is consistent with P and N if and only if F is satisfiable (more precisely, let $\sigma : \{y_i \mid 1 \leq i \leq n\} \to \{\text{true}, \text{false}\}$ 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 σ satisfies F if and only if α 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 Γ that is consistent with P and N. Since the *if* direction is obviously true, we shall now assume that there exists a pattern α that is consistent with P and N. Since $s_0 \in P$, we can conclude that α is a pattern over $(X \cup \{a\})^*$ of length at most $|s_0|$. If $|\alpha| = j < |s_0| = 2n$, then α 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 α 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 α. 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, $1 \leq i \leq n$, $z_i \in X$ and $|\alpha|_{z_i} = 1$, then t_i can be generated by α, which is a contradiction; thus, for every i, $1 \leq i \leq n$, either $z_i = a$ or $z_i \in X$ with $|\alpha|_{z_i} \geq 2$. We assume now that, for some i' with $1 \leq i' \leq n$, $z_{i'} \in X$ with $|\alpha|_{z_{i'}} \geq 2$. Since α is consistent, it can generate every s_i, $0 \leq i \leq n$, and since $|\alpha| = 2n$, $|s_i| \leq 2n+1$, $|s_i|_b \leq 1$, $1 \leq i \leq n$, 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 α 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 $|\alpha|_{z_i} \geq 2$ in α by a, we can transform α into a pattern $\alpha' \in \Gamma$ that is consistent with P and N. □

Theorem 11 is a strong negative result, since it implies the $\mathcal{NP}$-hardness of the consistency problem for all the classes $\textsc{Pat}_{\mathrm{reg}}$, $\textsc{Pat}_{\mathrm{nc}}$, $\textsc{Pat}^{\mathsf{r}}_{\mathrm{var}\leq k}$ and $\textsc{Pat}_{\mathrm{scd}\leq k}$, $k \in \mathbb{N}$, for which the membership problem is known to be solvable in polynomial time. As the set Γ from Theorem 11 is not a subset of $\textsc{Pat}_{\mathrm{var}\leq k}$, 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 Π-consistency problem can be intractable even though the membership problem for Π-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 6, i. e., if the membership problem for Π-pattern languages is $\mathcal{NP}$-hard, then the Π-consistency problem is $\mathcal{NP}$-hard as well (at least for classes Π of patterns with a bounded number of occurrences of terminal symbols). Before we state this result, we first cite the following two lemmas.

Lemma 12 (Angluin [1]). *Let Σ 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 β with $|\alpha| = |\beta|$, $S_\alpha \subseteq L_\Sigma(\beta)$ implies $L_\Sigma(\alpha) \subseteq L_\Sigma(\beta)$. Furthermore, S_α can be computed in polynomial time.*

Lemma 13 (Ko and Tzeng [26]). *Let Σ 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 β is consistent with P and N, then $|\beta| = |\alpha|$. The sets P_α and N_α 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 14. *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 Π-consistency problem is solvable in polynomial time, then the membership problem for Π-pattern languages is solvable in polynomial time.*

Proof. Let α 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_α, P_α and N_α are the sets given by Lemmas 12 and 13).

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 β is a pattern that is consistent with P and N. By Lemma 13, $P_\alpha \subseteq P$ and $N_\alpha \cap L_\Sigma(\beta) = \emptyset$ implies $|\alpha| = |\beta|$. Furthermore, since $S_\alpha \subseteq L_\Sigma(\beta)$, we conclude with Lemma 12 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 β 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 α is not consistent with P and N. By Lemmas 12 and 13, 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 α would be consistent with P and N. (Claim) $\square$

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 χ that solves the Π-consistency problem. We can now solve the membership problem for Π-pattern languages for an instance α 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 13). Then we use χ in order to decide whether or not there exists a pattern in Π 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 14 that the patterns have a bounded number of constants seems to be a strong restriction. However, the set $\textsc{Pat}_{\mathrm{tf}}$ 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 Π of patterns for which the consistency problem can be solved in polynomial time. To this end, we note that if, for some $\Pi \subseteq \textsc{Pat}$ and for every word w, all $\alpha \in \Pi$ with $w \in L_\Sigma(\alpha)$ can be enumerated in polynomial time, then the Π-consistency problem is solvable in polynomial time. The same holds for the situation that all $\beta \in \Pi$ with $w \notin L_\Sigma(\beta)$ can be enumerated in polynomial time.

While this is an obvious sufficient condition, it turns out that it is satisfied by structurally different examples of classes of patterns, e. g., the class $(\mathrm{PAT}_{\mathrm{nc}} \cap \mathrm{PAT}_{\mathrm{var}\leq k} \cap \mathrm{PAT}_{\mathrm{tf}})$. Other structurally simple, yet interesting examples of classes of patterns satisfying the condition from above are given by 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 polynomial time and they are relevant in other parts of discrete mathematics, e. g., combinatorics and algorithmics on words. Indeed, solving the Π-consistency problem for the class $\Pi = \{x_1 x_2^k x_3 \mid k \geq 2\}$ (respectively, $\Pi = \{x_1 x_2 x_3 x_2 (x_3 x_2)^k x_4 \mid k \geq 1\}$) and the input sets of words P and N means essentially finding the exponent k of a repetition such that each word of P contains a factor of the form t^k, while the pattern x^k is avoided (in the combinatorics on words sense) by all words of N, i.e., the repetitions occurring in the words of N have exponent strictly less than k.

Finally, observe that one might find Π-consistent patterns, for certain classes of patterns Π, more efficiently than enumerating all the patterns α of Π such that $P \subseteq L(\alpha)$ and testing for each of them whether $N \cap L(\alpha) = \emptyset$, or, alternatively, enumerating all the patterns α of Π such that $N \cap L(\alpha) = \emptyset$ and testing for each of them whether $P \subseteq L(\alpha)$. As an example, we consider again the class $\Pi = \{x_1 x_2^k x_3 \mid k \geq 2\}$, and assume that we are given two sets of words P and N. For each $w \in P \cup N$ we define the value $k_w = \max\{k \mid w$ has a factor t^k with $t \neq \varepsilon$, which is neither a prefix nor a suffix$\}$; let now $k_P = \min\{k_w \mid w \in P\}$ and $k_N = \max\{k_w \mid w \in N\}$. Let $\alpha = x_1 x_2^k x_3$ be a pattern from Π. It is not hard to see that $P \subseteq L(\alpha)$ if and only if $k \leq k_P$; similarly, $L(\alpha) \cap N = \emptyset$ if and only if $k > k_N$. Now, clearly, there exists $\alpha = x_1 x_2^k x_3$ such that $P \subseteq L(\alpha)$ and $L(\alpha) \cap N = \emptyset$ if and only if $k_N < k_P$. We only have to note that we can compute for each $w \in P \cup N$ the value k_w by computing the runs of w, i.e., the maximal periodic factors of w, and this can be done in $\mathcal{O}(|w|)$ time (see [29]); so k_P and k_N can be computed in $\mathcal{O}(\sum_{w\in P} |w| + \sum_{w \in N} |w|)$ time. Therefore, we can solve the Π-consistency problem, in this case, in linear time. A similar argument shows that for $\Pi = \{x_1 x_2 x_3 x_2 (x_3 x_2)^k x_4 \mid k \geq 1\}$ we can also find Π-consistent patterns in linear time.

Acknowledgements

We wish to thank the anonymous reviewers of this paper (and earlier versions of it) for their careful and detailed reviewing, which made us aware of several inaccuracies in the technical arguments that have now been resolved.

The work of Florin Manea was supported by by DFG grant number 596676. The work of Robert Mercaş was supported by the P.R.I.M.E. programme of DAAD with funds provided by the Federal Ministry of Education and Research (BMBF) and the European Unions Seventh Framework Programme for research, technological development and demonstration (grant agreement no. 605728).

References

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

[2] T. Shinohara, Polynomial time inference of extended regular pattern languages, in: Proceedings of RIMS Symposium on Software Science and Engineering, Vol. 147 of LNCS, 1982, pp. 115–127.

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

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

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

[6] R. Reischuk, T. Zeugmann, Learning one-variable pattern languages in linear average time, in: Proceedings of the 11th Annual Conference on Computational Learning Theory, COLT, 1998, pp. 198–208.

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

[8] T. Shinohara, H. Arimura, Inductive inference of unbounded unions of pattern languages from positive data, Theoretical Computer Science 241 (2000) 191–209.

[9] Z. Mazadi, Z. Gao, S. Zilles, Distinguishing pattern languages with membership examples, in: Proceedings of the 8th International Conference on Language and Automata Theory and Applications, LATA, Vol. 8370 of LNCS, 2014, pp. 528–540.

[10] T. Shinohara, S. Arikawa, Pattern inference, in: K. Jantke, S. Lange (Eds.), Algorithmic Learning for Knowledge-Based Systems, GOSLER Final Report, Vol. 961 of LNAI, 1995, pp. 259–291.

[11] S. Kobayashi, T. Yokomori, On approximately identifying concept classes in the limit, in: Proceedings of the 6th International Conference on Algorithmic Learning Theory, ALT, Vol. 997 of LNCS, 1995, pp. 298–312.

[12] D. D. Freydenberger, D. Reidenbach, Inferring descriptive generalisations of formal languages, Journal of Computer and System Sciences 79 (2013) 622–639.

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

[14] D. D. Freydenberger, D. Reidenbach, Existence and nonexistence of descriptive patterns, Theoretical Computer Science 411 (2010) 3274–3286.

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

[16] D. Reidenbach, M. L. Schmid, Patterns with bounded treewidth, Information and Computation 239 (2014) 87–99.

[17] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, T. Zeugmann, Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries, Theoretical Computer Science 261 (2001) 119–156.

[18] K.-I. Ko, C.-M. Hua, A note on the two-variable pattern-finding problem, Journal of Computer and System Sciences 34 (1987) 75–86.

[19] H. Fernau, F. Manea, R. Mercaş, M. L. Schmid, Pattern matching with variables: Fast algorithms and new hardness results, in: Proceedings of the 32nd Symposium on Theoretical Aspects of Computer Science, STACS, Vol. 30 of LIPIcs, 2015, pp. 302–315.

[20] J. Case, T. Kötzing, Difficulties in forcing fairness of polynomial time inductive inference, in: Proceedings of the 20th International Conference on Algorithmic Learning Theory, ALT, Vol. 5809 of LNCS, 2009, pp. 263–277.

[21] S. Lange, A note on polynominal-time inference of k-variable pattern languages, in: Proceedings of the 1st International Workshop on Nonmonotonic and Inductive Logic, NIL, Vol. 543 of LNCS, 1991, pp. 178–183.

[22] R. Wiehagen, T. Zeugmann, Ignoring data may be the only way to learn efficiently, Journal of Experimental and Theoretical Artificial Intelligence 6 (1) (1994) 131–144.

[23] L. Valiant, A theory of the learnable, Communications of ACM 27 (1984) 1134–1142.

[24] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, Journal of ACM 36 (4) (1989) 929–965.

[25] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[26] K.-I. Ko, W.-G. Tzeng, Three Σ_2^P-complete problems in computational learning theory, Computational Complexity 1 (1991) 269–310.

[27] K.-I. Ko, A. Marron, W.-G. Tzeng, Learning string patterns and tree patterns from examples, in: Proceedings of the 7th International Conference on Machine Learning, ICML, 1990, pp. 384–391.

[28] S. Miyano, A. Shinohara, T. Shinohara, Polynomial-time learning of elementary formal systems, New Generation Computing 18 (3) (2000) 217–242.

[29] R. M. Kolpakov, G. Kucherov, Finding maximal repetitions in a word in linear time, in: 40th Annual Symposium on Foundations of Computer Science, FOCS, 1999, pp. 596–604.