Polynomial-Time Algorithms for Learning Typed Pattern Languages*

Michael Geilke[1] and Sandra Zilles[2]

[1] Fachbereich Informatik, Technische Universität Kaiserslautern
D-67653 Kaiserslautern, Germany
geilke.michael@gmail.com
[2] Department of Computer Science, University of Regina
Regina, SK, Canada S4S 0A2
zilles@cs.uregina.ca

Abstract. This article proposes polynomial-time algorithms for learning typed pattern languages—formal languages that are generated by patterns consisting of terminal symbols and typed variables. A string is generated by a typed pattern by substituting all variables with strings of terminal symbols that belong to the corresponding types.
The algorithms presented consitute non-trivial generalizations of Lange and Wiehagen's efficient algorithm for learning patterns in which variables are not typed. This is achieved by defining *type witnesses* to impose structural conditions on the types used in the patterns. It is shown that Lange and Wiehagen's algorithm implicitly uses a special case of type witnesses. Moreover, the type witnesses for a typed pattern form characteristic sets whose size is linear in the length of the pattern; our algorithm, when processing any set of positive data containing such a characteristic set, will always generate a typed pattern equivalent to the target pattern. Thus our algorithms are of relevance to the area of grammatical inference, in which such characteristic sets are typically studied.

1 Introduction

Since Dana Angluin [1] introduced the pattern languages, they have been a popular object of study in the area of algorithmic learning theory and formal language theory. A pattern is a string consisting of terminal and variable symbols; its language is the set of all words that can be generated when replacing variables with non-empty strings of terminal symbols. Part of the reason for their popularity is that patterns provide an intuitive and simple way of representing rather complex but structured languages; moreover they can be thought of as a model for text mining applications.

It is known that the class of all pattern languages is learnable from positive data (i.e., if the learning algorithm sees only words contained in the target pattern language) in Gold's model of identification in the limit [7, 1]. The

* This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

first polynomial-time algorithm for learning pattern languages was presented by Lange and Wiehagen [10].

Unfortunately, most interesting text mining applications require a pattern model in which the variables are typed, i.e., the set of strings that can be substituted for a variable in a pattern is restricted. This is only natural since database entries are typically typed as well. Koshiba [9] hence introduced the model of typed pattern languages, in which each variable in a pattern has an associated type that defines the set of allowed substitutions for the variable. After Koshiba's initial results on learning typed pattern languages, recent progress was made on learning relational pattern languages (a generalization of typed pattern languages) [6]. In particular, learnability of relational and thus of typed pattern languages from positive data was shown in variations of Gold's learning model that at least partly address the issue of efficiency in terms of the amount of data presented to the learning algorithm [6].

The present paper focuses on the problem of learning typed pattern languages in polynomial time, in particular on generalizing Lange and Wiehagen's algorithm to the case of typed pattern languages. This is not only of interest for both practical and theoretical reasons, but it also constitutes a non-trivial task. Lange and Wiehagen's algorithm exploits only the shortest words contained in the given data and crucially relies on the fact that these words are formed when variables are replaced by strings of length one. Moreover, it relies on the fact that there are two distinct strings of length one that can be replaced for each variable (see Section 2.2 for more background on Lange and Wiehagen's algorithm). Neither property can be guaranteed when dealing with typed patterns.

We hence introduce the notion of *type witness* to impose certain structural conditions on the types used in the patterns. We demonstrate that

- Lange and Wiehagen's algorithm implicitly uses a special kind of type witness according to our definition,
- type witnesses allow for learning the corresponding typed pattern languages in polynomial time, using a generalization of Lange and Wiehagen's algorithm,
- type witnesses for a typed pattern form characteristic sets [3, 8] whose size is linear in the length of the pattern; our algorithm, when processing any set of positive data containing such a characteristic set, will always generate a typed pattern equivalent to the target pattern.[3]

Some of our results also address the important question of polynomial-time learning with a *consistent* algorithm that always hypothesizes patterns that generate all of the positive input data seen [4]. Neither Lange and Wiehagen's algorithm nor our proposed algorithms have this property in general, but we identify special cases in which consistent behaviour can be achieved without sacrificing runtime efficiency.

This article is the first one to design polynomial-time algorithms for learning typed pattern languages. It contributes to the state-of-the-art as follows:

[3] Characteristic sets play an important role in the area of grammatical inference.

– Our generalized study of Lange and Wiehagen's algorithm provides new theoretical insights into the difficulties of efficiently learning patterns in application-relevant scenarios, and it proposes new algorithmic solutions.
– The definition of type witnesses is likely to be of value for further studies on learning typed pattern languages and their generalizations, relational pattern languages.
– The connection to the use of characteristic sets bridges gaps between the inductive inference community (in which pattern languages have mostly been studied so far) and the grammatical inference community.

2 Preliminaries and Background

Languages are defined with respect to a non-empty *alphabet* Σ. A *word* w is a finite, possibly empty, sequence of symbols from Σ the length of which is denoted by $|w|$. ϵ refers to the empty word, *i.e.*, the word of length 0. The set of all words over Σ is denoted by Σ^*, and the set of all non-empty words over Σ by Σ^+; hence $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. A *language* L is a subset of Σ^*. By $w_1 \circ w_2$ we denote the concatenation of two words w_1 and w_2 (where, for ease of presentation, we allow w_1 and/or w_2 to be written as $\sigma \in \Sigma$ rather than a word (σ) of length 1). In what follows, we always assume Σ to be a finite set of cardinality at least 2.

For any $w, v \in \Sigma^*$, we say that v is a *prefix* of w (*suffix* of w, resp.), if $w = v \circ w'$ ($w = w' \circ v$, resp.) for some $w' \in \Sigma^*$. A prefix (suffix, resp.) of w is called *proper* if it is distinct from w. If $A \subseteq \Sigma^*$ and $w \in \Sigma^*$, we write $w \sqsubseteq A$ ($w \sqsubset A$, resp.) if w is a prefix (proper prefix, resp.) of some word in A.

For $A \subseteq \Sigma^*$ and $w \in \Sigma^*$, we call w *A-decomposable*, if w can be written as $w = w_1 \circ \cdots \circ w_m$ for some $m \geq 1$, where $w_1, \ldots, w_m \in A$. $(w_1, \ldots w_m)$ is then called an *A-decomposition* of w; the length of this A-decomposition is m.

2.1 Pattern Languages and Typed Pattern Languages

A class of languages that has been studied in the formal language theory community as well as in the learning theory community is Angluin's class of non-erasing pattern languages [1], defined as follows. Let $X = \{x_1, x_2, \ldots\}$ be a countable set of symbols called variables; we require that X be disjoint from Σ. A *pattern* is a non-empty finite string over $\Sigma \cup X$. The set of all patterns over $\Sigma \cup X$ will be denoted by Pat_Σ. A *substitution* is a string homomorphism $\theta : Pat_\Sigma \to \Sigma^+$ that is the identity when restricted to Σ. The set of all substitutions with respect to Σ is denoted by Θ_Σ. The *(non-erasing) language* $L(p)$ of a pattern $p \in (\Sigma \cup X)^+$ is defined by $L(p) = \{w \in \Sigma^+ \mid \exists \theta \in \Theta_\Sigma \, [\theta(p) = w]\}$, *i.e.*, it consists of all words that result from substituting all variables in p by non-empty words over Σ. The class $\mathcal{L}_\Sigma$ of (non-erasing) pattern languages is defined by $\mathcal{L}_\Sigma = \{L(p) \mid p \in Pat_\Sigma\}$.

Thus patterns constitute a simple and intuitive means of describing formal languages of a particular kind of structure. From a practical point of view, pattern languages are interesting because in applications one can think of a set of database entries as being generated by a fixed underlying pattern.

Shinohara [13] defined erasing pattern languages by allowing substitutions by the empty word, but in this article we focus on non-erasing languages exclusively. Wright [14] studied a subclass of erasing pattern languages under restricted substitutions, leading to Koshiba's typed pattern languages [9]. In Koshiba's model, each variable x in a pattern p is assigned a particular *type* $T(x)$, which is a decidable subset of Σ^+. The words generated by p, when respecting the types, are formed by substituting any variable x in p only with words from $T(x)$, resulting in the typed pattern language $L(p,T)$.

Definition 1 *A* typed pattern *over Σ is a pair (p,T), where p is a pattern over Σ and $T : X \to \{S \subseteq \Sigma^+ \mid S \text{ is decidable}\}$ is a mapping that associates each variable with a recursive language. For each $x \in X$, $T(x)$ is called the* type *of x. A T-typed substitution is a string homomorphism $\theta : Pat_\Sigma \to \Sigma^+$ that is the identity when restricted to Σ and that fulfills $\theta(x) \in T(x)$ for all $x \in X$. The set of all T-typed substitutions with respect to Σ is denoted by $\Theta_{\Sigma,T}$. The language of a typed pattern (p,T), denoted by $L(p,T)$, is defined by $L(p,T) = \{w \in \Sigma^+ \mid \exists \theta \in \Theta_{\Sigma,T}\ [\theta(p) = w]\}$.*

Types make pattern languages more expressive and more suitable for applications. For example, a system for entering bibliographic data as described by Shinohara [13] might use patterns like $p = \texttt{author:}\ x_1\ \texttt{title:}\ x_2\ \texttt{year:}\ x_3$. One would expect x_3 to be substituted only by certain two or four digit integers—a property that becomes expressible when using types.

For the sake of simplicity, we assume that the range of T is finite. However, all of the results we present below generalize very easily to the case that the range of T is infinite, under minor conditions explained in Section 5.1.

Definition 2 *Let $\mathcal{T}$ be a finite set of decidable subsets of Σ^+. The class of all $\mathcal{T}$-typed pattern languages is the class of all languages $L(p,T)$ where (p,T) is a typed pattern and $T(x) \in \mathcal{T}$ for all $x \in X$.*

2.2 Learnability

In Gold's model of learning in the limit from positive data [7], a class of languages is learnable if there is a learner that "identifies" every language in the class from any of its texts, where a text for a language L is an infinite sequence $\tau(0), \tau(1), \tau(2), \ldots$ of words such that $\{\tau(i) \mid i \in \mathbb{N}\} = L$.

Definition 3 (Gold [7]) *Let $\mathcal{L}$ be a class of languages. $\mathcal{L}$ is learnable in the limit from positive data if there is a hypothesis space $\{L_i \mid i \in \mathbb{N}\} \supseteq \mathcal{L}$ and a partial recursive mapping $\mathcal{A}$ such that, for any $L \in \mathcal{L}$ and any text $(\tau(i))_{i\in\mathbb{N}}$ for L, $\mathcal{A}(\tau(0), \ldots, \tau(n))$ is defined for all n and there is a $j \in \mathbb{N}$ with $L_j = L$ and $\mathcal{A}(\tau(0), \ldots, \tau(n)) = j$ for all but finitely many n.*

Below we will focus on *polynomial-time learning*, i.e., learning by an algorithm that computes its hypotheses in time polynomial in the length of its input.[4]

[4] This notion of efficient learning does not refer to the number of text examples needed before convergence; for a critical treatment of notions of efficiency in learning, the reader is referred to the work by Pitt [11] and Case and Kötzing [5].

Angluin showed that $\mathcal{L}_\Sigma$ is learnable [1]. Most intuitive learning algorithms, including the one proposed by Angluin, conduct a large number of membership tests ("does word w belong to the language $L(p)$?") in order to identify pattern languages from text. Since the membership problem for non-erasing pattern languages is NP-complete [1], polynomial-time learning cannot be guaranteed by this kind of algorithm.

Lange and Wiehagen [10] designed an algorithm that learns $\mathcal{L}_\Sigma$ in polynomial time. At any point in the learning process, their algorithm uses only the shortest words presented in the text so far to build a hypothesis pattern p_h; all other words in the text are ignored. For any i, if a symbol $\sigma \in \Sigma$ occurs in the i-th position in all the shortest words presented so far, the algorithm sets the i-th position of p_h to σ. Otherwise, the i-th position of p_h is a variable. If both position i and position j in p_h contain a variable and the symbol in position i of any shortest word w in the given text segment equals the symbol in position j of w, then the variables in positions i and j in p_h are chosen to be identical. For example, if the shortest words collected are

$$\begin{array}{llllll} a & a & a & bb & a & ab \\ a & a & b & bb & a & ab \\ a & b & a & bb & b & ab \end{array}$$

then the pattern p_h hypothesized by the algorithm is $ax_1x_2bbx_1ab$.

This algorithm does not work for erasing pattern languages; in fact, for $|\Sigma| \in \{2, 3, 4\}$, the class of all erasing pattern languages is not learnable at all [12], not even by an inefficient learning algorithm. Moreover, in general, neither erasing nor non-erasing typed pattern languages are learnable (cf. [9]), let alone polynomial-time learnable. The main contribution of this paper is the design of polynomial-time algorithms for learning interesting subclasses of (non-erasing) typed pattern languages, where certain structural properties of the potential target patterns are assumed. In order to define appropriate structural properties, we introduce the notion of *type witness* below.

Unfortunately, our above definition of polynomial-time learning is of no consequence. Pitt [11] showed that by repeating previous hypotheses until the input has grown long enough to afford computing a new hypothesis, a polynomial-time learner can be constructed from any learner successful according to Definition 3. But Lange and Wiehagen's algorithm $\mathcal{A}$ learns efficiently in a much stronger sense: every non-erasing pattern language L possesses a finite subset S_L of size polynomial in the length of any pattern p with $L = L(p)$ such that $\mathcal{A}$, on input of any text segment for L containing at least all words in S_L, returns a correct hypothesis for L. Such a subset S_L is what de la Higuera calls a characteristic set of polynomial size [8, 3], which, for learning from positive data, corresponds to a tell-tale set [2]. Deviating slightly from de la Higuera's model, we define:

Definition 4 (cf. de la Higuera [8]) *Let $\mathcal{L}$ be a class of languages and $\mathcal{H}$ a set of representations such that for each $H \in \mathcal{H}$ there is exactly one $L_H \in \mathcal{L}$ associated with H. Let $||H||$ denote the size of a representation $H \in \mathcal{H}$. $\mathcal{H}$ is*

polynomially learnable from positive data if there exist two polynomials χ and ξ and an algorithm $\mathcal{A}$ such that the following conditions are fulfilled.

1. *Given any finite set $S \subseteq \Sigma^*$, $\mathcal{A}$ returns some $H \in \mathcal{H}$ in time at most $\chi(|S|)$.*
2. *For any $H \in \mathcal{H}$ there is a finite set $S_H \subseteq L_H$ with $|S_H| \leq \xi(||H||)$ such that, on input of any set S with $S_H \subseteq S \subseteq L_H$, $\mathcal{A}$ returns some $H' \in \mathcal{H}$ with $L_{H'} = L_H$. S_H is called a characteristic set for H with respect to $\mathcal{H}$.*

De la Higuera additionally requires $\mathcal{A}$ to be consistent [4], i.e., to produce only hypotheses that contain the input set S. We drop this condition in Definition 4. Obviously, if $\mathcal{L}$ has an effectively enumerable representation set $\mathcal{H}$ that is polynomially learnable from positive data according to Definition 4, then $\mathcal{L}$ is also learnable in the limit from positive data. For example, the set of all patterns may serve as a representation set for the set of all pattern languages.

A characteristic set S_p for a pattern p, for each variable occurring in the target pattern p, contains two shortest words in which that variable is replaced differently and all other variables are replaced by some fixed $\sigma \in \Sigma$:

Proposition 5 *Let $p \in Pat_\Sigma$. Then there is a subset S_p of $L(p)$ with $|S_p| \leq 2|p|$ such that S_p is a characteristic set with respect to Pat_Σ for Lange and Wiehagen's algorithm.*

Proof. Let σ_1, $\sigma_2 \in \Sigma$, $\sigma_1 \neq \sigma_2$. Let S_p be a subset of $L(p)$ consisting of two words $v_1(x) = \theta_1^x(p)$ and $v_2(x) = \theta_2^x(p)$ per variable x occurring in p, where

$$\theta_\ell^x(x_i) = \begin{cases} \sigma_\ell\,, & \text{if } x_i = x\,, \\ \sigma_1\,, & \text{if } x_i \neq x\,, \end{cases}$$

for $\ell \in \{1, 2\}$. Clearly, $|S_p| \leq 2|p|$ and S_p contains only shortest words from $L(p)$.

Lange and Wiehagen's algorithm on input S, $S_p \subseteq S \subseteq L(p)$, can identify the positions in which variables reside (because in each such position in the words in S_p both σ_1 and σ_2 occur) and the pairs of positions in which two distinct variables reside (because in these pairs of positions at least one word in S_p has distinct entries). Thus S_p is a characteristic set for p with respect to Pat_Σ. □

3 Type Witnesses

The key to Lange and Wiehagen's algorithm is that a position in a shortest word of a non-erasing pattern language corresponds to the same position in the underlying pattern. For typed pattern languages, this is no longer the case, since the types might not contain any words of length one, and thus the shortest words of the language might be longer than the underlying pattern itself.

The shortest words in Lange and Wiehagen's case are those that are generated by substitutions replacing a variable with a word of length 1. Since, for $\Sigma \geq 2$, there are at least 2 distinct words of length 1, a learner can eventually figure out which positions of the shortest words in the given text correspond

to variables and which variable positions correspond to repeated variables. We generalize this idea by using *type witness* words instead of words of length 1. These words will allow the learner to recognize repetitions of variables. Just as there are 2 distinct words of length 1, we need 2 distinct type witness words per type. For the learner to be able to distinguish types, we require the type witness words to belong only to their corresponding type. This is formalized as follows.

Definition 6 *Let $\mathcal{T}$ be a set of subsets of Σ^+. Let $\omega_1, \omega_2 : \mathcal{T} \to \Sigma^+$ be mappings and $W = \bigcup_{t \in \mathcal{T}} \{\omega_1(t), \omega_2(t)\}$. (ω_1, ω_2) is a* type witness *for $\mathcal{T}$ if the following properties are fulfilled.*

1. *$\omega_1(t) \neq \omega_2(t)$ and $\{\omega_1(t), \omega_2(t)\} \subseteq t \setminus \bigcup_{t' \in \mathcal{T} \setminus \{t\}} t'$ for all $t \in \mathcal{T}$,*
2. *If $w \sqsubseteq \alpha \circ w_1 \circ \cdots \circ w_m$ for $w_1, \ldots, w_m \in \bigcup_{t \in \mathcal{T}} t$, $\alpha \in \Sigma^*$, and $w \in W$, then $w \sqsubseteq \alpha \circ w_1$.*
3. *If $w \in W$ and $w' \in \bigcup_{t \in \mathcal{T}} t$ then w' is not a proper suffix of w.*
4. *If $w, w' \in W$ then w' is not a proper prefix of w.*

Conditions 2 through 4 in this definition are included for technical reasons that will become obvious in the results proven below. In Lange and Wiehagen's case, there was only one type in $\mathcal{T}$, namely Σ^+. As type witnesses, two distinct words of length 1 in Σ^+ were used. This agrees with all conditions in Definition 6.

Remark: For some applications, Definition 6 may be inappropriate. However, there are other options to define type witnesses to yield sufficient conditions for learnability. We see Definition 6 as one non-trivial example of creating sufficient conditions for the design of non-trivial efficient learning algorithms for typed pattern languages.

Since the length of a witness word no longer has to be equal to 1, it might be impossible for the learner to identify which parts of a word in the text correspond to constants in the underlying target pattern. This obstacle is avoided if type witnesses are *short* in the sense of the following definition.

Definition 7 *Let $\mathcal{T}$ be a set of subsets of Σ^+. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$. (ω_1, ω_2) is a* short type witness *for $\mathcal{T}$ if, for all $t \in T$ and all $w \in t$,*

$$|\omega_1(t)| = |\omega_2(t)| \leq |w| \,.$$

The following example demonstrates how short type witnesses can be defined for Lange and Wiehagen's case of learning non-erasing pattern languages and for a text mining case in which variables can be either of type "positive integer" or of type "float" or of type "string."

Example 8 1. *Let $\Sigma = \{a, b\}$, $t = \Sigma^+$, $\mathcal{T} = \{t\}$. Let $\omega_1(t) = a$ and $\omega_2(t) = b$. Then (ω_1, ω_2) is a short type witness for $\mathcal{T}$.*
2. *Let $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ where $\Sigma_1 = \{0, 1, \ldots, 9\}$, $\Sigma_2 = \{`.`, `,`, SPACE\}$, $\Sigma_3 = \{a, b, \ldots, z\}$. Let $\mathcal{T} = \{t_1, t_2, t_3\}$, where $t_1 = \{w \in \Sigma_1^+ \mid w$ does not start with $0\}$, $t_2 = \{w \circ . \circ w' \mid w, w' \in \Sigma_1^+\}$, and $t_3 = \{w \circ \sigma \circ w' \mid w, w' \in \Sigma^*$, $\sigma \in \Sigma_3$, w does not contain any substring $\alpha.\sigma'$ with $\alpha \in t_1$ and $\sigma' \in \Sigma \setminus \{SPACE\}\} \setminus \{1, 2\}$. Define $(\omega_1(t_1), \omega_2(t_1)) = (1, 2)$, $(\omega_1(t_2), \omega_2(t_2)) = (3.0, 4.0)$, $(\omega_1(t_3), \omega_2(t_3)) = (a, b)$. Then (ω_1, ω_2) is a short type witness for $\mathcal{T}$.*

4 Polynomial-time Learning of Typed Pattern Languages

In what follows, polynomial learning of typed pattern languages implicitly uses a representation class in which each language is represented by a pattern p and a table assigning a type name to each variable occurring in p.

Our first theorem states that the class of all non-erasing typed pattern languages generated by terminal-free patterns is polynomially learnable from positive data, if the underlying type collection has a type witness. By terminal-free patterns we mean patterns consisting only of variables.

Theorem 9 *Let $\mathcal{T}$ be a finite set of decidable subsets of Σ^+ that has a type witness. Then the class of all non-erasing $\mathcal{T}$-typed pattern languages that are generated by terminal-free patterns is polynomially learnable from positive data.*

To prove this theorem, we first prove some helpful lemmas.

Lemma 10 *Let $\mathcal{T}$ be a set of subsets of Σ^+. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$, $W = \bigcup_{t\in\mathcal{T}}\{\omega_1(t), \omega_2(t)\}$. Let $\alpha, \alpha' \in W$ and $\beta_1, \ldots, \beta_m \in \bigcup_{t\in\mathcal{T}} t$ be such that $\alpha' \sqsubseteq \alpha \circ \beta_1 \circ \cdots \circ \beta_m$. Then $\alpha = \alpha'$. In particular, every W-decomposable word has a unique W-decomposition.*

Proof. Since $\alpha' \sqsubseteq \alpha \circ \beta_1 \circ \cdots \circ \beta_m$, where $\alpha, \beta_1, \ldots, \beta_m \in \bigcup_{t\in\mathcal{T}} t$, and $\alpha' \in W$, Definition 6.2 implies $\alpha' \sqsubseteq \alpha$. Definition 6.4 then yields $\alpha = \alpha'$. □

Lemma 10 helps to prove that the unique W-decomposition of a W-decomposable word $w \in \Sigma^+$ can be found in time polynomial in the length of w (for a fixed collection of polynomial-time decidable types with type witnesses).

Lemma 11 *Let $\mathcal{T}$ be a set of decidable subsets of Σ^+. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$, $W = \bigcup_{t\in\mathcal{T}}\{\omega_1(t), \omega_2(t)\}$. There is an algorithm that, given a $(\bigcup_{t\in\mathcal{T}} t)$-decomposable word $w \in \Sigma^+$, runs in time polynomial in $|w|$, returns the W-decomposition of w if w is W-decomposable and returns "no" otherwise.*

Proof. Because of Lemma 10, it suffices to traverse w in steps in the following way. In the i-th step ($i \geq 1$), one picks any word $w_i \in W$ such that $w_1 \circ \cdots \circ w_{i-1} \circ w_i \sqsubseteq w$. If, after some number m of steps, no more such word w_{m+1} is found, one returns $(w_1, \ldots, w_m)$ in case $w_1 \circ \cdots \circ w_m = w$ and returns "no" otherwise. Correctness and efficiency are evident. □

In order to study W-decomposable words generated by a terminal-free pattern, we introduce some more notation.

Definition 12 *Let $\mathcal{T}$ be a set of subsets of Σ^+. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$, $W = \bigcup_{t\in\mathcal{T}}\{\omega_1(t), \omega_2(t)\}$. For any $\mathcal{T}$-typed pattern (p, T), define*

$$L_W(p, T) = \{w \in \Sigma^+ \mid \exists \theta \in \Theta_{\Sigma,T}\ [\theta(p) = w \text{ and } \theta(x) \in W \text{ for all } x \in X]\}.$$

Lemma 13 *Let $\mathcal{T}$ be a set of subsets of Σ^+. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$, $W = \bigcup_{t\in\mathcal{T}}\{\omega_1(t), \omega_2(t)\}$. Let (p, T) be a terminal-free $\mathcal{T}$-typed pattern, $|p| = m$. Let $w \in L_W(p, T)$. Then the following statements are fulfilled.*

1. *w has a W-decomposition of length m.*
2. *If $w' \in L(p,T) \setminus L_W(p,T)$ and w' is W-decomposable, then the W-decomposition of w' has length greater than m.*

Proof sketch. Statement 1 is straightforward to show: since p is terminal-free, if $w \in L_W(p,T)$ then w is obviously W-decomposable into m words.

To prove Statement 2, let $p = y_1 \dots y_m$, where the y_i are (not necessarily distinct) variables. Let $\theta \in \Theta_{\Sigma,T}$ be such that $w' = \theta(p) = \theta(y_1) \circ \dots \circ \theta(y_m)$. Let $(v_1, \dots, v_n)$ be the unique W-decomposition of w'. We need to show that $n > m$. Assume $n \leq m$. The rest of the proof can be sketched as follows.

First, repeated application of Property 2 in Definition 6 yields $v_1 \circ \dots \circ v_i \sqsubseteq \theta(y_1) \circ \dots \circ \theta(y_i)$ for $1 \leq i \leq n$, which implies $n = m$ and $v_1 \circ \dots \circ v_n = \theta(y_1) \circ \dots \circ \theta(y_n)$.

Second, $v_1 \circ \dots \circ v_i \sqsubseteq \theta(y_1) \circ \dots \circ \theta(y_i)$ for $1 \leq i \leq n$ and $v_1 \circ \dots \circ v_n = \theta(y_1) \circ \dots \circ \theta(y_n)$ implies that either (i) $\theta(y_i) = v_i$ for all $i \in \{1, \dots, n\}$ or (ii) $\theta(y_i)$ is a proper suffix of v_i for some $i \in \{1, \dots, n\}$. (i) would contradict the choice of $w' \notin L_W(p,T)$, (ii) would contradict Property 3 in Definition 6. Since the assumption $n \leq m$ leads to a contradiction, we have $n > m$. □

Lemma 13 is crucial for our proof of Theorem 9. It further implies the existence of normal forms for terminal-free $\mathcal{T}$-typed patterns, for any type collection $\mathcal{T}$ that has a type witness. In particular, if two terminal-free $\mathcal{T}$-typed patterns generate the same language, they are equal modulo renaming of variables.

Proof of Theorem 9. Let $\mathcal{T}$ be a finite set of decidable sets of words. Let (ω_1, ω_2) be a type witness for $\mathcal{T}$ and let $W = \bigcup_{t \in \mathcal{T}} \{\omega_1(t), \omega_2(t)\}$. The required learner $\mathcal{A}$, given a set $S \subseteq \Sigma^+$, obeys the following algorithm.

Algorithm 1.

1. *Compute the set C of all W-decomposable words in S with the shortest W-decompositions. Let m be the length of such a shortest decomposition.*
2. *Initialize $p = y_1 \dots y_m$. For each $i \in \{1, \dots, m\}$, let $T(y_i)$ be the type $t \in \mathcal{T}$ for which the i-th word in the W-decomposition of any $z \in C$ is contained in $\{\omega_1(t), \omega_2(t)\}$.*
3. *For each $i \in \{1, \dots, m\}$ and each $j \in \{i, \dots, m\}$, if $T(y_i) = T(y_j)$ and, for all $z \in S$, the i-th word equals the j-th word in the W-decomposition of z, then replace y_i and y_j in p by x_i.*
4. *Replace all remaining y_i in P by x_i. $T(x_i) = T(y_i)$ for each $i \in \{1, \dots, m\}$.*
5. *Return (p,T).*

Step 1 can be done in polynomial time according to Lemma 11. Step 2 can be executed since $\mathcal{T}$ is finite and because of Definition 6. Steps 3 and 4 can obviously be executed in polynomial time.

Assume the target pattern is (p^*, T^*). Note that $L_W(p^*, T^*)$ is finite. Let S^* be a subset of $L_W(p^*, T^*)$ consisting of two words $v_1(x) = \theta_1^x(p^*)$ and $v_2(x) = \theta_2^x(p^*)$ per variable x occurring in p^* where, for $\ell \in \{1, 2\}$,

$$\theta_\ell^x(x_i) = \begin{cases} \omega_\ell(T^*(x)), & \text{if } x_i = x, \\ \omega_1(T^*(x_i)), & \text{if } x_i \neq x. \end{cases}$$

If $S^* \subseteq S$, it is not hard to verify, using Lemma 13, that there is a terminal-free pattern (p, T) with $L(p^*, T^*) = L(p, T)$, such that $\mathcal{A}$ on input S returns (p, T).

Finally, note that the size of the characteristic set S^* defined above is polynomial in the length of p^*. □

Theorem 9 can be generalized to the case of all non-erasing typed pattern languages if we require the underlying type witnesses to be short.

Theorem 14 *Let $\mathcal{T}$ be a finite set of decidable subsets of Σ^+ that has a short type witness. Then the class of all non-erasing $\mathcal{T}$-typed pattern languages is polynomially learnable from positive data.*

Proof sketch. The proof is based on the easily verifiable fact that, for any target pattern (p, T), the set $L_W(p, T)$ is contained in the set of shortest words in $L(p, T)$, which is obviously finite. The required learner $\mathcal{A}$, given a finite set $S \subseteq \Sigma^+$, obeys the following algorithm.

Algorithm 2.

1. *Compute the set C of all shortest words in S, where m is their length.*
2. *Initialize $p = y_1 \dots y_m$. For each $i \in \{1, \dots, m\}$, if there is a $\sigma \in \Sigma$ such that the i-th position of each word in C is σ, change the i-th position of p to σ and mark the i-th position in p "constant."*
3. *Remove all words from C for which any of the maximal substrings corresponding to non-constant positions in p are not W-decomposable.*
4. *Apply Algorithm 1 to the left-to-right concatenations of maximal substrings of words in C corresponding to non-constant positions in p. Replace the non-constant parts in p with the corresponding parts of the pattern obtained from Algorithm 1.*
5. *Return (p, T).*

It is not hard to verify that this algorithm runs in polynomial time.

Assume the non-erasing target pattern is (p^*, T^*). Let S^* be the subset of all shortest words in $L(p^*, T^*)$ that is defined by analogy with the set S^* used in the proof of Theorem 9. The size of S^* is polynomial in $|p^*|$. If $S^* \subseteq S$, one can prove that there is a pattern (p, T) with $L(p^*, T^*) = L(p, T)$, such that $\mathcal{A}$ on input S returns (p, T). Details are omitted due to space constraints. □

5 Extensions of the Presented Results

This section addresses several possible ways of strengthening the above results.

5.1 Learning Typed Pattern Languages over Infinitely Many Types

For some applications, it might be required to model the learning problem using an infinite set of types, i.e., an infinite set $\mathcal{T}$. Our learnability results immediately transfer to this case if the type witness is equipped with two appropriate mappings that (i) assign type witness words to a type index number and (ii) find a type index for each witness word.

Theorem 15 *Let $\mathcal{T}$ be a set of decidable subsets of Σ^+ that has a short type witness (ω_1, ω_2) fulfilling the following property.*

> *There is an efficiently computable one-to-one mapping* $\text{witness} : \mathbb{N} \to (\Sigma^+)^2$ *and an efficiently computable mapping* $\text{typeIndex}: \bigcup_{t \in \mathcal{T}} \{\omega_1(t), \omega_2(t)\} \to \mathbb{N}$ *such that, for all $t \in \mathcal{T}$ and $w \in \{\omega_1(t), \omega_2(t)\}$,* $\text{witness}(\text{typeIndex}(w)) = (\omega_1(t), \omega_2(t))$.

Then the class of all non-erasing $\mathcal{T}$-typed pattern languages is polynomially learnable from positive data.

We get an analogous generalization of Theorem 9 for terminal-free patterns when dropping the shortness constraint on the type witness.

5.2 Consistent Learning of Typed Pattern Languages

Algorithms 1 and 2 presented above, just like Lange and Wiehagen's algorithm, ignore most of the input words and build a hypothesis based just on a special subset of the given words. A side-effect is that the hypotheses returned on any set not containing a characteristic set may be inconsistent with the given data.

One way of making our learning algorithms consistent would be to test for each input word whether it is generated by the pattern that the algorithm would normally return. If not, then (x_1, T) with $T(x_1) = \Sigma^+$ could be returned instead. However, there is no known polynomial-time algorithm deciding membership for (typed) pattern languages, even if all types are polynomial-time decidable: the membership problem for non-erasing pattern languages is NP-complete [1].[5]

However, the membership problem for non-erasing pattern languages is in P, when restricted to patterns containing a bounded number of distinct variables [1]. This was generalized to typed pattern languages with finitely many polynomial-time decidable types [6]. A similar result holds for all typed pattern languages with finitely many polynomial-time decidable types when bounding the length of words from above [6]. These results imply the following theorem.

Theorem 16 *Let $\mathcal{T}$ be a finite set of polynomial-time decidable subsets of Σ^+ that has a short type witness (ω_1, ω_2). Let $k \in \mathbb{N}$. Then the following holds.*

1. *The class of all non-erasing $\mathcal{T}$-typed pattern languages that are generated by typed patterns containing at most k distinct variables is polynomially learnable from positive data by a consistent learning algorithm.*
2. *The class of all non-erasing $\mathcal{T}$-typed pattern languages with shortest words of length at most k is polynomially learnable from positive data by a consistent learning algorithm, if the input to the learning algorithm is restricted to sets of words of length at most $c \cdot k$ for some constant c.*

A detailed proof is omitted due to space constraints. Again we get an analogous generalization of Theorem 9 for terminal-free patterns when dropping the shortness constraint on the type witness.

[5] The membership problem for (non-erasing) pattern languages is to decide, given a pattern p and a word w, whether or not $w \in L(p)$.

6 Conclusion

We generalized the method of Lange and Wiehagen's polynomial-time algorithm for learning pattern languages to be applicable to typed pattern languages. This required the introduction of type witnesses, structurally restricting the class of typed pattern languages for which we can prove polynomial-time learnability. We established a connection to the study of characteristic sets, though in general without being able to guarantee consistent learning algorithms. Our preliminary considerations in Section 5.2 are a first step towards future work on the problem of achieving consistency without sacrificing runtime efficiency.

Type witnesses are likely to be of further use for the study of efficient algorithms for the identification of (typed or relational) patterns. In particular, specific application scenarios will allow for modeling specific sets of type witnesses; in all such cases the algorithms we proposed can be a basis for efficient solutions to machine learning and data mining problems.

References

1. Angluin, D.: Finding patterns common to a set of strings. J. Comput. Syst. Sci. 21, 46–62 (1980)
2. Angluin, D.: Inductive inference of formal languages from positive data. Inform. Control 45, 117–135 (1980)
3. Angluin, D.: Inference of reversible languages. J. ACM 29, 741–765 (1982)
4. Blum, L., Blum, M.: Toward a mathematical theory of inductive inference. Inform. Control 28, 125–155 (1975)
5. Case, J., Kötzing, T.: Difficulties in forcing fairness of polynomial time inductive inference. In: ALT. pp. 263–277 (2009)
6. Geilke, M., Zilles, S.: Learning relational patterns. In: ALT. pp. 84–98 (2011)
7. Gold, E.: Language identification in the limit. Inform. Control 10, 447–474 (1967)
8. de la Higuera, C.: Characteristic sets for polynominal grammatical inference. In: ICGI. pp. 59–71 (1996)
9. Koshiba, T.: Typed pattern languages and their learnability. In: EuroCOLT. pp. 367–379 (1995)
10. Lange, S., Wiehagen, R.: Polynomial-time inference of arbitrary pattern languages. New Generation Comput. 8, 361–370 (1991)
11. Pitt, L.: Inductive inference, DFAs, and computational complexity. In: AII. pp. 18–44 (1989)
12. Reidenbach, D.: Discontinuities in pattern inference. Theor. Comput. Sci. 397, 166–193 (2008)
13. Shinohara, T.: Polynomial time inference of extended regular pattern languages. In: RIMS Symposium on Software Science and Engineering. pp. 115–127 (1982)
14. Wright, K.: Inductive identification of pattern languages with restricted substitutions. In: COLT. pp. 111–121 (1990)