Pattern Learning Revisited

Frank Stephan[⋆,1], Ryo Yoshinaka[2] and Thomas Zeugmann[2]

[1] Department of Mathematics and Department Computer Science
National University of Singapore, Singapore 117417, Republic of Singapore
fstephan@comp.nus.edu.sg
[2] Division of Computer Science, Graduate School of Information Science and Technology
Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan
ry@ist.hokudai.ac.jp and thomas@ist.hokudai.ac.jp

Abstract. Angluin showed that there is a consistent and conservative learner for the class of non-erasing pattern language; however, most of these learners are *NP*-hard. In the current work, the complexity of consistent polynomial time learners for the class of non-erasing pattern languages is revisited, with the goal to close one gap left by Angluin, namely the question on what happens if the learner is not required to output each time a consistent pattern of maximum possible length. It is shown that consistent learners are non-uniformly $W[1]$-hard inside the fixed-parameter hierarchy of Downey and Fellows (1999), and that there is also a $W[1]$-complete such learner. Only when one requires that the learner is in addition both, conservative and class-preserving, then one can show that the learning task is *NP*-hard for certain alphabet-sizes.

1 Introduction

Angluin [1] introduced pattern languages as an example for an interesting class that is learnable in the limit from positive data. A *pattern* is a finite string over a finite alphabet Σ and a countably infinite set X of variables, where $\Sigma \cap X \neq \emptyset$. The *language* generated by the pattern is the set of strings which can be obtained by substituting each variable in the pattern by a non-empty string over Σ.

Note that further investigations also studied the use of empty strings as possible values of the variables, resulting in the notion of the *erasing pattern languages* (cf., e.g., Shinohara [15], Reidenbach [13, 12], and the references therein). However, as the erasing pattern languages are not learnable from positive data [13, 12], the present work follows Angluin's [1] approach and requires that every variable takes a non-empty string as a value. Such pattern languages are called *non-erasing pattern languages*.

Next we shortly describe the learning model considered in this paper, i.e., learning in the limit from positive data (cf. Gold [8]). Let L be any target language. A *text* of a language is an infinite sequence of words that eventually contains all words of L and possibly a special symbol called *pause symbol* which does not belong to any of the languages to be learned. To see that the pause symbol is necessary we may just look at the case where a target language is the empty

⋆ Supported in part by NUS grant numbers R146-000-114-112 and R252-000-420-112.

one. So, a text is a *positive presentation* of the target, and therefore we talk about learnability from positive data.

An algorithmic learner takes as input growing initial segments of a text t and computes and outputs an hypothesis about the target language. The set $\mathcal{H}$ of all admissible hypotheses is called *hypothesis space*. The sequence of hypotheses has to converge to a hypothesis correctly describing the target language. That is, after some point the sequence of hypotheses stabilizes to an accurate hypothesis. Then we say that the learner learns the target language from text t. If there is a learner that learns the target language L from all texts for it then L is said to be *learnable in the limit from positive data* with respect to $\mathcal{H}$. A class of languages is said to be *learnable in the limit from positive data* if there is one learner that learns all the languages from the class with respect to $\mathcal{H}$.

Furthermore, a learner is *consistent* if it exclusively outputs hypotheses that contain all the words seen so far by the learner. A learner is *conservative* if it exclusively performs so-called justified mind changes. That is, whenever two consecutively hypotheses, say i and j (computed on input σ and τ, respectively) output by the learner differ, then the content of τ is not a subset of the language described by j. For more information we refer the reader to [18, 11], and the references therein.

Already Angluin [1, 2] showed the class of all non-erasing pattern languages to be learnable in the limit from positive data. Note that an easy modification of Angluin's [1, 2] algorithm for learning the pattern languages is consistent and conservative (cf., e.g., [18]) but not polynomial-time computable if $P \neq NP$ (cf. Angluin [1], Theorem 3.6).

Lange and Wiehagen [10] sacrificed the consistency requirement and provided a polynomial time learner. Zeugmann [17] investigated the learning properties of Lange and Wiehagen's algorithm from a statistical perspective. Other investigations on various approaches to learn pattern languages followed [7, 9, 14, 16].

Angluin [1] showed that it is *NP*-complete to test whether a pattern generates a certain string, i.e., the membership problem for the non-erasing pattern languages is *NP*-complete. She concluded that there is no consistent pattern learner which outputs consistent patterns of maximum possible length unless $P = NP$. The present work returns to Angluin's [1] question and asks what happens if the condition on the maximum possible length of the patterns is sacrificed. It is shown that the following holds:

- The existence of a polynomial-time consistent learner for the class of non-erasing pattern languages is equivalent to the condition that *CLIQUE* is non-uniformly fixed-parameter tractable, that is, $W[1] \subseteq FPT^n$;
- The existence of a polynomial-time consistent and conservative learner using a hypothesis space with some minimum decidability requirements (one can decide whether $H_i = L(\pi)$ for an index i and pattern π) implies that *CLIQUE* is uniformly fixed-parameter tractable, that is, $W[1] \subseteq FPT$.

Note that there is an oracle A relative to which $FPT = W[Poly]$ and $P \neq NP$. This result gives some evidence that one cannot show that the existence of a polynomial time consistent and class-preserving pattern language learner would imply $P = NP$.

Angluin [1] had already shown that the word problem of the pattern languages is in $W[poly]$ which is the highest class of the W-hierarchy of Downey and Fellows [6]. Angluin did not locate the exact level of that hierarchy as the corresponding framework was not there when she investigated the problem [1]. Case and Kötzing [4, 5] also provided polynomial-time consistent and conservative class-comprising learning algorithms for classes where the membership-problem is uniformly polynomial-time computable and some mild other conditions hold; in the current work, it is easier to go directly back to the learning algorithm of Lange and Wiehagen [10] in place of the more recent work of Case and Kötzing, but the algorithm of Theorem 4 below could also be obtained by their methods.

For getting better learnability properties, the learnability of subclasses of the pattern languages was studied. For example, patterns generated by only a few variables were considered [7, 9]. Another approach is due to Shinohara [15]: he introduced the concept of regular patterns and regular pattern languages, where each variable appears at most once in the pattern. In these subcases, polynomial time learners can be achieved. The present work returns to the more general task of learning all non-erasing pattern languages.

2 The Complexity of the Pattern-Membership Problem

Angluin [1] showed that the word-membership problem of the pattern languages is NP-complete. The result, however, was not checking the membership-problem for a fixed pattern but showed only that the uniform membership-problem $\{(\pi, w) : w \in L(\pi)\}$ is NP-complete. One might now ask what happens if the underlying numbering of all the pattern languages does not follow the default-numbering but is given in some other way. This would indeed weaken the requirements on a learning algorithm, as certain information could be coded into the hypothesis space and the learner could therefore gain power by using that coding and delaying convergence.

In order to deal with these phenomena, it is adequate to look at the parameterized complexity as introduced by Downey and Fellows [6]. They defined that a parameterized set A is in FPT iff there is a recursive function f and a polynomial p such that the question whether $(k, x) \in A$ can be decided in $p(f(k) + |x|)$ time.

Downey and Fellows [6] defined that a parameterized set A is *fixed-parameter many-one reducible* to a parameterized set B (written $A \leq_m^n B$) iff there is a polynomial p such that for each parameter k there is a parameter k', a factor d_k and a reduction ψ_k such that ψ_k translates every $\langle x, k\rangle$ in time $d_k \cdot p(k + |x|)$ into $\langle x', k'\rangle$ with $\langle x, k\rangle \in A \Leftrightarrow \langle x', k'\rangle \in B$.

Furthermore, A is *strongly uniformly fixed-parameter many-one reducible* to B (written $A \leq_m^s B$) iff there is a recursive function which computes k', the factor d_k and a program for ψ_k from k.

Note that Downey and Fellow [6] also relaxed the class FPT to a non-uniform version FPT^n which contains all parameterized sets A for which there exists a set $B \in FPT$ with $A \leq_m^n B$.

Let PMO denote the membership problem of pattern languages with the parameter k being the number of occurrences of variables in the pattern on hand. For example, consider a pattern π such that one variable occurs twice in π and two other variables occur just one time each in π, then the parameter is 4. For *CLIQUE* the parameter is the size of the clique requested. $W[1]$ is a complexity class which can be characterized as the those sets which are strongly uniformly fixed-parameter many-one reducible to $CLIQUE$.

Theorem 1. *The problem PMO is $W[1]$-complete for strongly uniform fixed-parameter many-one reducibility. In particular, PMO $\equiv^s_m$ CLIQUE.*

Proof. It is sufficient to show the second statement; the first one then follows from the fact that *CLIQUE* is a complete problem for $W[1]$.

First, we show that *CLIQUE* $\leq^s_m$ *PMO*.

In order to show this, let k be the target size of the clique.

One makes the following reduction which produces for each k a pattern π_k and for each graph (V, E) a word $w_{V,E}$ of length polynomial in $|V| + |E|$ such that $w_{V,E} \in L(\pi_k)$ iff (V, E) has a k-clique. In the following, the alphabet $\{0, 1, 2, 3, 4\}$ is used for better readability, but one could do the replacements $0 \to 00000010, 1 \to 00000110, 2 \to 00001110, 3 \to 00011110, 4 \to 00111110$ in order to code everything using the binary alphabet. The pattern π_k is

$$\Sigma^+ 2x_1 2\Sigma^+ 2x_2 2\Sigma^+ 2x_3 2\ \ldots\ 2x_k 2\ 4\ \Sigma^+ 3x_1x_2 3\Sigma^+ 3x_1x_3 3\Sigma^+\ \ldots\ 3x_1x_k 3\Sigma^+$$
$$\ldots\ \Sigma^+ 3x_2x_3 3\Sigma^+\ \ldots\ 3x_2x_k 3\Sigma^+\ \ldots\ \Sigma^+ 3x_{k-1}x_k 3\Sigma^+$$

where Σ^+ stands for variables occurring only once. The string $w_{V,E}$ has one 4 separating the first and second part of the coding and uses binary coding for each vertex (say as $00, 01, 10, 11$) which in the first entry are surrounded by 222 before, between and after the variables and the follows a list of all edges in E in lexicographically ascending order where again each pair of vertices is surrounded by 333. So in the case of $V = \{00, 01, 10, 11\}$ (the elements in V have all to have the same length) and $E = \{(00, 01), (00, 10), (01, 10), (10, 11)\}$ the string $w_{V,E,}$ would be

222 00 222 01 222 10 222 11 222 4 333 0001 333 0010 333 0110 333 1011 333

without the spaces which are placed for easier readability. The corresponding pattern π_3 is

$$\Sigma^+ 2x_1 2\Sigma^+ 2x_2 2\Sigma^+ 2x_3 2\Sigma^+ 4\Sigma^+ 3x_1x_2 3\Sigma^+ 3x_1x_3 3\Sigma^+ 3x_2x_3 3\Sigma^+$$

and now $w_{V,E}$ is in $L(\pi_3)$ iff x_1, x_2, x_3 take in lexicographic order the values the nodes forming a 3-clique. Note that in a solution x_1, x_2, x_3 cannot have the symbol 2 or 3 as they appear before and after the 4 while the symbol 2 only appears before and the symbol 3 only appears after the 4. Furthermore, x_1, x_2, x_3 must take the binary values of nodes as they appear between two 2 and are not empty; so they single out in the third part nodes for which $3x_1x_23$, $3x_1x_3$ and $3x_2x_33$ are substrings of the word $w_{V,E}$, hence all three edges (x_1, x_2), (x_1, x_3) and (x_2, x_3) are in E and occur in the string in the given ordering by lexicographic order. Therefore the membership problem for $w_{V,E} \in L(\pi_3)$ is positive only if (V, E) contains a 3-clique.

On the other hand, it is easy to see that by selecting for x_1, x_2, x_3 the values of the nodes of the 3-clique then the membership problem is positive. This verification generalizes to the question whether $w_{V,E} \in L(\pi_k)$. Note that the string $w_{V,E}$ has at most length $O((|V| + |E|) \cdot \log(|V|))$

which is polynomial in $|V|+|E|$. Furthermore, the pattern π_k contains k^2 occurrences of variables $x_1, x_2, \ldots, x_k$ and $k(k+1)/2+2$ occurrences of Σ^+ representing variables which occur only once; hence there are $k(3k+1)/2+1$ occurrences of variables in π_k and the length is at most twice this length, as constants appear only between two variables. So the length of an instance query "Is $w_{V,E} \in L(\pi_k)$?" is bounded by a polynomial in $k+|V|+|E|$ and the translations maps the parameter of the clique to a corresponding parameter of the pattern-membership problem independently of V and E. So the reduction witnesses $\mathit{CLIQUE} \leq_m^s \mathit{PMO}$.

For the converse direction, assume that a pattern π with k occurrences of variables is given and let w be any word. Now a graph $(V_{\pi,w}, E_{\pi,w})$ is constructed such that this graph has a k-clique iff $w \in L(\pi)$.

The set $V_{\pi,w}$ is that of all triples (i, j, ℓ), where i, j refer to positions in w and ℓ to the occurrence in π. For a pair of nodes, let (i, j, ℓ) and (i', j', ℓ') denote the two triples such that $\ell \leq \ell'$. Now the pair is in the set $E_{\pi,w}$ iff the following conditions are satisfied:

- $\ell < \ell'$;
- $j < i'$;
- if $\ell' = \ell + 1$ then the part between the positions j and i' in w matches the constants in π between the occurrences of variables at ℓ and ℓ';
- if $\ell = 1$ then the initial part of w before position i matches the constants in π before the first variable occurrence;
- if $\ell' = k$ then the final part of w after position j' matches the constants in π after the last variable occurrence;
- if at ℓ and ℓ' the same variable occurs in π then w at the positions from i to j equals w at the positions from i' to j'.

Now $(V_{\pi,w}, E_{\pi,w})$ has a k-clique iff $w \in L(\pi)$. Furthermore, the reduction is a FPT-reduction: $|V_{\pi,w}| \leq |\pi| \cdot |w|^2$ and $(V_{\pi,w}, E_{\pi,w})$ is constructed from π and w in time polynomial in $|\pi| + |w|$. This proves that $\mathit{PMO} \leq_m^s \mathit{CLIQUE}$. $\square$

3 Learning Theory

In this section we shall apply Theorem 1 to learning theory. Our goal is to show the following. The class of all pattern languages is consistently polynomial time learnable iff $W[1] \subseteq \mathit{FPT}^n$.

Before we can come to the results, we have to introduce some notions from learning theory. As already explained in the introduction, the source of information we wish to learn from are texts. A *text* is an infinite sequence that eventually contains all words of the language to be learned, and which besides the members of the target language can contain some pause symbols but no non-members. Further more, the order of the members in the text is arbitrary.

The learner is successively fed incrementally growing initial segments of text. From the initial segment that contains precisely the first n members of the text the learner produces the hypothesis i_n. The hypotheses are interpreted with respect to a chosen *hypothesis space* $\{H_i : i \in I\}$. The sequence $(i_n)_{n \in \mathbb{N}}$ (here $\mathbb{N}$ denotes the set of all natural numbers) of all created hypotheses has to *converge* to an $i \in I$ which satisfies $H_i = L$. Here by converge we mean that $i_n = i$ for all but finitely many $n \in \mathbb{N}$.

A learner learns a target language L from text if it learns L from all text for it. Finally, a learner learns a class C with respect to a hypothesis space $\{H_i : i \in I\}$ from text iff it learns every language from C from text.

This model of learning is called *learning in the limit* from text, or synonymously referred to as *explanatory learning* from text, and it was introduced by Gold [8]. Since we deal here exclusively with learning from text, in the following we refer to it just as *learning in the limit*.

Angluin [1, 2] introduced additional requirements for the learning process which are important in the present context, i.e., consistency and conservativeness. A learner M is *consistent* iff for every input σ it outputs a hypothesis i such that the range of σ is contained in H_i; if such an i does not exist the learner is permitted to signal this by outputting a special no-conjecture symbol.

A learner M is *conservative* iff for every two different subsequent hypotheses i based on σ and j based on $\sigma \diamond \tau$ it holds that there is some data item $x \in \mathit{range}(\sigma \diamond \tau)$ which is not contained in H_i. That is, a conservative learner replaces only inconsistent hypotheses and keeps a consistent hypothesis as long as only data from it appear. In this context, the notation $\sigma \diamond \tau$ denotes the concatenation of the sequences σ and τ. Note that the above definition implies that a conservative learner never overgeneralizes, that is, it never happens that M outputs on $\mathit{range}(\sigma)$ a hypothesis for a language L while there is a language L' in the class with $\mathit{range}\,\sigma) \subseteq L' \subset L$.

A hypothesis space $\{H_i : i \in I\}$ is *class-preserving* (with respect to the class C to be learned) iff $\{H_i : i \in I\} = C$. The hypothesis space $\{H_i : i \in I\}$ is called *class-comprising* iff $\{H_i : i \in I\} \supseteq C$. Note that every hypothesis space is required to be class-comprising; ideally it should also be class-preserving, although this restricts the learnability in certain cases (cf., e.g., [18]).

In order to relate our results to the ones previously obtained by Angluin [1, 2], we shortly recall her approach here. The key ingredient is the notion of a descriptive pattern. By *sample* we mean any finite set of words over the underlying alphabet Σ. A pattern π is said to be *descriptive* of a sample S iff $S \subseteq L(\pi)$ and for every pattern τ such that $S \subseteq L(\tau)$, the condition that $L(\tau)$ is not a proper subset of $L(\pi)$ must be satisfied. She then showed that every learner that outputs on every input a descriptive pattern of the range of the input learns the class of all non-erasing pattern languages from text. Finally she showed that there is an algorithm computing on input any sample S a pattern that is descriptive of S.

Our first result in this section shows that this is essentially the only way to obtain a consistent and conservative learner for the class of all non-erasing pattern languages if we require learnability with respect to a class-preserving hypothesis space.

Theorem 2. *Let M be a consistent and conservative learner that learns the class of all non-erasing pattern languages with respect to a class-preserving hypothesis space. Then M must output in every step a hypothesis that is descriptive for the content of the text seen so far.*

Proof. Assume $\mathcal{H} = \{H_i : i \in I\}$ to be a class-preserving hypothesis space for the class of all non-erasing pattern languages. By $L(H_i)$ we denote the (pattern) language described by H_i. Let

π be the target pattern and let t be any text for $L(\pi)$. Furthermore, let $H_\sigma = M(\sigma)$ be the hypothesis output by M when fed σ, where σ is any initial segment of t. Since M is consistent, we already know that $\mathit{range}(\sigma) \subseteq L(H_\sigma)$.

In order to verify that H_σ is descriptive of $\mathit{range}(\sigma)$ we have to show that there is no hypothesis $H_j \in \mathcal{H}$ such that $\mathit{range}(\sigma) \subseteq L(H_j) \subset L(H_\sigma)$.

Suppose the converse. Then σ is also an initial segment of some text $\hat{t}$ for $L(H_j)$. Since $L(H_j) \subset L(H_\sigma)$, we may conclude that M, when fed σ, did overgeneralize, a contradiction. □

The results presented below will show the following. Under the condition that $W[1] = \mathit{FPT}$ and $P \neq \mathit{NP}$ the class of all non-erasing pattern languages over a countable infinite alphabet is consistently and conservatively learnable using some class-comprising hypothesis space but not using a class-preserving hypothesis space.

An important tool in learning theory is that of the locking sequence; here a *locking-sequence* σ of a learner M on a language L is a finite sequence of members from L such that for every further finite sequence τ of members from L it holds that $M(\sigma \diamond \tau) = M(\sigma)$ and furthermore $M(\sigma)$ is an index for the set L. Blum and Blum [3] showed that whenever M learns a language L then M has a locking sequence on L. Locking-sequences can normally be only determined in the limit; however, if the learner is consistent and conservative and the hypothesis space satisfies certain decidability requirements then one can find the locking-sequences also using a recursive function.

The complexity analysis often works with decision problems instead of functions (as learners are). For the purpose of this work it is sufficient to look at the output-equality problem of the learner, that is, given a learner M one shows that the set $\{(\sigma, \tau) : M(\sigma) = M(\tau)\}$ is hard for the corresponding complexity class.

Theorem 3. *The class of all patterns has a consistent polynomial-time learner if and only if* $W[1] \subseteq \mathit{FPT}^n$.

Proof. Assume that the class of non-erasing pattern languages has a consistent polynomial-time learner M and let π_k be as in Theorem 1. There is a locking-sequence σ_k for every language $L(\pi_k)$ and it holds now that (V, E) has a k-clique iff $M(\sigma_k \diamond w_{V,E}) = M(\sigma_k)$. This is now an FPT^n-algorithm to decide whether (V, E) has a k-clique; it is polynomial in $|\sigma_k| + |V| + |E|$ and as σ_k depends only on k and not on V, E the algorithm is an FPT^n-algorithm. It follows that $W[1] \subseteq \mathit{FPT}^n$.

On the other hand, if $W[1] \subseteq \mathit{FPT}^n$ then there is a uniformly polynomial time computable class $\{L_i : i \in I\}$ containing all non-erasing pattern languages. Now one can define the following consistent learner for this class: On input σ, the learner selects the least index i for which there is a pattern π satisfying the following conditions:

- $\mathit{range}(\sigma) \subseteq L_i$;
- $|i| \leq \log(|\sigma|)$ and $|\pi| \leq \log(|\sigma|)$;
- $y \in \mathit{range}(\sigma) \Leftrightarrow y \in L(\pi)$ for all y with $|y| \leq |\pi|$;
- $y \in L_i \Leftrightarrow y \in L(\pi)$ for all y with $|y| \leq \log(|\sigma|)$.

If no index i qualifies for these conditions then the learner outputs the smallest index of Σ^+ which is generated by the single-variable pattern. Furthermore, there is the special case that an index for the language of the empty word is output iff the empty word is seen on the input. Note that no non-erasing pattern language contains the empty word and a nonempty word, hence the learner does not need to make a conjecture in that case.

It is easy to see that the above learner is in polynomial time and consistent. Furthermore, once all the shortest words in the pattern language are seen and the input is sufficiently long, the π in the above algorithm is correct and kept forever; the corresponding index i matching π is found in the limit. In other words, the algorithm given here is just a consistent variant of the algorithm of Lange and Wiehagen [10]; for working the variant just requires that the pattern languages are in FPT^n. □

The next result shows that for consistent and conservative learning with a minimum decidability requirement on the hypothesis space, polynomial-time learnability becomes linked to the more restrictive condition $W[1] \subseteq FPT$.

Theorem 4. *$W[1] \subseteq FPT$ if and only if the class of non-erasing pattern languages has a consistent and conservative polynomial-time learner which uses a hypothesis space $\{H_i : i \in I\}$ such that $\{(\pi, i) : L(\pi) = H_i\}$ is decidable.*

Proof. In the case that $W[1] \subseteq FPT$, there is a recursive function f and a uniformly polynomial-time computable hypothesis space $\{H_i : i \in I\}$ such that $H_{f(\pi)} = L(\pi)$ for all non-erasing patterns π. Furthermore it is decidable in polynomial time whether an index i is of the form $f(\pi)$ or not. Without loss of generality the other indices form a canonical enumeration of the finite sets. Now the learner outputs on input σ a hypothesis of the form $H_{f(\pi)}$ iff $range(\sigma) \subseteq H_{f(\pi)}$ and every shortest string in $L(\pi)$ has occurred in $range(\sigma)$ and $f(\pi)$ has been computed in time polynomial in the size of σ. In all other cases, the learner outputs an index i with $H_i = range(\sigma)$.

It follows again from ideas of Lange and Wiehagen [10] that this algorithm learns all non-erasing pattern languages. Furthermore, it is polynomial-time, consistent and conservative.

Now assume that there is a consistent and conservative polynomial-time learner M using a uniformly polynomial-time computable hypothesis space $\{H_i : i \in I\}$ for which one can decide whether $H_i = L(\pi)$. Then one can find effectively for each k the locking σ_k for $L(\pi_k)$ where π_k is as in Theorem 1. Now, as the learner is conservative, it would not give up a conjecture consistent with all data seen so far. Hence one has again that

$$(k, V, E) \in \mathit{CLIQUE} \Leftrightarrow M(\sigma_k \diamond w_{V,E}) = M(\sigma_k)$$

and this time the condition shows that $\mathit{CLIQUE} \in FPT$ as the mapping $k \mapsto \sigma_k$ is recursive and the test whether $M(\sigma_k \diamond w_{V,E}) = M(\sigma_k)$ is computable in time polynomial in $|\sigma_k| + |V| + |E|$. Hence $W[1] \subseteq FPT$. □

In the following, an infinite alphabet Σ is used, for example Σ could be the set of all natural numbers. It is shown that in this case it is NP-hard to make a consistent and conservative learner using a class-preserving hypothesis space. This shows that the weakening from class-preserving to class-comprising hypotheses spaces is indeed an important ingredient for the general polynomial-time learners obtained by Case and Kötzing [4, 5] for many uniformly polynomial-time decidable classes.

Theorem 5. *Let $\Sigma = \{0, 1, 2, \ldots\}$. Now $P = NP$ if and only if there is a consistent and conservative class-preserving polynomial-time learner of the class of all non-erasing pattern languages over Σ.*

Proof. The sufficiency of the condition $P = NP$ can be straightforwardly obtained by combining the techniques of Angluin [1] and the ones outlined in Shinohara and Arikawa [16].

For the necessity, she added the condition that the learner should output patterns of maximum possible length that are descriptive of the range of the text seen so far. This cannot be enforced in general and the fact that the inclusion-problem for pattern languages is undecidable hinders to produce polynomially sized characteristic samples for unknown learners in general. In the case of an infinite alphabet one can, however, construct a pattern language where every consistent and conservative learner outputs the correct pattern after having seen two examples. The idea is to code the following *NP*-complete problem *RSAT* (restricted *SAT*):

- An *RSAT*-instance consists of a list of clauses of non-negated variables;
- Every clause in an *RSAT*-instance contains at least 2 and at most 3 variables;
- Each variable occurs in at most three clauses;
- Each pair of variables occurs at most in one clause together;
- An *RSAT* instance is satisfiable iff there is an assignment to the variables which has in every clause exactly one true variable.

For example, $(x_1+x_2+x_3, x_1+x_4, x_2+x_4, x_3+x_4)$ is an unsatisfiable *RSAT*-instance as exactly one x_i of the x_1, x_2, x_3 is true and forces x_4 to be false as $x_i + x_4$ is a clause in the instance; then there are also clauses $x_j + x_4$ with $x_j \in \{x_1, x_2, x_3\} - \{x_i\}$ which are not satisfied in the instance. Another example is $(x_1 + x_2, x_2 + x_3, x_3 + x_4, x_4 + x_1)$ which is satisfiable. The idea is now to make two code an *RSAT*-instance by separating the variables by multiples of 3. For example, the instance $(x_1 + x_2, x_1 + x_3, x_1 + x_2 + x_3)$ is coded by the pattern

$$\pi \;=\; x_1x_23x_1x_36x_2x_3$$

and now one considers the two patterns which are obtained by assigning to each variable x_i the values $3i - 2$ and $3i - 1$, respectively; so the two words would be

$$v \;=\; 14\,3\,17\,6\,47 \;\text{ and }\; w \;=\; 25\,3\,28\,6\,58$$

where again the spacing is just for better readability. Now it is argued that a consistent and conservative learner has to output the correct pattern (up to renaming of the variables) after having seen these two examples. So let $L(\rho)$ be the hypothesis of M after having seen the two examples. The pattern ρ can only have multiples of 3 as constants as the other constants appear only in one of the words v and w. Fix now a substitution of the variables in ρ which produces

v, note that a similar substitution with all values $3k-1$ replaced by $3k-2$ produces w. If now a variable in ρ produces at least two symbols then it cannot appear twice, as the multiples of 3 occur only once in v and are therefore not in the values of twice occurring variables; furthermore neighboring instance variables $x_i x_j$ occur also only once in the pattern π and therefore the corresponding sequence $3i-1\,3j-1$ occurs in v also only once. Hence one can conclude that this case does not occur and every variable which occurs at least twice produces in v exactly one symbol. Furthermore, one can replace every variable which occurs once and produces k symbols by k neighboring variables occurring once and each producing one symbol in v; the language generated by the new pattern ρ' is a subset of the language generated by ρ and it still contains v and w; but by the conservativeness of the learner it does not overgeneralize when learning $L(\rho')$ and therefore it would not output $L(\rho)$ on the data v, w from $L(\rho')$. Hence one can conclude that this case does not occur and that every variable in ρ produces exactly one symbol in v per occurrence. It follows that for every variable x_k in the instance there are at each occurrence variables $y_{k,1}, y_{k,2}, y_{k,3}$ in ρ which produce the value $3k-1$ in v and $3k-2$ in w, respectively; these variables do not occur at other places. Consequently, one can conclude again by using that $L(\rho)$ does not overgeneralize, that the values $3k$ in v and w are not generated by variables but by constants in ρ and that the variables $y_{k,1}, y_{k,2}, y_{k,3}$ are equal. It follows that $L(\rho) = L(\pi)$.

Now one takes as a third word u, where u contains the multiples of 3 to separate the clauses (as before) and each clause containing k variables is replaced by 0^{k+1}. As a consequence, $u \in L(\rho)$ iff there is an assignment of the variables which makes in each clause exactly one variable to 00 and the others to 0. This means, $u \in L(\rho)$ iff the coded $RSAT$-instance is satisfiable by taking those variables to true which are mapped to 00 and those to false which are mapped to 0. Thus one obtains the result that $M(v \diamond w) = M(v \diamond w \diamond u)$ iff the corresponding $RSAT$-instance is satisfiable and this algorithm provides now a polynomial-time many-one reduction from $RSAT$ to the corresponding learnability problem. Thus the learner M can only exist if $P = NP$. $\square$

Note that the same result holds if one works with finite alphabets and the learner has to learn the pattern plus the alphabet from the data; the latter is easy as the alphabet is just the set of all symbols in observed data (unless the pattern consists only of constants in which it is irrelevant to know the non-occurring symbols in the pattern). So in this case it is shown that the learnability problem uniform over finite alphabets is *NP*-complete. It would be interesting to know whether larger and larger (or infinite) alphabets are really needed for the *NP*-hardness result.

Open Problem 6. *Let Σ be a fixed finite alphabet. Is the task to learn the class of all non-erasing pattern languages over Σ in a consistent and conservative way using a class-preserving hypothesis space NP-hard?*

References

1. Dana Angluin: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21(1), 46–62 (1980)
2. Dana Angluin: Inductive inference of formal languages from positive data. Information and Control 45(2), 117–135 (May 1980)

3. Lenore Blum and Manuel Blum: Toward a mathematical theory of inductive inference. Information and Control 28(2), 125–155 (1975)
4. John Case and Timo Kötzing: Dynamically delayed postdictive completeness and consistency in learning. In: Algorithmic Learning Theory, 19th International Conference, ALT 2008, Budapest, Hungary, October 2008, Proceedings. Lecture Notes in Artificial Intelligence, vol. 5254, pp. 389–403. Springer, Berlin (2008)
5. John Case and Timo Kötzing: Difficulties in forcing fairness of polynomial time inductive inference. In: Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 2009, Proceedings. Lecture Notes in Artificial Intelligence, vol. 5809, pp. 263–277. Springer, Berlin/Heidelberg (2009)
6. Rodney G. Downey and Micheal R. Fellows: Parameterized Complexity. Monographs in Computer Science, Springer, Berlin (1999)
7. Thomas Erlebach, Peter Rossmanith, Hans Stadtherr, Angelika Steger and Thomas Zeugmann: Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. Theoret. Comput. Sci. 261(1), 119–156 (2001)
8. E. Mark Gold: Language identification in the limit. Inform. Control 10(5), 447–474 (1967)
9. Michael J. Kearns and Leonard Pitt: A polynomial-time algorithm for learning k-variable pattern languages from examples. In: Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, CA. pp. 57–71. Morgan Kaufmann, San Mateo, CA (1989)
10. Steffen Lange and Rolf Wiehagen: Polynomial-time inference of arbitrary pattern languages. New Generation Computing 8(4), 361–370 (1991)
11. Steffen Lange, Thomas Zeugmann and Sandra Zilles: Learning indexed families of recursive languages from positive data: A survey. Theoretical Computer Science 397(1-3), 194–232 (2008)
12. Daniel Reidenbach: The Ambiguity of Morphisms in Free Monoids and its Impact on Algorithmic Properties of Pattern Languages. Ph.D. thesis, Department of Computer Science, University of Kaiserslautern, Germany, Logos Verlag, Berlin (2006)
13. Daniel Reidenbach: A non-learnable class of E-pattern languages. Theoretical Computer Science 350(1), 91–102 (2006)
14. Peter Rossmanith and Thomas Zeugmann: Stochastic finite learning of the pattern languages. Machine Learning 44(1/2), 67–91 (2001)
15. Takeshi Shinohara: Polynomial time inference of extended regular pattern languages. In: RIMS Symposium on Software Science and Engineering, Kyoto, 1982, Proceedings. Lecture Notes in Computer Science, vol. 147, pp. 115–127. Springer-Verlag (1983)
16. Takeshi Shinohara and Setsuo Arikawa: Pattern inference. In: Algorithmic Learning for Knowledge-Based Systems, Lecture Notes in Artificial Intelligence, vol. 961, pp. 259–291. Springer (1995)
17. Thomas Zeugmann: Lange and Wiehagen's pattern language learning algorithm: An average-case analysis with respect to its total learning time. Annals of Mathematics and Artificial Intelligence 23(1/2), 117–145 (1998)

18. Thomas Zeugmann and Steffen Lange: A guided tour across the boundaries of learning recursive languages. In: Algorithmic Learning for Knowledge-Based Systems, Lecture Notes in Artificial Intelligence, vol. 961, pp. 190–258. Springer (1995)