Learning k-Variable Pattern Languages Efficiently Stochastically Finite on Average from Positive Data

Rossmanith, Peter
Institut für Informatik Technische Universität München

Zeugmann, Thomas
Department of Informatics Kyushu University

出版情報：DOI Technical Report. 145, 1998-01-17. Department of Informatics, Kyushu University
バージョン：
権利関係：

DOI-TR-145

DOI Technical Report

Learning k-Variable Pattern Languages Efficiently Stochastically Finite on Average from Positive Data

by

Peter Rossmanith and Thomas Zeugmann

January 17, 1998

Department of Informatics
Kyushu University
Fukuoka 812-81, Japan

Email: thomas@i.kyushu-u.ac.jp Phone: +81-92-543-7640

Learning k-Variable Pattern Languages Efficiently Stochastically Finite on Average from Positive Data

Peter Rossmanith
Institut für Informatik
Technische Universität München
D–80290 München
Germany
rossmani@informatik.tu-muenchen.de

Thomas Zeugmann
Department of Informatics
Kyushu University
Kasuga 816
Japan
thomas@i.kyushu-u.ac.jp

Abstract

The present paper deals with the average-case analysis of the Lange–Wiehagen (1991) algorithm learning the class of all pattern languages in the limit from positive data.

Let $\mathcal{A} = \{0, 1, \ldots\}$ be any non–empty finite alphabet containing at least two elements. Furthermore, let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathcal{A} \cap X = \emptyset$. *Patterns* are non–empty strings over $\mathcal{A} \cup X$. $L(\pi)$, the language generated by pattern π is the set of strings which can be obtained by substituting non-null strings from $\mathcal{A}^*$ for the variables of the pattern π.

The Lange–Wiehagen (1991) algorithm is analyzed with respect to its *total learning time*, i.e., the overall time taken by the algorithm until *convergence*. The expectation of the total learning time is carefully analyzed and *exponentially* shrinking tail bounds for it are established for a large class of probability distributions. For every pattern π containing k different variables it is shown that Lange and Wiehagen's algorithm possesses an expected total learning time of $O\big(\hat{\alpha}^k(\log_{1/\beta}(k) + 2)E[\Lambda]\big)$, where $\hat{\alpha}$ and β are two easy computable parameters arising naturally from the underlying probability distributions, and $E[\Lambda]$ is the expected example string length. In particular, for the *uniform distribution* the expected total learning time is $O(2^k|\pi| \log_{|\mathcal{A}|}(k))$ which is a slight improvement over previously known results.

Finally, we show how the improved analysis can be used to arrive at a *new learning model*. In this model, learning in the limit is transformed into *finite* learning with *high confidence*.

1. Introduction

The present paper deals with the average-case analysis of the Lange–Wiehagen [10] algorithm (LWA for short) that learns the class of all pattern languages in the limit from positive data. That means the learner is fed successively example strings and its previously made hypothesis, and it computes from these input data a new pattern as its hypothesis. The sequence of all pattern output stabilized to a single pattern which generates the target pattern language.

Our goal is twofold. On the one hand, we *generalize* and *improve* the average-case analysis of the same algorithm performed by Zeugmann [19] for its expected *total learning time*. The time taken by a learner for computing a single hypothesis from its input data is usually called *update time*. The total learning time is the time taken by the learner until convergence, i.e., the sum of all update times until successful learning. Note that it is a highly non-trivial task to define an appropriate complexity measure for learning in the limit (cf. [12]). The total learning time has been introduced by Daley and Smith [3]. As Pitt [12] pointed out, allowing the total learning time to depend on the length of the examples seen so far is unsatisfactory, since the learner may delay convergence until a sufficiently long example appears so that the algorithm may meet the wanted polynomial time bound. We therefore measure the total learning time *only* in dependence on the length of the target pattern. As shown in Zeugmann [19], then the total learning time is unbounded in the worst-case. Thus analyzing the total learning in such a worst-case setting does yield much insight. Moreover, such a worst-case scenario is too pessimistic for many applications. Zeugmann [19] therefore initialized the average-case analysis of limit learners with respect to their total learning time. We continue along this line.

On the other hand, we show how the improved analysis can be used to arrive at a *new learning model*. In this model, learning in the limit is transformed into *finite* learning with *high confidence*. Since this transformation requires a certain amount of knowledge concerning the underlying class of admissible probability distributions we refer to the new learning model as *stochastically finite learning*. The basic idea can be described as follows. Based on exponentially shrinking tail bounds obtained from our average case analysis for the expected *total learning time*, the new learner takes as input a text and a *confidence parameter* δ. It then computes internally hypotheses until the time bound established is met for the first time. Intuitively, now the algorithm has already learned on average. However, in order to establish the wanted confidence the learner proceeds until the probability to have missed the successful learning is smaller than or equal to the desired confidence. It then outputs its last internally computed guess and stops thereafter (cf. Section 4, Definition 2).

Our research derives its motivation from the fact that the pattern languages (*PAT* for short) are a prominent and important concept class that can be learned from positive data (cf., e.g., Salomaa [14, 15], and Shinohara and Arikawa [17] for an overview). Nevertheless, despite its importance there is still a bottleneck concerning efficient learning algorithms. Kearns and Pitt [8], Ko, Marron and Tzeng [9]] and Schapire S90 intensively studied the learnability of pattern languages in the PAC–learning model. In particular, Schapire [16] proved that the class *PAT* is not PAC-learnable regardless of the representation used by the learning algorithm, provided only that the learner

is requested to output a polynomial-size hypothesis that can be evaluated in polynomial time, unless $\mathcal{P}_{/poly} = \mathcal{NP}_{/poly}$. However, the class *Pat* of all patterns is not a polynomial time representation for *PAT*, since the membership problem for *PAT* with respect to *Pat* is $\mathcal{NP}$-complete (cf. [1]). On the other hand, Kearns and Pitt [8] designed a polynomial-time PAC-learner for the set of all k-variable pattern languages (k arbitrarily fixed) under the assumption the only product distributions are allowed. Unfortunately, the constant in the running time achieved depends *doubly exponential* on k, and thus, their algorithm becomes rapidly impractical when k increases.

In contrast, our stochastically finite learner achieves a running time whose constant depends only exponentially on the number k of different variables occurring in the target pattern and is otherwise linearly bounded in the expected length of sample strings fed to the learner (cf. Theorem 2). The price paid is rather small. We restrict the class of all probability distributions to a subclass that has an arbitrary but fixed bound on two parameters arising naturally. In essence, that means at least two letters from the underlying probability distribution have a *known* lower bound on their probability.

Finally, we shortly summarize the improvements obtained concerning the average-case analysis compared to the analysis undertaken in [19].

(1) Let Λ be the length of random positive example according to some distribution. Zeugmann's [19] estimate of the expected total learning time contains the variance of Λ as a factor. While this variance is small for many distribution it is sometimes infinity and sometimes very big. The new analysis does not use variances or other higher moments of Λ at all. Thus it turns out that the expected total learning time is bounded iff the expectation of Λ is bounded.

(2) We present exponentially fast shrinking tail bounds for the expected total learning time. Previous work did not study tail bounds at all.

(3) The new analysis presents the bound on the expected total learning time as a simple formula that contains only three parameters: α, β, and $E[\Lambda]$. The parameters α and β are very simple to compute for each probability distribution.

(4) The new analysis is slightly tighter: For the "uniform" distribution, e.g., the upper bound is by a factor of $k^2|\pi|$ smaller, where $|\pi|$ is the length of the pattern and k again the number of different variables occurring in π. We also give all bounds as exact formulas without hiding constant factors in a big O. These constants, however, are not the best possible in order to keep the proofs simple.

(5) Besides the total learning time we give separate estimates on the number of iterations until convergence, the number of union operations performed by the LWA, and the time spent in union operations. The union operation is the most difficult part in the LWA. On the one hand, we provide a new algorithm for computing the union operation that achieves linear time, while all previously known algorithms take quadratic time. Nevertheless, our algorithm needs quadratic space. Thus, these extra estimates help to judge whether this optimization is worthwhile. It turns out that except for rather pathological distributions the time spent for union operations is quite small even when each union operation takes quadratic time. Hence, if space is a serious matter of concern, one can still stick to the naïve implementation without worsening the overall time bound too much.

2. Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of all natural numbers, and let $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. For all real numbers x we define $\lfloor x \rfloor$, the *floor function*, to be the greatest integer less than or equal to x. Furthermore, by $\lceil x \rceil$ we denote the *ceiling function*, i.e., the least integer greater than or equal to x.

Following Angluin [1] we define patterns and pattern languages as follows. Let $\mathcal{A} = \{0, 1, \ldots\}$ be any non-empty finite alphabet containing at least two elements. By $\mathcal{A}^*$ we denote the free monoid over $\mathcal{A}$ (cf. Hopcroft and Ullman [7]). The set of all finite non-null strings of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^+$, i.e., $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$, where ε denotes the empty string. By $|\mathcal{A}|$ we denote the cardinality of $\mathcal{A}$. Furthermore, let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathcal{A} \cap X = \emptyset$. *Patterns* are non-empty strings over $\mathcal{A} \cup X$, e.g., 01, $0x_0111$, $1x_0x_00x_1x_2x_0$ are patterns. The length of a string $s \in \mathcal{A}^*$ and of a pattern π is denoted by $|s|$ and $|\pi|$, respectively. A pattern π is in *canonical form* provided that if k is the number of different variables in π then the variables occurring in π are precisely $x_0, \ldots, x_{k-1}$. Moreover, for every j with $0 \leq j < k-1$, the leftmost occurrence of x_j in π is left to the leftmost occurrence of x_{j+1} in π. The examples given above are patterns in canonical form. In the sequel we assume, without loss of generality, that all patterns are in canonical form. By Pat we denote the set of all patterns in canonical form.

Let $\pi \in Pat$, $1 \leq i \leq |\pi|$; we use $\pi(i)$ to denote the i-th symbol in π. If $\pi(i) \in \mathcal{A}$, then we refer to $\pi(i)$ as to a *constant*; otherwise $\pi(i) \in X$, and we refer to $\pi(i)$ as to a *variable*. Analogously, by $s(i)$ we denote the i-th symbol in s for every string $s \in \mathcal{A}^+$ and all $i = 1, \ldots, |s|$. By $\#\mathrm{var}(\pi)$ we denote the number of different variables occurring in π, and by $\#_{x_i}(\pi)$ we denote the number of occurrences of variable x_i in π. If $\#\mathrm{var}(\pi) = k$, then we refer to π as to a *k-variable pattern*. Let $k \in \mathbb{N}$, by Pat_k we denote the set of all *k-variable patterns*. Furthermore, let $\pi \in Pat_k$, and let $u_0, \ldots, u_{k-1} \in \mathcal{A}^+$; then we denote by $\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}]$ the string $w \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \ldots, k-1$, in the pattern π. For example, let $\pi = 0x_01x_1x_0$. Then $\pi[x_0/10, x_1/01] = 01010110$. The tuple $(u_0, \ldots, u_{k-1})$ is called *substitution*. Furthermore, if $|u_0| = \cdots = |u_{k-1}| = 1$, then we refer to $(u_0, \ldots, u_{k-1})$ as to a *shortest substitution*. Now, let $\pi \in Pat_k$, and let $S = \{(u_0, \ldots, u_{k-1}) \mid u_j \in \mathcal{A}^+,\ j = 0, \ldots, k-1\}$ be any finite set of substitutions. Then we set $S(\pi) = \{\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}] \mid (u_0, \ldots, u_{k-1}) \in S\}$, i.e., $S(\pi)$ is the set of all strings obtained from pattern π by applying all the substitutions from S to it. For every $\pi \in Pat_k$ we define the *language generated by pattern* π by $L(\pi) = \{\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}] \mid u_0, \ldots, u_{k-1} \in \mathcal{A}^+\}$. By PAT_k we denote the set of all *k-variable pattern languages*. Finally, $PAT = \bigcup_{k \in \mathbb{N}} PAT_k$ denotes the set of all pattern languages over $\mathcal{A}$. Note that for every $L \in PAT$ there is precisely one pattern $\pi \in Pat$ such that $L = L(\pi)$ (cf. Angluin [1]).

We are interested in *inductive inference*, which means to gradually learn a concept from successively growing sequences of examples. If L is a language to be identified, a sequence $(s_1, s_2, s_3, \ldots)$ is called a *text* for L if $L = \{s_1, s_2, s_3, \ldots\}$ (cf. [5]). However, in practical applications, the requirement to exhaust the language to be learned will be hardly fulfilled. We therefore *omit* this assumption here. Instead, we generalize the notion of text to the case that the sequence $t = s_1, s_2, s_3, \ldots$ contains "enough" information to recognize the target pattern. As for the LWA, "enough"

can be made precise by requesting that sufficiently many shortest strings appear in the text. We shall come back to this point when defining admissible probability distributions.

As introduced by Gold [5] an *inductive inference machine* is an algorithm that takes as input larger and larger initial segments of a text and outputs, after each input, a hypothesis from a prespecified *hypothesis space*. In the case of pattern languages the hypothesis space is Pat.

DEFINITION 1. PAT is called *learnable in the limit from text* iff there is an IIM M such that for every $L \in PAT$ and every text t for L,

(1) for all $n \in \mathbb{N}^+$, $M(t_n)$ is defined,

(2) there is a pattern $\pi \in Pat$ such that $L(\pi) = L$ and for almost all $n \in \mathbb{N}^+$, $M(t_n) = \pi$.

It is well known that pattern languages are learnable in the limit from text (cf. [1]).

Whenever one deals with the average case analysis of algorithms one has to consider probability distributions over the relevant input domain. For learning from text, we have the following scenario. Every string of a particular pattern language is generated by a substitution. Therefore, it is convenient to consider probability distributions over the set of all possible substitutions. That is, if $\pi \in Pat_k$, then it suffices to consider any probability distribution D over $\underbrace{\mathcal{A}^+ \times \cdots \times \mathcal{A}^+}_{k-\text{times}}$. For $(u_0, \ldots, u_{k-1}) \in \mathcal{A}^+ \times \cdots \times \mathcal{A}^+$ we denote by $D(u_0, \ldots, u_{k-1})$ the probability that variable x_0 is substituted by u_0, variable x_1 is substituted by u_1, ..., and variable x_{k-1} is substituted by u_{k-1}.

In particular, we mainly consider a special class of distributions, i.e., *product distributions*. Let $k \in \mathbb{N}^+$, then the class of all product distributions for Pat_k is defined as follows. For each variable x_j, $0 \leq j \leq k-1$, we assume an arbitrary probability distribution D_j over $\mathcal{A}^+$ on substitution strings. Then we call $D = D_0 \times \cdots \times D_{k-1}$ product distribution over $\mathcal{A}^+ \times \cdots \times \mathcal{A}^+$, i.e., $D(u_0, \ldots, u_{k-1}) = \prod_{j=0}^{k-1} D_j(u_j)$. Moreover, we call a product distribution *regular* if $D_0 = \cdots = D_{k-1}$. Throughout this paper, we restrict ourselves to deal with *regular* distributions. We therefore use d to denote the distribution over $\mathcal{A}^+$ on substitution strings, i.e, $D(u_0, \ldots, u_{k-1}) = \prod_{j=0}^{k-1} d(u_j)$. Note, however, that most of our results can be generalized to larger classes of distributions. Finally, we can provide the announced specification of what is meant by "enough" information. We call a regular distribution *admissible* provided $d(a) > 0$ for at least two different elements $a \in \mathcal{A}$.

Following Daley and Smith [3] we define the total learning time as follows. Let M be any IIM that learns all the pattern languages. Then, for every $L \in PAT$ and every text t for L, let

$$Conv(M, t) =_{df} \text{the least number } m \in \mathbb{N}^+ \text{ such that for all } n \geq m,\ M(t_n) = M(t_m)$$

denote the *stage of convergence* of M on t. Moreover, by $T_M(t_n)$ we denote the time to compute $M(t_n)$. We measure this time as a function of the length of the input

and refer to it as to the *update time*. Finally, the total learning time taken by the IIM M on successive input t is defined as

$$TT(M,t) =_{df} \sum_{n=1}^{Conv(M,t)} T_M(t_n).$$

Assuming any fixed admissible probability distribution D as described above, we aim to evaluate the *expectation* of $TT(M,t)$ with respect to D which we refer to as to the *total average learning time*.

The model of computation as well as the representation of patterns we assume is the same as in Angluin [1]. In particular, we assume a random access machine that performs a reasonable menu of operations each in unit time on registers of length $O(\log n)$ bits, where n is the input length.

Finally, we recall the LWA. The LWA works as follows. Let h_n be the hypothesis computed after reading $s_1, \ldots, s_n$, i.e., $h_n = M(s_1, \ldots, s_n)$. Then $h_1 = s_1$ and for all $n > 1$:

$$h_n = \begin{cases} h_{n-1}, & \text{if } |h_{n-1}| < |s_n| \\ s_n, & \text{if } |h_{n-1}| > |s_n| \\ h_{n-1} \cup s_n, & \text{if } |h_{n-1}| = |s_n| \end{cases}$$

The algorithm computes the new hypothesis only from the latest example and the old hypothesis. If the latest example is longer than the old hypothesis, the example is ignored, i.e., the hypothesis does not change. If the latest example is shorter than the old hypothesis, the old hypothesis is ignored and the new example becomes the new hypothesis. Hence, the LWA is quite simple and the update time will be very fast for these two possibilities.

If, however, $|h_{n-1}| = |s_n|$ the new hypothesis is the *union* of h_{n-1} and s_n. The union $\varrho = \pi \cup s$ of a canonical pattern π and a string s of the same length is defined as

$$\varrho(i) = \begin{cases} \pi(i), & \text{if } \pi(i) = s(i) \\ x_j, & \text{if } \pi(i) \neq s(i) \ \& \ \exists k < i : \\ & [\varrho(k) = x_j,\ s(k) = s(i),\ \pi(k) = \pi(i)] \\ x_m, & \text{otherwise, where } m = \#var(\varrho(1) \ldots \varrho(i-1)) \end{cases}$$

where $\varrho(0) = \varepsilon$ for notational convenience. Note that the resulting pattern is again canonical.

Obviously, the union operation can be computed in quadratic time. We finish the section by providing a linear-time algorithm computing the union operation. The only crucial part is to determine whether or not there is some $k < i$ with $\varrho(k) = x_j$, $s(k) = s(i)$, and $\pi(k) = \pi(i)$. The new algorithm uses an array $I = \{1, \ldots, |s|\}^{\mathcal{A} \times (\mathcal{A} \cup \{x_0, \ldots, x_{|\pi|-1}\})}$ for finding the correct k, if any, in *constant* time. The array I is *partially* initialized by writing the first position into it at which $s(i), \pi(i)$ occurs. Then, for each position i, the algorithm checks whether or not

$Is(i), \pi(i) = i$. Suppose it is, thus $s(i), \pi(i)$ did not occur left to i. Hence, it remains to check whether or not $\pi(i) = s(i)$ and $\varrho(i)$ can be immediately output.

If $I_{s(i),\pi(i)} \neq i$, then $s(i), \pi(i)$ did occur left to i. Hence, in this case it suffices to output $\varrho(j)$ where $j = I_{s(i),\pi(i)}$.

THEOREM 1. *The union operation can be computed in linear time.*

Proof. The following algorithm constructs $\varrho = \pi \cup s$ in linear time.

***Algorithm* 1**

Input: A pattern π and a string $s \in \mathcal{A}^+$ such that $|\pi| = |s|$.
Output: $\pi \cup s$
Method:
for $i = 1, \ldots, |s|$ **do** $I_{s(i),\pi(i)} = 0$ **od**
for $i = 1, \ldots, |s|$ **do**
  if $I_{s(i),\pi(i)} = 0$ **then** $I_{s(i),\pi(i)} \leftarrow i$ **fi**
$m \leftarrow 0$;
for $i = 1, \ldots, |s|$ **do**
  $j \leftarrow I_{s(i),\pi(i)}$
  if $i = j$ **then**
    if $\pi(i) = s(i)$ **then** $\varrho(i) = \pi(i)$
    else $\varrho(i) \leftarrow x_m$, $m \leftarrow m + 1$ **fi**
  else $\varrho(i) = \varrho(j)$
  fi
od

The correctness of this algorithm can be easily proved inductively by formalizing the argument given above. We omit the details. ∎

We continue with a summary of the results obtained.

3. Results

Following [19] we perform the desired analysis in dependence on the number k of different variables occurring in the target pattern π. If $k = 0$, then the LWA immediately converges. Therefore, in the following we assume $k \in \mathbb{N}^+$, and $\pi \in Pat_k$. Taking into account that $|w| \geq |\pi|$ for every $w \in L(\pi)$, it is obvious that the LWA can only converge if it has been fed sufficiently many strings from $L(\pi)$ having minimal length. Therefore let

$$L(\pi)_{min} = \{w \mid w \in L(\pi),\ |w| = |\pi|\}.$$

Zeugmann [19] found an exact formula for the minimum number of examples that the LWA needs to converge:

PROPOSITION 1 (Zeugmann [19]). *To learn a pattern $\pi \in Pat_k$ the LWA needs exactly $\lfloor \log_{|\mathcal{A}|}(|\mathcal{A}| + k - 1) \rfloor + 1$ examples in the best case.*

Clearly, in order to match this bound all examples must have been drawn from $L(\pi)_{min}$. In the worst case there is no upper bound on the number of examples.

For analyzing the *average-case* behavior of the LWA, in the following we let $t = s_1, s_2, s_3, \ldots$ range over all randomly generated texts with respect to some arbitrarily fixed admissible probability distribution D. Then the stage of convergence is a random variable which we denote by C. Note that the distribution of C depends on π and on D. We introduce several more random variables. By Λ_i we denote the length of the example string s_i, i.e., $\Lambda_i = |s_i|$. Since all Λ_i are independent and identically distributed, we can assume that the random variable has the same distribution as Λ. We will use Λ when talking about the length of an example when the number of the example is not important. Particularly, we will often use the expected length of a random example $E[\Lambda]$.

Let T be the *total length* of examples processed until convergence, i.e.,

$$T = \Lambda_1 + \Lambda_1 + \ldots + \Lambda_C.$$

Whether the LWA converges on $s_1, \ldots, s_r$ depends only on those examples s_i with $s_i \in L(\pi)_{min}$. Let $r \in \mathbb{N}^+$; by M_r we denote the number of minimum length examples among the first r strings, i.e.,

$$M_r = |\{\, i \mid 1 \leq i \leq r \text{ and } \Lambda_i = |\pi|\,\}|.$$

In particular, M_C is the number of minimum length examples read until convergence. We assume that reading and processing one character takes exactly one time step in the LWA unless union operations are performed. Disregarding the union operations, the total learning time is then T. The number of union operations until convergence is denoted by U. The time spent in union operations until convergence is V. The total learning time is therefore $TT = T + V$. We assume that computing $\varrho \cup s$ takes at most $c \cdot |\rho|$ steps, where c is a constant that depends on the implementation of the union operation.

We will express all estimates with the help of the following parameters: $E[\Lambda]$, c, α and β. To get concrete bounds for a concrete implementation one has to obtain c from the algorithm and has to compute $E[\Lambda]$, α, and β from the admissible probability distribution D. Let $u_0, \ldots, u_{k-1}$ be independent random variables with distribution d for substitution strings. Whenever the index i of u_i does not matter, we simply write u or u'.

The two parameters α and β are now defined via d. First, α is simply the probability that u has length 1, i.e.,

$$\alpha = \Pr(|u| = 1) = \sum_{a \in \mathcal{A}} d(a).$$

Second, β is the conditional probability that two random words that get substituted into π are identical under the condition that both their length are 1, i.e.,

$$\beta = \Pr\big(u = u' \;\big|\; |u| = |u'| = 1\big) = \sum_{a \in \mathcal{A}} d(a)^2 \Big/ \Big(\sum_{a \in \mathcal{A}} d(a)\Big)^2.$$

The parameter α and β are therefore quite easy to compute even for complicated distribution since they depend only on $|\mathcal{A}|$ point probabilities. We can also compute $E[\Lambda]$ for a pattern π from d quite easily.

Let $u = (u_0, \ldots, u_{k-1})$ be any substitution. Because of

$$|\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}]| = |\pi| + \sum_{i=0}^{k-1} \#_{x_i}(\pi)(|u_i| - 1),$$

we have

$$E[\Lambda] = |\pi| + v(E[|u|] - 1)$$

where v is the total number of variable occurrences in π, i.e., $v = \sum_{i=0}^{k-1} \#_{x_i}(\pi)$

In our analysis we will use often the *median* of a random variable. If X is a random variable then μX is a median of X iff

$$\Pr(X \geq \mu X) \leq 1/2 \text{ and } \Pr(X \leq \mu X) \leq 1/2.$$

A nonempty set of medians exists for each random variable and consists either of a single real number or of a closed real interval. We will denote the smallest median of X by μX, since this choice gives the best upper bounds.

Next, we present the main results and compare them to Zeugmann's [19] analysis. His distribution independent results read as follows in our notation:

PROPOSITION 2 (Zeugmann [19], Theorem 8). $E[TT] = O\big(E[C] \cdot (V[\Lambda] + E^2[\Lambda])\big)$.

The variance of Λ is herein denoted by $V[\Lambda]$. The parameters $V[\Lambda]$ and $E[\Lambda]$ have to be computed for a given distribution. For $E[C]$ he gives an estimate with the help of $E[M_C]$ and $E[T_j]$, which is the expected time to receive the first j pairwise different elements from $L(\pi)_{min}$:

PROPOSITION 3 (Zeugmann [19], Theorem 8).

$$E[C] \leq E[M_C] \cdot \max\left\{E[T_1],\ \frac{1}{2}\sum_{j=1}^{2} E[T_j],\ \ldots,\ \frac{1}{|\mathcal{A}|^{k-1}+1}\sum_{j=1}^{|\mathcal{A}|^{k-1}+1} E[T_j]\right\}$$

He then proceeds to estimate these parameters for the *uniform distribution*, where $d(u) = 2^{-|u|}|\mathcal{A}|^{-|u|}$. Here his estimate is as follows:

PROPOSITION 4 (Zeugmann [19], Theorem 11). $E[TT] = O(2^k |\pi|^2 \log_{|\mathcal{A}|}(k|\mathcal{A}|))$ for the uniform distribution.

The main difference to our analysis are the parameters that have to be evaluated for a given distribution. Instead of $E[\Lambda]$, $V[\Lambda]$, $E[C_N]$, $E[T_j]$, and $|\mathcal{A}|$ we use α, β, and $E[\Lambda]$, which are easier to obtain. Setting $\hat{\alpha} = 1/\alpha$, we can estimate the total learning time as follows:

THEOREM 2. $E[TT] = O\big(\hat{\alpha}^k(\log_{1/\beta}(k) + 2)E[\Lambda]\big)$.

At first sight Theorem 2 looks complicated, but it is rather simple to evaluate. One further advantage is that the variance of Λ is not used at all. Take for example some distribution with $\Pr(|u| = 2^i) = 3 \cdot 4^{-i-1}$ and $\Pr(|u| = n) = 0$ if n is not a power of 2. Then $E[\Lambda] \leq \frac{3}{2}|\pi|$, but $V[\Lambda] = \infty$. Hence, Proposition 2 just says $E[TT] \leq \infty$. Since $\hat{\alpha} = 4/3$, Theorem 2 yields a very good upper bound, i.e.,

$E[TT] = O\big((4/3)^k(\log_{1/\beta}(k)+2)|\pi|\big)$. Even if $V[\Lambda]$ exists it can be much bigger than $E^2[\Lambda]$.

Next, we insert the parameter for the uniform distribution into Theorem 2. For the uniform distribution we get $\hat{\alpha} = 2$, $\beta = 1/|\mathcal{A}|$, and $E[\Lambda] \leq 2|\pi|$.

THEOREM 3. $E[TT] = O(2^k|\pi|\log_{|\mathcal{A}|}(k))$ *for the uniform distribution.*

This estimate is slightly better than Proposition 4.

We continue by investigating other expected values of interest. Often time is the most precious resource and then we have to minimize the total learning time. The number of examples until convergence can also be interesting, if the gathering of examples is expensive. Then the average number of examples is the critical parameter and we are interested in $E[C]$.

THEOREM 4. $E[C] = O(\hat{\alpha}^k \cdot \log_{1/\beta}(k))$.

If we compare Theorems 2 and 4 it turns out that in many cases the total learning time is by a factor of about $E[\Lambda]$ larger than the number of examples read until convergence. This is about the same time an algorithm uses that just reads $E[C]$ random positive examples.

We can even get a better understanding of the behavior if we examine the union operations by themselves. Is it worthwhile to optimize the computation of $w \cup \pi$? It turns out that union operations are responsible only for a small part of the overall computation time. Remember that U is the number of union operations and V is the time spent in union operations.

THEOREM 5.

(1) $E[U] = O(\hat{\alpha}k + \log_{1/\beta}(k))$

(2) $E[U] = O(\hat{\alpha}kE[\Lambda] + \log_{1/\beta}(k)|\pi|)$ *provided the union operation is performed by Algorithm 1,*

(3) $E[V] = O(\hat{\alpha}kE^2[\Lambda] + \log_{1/\beta}(k)|\pi|^2)$ *if the union operation is performed by the naïve algorithm.*

Consequently, if space is a serious matter of concern, e.g., if the patterns to be learned are very long, one may easily trade a bit more time by using the naïve, quadratic time algorithm instead of Algorithm 1 above. We shall come back to this point in Section 5.

3.1. *Tail Bounds*

Finally we have to ask whether the average total learning time is sufficient for judging the LWA. The expected value of a random variable is only one aspect of its distribution. In general we might also be interested on how often the learning time exceeds the average substantially. Again this is a question motivated mainly by practical considerations. Equivalently we can ask, how good the distribution is concentrated around its expected value. Often this question is answered by estimating

the *variance*, which enables the use of Chebyshev's inequality. If the variance is not available, Markov's inequality provides us with (worse) tail bounds:

$$\Pr(X \geq t \cdot E[X]) \leq \frac{1}{t}$$

The Markov inequality is quite general but produces only weak bounds. The next theorem gives better tail bounds for a large class of learning algorithms including the LWA. The point here is, that the LWA possesses two additional desirable properties, i.e., it is *set-driven* (cf. Zeugmann [19]) and *conservative*. A learner is said to be set-driven, if its outputs depends only on the range of its input (cf. [18]). Conservative learners maintain their actual hypotheses at least as long as they have not seen data contradicting them (cf. Angluin [2]).

THEOREM 6. *Let X be the sample complexity of a conservative and set-driven learning algorithm. Then $\Pr(X \geq t \cdot \mu X) \leq 2^{-t}$ for all $t \in \mathbb{N}$.*

Proof. We divide the text $(s_1, s_2, \ldots)$ into blocks of length μC. The probability that the algorithm converges after reading any of the blocks is then at least $1/2$. Since the algorithm is set-driven the order of the blocks does not matter and since the algorithm is conservative it does not change its hypothesis after computing once the right pattern. ∎

COROLLARY 7. *Let X be the sample complexity of a conservative and set-driven learning algorithm. Then $\Pr(X \geq 2t \cdot E[X]) \leq 2^{-t}$ for all $t \in \mathbb{N}$.*

Proof. Since $\mu X \leq 2E[X]$ for every positive random variable X by the Markov inequality, we get immediately that $\Pr(X \geq 2t \cdot E[X]) \leq 2^{-t}$ for every $t \in \mathbb{N}$. ∎

Theorem 6 and Corollary 7 put the importance of conservative and set-driven learners into the right perspective. Moreover, both results remain true for conservative and *rearrangement-independent* learners. A learner is said to be rearrangement-independent if its outputs depends exclusively of the range and length of its input. Thus, Theorem 6 and Corollary 7 have wide range of potential applications as long as the learnability of indexed families is concerned, since every conservative learner can be transformed into a learner that is both conservative and rearrangement-independent provided the hypothesis space is appropriately chosen (cf. Lange and Zeugmann [11]).

Since the distribution of X decreases geometrically, all higher moments of X exist. The next two theorems estimate the expected value and the variance of X in terms of the median. Similar results hold for higher moments.

THEOREM 8. *Let X be the sample complexity of a conservative and set-driven learning algorithm. Then $E[X] \leq 2\mu X$.*

Proof.

$$\begin{aligned} E[X] = \sum_{i=1}^{\infty} \Pr(X \geq i) \leq \sum_{i=1}^{\infty} 2^{-\lfloor i/\mu X \rfloor} &\leq \sum_{i=0}^{\infty} \sum_{j=0}^{\mu X - 1} 2^{-\lfloor \frac{i\mu X + j}{\mu X} \rfloor} \\ &= \sum_{i=0}^{\infty} \mu X \cdot 2^{-i} = 2\mu X \end{aligned}$$

∎

THEOREM 9. *Let X be the sample complexity of a conservative and set-driven learning algorithm. Then $V[X] \leq 2\mu X(3\mu X - E[X]) \leq 5(\mu X)^2$.*

Proof.

$$\begin{aligned}
V[X] = E[X^2] - E^2[X] &= \sum_{i=0}^{\infty} i^2 \cdot \Pr(X = i) - E^2[X] \\
&= \sum_{i=0}^{\infty} i^2 \cdot \big(\Pr(X \geq i) - \Pr(X \geq i+1)\big) - E^2[X] \\
&= \underbrace{\sum_{i=0}^{\infty} i^2 \Pr(X \geq i) - \sum_{i=0}^{\infty} (i+1)^2 \Pr(X \geq i+1)}_{=0} \\
&+ \sum_{i=0}^{\infty} (2i+1) \Pr(X \geq i+1) - E^2[X] \\
&= \sum_{i=1}^{\infty} (2i-1) \cdot \Pr(X \geq i) - E[X] \sum_{i=1}^{\infty} \Pr(X \geq i) \\
&= \sum_{i=1}^{\infty} (2i - 1 - E[X]) \Pr(X \geq i) \\
&\leq \sum_{i=1}^{\infty} (2i - 1 - E[X]) \cdot 2^{-\lfloor i/\mu X \rfloor} \text{ (by Theorem 6)} \\
&\leq \sum_{i=0}^{\infty} \sum_{j=0}^{\mu X - 1} (2(i\mu X + j) - 1 - E[X]) \cdot 2^{-\left\lfloor \frac{i\mu X + j}{\mu X} \right\rfloor} \\
&= \sum_{i=0}^{\infty} \mu X (2(i+1)\mu X - E[X] - 2) \cdot 2^{-i} \\
&= 2\mu X(3\mu X - E[X] - 2) \\
&\leq 2\mu X(3\mu X - E[X]) \leq 5(\mu X)^2
\end{aligned}$$

∎

3.2. *The Sample Complexity*

In this section we estimate the sample complexity. While being of interest itself, whenever acquiring examples is expensive, $E[C]$ is also an important ingredient in the estimation of the total learning time. In estimating $E[C]$, we first need $E[M_C]$, which we get from tail bounds of M_C given in the following lemma.

LEMMA 1. $\Pr(M_C > m) = \Pr(C > r \mid M_r = m) \leq \binom{k}{2}\beta^m + k\beta^{m/2}$ *for all* $m, r \in \mathbb{N}^+$ *with* $r \geq m$.

Proof. Without loss of generality, let $S_r = \{s_1, \ldots, s_m\}$, i.e., $m = r$. Additionally, we can make the assumption that all strings $s_i \in S_r$ have length k, since we need to consider only shortest words for M_C and we can assume that $\pi = x_0 x_2 \ldots x_{k-1}$ (cf.

[19]). For $1 \leq j \leq k$ let $c_j = s_0(j)s_1(j)\ldots s_{m-1}(j)$ be the jth *column* of a matrix whose rows are $s_1, \ldots, s_m$.

The algorithms computes the hypothesis π on input S_r iff no column is constant and there are no identical columns. The probability that c_j is constant is at most $\beta^{m/2}$, since $m/2$ pairs have to be identical. But this short argument works only for even m.

The probability that a column is constant, i.e., the probability that m independent random words are identical under the condition that each length is exactly 1, is

$$\sum_{a\in\mathcal{A}} d(a)^m \Big/ \left(\sum_{a\in\mathcal{A}} d(a)\right)^m.$$

In the following we show that this quantity is at most $\beta^{m/2}$. We start with the following inequality obtained by the multinomial theorem.

$$\sum_{a\in\mathcal{A}} d(a)^m \;=\; \sum_{a\in\mathcal{A}} d(a)^{2\cdot\frac{m}{2}} \;\leq\; \left(\sum_{a\in\mathcal{A}} d(a)^2\right)^{m/2}$$

Dividing both sides by $\left(\sum_{a\in\mathcal{A}} d(a)\right)^m$ yields

$$\begin{aligned}\frac{\sum_{a\in\mathcal{A}} d(a)^m}{\left(\sum_{a\in\mathcal{A}} d(a)\right)^m} &\leq \frac{\left(\sum_{a\in\mathcal{A}} d(a)^2\right)^{m/2}}{\left(\left(\sum_{a\in\mathcal{A}} d(a)\right)^2\right)^{m/2}} \\ &= \left(\sum_{a\in\mathcal{A}} d(a)^2 \Big/ \left(\sum_{a\in\mathcal{A}} d(a)\right)^2\right)^{m/2} = \beta^{m/2}\end{aligned}$$

The probability that at least one of the k columns is constant is then at most $k\beta^{m/2}$.

The probability that $c_i = c_j$ is β^m if $i \neq j$. The probability that some columns are equal is therefore at most $\binom{k}{2}\beta^m$. The probability that there is a constant column *or* that there are identical columns is at most $\binom{k}{2}\beta^m + k\beta^{m/2}$. ∎

Inserting the above tail bounds into the definition of the expected value yields an upper bound on $E[M_C]$.

LEMMA 2. $E[M_C] \leq \dfrac{2\ln(k)+3}{\ln(1/\beta)} + 2 \leq 7\log_{1/\beta}(k) + 2 = O(\log_{1/\beta}(k))$.

Proof. M_C is the number of shortest words read until convergence. By Lemma 1 we have $\Pr(M_C > m) \leq \binom{k}{2}\beta^m + k\beta^{m/2}$.

$$\begin{aligned}E[M_C] &= \sum_{m=0}^{\infty} \Pr(M_C > m) \\ &\leq \ell + \sum_{m=l}^{\infty}\left(\binom{k}{2}\beta^m + k\beta^{m/2}\right) \\ &= \ell + \binom{k}{2}\frac{\beta^\ell}{1-\beta} + k\frac{\sqrt{\beta}^\ell}{1-\sqrt{\beta}}\end{aligned}$$

for each natural number ℓ. We choose

$$\ell = \lceil 2\log_{1/\beta}(k) \rceil + 1,$$

which yields when inserted in above inequality

$$E[M_C] \quad \leq \quad \ell + \frac{\beta}{1-\beta} + \frac{\sqrt{\beta}}{1-\sqrt{\beta}}.$$

The lemma now follows from the inequality

$$\frac{\beta}{1-\beta} + \frac{\sqrt{\beta}}{1-\sqrt{\beta}} \quad \leq \quad \frac{3}{\ln(1/\beta)},$$

which can be proved by standard methods from calculus. ∎

Now, we already know the expectation for the number of strings from $L(\pi)_{min}$) the LWA has to read until convergence. Our next major goal is to establish an upper bound on the overall number of examples to be read on average by the LWA until convergence. This is done by the next theorem.

THEOREM 10. $E[C] = \hat{\alpha}^k E[M_C] \leq \hat{\alpha}(7\log_{1/\beta}(k) + 2) = O(\hat{\alpha}^k \log_{1/\beta}(k))$.

Proof. The LWA converges after reading exactly C example strings. Among these examples are M_C many of minimum length. Prior to these minimum length words come M_C possibly empty blocks of words whose length is bigger than $|\pi|$. Let us call the numbers of those words in the i th block G_i. Then $C = G_1 + G_2 + \cdots + G_{M_C} + M_C$. It is easy to compute the distribution of G_i:

$$\Pr(G_i = m) = \Pr(\Lambda > |\pi|)^m \Pr(\Lambda = |\pi|) = (1-\alpha^k)^m \alpha^k \tag{1}$$

Of course, all G_i are identically distributed and independent. The expected value of C is therefore

$$\begin{aligned} E[C] &= E[M_C] + E[G_1 + \cdots + G_{M_C}] \\ &= E[M_C] + \sum_{m=0}^{\infty} E[G_1 + \cdots + G_m \mid M_C = m] \cdot \Pr(M_C = m) \\ &= E[M_C] + \sum_{m=0}^{\infty} m \cdot E[G_1] \cdot \Pr(M_C = m) \\ &= E[M_C] + E[M_C] \cdot E[G_1] \end{aligned} \tag{2}$$

The expected value of G_1 is

$$E[G_1] = \sum_{m=0}^{\infty} m \cdot \Pr(\Lambda > |\pi|)^m \cdot \Pr(\Lambda = |\pi|) = \frac{\Pr(\Lambda > |\pi|)}{\Pr(\Lambda = |\pi|)} = \frac{1-\alpha^k}{\alpha^k} \tag{3}$$

Combining (2) and (3) with $E[M_C] \leq 7\log_{1/\beta}(k) + 2$ from Lemma 2 proves the theorem. ∎

3.3. *The Length of the Text until Convergence*

Now, we are almost done. For establishing the main theorem, i.e., the expected total learning time, it suffices to calculate the expected length of a randomly generated text until the LWA converges.

LEMMA 3. Let $m \geq 1$. Then $E[\Lambda_1 \mid G_1 = m] = (E[\Lambda] - \alpha^k)/(1 - \alpha^k)$.

Proof. This proof is based on $\Pr(\Lambda_1 = i \mid G_1 = m) = \Pr(\Lambda_1 \mid \Lambda_1 > |\pi|)$, which intuitively holds because for Λ_1 the condition $G_1 = m$ means that the first block of non-minimum length strings is not empty and this holds iff Λ_1 has not minimum length. More formally note that $G_1 = m$ is equivalent to $\Lambda_i > |\pi|$ for $1 \leq i \leq m$ and $\Lambda_{m+1} = |\pi|$ and therefore

$$E[\Lambda_1 \mid G_1 = m] = E[\Lambda_1 \mid \Lambda_1 > |\pi| \wedge \cdots \wedge \Lambda_m > |\pi| \wedge \Lambda_{m+1} = |\pi|],$$

but since all Λ_i are independent it boils down to $E[\Lambda_1 \mid G_1 = m] = E[\Lambda_1 \mid \Lambda_1 > |\pi|]$.

Now it is easy to compute $E[\Lambda_1 \mid G_1 = m]$:

$$\begin{aligned}
E[\Lambda_1 \mid G_1 = m] &= \sum_{i=|\pi|+1}^{\infty} i \cdot \Pr(\Lambda_1 = i \mid G_1 = m) \\
&= \sum_{i=|\pi|+1}^{\infty} i \cdot \Pr(\Lambda_1 = i \mid \Lambda_1 > |\pi|) = \sum_{i=|\pi|+1}^{\infty} i \cdot \frac{\Pr(\Lambda_1 = i)}{\Pr(\Lambda_1 > |\pi|)} \\
&= \frac{E[\Lambda_1]}{\Pr(\Lambda_1 > |\pi|)} - \frac{\Pr(\Lambda_1 = |\pi|)}{\Pr(\Lambda_1 > |\pi|)} \\
&= \frac{E[\Lambda] - \alpha^k}{1 - \alpha^k}
\end{aligned}$$

∎

THEOREM 11.

$$\begin{aligned}
E[T] &= E[M_C] \cdot \big(|\pi| + \hat{\alpha}^k (E[\Lambda] - 1)\big) \\
&\leq (7 \log_{1/\beta}(k) + 2)\big(|\pi| + \hat{\alpha}^k (E[\Lambda] - 1)\big) = O\big(\hat{\alpha}^k (\log_{1/\beta}(k) + 2) E[\Lambda]\big).
\end{aligned}$$

Proof. We can write the length of text read until convergence as $T = T_1 + T_2 + \cdots + T_{M_C} + |\pi| M_C$. Exactly M_C strings of length $|\pi|$ are read; all other strings are longer and are contained in blocks in front of those minimum length strings. The ith blocks contains G_i strings and we denote the total length of these G_i strings by T_i. In order to get $E[T]$ we start by computing $E[T_1]$.

$$\begin{aligned}
E[T_1] &= \sum_{m=0}^{\infty} E[\Lambda_1 + \cdots + \Lambda_m \mid G_1 = m] \cdot \Pr(G_1 = m) \\
&= \sum_{m=1}^{\infty} m \cdot E[\Lambda_1 \mid G_1 = m] \cdot \Pr(G_1 = m) \\
&= \sum_{m=1}^{\infty} m \cdot \frac{E[\Lambda] - \alpha^k}{1 - \alpha^k} \cdot (1 - \alpha^k)^m \alpha^k \quad \text{(by Lemma 3 and (1))} \\
&= (E[\Lambda] - \alpha^k)\alpha^k \sum_{m=1}^{\infty} m (1 - \alpha^k)^{m-1} \\
&= \hat{\alpha}^k E[\Lambda] - 1
\end{aligned}$$

Now it is easy to estimate $E[T]$. We use that T_1 and M_C are independent.

$$\begin{aligned} E[T] - |\pi| E[M_C] &= E[T_1 + \cdots + T_{M_C}] \\ &= \sum_{m=0}^{\infty} m \cdot E[T_1] \cdot \Pr(M_C = m) = E[M_C] \cdot E[T_1] \end{aligned}$$

and thus

$$E[T] = E[M_C]\big(|\pi| + \hat{\alpha}^k E[\Lambda] - 1\big).$$

Finally insert the estimation of $E[M_C]$ from Lemma 2. ∎

4. Learning Stochastically Finite with High Confidence

Now we are ready to introduce the new learning model mentioned in the Introduction.

DEFINITION 2. Let D be an admissible probability distribution. PAT is called *stochastically finite learnable with high confidence from text* iff there is an IIM M such that for every $L \in PAT$, every randomly generated text t with respect to D for L, and every number $\delta \in (0,1)$ there exists a pattern $\pi \in Pat$ such that M, when successively fed t, outputs the single hypothesis π, $L(\pi) = L$ with probability at least δ, and stops thereafter.

Note that the learner in the definition above takes δ as additional input.

Next, we show how the LWA can be transformed into a stochastically finite learner that identifies all the pattern languages with high confidence provided we have a bit of prior knowledge about the class of admissible distributions that may actually be used to generate the information sequences.

THEOREM 12. *Let α_*, $\beta_* \in (0,1)$. Assume $\mathcal{D}$ to be a class of admissible probability distributions over $\mathcal{A}^+$ such that $\alpha \geq \alpha_*$, $\beta \geq \beta_*$ and $E(d)$ finite for all distributions $d \in \mathcal{D}$. Then PAT is stochastically finitely learnable with high confidence for all admissible probability distributions D generated by any $d \in \mathcal{D}$ from text.*

Proof. Let $d \in \mathcal{D}$ and $\delta \in (0,1)$ be arbitrarily fixed. Note that d induces an admissible probability distribution D. Furthermore, let $t = s_1, s_2, s_3, \ldots$ be any randomly generated text with respect to D for the target pattern language. The wanted learner M uses the the LWA as a subroutine. Additionally, it has a counter for memorizing the number of examples already seen. Now, we exploit the fact that the LWA produces a sequence $(\tau_n)_{n \in \mathbb{N}^+}$ of hypotheses such that $|\tau_n| \geq |\tau_{n+1}|$ for all $n \in \mathbb{N}^+$.

The learner runs the LWA until for the first time C many examples have been processed, where

$$C = O(\hat{\alpha_*}^{|\tau|} \log_{1/\beta_*}(|\tau|)) \qquad (A)$$

and τ is the actual output made by the LWA. The precise constants hidden in the expression given in (A) can be readily computed as soon as the precise machine is known on which we want to run the algorithm.

Finally, in order to achieve the desired confidence, the learner computes the least $t \in \mathbb{N}^+$ such that $1-2^{-t} \geq \delta$. It then continues running the LWA for $2t \cdot C$ additional inputs. This is the reason we need the counter for the numer of examples processed. Now, it outputs the last hypothesis τ produced by the LWA, and stops thereafter.

Clearly, the learner described above is finite. Let L be the target language and let $\pi \in Pat_k$ be the unique pattern such that $L = L(\pi)$. It remains to argue that $L(\pi) = L(\tau)$ with probability at least δ. First, the bound in (A) upper bounds the expected number of examples needed for convergence by the LWA that has been established in Theorem 10. This follows from our assumptions about the allowed α and β as well as from the fact that $|\tau| \geq |\pi|$ for every hypothesis output and because of $\#var(\pi) \leq |\pi|$. Therefore, after having processed C many examples the LWA has already converged on average. The desired confidence is then an immediate consequence of Corollary 7. ∎

The latter theorem allows a nice corollary which we state next. Making the same assumption as done by Kearns and Pitt [8], i.e., assuming the additional prior knowledge that the target pattern belongs to Pat_k, we arrive at a stochastically finite linear-time learner for PAT_k. This is a major improvement, since the constant depending on k grows only exponentially in k in contrast to the doubly exponentially growing constant in Kearns and Pitt's [8] algorithm.

COROLLARY 13. *Let α_*, $\beta_* \in (0,1)$. Assume $\mathcal{D}$ to be a class of admissible probability distributions over $\mathcal{A}^+$ such that $\alpha \geq \alpha_*$, $\beta \geq \beta_*$ and $E(d)$ finite for all distributions $d \in \mathcal{D}$. Furthermore, let $k \in \mathbb{N}^+$ be arbitrarily fixed. Then there exist a learner M such that*

(1) *M learns PAT_k stochastically finitely with high confidence from text for all admissible probability distributions D generated by any $d \in \mathcal{D}$, and*

(2) *The running time of M is bounded by $O\big(\hat{\alpha}^k(\log_{1/\beta}(k)+2)E[\Lambda]\big)$, where the constants hidden depend exponentially on k and linearly on $\log(1/\delta)$.*

Proof. The learner works precisely as in the proof of Theorem 12 except that (A) is replaced by

$$C = O(\hat{\alpha_*}^k \log_{1/\beta_*}(k)) \qquad (A')$$

The correctness follows as above by Theorem 10 and Corollary 7, since the target belongs to Pat_k. The running time is a direct consequence of Theorem 11 and the choice of t. ∎

One more remark is mandatory here. The learners described above can be made more efficiently by using even better tail bounds. We therefore continue to establish some more tail bounds. Note that each of these bounds has a special range where it outperforms the other ones. Hence, the concrete choice in an actual implementation of the algorithm above depends on the precise values of α and β. Since these values are usually not known precisely, it is advantageous to take the minimum of all three.

LEMMA 4. $\Pr(M_r \geq er\alpha^k) \leq e^{-r\alpha^k}$ and $\Pr(M_r \leq \frac{1}{2}r\alpha^k) \leq (e/2)^{-r\alpha^k/2}$.

Proof. The expected value of M_r is $r\alpha^k$, since $\Pr(\Lambda = k) = \alpha^k$. Chernoff bounds [6, (12)] yield

$$\Pr(M_r \geq er\alpha^k) \leq \left(\frac{r\alpha^k}{er\alpha^k}\right)^{er\alpha^k} e^{er\alpha^k - r\alpha^k} = e^{-r\alpha^k}$$

and

$$\Pr(M_r \leq \tfrac{1}{2}r\alpha^k) \leq \left(\frac{r\alpha^k}{r\alpha^k/2}\right)^{r\alpha^k/2} e^{r\alpha^k/2 - r\alpha^k} = (e/2)^{-r\alpha^k/2}.$$

∎

THEOREM 14. $\Pr(C > r) \leq k^2\beta^{\frac{1}{4}r\alpha^k} + (e/2)^{-r\alpha^k/2}$.

Proof. We split $\Pr(C > r)$ into a sum of conditional probabilities according to the conditions $M_r \geq m$ and $M_r < m$ for a well chosen parameter m. We use the fact that

$$\Pr(C > r \mid C \geq m) \geq \Pr(C > r \mid M_r = m), \tag{4}$$

which is quite obvious.

$$\begin{aligned}
\Pr(C > r) &= \Pr(C > r \mid M_r \geq m)\Pr(M_r \geq m) \\
&+ \Pr(C > r \mid M_r < m)\Pr(M_r < m) \\
&\leq \Pr(C > r \mid M_r = m) + \Pr(M_r \leq m) \\
&= \Pr(M_C > m) + \Pr(M_r \leq m) \quad \text{(by the first part of Lemma 1)}
\end{aligned}$$

We choose $m = \frac{1}{2}r\alpha^k$ and get

$$\Pr(C > r) \leq k^2\beta^{\frac{1}{4}r\alpha^k} + (e/2)^{-r\alpha^k/2}$$

by Lemma 1 and 4. ∎

THEOREM 15. $\Pr(C > r) \leq k^2 e^{-\alpha^k(1-\beta)r}$.

Proof. Using $\Pr(C > r \mid M_r = m) = \Pr(M_C > m \mid M_r = m) = \Pr(M_C > m)$ we can write $\Pr(C > r)$ as a sum of products:

$$\Pr(C > r) = \sum_{m=0}^{r} \Pr(M_C > m) \cdot \Pr(M_r = m).$$

Now $\Pr(M_C > m) \leq k^2\beta^{m/2}$ by Lemma 1 and $\Pr(M_r = m) = \binom{r}{m}\alpha^{km}(1-\alpha^k)^{r-m}$, since M_r has a binomial distribution with parameters α^k and $1-\alpha^k$. Using these estimates we get immediately

$$\begin{aligned}
\Pr(C > r) &\leq \sum_{m=0}^{r} \binom{r}{m} k^2\beta^{m/2}\alpha^{km}(1-\alpha^k)^{r-m} \\
&= k^2\left(\sqrt{\beta}\alpha^k + 1 - \alpha^k\right)^r \\
&= k^2\left(1 - \alpha^k(1-\sqrt{\beta})\right)^r \\
&\leq k^2\left(\frac{1}{e}\right)^{\alpha^k(1-\sqrt{\beta})r}
\end{aligned}$$

and the theorem is proved. ∎

5. A closer Look at the Union Operations

Until now we have assumed that the union operations are performed by our new linear time algorithm. While this algorithm optimizes the running time needed to perform the union operations it does not simultaneously optimize the space needed. Implementing it naïvely will result in using quadratic space. More sophisticated techniques like hashing may reduce this space bound on average. But is it really worthwhile to do this? For answering this question we continue with a closer look at both the number of union operations performed and the total time spent for performing union operations.

5.1. *The Number of Union Operations*

LEMMA 5. If $m > |\pi|$ then $\Pr(\Lambda = m)/\Pr(\Lambda < m) \leq \hat{\alpha}$.

Proof. Remember that Λ is the length of a random string $\pi[x_1/u_1, \ldots, x_k/u_k]$, where the u_i's are random strings from $\mathcal{A}^+$ according to the probability distribution d. Whether $\Lambda = m$ depends only on the lengths of the substituted strings and we will work only with $|u_1|, \ldots, |u_k|$ and with *length vectors* from $\mathbb{N}^k$. Let M be the set of length vectors that lead to $\Lambda = m$, i.e.,

$$M = \{\, (|v_1|, \ldots, |v_k|) \mid v_i \in \mathcal{A}^+ \text{ and } |\pi[x_1/v_1, \ldots, x_k/v_k]| = m \,\}.$$

Let $\mathbf{n} = (n_1, \ldots, n_k) \in \mathbb{N}^k$ and $[\mathbf{n}] = \min\{\, i \mid n_i \neq 1 \,\}$ ($[\mathbf{n}]$ is well defined for $\mathbf{n} \in M$ since $m > |\pi|$). Define $\mathbf{u} = (|u_1|, \ldots, |u_k|)$ and

$$f(\mathbf{n}) = (n_1, n_2, \ldots, n_{[\mathbf{n}]-1}, 1, n_{[\mathbf{n}]+1}, \ldots, n_l) \text{ and } M' = \{\, f(\mathbf{n}) \mid \mathbf{n} \in M \,\}.$$

Of course, $\Pr(\mathbf{u} = \mathbf{n}) = d(\mathcal{A}^{n_1})d(\mathcal{A}^{n_2}) \cdots d(\mathcal{A}^{n_k})$ and in the same way $\Pr(\mathbf{u} = f(\mathbf{n})) = d(\mathcal{A}^{n_1})d(\mathcal{A}^{n_2}) \cdots d(\mathcal{A}^{n_k}) \cdot d(\mathcal{A}^1)/d(\mathcal{A}^{n_{[\mathbf{n}]}})$ and therefore

$$\frac{\Pr(\mathbf{u} = \mathbf{n})}{\Pr(\mathbf{u} = f(\mathbf{n}))} = \frac{d(\mathcal{A}^{n_{[\mathbf{n}]}})}{d(\mathcal{A}^1)} = \frac{d(\mathcal{A}^{n_{[\mathbf{n}]}})}{\alpha} \leq \hat{\alpha}$$

and consequently $\Pr(\mathbf{u} \in M)/\Pr(\mathbf{u} \in f(M)) \leq \hat{\alpha}$. Now note that $\Pr(\mathbf{u} \in M) = \Pr(\Lambda = m)$ and $\Pr(\mathbf{u} \in f(M)) \leq \Pr(\Lambda < m)$. ∎

LEMMA 6. Let $0 \leq z_i \leq \hat{\alpha}$ for $1 \leq i \leq m$ and $s = z_1 + z_2 + \cdots + z_m$. Then

$$(1 + \hat{\alpha})^{s/\hat{\alpha}} \leq (1 + z_1)(1 + z_2) \cdots (1 + z_m). \tag{5}$$

Proof. Since $(1 + 1/x)^x$ grows monotonically for $x > 0$ (with limit e), $(1 + x)^{1/x}$ decreases monotonically and for $0 \leq z_i \leq \hat{\alpha}$ we get

$$(1 + \hat{\alpha})^{1/\hat{\alpha}} \leq (1 + z_i)^{1/z_i}.$$

Raising both sides to the power z_i yields

$$(1 + \hat{\alpha})^{z_i/\hat{\alpha}} \leq 1 + z_i$$

and forming the product for $1 \leq i \leq m$ on both sides gives finally

$$\prod_{i=1}^{m} (1 + \hat{\alpha})^{z_i/\hat{\alpha}} \leq \prod_{i=1}^{m} (1 + z_i),$$

which is exactly the claim made by the lemma. ∎

LEMMA 7. $\displaystyle\sum_{i=|\pi|+1}^{\mu(\Lambda)} \frac{\Pr(\Lambda = i)}{\Pr(\Lambda < i)} < \hat{\alpha} \cdot k\,.$

Proof. Let $z_i = \dfrac{\Pr(\Lambda = i)}{\Pr(\Lambda < i)}$. Then

$$\begin{aligned}
&\Pr(\Lambda = |\pi|) \cdot (1 + z_{|\pi|+1})(1 + z_{|\pi|+2}) \cdots (1 + z_{\mu L}) \\
= \quad & \Pr(\Lambda = |\pi|) \cdot \frac{\Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1)}{\Pr(\Lambda = |\pi|)} \\
\cdot \quad & \frac{\Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1) + \Pr(\Lambda = |\pi| + 2)}{\Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1)} \cdots \\
\cdots \quad & \frac{\Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1) + \cdots + \Pr(\Lambda = \mu L)}{\Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1) + \cdots + \Pr(\Lambda = \mu L - 1)} \\
= \quad & \Pr(\Lambda = |\pi|) + \Pr(\Lambda = |\pi| + 1) + \cdots + \Pr(\Lambda = \mu L) = \Pr(\Lambda \leq \mu L) \leq 1 \\
\Longrightarrow \quad & (1 + z_{|\pi|+1})(1 + z_{|\pi|+2}) \cdots (1 + z_{\mu L}) \leq \frac{1}{\Pr(\Lambda = |\pi|)} = \hat{\alpha}^k \qquad (6)
\end{aligned}$$

By Lemma 5 we know $0 \leq z_i \leq \hat{\alpha}$ for $|\pi| + 1 \leq i \leq \mu L$. We can therefore apply Lemma 6 to (6) and get

$$\hat{\alpha}^k \geq (1 + z_{|\pi|+1})(1 + z_{|\pi|+2}) \cdots (1 + z_{\mu L}) \geq (1 + \hat{\alpha})^{s/\hat{\alpha}}, \qquad (7)$$

where $s = \displaystyle\sum_{i=|\pi|+1}^{\mu(\Lambda)} \frac{\Pr(\Lambda = i)}{\Pr(\Lambda < i)}$.

It remains to be shown that $s < \hat{\alpha} k$. In order to do so we start with (7) and solve for s:

$$\hat{\alpha}^k \geq (1 + \hat{\alpha})^{s/\hat{\alpha}} > \hat{\alpha}^{s/\hat{\alpha}} \quad \Longrightarrow \quad k > s/\hat{\alpha} \quad \Longrightarrow \quad s < \hat{\alpha} k$$

∎

If $\hat{\alpha}$ is very small, then Lemma 7 is quite pessimistic. Solving $\hat{\alpha}^k \geq (1 + \hat{\alpha})^{s/\hat{\alpha}}$ exactly, yields $s \leq \hat{\alpha} \cdot k \cdot \ln(\hat{\alpha}) / \ln(1 + \hat{\alpha})$. We will not use this better approximation, since it is more complicated. A small $\hat{\alpha}$ or, equivalently, an α near 1 means that the random strings according to u have length 1 with high probability. If this is the case, the above better approximation might be useful. Because then $\hat{\alpha} \approx \ln(1 + \hat{\alpha})$, thus we get $s \leq k \ln(\hat{\alpha})$.

Let $U = U_{|\pi|} + U' + U''$ with

$$U' = \sum_{l=|\pi|+1}^{\mu L} U_l \qquad \text{and} \qquad U'' = \sum_{l=\mu L+1}^{\infty} U_l.$$

We introduce a new random variable to be used in the next three lemmata. By N_l we denote the number of examples generated until a string with length smaller than l occurs, i.e.,

$$N_l = \min\{\, i \mid \Lambda_i < l \,\}.$$

LEMMA 8. If $l > |\pi|$, then $E[N_l] = 1/\Pr(N < l)$.

Proof. Trivial, since the expected number for carrying out identical experiments until a chosen result occurs is $1/p$, where p is the probability of the result. ∎

LEMMA 9. $E[U'] \leq 2\hat{\alpha}k$.

Proof. Let $|\pi| < l \leq \mu L$.

$$\begin{aligned} E[U_l \mid N_l = n] &= (n-1) \cdot \Pr(\Lambda_1 = l \mid N_l = n) \\ &= (n-1) \cdot \Pr(\Lambda_1 = l \mid \Lambda_1 \geq l) \\ &= (n-1) \cdot \Pr(\Lambda = l) / \Pr(\Lambda \geq l) \\ &\leq 2(n-1) \Pr(\Lambda = l) \end{aligned} \tag{8}$$

To get (8) we used $\Pr(\Lambda_1 = l \mid N_l = n) = \Pr(\Lambda = l \mid \Lambda_1 \geq l)$ and $\Pr(\Lambda_1 \geq l) \geq 1/2$.

Now

$$\begin{aligned} E[U_l] &= \sum_{n=1}^{\infty} E[U_l \mid N_l = n] \cdot \Pr(N_l = n) \\ &\leq 2 \Pr(\Lambda_1 = l) \cdot \sum_{n=1}^{\infty} (n-1) \Pr(N_l = n) \quad \text{(by (8))} \\ &\leq 2 \Pr(\Lambda_1 = l) \cdot E[N_l] \\ &= 2 \frac{\Pr(\Lambda = l)}{\Pr(\Lambda < l)} \quad \text{(by Lemma 8)} \end{aligned} \tag{9}$$

Lemma 7 together with (9) yields the desired bound on U':

$$U' = \sum_{l=|\pi|+1}^{\mu L} E[U_l] \leq \sum_{l=|\pi|+1}^{\mu L} 2 \frac{\Pr(\Lambda = l)}{\Pr(\Lambda < l)} < 2\hat{\alpha}k$$

∎

LEMMA 10. $E[U''] \leq 1$.

Proof. A union operation belonging to U'' is performed on a string longer than μL. Only the first $N_{\mu L+1} - 1$ strings are longer than μL. So clearly $U'' \leq N_{\mu L+1} - 1$ and consequently $E[U''] \leq E[N_{\mu L+1}] - 1$. Lemma 8 yields $E[N_{\mu L+1}] = 1/\Pr(\Lambda \leq \mu L) \leq 2$. ∎

Combining Lemmata 9, 10, and 2 and the fact that $C_N = U_{|\pi|} + 1$ we get the following estimate of the average number of union operations.

THEOREM 16. $E[U] \leq 2\hat{\alpha}k + E[C_N] \leq 2\hat{\alpha}k + 7\log_{1/\beta}(k) + 2$.

5.2. *The Time Spent for Union Operations*

THEOREM 17. $E[V] \leq c \cdot \left(2\hat{\alpha}k(\mu L)^2 + E[C_N]|\pi|^2 + 12E^2[\Lambda]\right)$.

Proof. We estimate $E[V] = E[V' + V'' + V_{|\pi|}]$ by looking at V', V'', and $V_{|\pi|}$ separately. In the case of V' the union operation is performed on patterns and strings whose length is at most μL. Therefore $V' \leq c(\mu L)^2 \cdot U'$. In the case of $V_{|\pi|}$ unions are performed on patterns and strings of length exactly $|\pi|$ and thus $V_{|\pi|} \leq c|\pi|^2 U_{|\pi|}$. Hence, $E[V'] \leq 2c\hat{\alpha}k(\mu L)^2$ (Lemma 9) and $E[V_{|\pi|}] \leq (3c\log_{1/\beta(k)} + 1)|\pi|^2$ (Lemma 2 and $U_{|\pi|} = C_N - 1$).

To estimate $E[V'']$ is more complicated. Let $l > \mu L$. We will need the following inequality

$$\frac{\Pr(\Lambda = l)}{\Pr(\Lambda \leq l)} \leq \frac{1}{2} \tag{10}$$

which follows easily from $\Pr(\Lambda \leq l) \geq \Pr(\Lambda \leq \mu L) + \Pr(\Lambda = l) \geq 1/2 + \Pr(\Lambda = l)$ and $\Pr(\Lambda = l) \leq \frac{1}{2}$.

Now let U_l be the number of union operations performed on patterns of length l. Then $U_l = t$ iff exactly $t+1$ positive examples of length t precede the first example of length shorter than t. In between there are $t+2$ possibly empty blocks of examples whose length is more than t. From this observation we can compute the probability of $U_l = t$:

$$\begin{aligned}
\Pr(U_l = t) &= \Pr(\Lambda = l)^{t+1} \cdot \left(\sum_{i=0}^{\infty} \Pr(\Lambda > l)^i\right)^{t+2} \cdot \Pr(\Lambda < l) \\
&= \frac{\Pr(\Lambda = l)^{t+1}}{(1 - \Pr(\Lambda > l))^{t+2}} \cdot \Pr(\Lambda < l) \\
&\leq \frac{\Pr(\Lambda = l)^{t-1}}{\Pr(\Lambda \leq l)^{t-1}} \cdot \frac{\Pr(\Lambda < l)}{\Pr(\Lambda \leq l)^3} \cdot \Pr(\Lambda = l)^2 \\
&\leq 2^{-(t-1)} \cdot 4 \cdot \Pr(\Lambda = l)^2 \\
&= 2^{-(t-3)} \cdot \Pr(\Lambda = l)^2
\end{aligned}$$

Next we estimate the expected value of U_l.

$$E[U_l] = \sum_{t=2}^{\infty} t \cdot \Pr(U_l = t) \leq \sum_{t=2}^{\infty} t \cdot 2^{-(t-3)} \cdot \Pr(\Lambda = l)^2 = 12 \cdot \Pr(\Lambda = l)^2$$

The expectation of V'' is then at most

$$\begin{aligned}
E[V''] &= \sum_{l > \mu L} cl^2 \cdot E[U_l] \leq \sum_{l > \mu L} 12cl^2 \cdot \Pr(\Lambda = l)^2 \\
&\leq 12c \cdot \left(\sum_{l=1}^{\infty} l \cdot \Pr(\Lambda = l)\right)^2 = 12c \cdot E^2[\Lambda]
\end{aligned}$$

Note that the above estimate is rather pessimistic unless the distribution of L above μL is nearly concentrated at one point. ∎

We finish this subsection by restating the bounds given in Theorem 2 and refuniformTT for the case that the union operations are performed by the naïve algorithm.

THEOREM 18. *$E[TT] = O\big(\log_{1/\beta}(k) \cdot (\hat{\alpha}^k E[\Lambda] + |\pi|^2) + k\hat{\alpha}E^2[\Lambda]\big)$ provided the union operations are performed by the naïve quadratic-time algorithm.*

Proof. The proof is a direct consequence of Theorems 17 and 11. ∎

THEOREM 19. *$E[TT] = O(2^k|\pi|\log_{|\mathcal{A}|}(k) + k|\pi|^2)$ for the uniform distribution provided the union operations are performed by the naïve quadratic-time algorithm.*

A comparison of the latter two theorems and their counterparts, i.e., Theorem 2 and refuniformTT show that, if space is a serious matter of concern, one may easily

trade a bit more time by using the naïve, quadratic time algorithm instead of Algorithm 1. The difference between the established bounds helps to make this decision.

6. Conclusions

The present paper dealt with the average-case analysis of Lange and Wiehagen's pattern language learning algorithm with respect to its total learning time. The results presented considerably improved the analysis made by Zeugmann [19]. Clearly, the question arises whether the improvement is worth the effort undertaken to obtain it. This question has been naturally answered by the introduction of our new model of stochastically finite learnability with high confidence. Thus, the present paper provides evidence that analyzing the average-case behavior of limit learners with respect to their total learning time may be considered as a promising path towards a theory of efficient algorithmic learning. Recently obtained results along the same path as outlined in Erlebach *et al.* [4] as well as in Reischuk and Zeugmann [13] provide further support for the fruitfulness of this approach.

Moreover, the approach undertaken may also provide the necessary tools to perform the average-case analysis of a wider variety of learning algorithms. In particular, the new approach undertaken to estimate the average-case behavior via showing very useful tail bounds seems to be generalizable to the large class of conservative and rearrangement-independent learning algorithms.

References

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

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

[3] R. Daley and C.H. Smith. On the complexity of inductive inference. *Information and Control*, 69:12–40, 1986.

[4] T. Erlebach, P. Rossmanith, H. Stadtherr, A. Steger, and T. Zeugmann. Efficient learning of one-variable pattern languages from positive examples. DOI Technical Report DOI-TR-128, Department of Informatics, Kyushu University, December 1996.

[5] E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[6] T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33:305–308, 1990.

[7] J.E. Hopcroft and J.D. Ullman. *Formal Languages and their Relation to Automata.* Addison-Wesley, Reading, Massachusetts, 1969.

[8] M. Kearns and L. Pitt. A polynomial-time algorithm for learning k–variable pattern languages from examples. In R. Rivest, D. Haussler and M.K. Warmuth, editors, *Proc. 2nd Annual ACM Workshop on Computational Learning Theory* pp. 57–71, 1991, Morgan Kaufmann Publishers Inc., San Mateo.

[9] Ker-I Ko, A. Marron and W.G. Tzeng. Learning string patterns and tree patterns from examples. In B.W. Porter and R.J. Mooney, editors, *Proc. 7th International Conference on Machine Learning*, pp. 384–391, 1990, Morgan-Kaufmann Publishers Inc., San Mateo.

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

[11] S. Lange and T. Zeugmann. Set-driven and rearrangement-independent learning of recursive languages. *Mathematical Systems Theory*, 29:599–634, 1996.

[12] L. Pitt. Inductive inference, DFAs and computational complexity. In K.P. Jantke, editor, *Proc. Analogical and Inductive Inference*, Lecture Notes in Artificial Intelligence 397, pp. 18–44, Berlin, 1989, Springer-Verlag.

[13] R. Reischuk and T. Zeugmann. Learning One-Variable Pattern Languages in Linear Average Time DOI Technical Report DOI-TR-140, Department of Informatics, Kyushu University, September 1997.

[14] A. Salomaa. Patterns. (The Formal Language Theory Column). EATCS Bulletin, 54:46–62, 1994.

[15] A. Salomaa. Return to patterns. (The Formal Language Theory Column). EATCS Bulletin, 55:144–157, 1994.

[16] R.E. Schapire. Pattern languages are not learnable. In M.A. Fulk and J. Case, editors, *Proc. 3rd Annual ACM Workshop on Computational Learning Theory*, pp. 122–129, 1990. Morgan Kaufmann Publishers Inc., San Mateo.

[17] T. Shinohara and S. Arikawa. Pattern inference. In "Algorithmic Learning for Knowledge-Based Systems" K. P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 259–291, Berlin, 1995. Springer-Verlag.

[18] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition.* MIT Press, Cambridge, MA, 1980.

[19] T. Zeugmann. Lange and Wiehagen's pattern 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.