On the Amount of Nonconstructivity in Learning Formal Languages from Positive Data

Sanjay Jain[1], Frank Stephan[2], and Thomas Zeugmann[3]

[1] Department of Computer Science
National University of Singapore, Singapore 117417
sanjay@comp.nus.edu.sg
[2] Department of Computer Science and Department of Mathematics
National University of Singapore, Singapore 117543
fstephan@comp.nus.edu.sg
[3] Division of Computer Science, Hokkaido University
N-14, W-9, Sapporo 060-0814, Japan
thomas@ist.hokudai.ac.jp

Abstract. Nonconstructive computations by various types of machines and automata have been considered by e.g., Karp and Lipton [18] and Freivalds [9, 10]. They allow to regard more complicated algorithms from the viewpoint of more primitive computational devices. The amount of nonconstructivity is a quantitative characterization of the distance between types of computational devices with respect to solving a specific problem.
This paper studies the amount of nonconstructivity needed to learn classes of formal languages from positive data. Different learning types are compared with respect to the amount of nonconstructivity needed to learn indexable classes and recursively enumerable classes, respectively, of formal languages from positive data. Matching upper and lower bounds for the amount of nonconstructivity needed are shown.

1 Introduction

The research subject studied in this paper derives its motivation from various sources which we shortly present below. Nonconstructive methods of proof in mathematics have a rather long and dramatic history. The debate was especially passionate when mathematicians tried to overcome the crisis concerning the foundations of mathematics.

The situation changed slightly in the forties of the last century, when nonconstructive methods found their way to discrete mathematics. In particular, Paul Erdős used nonconstructive proofs masterly, beginning with the paper [8].

Another influential paper was Bārzdiņš [4], who introduced the notion of advice in the setting of Kolmogorov complexity of recursively enumerable sets. Karp and Lipton [18] introduced the notion of a Turing machine that takes advice to understand under what circumstances nonuniform upper bounds can be used to obtain uniform upper bounds. Damm and Holzer [7] adapted the

notion of advice for finite automata. Later Cook and Krajiček [6] initiated the study of proof systems that use advice for the verification of proofs. Even more recently, Beyersdorff *et al.* [5] continued along this line of research.

Quite often, we experience that finding a proof for a new deep theorem is triggered by a certain amount of inspiration. Being inspired does not mean that we do not have to work hard in order to complete the proof and to elaborate all the technical details. However, this work is quite different from enumerating all possible proofs until we have found the one sought for. Also, as experience shows, the more complicated the proof, the higher is the amount of inspiration needed. These observations motivated Freivalds [9, 10] to introduce a qualitative approach to measure the amount of nonconstructivity (or advice) in a proof. Analyzing three examples of nonconstructive proofs led him to a notion of nonconstructive computation which can be used for many types of automata and machines and which essentially coincides with Karp and Lipton's [18] notion when applied to Turing machines.

As outlined by Freivalds [9, 10], there are several results in the theory of inductive inference of recursive functions which suggest that the notion of nonconstructivity may be worth a deeper study in this setting, too. Subsequently, Freivalds and Zeugmann [11] introduced a model to study the amount of nonconstructivity needed to learn recursive functions.

The present paper generalizes the model of Freivalds and Zeugmann [11] to the inductive inference of formal languages. We aim to characterize the difficulty to learn classes of formal languages from positive data by using the *amount of nonconstructivity* needed to learn these classes. We shortly describe this model. The learner receives growing initial segments of a text for the target language L, where a text is any infinite sequence of strings and a special pause symbol # such that the range of the text minus the pause symbol contains all strings of L and nothing else. In addition, the learner receives as a second input a bitstring of finite length which we call *help-word*. If the help-word is correct, the learner learns in the desired sense. Since there are infinitely many languages to learn, a parameterization is necessary, i.e., we allow for every n a possibly different help-word and we require the learner to learn every language contained in $\{L_0, \ldots, L_n\}$ with respect to the hypothesis space $(L_i)_{i \in \mathbb{N}}$ chosen (cf. Definition 6). The difficulty of the learning problem is then measured by the length of the help-words needed, i.e., in terms of the growth rate of the function d bounding this length. As in previous approaches, the help-word does *not* just provide an answer to the learning problem. There is still much work to be done by the learner.

First, we consider the learnability of indexable classes in the limit from positive data and ask for the amount of nonconstructivity needed to learn them. This is a natural choice, since even simple indexable subclasses of the class of all regular languages are known *not* to be inferable in the limit from positive data (cf. [13, 15, 23]). Second we investigate the amount of nonconstructivity needed to infer recursively enumerable classes of recursively enumerable languages. Moreover, several variations of Gold's [13] model of learning in the limit have been considered (cf., e.g., [15, 21, 23] and the references therein). Thus, it

is only natural to consider some of these variations, too. In particular, we shall study *conservative* learning and *strong-monotonic* inference.

We show upper and lower bounds for the amount of nonconstructivity in learning classes of languages from positive data. The usefulness of this approach is nicely reflected by our results which show that the function d may considerably vary. In particular, the function d may be arbitrarily slow growing for learning indexable classes in the limit from positive data (cf. Theorem 1), while we have an upper bound of $\log n$ and a lower bound of $\log n - 2$ for conservative learning of indexable classes from positive data (cf. Theorems 2 and 3). Furthermore, we have a $2 \log n$ upper bound and a $2 \log n - 4$ lower bound for strong-monotonic inference of indexable classes from positive data (cf. Theorems 4 and 5).

Moreover, the situation changes considerably when looking at recursively enumerable classes of recursively enumerable languages. For learning in the limit from positive data we have an upper bound of $\log n$ and a lower bound of $\log n - 2$, while for conservative learning even any limiting recursive bound on the growth of the function d is not sufficient to learn all recursively enumerable classes of recursively enumerable languages from positive data (cf. Theorems 7, 8 and 9). Due to the lack of space several proofs and details are omitted. A full version of this paper is available as technical report (cf. [16]).

2 Preliminaries

Any unspecified notations follow Rogers [22]. In addition to or in contrast with Rogers [22] we use the following. By $\mathbb{N} = \{0, 1, 2, \dots\}$ we denote the set of all natural numbers, and we set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$.

The cardinality of a set S is denoted by $|S|$. We write $\wp(S)$ for the power set of set S. Let $\emptyset$, $\in$, $\subset$, and $\subseteq$ denote the empty set, element of, proper subset, and subset, respectively. Let S_1, S_2 be any sets; then we write $S_1 \triangle S_2$ to denote the symmetric difference of S_1 and S_2, i.e., $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. By $\max S$ and $\min S$ we denote the maximum and minimum of a set S, respectively, where, by convention, $\max \emptyset = 0$ and $\min \emptyset = \infty$.

We use $\mathfrak{T}$ to denote the set of all total functions of one variable over $\mathbb{N}$. Let $n \in \mathbb{N}^+$; then the set of all partial recursive functions and of all recursive functions of one and n variables over $\mathbb{N}$ is denoted by $\mathcal{P}$, $\mathcal{R}$, $\mathcal{P}^n$, $\mathcal{R}^n$, respectively. Let $f \in \mathcal{P}$, then we use $\mathtt{dom}(f)$ to denote the *domain* of the function f, i.e., $\mathtt{dom}(f) = \{x \mid x \in \mathbb{N},\ f(x) \text{ is defined}\}$. By $\mathtt{range}(f)$ we denote the *range* of f, i.e., $\mathtt{range}(f) = \{f(x) \mid x \in \mathtt{dom}(f)\}$.

It is technically most convenient to define recursively enumerable families of recursively enumerable languages as follows. Any $\psi \in \mathcal{P}^2$ is called a numbering. Let $\psi \in \mathcal{P}^2$, then we write ψ_i instead of $\lambda x.\psi(i, x)$. We set $W_i^\psi = \mathtt{dom}(\psi_i)$ and refer to it as the ith enumerated language. Clearly, the sets $W_i^\psi \subseteq \mathbb{N}$ are recursively enumerable.

A function $f \in \mathcal{P}$ is said to be *strictly monotonic* provided for all $x, y \in \mathbb{N}$ with $x < y$ we have, if both $f(x)$ and $f(y)$ are defined then $f(x) < f(y)$. By $\mathcal{R}_{mon}$ we denote the set of all strictly monotonic recursive functions.

Let Σ be any fixed finite alphabet, and let Σ^* be the free monoid over Σ. Any $L \subseteq \Sigma^*$ is a language. Furthermore, we fix a symbol $\#$ such that $\# \notin \Sigma$. By $\mathcal{REG}$ we denote the class of all *regular* languages (cf., e.g., [24]). Furthermore, we use $\mathcal{C}$ or $\mathcal{L}$ to denote any (infinite) class and family of languages, respectively.

Definition 1 (Gold [13]). *Let L be any language. Every total function $t\colon \mathbb{N} \to \Sigma^* \cup \{\#\}$ with $\{t(j) \mid j \in \mathbb{N}\} \setminus \{\#\} = L$ is called a* text *for L.*

Note that the symbol $\#$ denotes pauses in the presentation of data. Furthermore, there is no requirement concerning the computability of a text. So, any order and any number of repetitions is allowed. For any $n \in \mathbb{N}$ we use $t[n]$ to denote the *initial segment* $(t(0), \ldots, t(n))$. Additionally we use $content(t[n]) =_{df} \{t(0), \ldots, t(n)\} \setminus \{\#\}$ and $content(t) =_{df} \{t(j) \mid j \in \mathbb{N}\} \setminus \{\#\}$ to denote the content of an initial segment and of a text, respectively.

An algorithmic *learner* M finds a rule (grammar) from growing initial segments of a text. On each initial segment the learner M has to output a hypothesis which is a natural number, i.e., $M(t[n]) \in \mathbb{N}$. Then the sequence $(M(t[n]))_{n\in\mathbb{N}}$ has to *converge* (to some representation of the input), i.e., there is a $j \in \mathbb{N}$ such that $M(t[n]) = j$ for all but finitely many $n \in \mathbb{N}$.

So, we still have to specify the semantics of the numbers output by M. In order to do so, we need the following.

Definition 2 (Angluin [2]). *A family $(L_j)_{j\in\mathbb{N}}$ of languages is said to be* uniformly recursive *if there exists a recursive function $f\colon \mathbb{N} \times \Sigma^* \to \{0,1\}$ such that $L_j = \{w \mid w \in \Sigma^*,\ f(j,w) = 1\}$ for all $j \in \mathbb{N}$. We call f a* decision function.

Definition 3. *A class $\mathcal{C}$ of non-empty recursive languages is said to be* indexable *if there is a family $(L_j)_{j\in\mathbb{N}}$ of uniformly recursive languages such that $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$. Such a family is said to be an* indexing *of $\mathcal{C}$.*
By $\mathcal{ID}$ we denote the collection of all indexable classes.

Note that $\mathcal{REG}$, and also the class of all context-free languages and the class of all context-sensitive languages form an indexable class. Further information concerning indexable classes and their learnability can be found in [21, 23].

So, when dealing with the learnability of indexable classes, it is only natural to interpret the hypotheses output by M with respect to a chosen indexing of a class containing the target class $\mathcal{C}$ (cf. Definition 4 below). On the other hand, when considering recursively enumerable classes $\mathcal{C}$ of recursively enumerable languages, then we always take as hypothesis space the family $(W_i^{\psi})_{i\in\mathbb{N}}$, where $\psi \in \mathcal{P}^2$ is the numbering defining the class $\mathcal{C}$.

Definition 4. *Let $\mathcal{C}$ be an indexable class. A family $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ is said to be an* indexed hypothesis space *for $\mathcal{C}$ if $(L_j)_{j\in\mathbb{N}}$ is uniformly recursive and $\mathcal{C} \subseteq \{L_j \mid j \in \mathbb{N}\}$.*

Following [20], if $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$ then we call $\mathcal{H}$ *class preserving* and if $\mathcal{C} \subseteq \{L_j \mid j \in \mathbb{N}\}$ then the hypothesis space $\mathcal{H}$ is said to be *class comprising.*

Now we are ready to provide the formal definition of learning in the limit from text. Following Gold [13] we call our learners *inductive inference machines* (abbr. IIM). To unify notations, in the definitions below we use $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ to denote our hypothesis spaces, where we assume the interpretation given above.

Definition 5 (Gold [13]). *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{C}$, and let $L \in \mathcal{C}$. An IIM M is said to* learn L in the limit from text with respect to $\mathcal{H}$ *if*

(1) *for every text t for L there is a $j \in \mathbb{N}$ such that the sequence $(M(t[n]))_{n\in\mathbb{N}}$ converges to j, and*
(2) $L = h_j$.

An IIM M learns $\mathcal{C}$ in the limit from text with respect to $\mathcal{H}$ *if M learns all $L \in \mathcal{C}$ in the limit from text with respect to $\mathcal{H}$.*

The collection of all classes $\mathcal{C}$ for which there is an IIM M and a hypothesis space $\mathcal{H}$ such that M learns $\mathcal{C}$ in the limit from text with respect to $\mathcal{H}$ is denoted by LimTxt.

In the following modifications of Definition 5 additional requirements are made. An IIM M is said to be *consistent* if for all relevant texts t and all $n \in \mathbb{N}$ the condition $content(t[n]) \subseteq h_{M(t[n])}$ is satisfied (cf. Angluin [1], Barzdin [3]).

An IIM M is said to be *conservative* if for all relevant texts t and all $n, m \in \mathbb{N}$ the following condition is satisfied. If $j = M(t[n]) \neq M(t[n+m])$ then $content(t[n+m]) \not\subseteq h_j$ (cf. Angluin [2]).

We call an IIM M *strong-monotonic* if for all relevant texts t and all numbers $n, m \in \mathbb{N}$ the following condition is satisfied. If $j = M(t[n]) \neq M(t[n+m]) = k$ then $h_j \subseteq h_k$ must hold (cf. Jantke [17], Lange and Zeugmann [19]).

We denote the resulting learning types by *ConsTxt*, *ConsvTxt*, and *SmonTxt*, respectively.

After having defined several learning models, it is only natural to ask why should we study learning with nonconstructivity. The answer is given by the fact that many interesting language classes are *not learnable from text*. As shown in [23], even quite simple classes cannot be learned from text, e.g., the class

$$\mathcal{C} = \{a^j \mid j \in \mathbb{N}^+\} \bigcup_{k\in\mathbb{N}^+} \{a^\ell \mid 1 \leq \ell \leq k\} \; . \tag{1}$$

We aim to characterize quantitatively the difficulty of such learning problems by measuring the amount of nonconstructivity needed to solve them.

The learners used for *nonconstructive* inductive inference take as input not only growing initial segments $t[n]$ of a text t but also a *help-word* w. The help-words are assumed to be encoded in binary. So, for such learners we write $M(t[n], w)$ to denote the hypothesis output by M. Then, for all the learning types defined above, we say that M nonconstructively identifies L with the help-word w provided that for every text t for L the sequence $(M(t[n], w))_{n\in\mathbb{N}}$ converges to a number j such that $h_j = L$ (for *LimTxt*) and M is consistent (conservative, strong-monotonic) for *ConsTxt* (for *ConsvTxt*, and *SmonTxt*), respectively. More formally we have the following definition.

Definition 6. *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{C}$, and let $d \in \mathcal{R}$. An IIM M* infers $\mathcal{C}$ with nonconstructivity $d(n)$ in the limit with respect to $\mathcal{H}$, *if for each $n \in \mathbb{N}$ there is a help-word w of length at most $d(n)$ such that for every $L \in \mathcal{C} \cap \{h_0, h_1, \ldots, h_n\}$ and every text t for L the sequence $(M(t[m], w))_{m\in\mathbb{N}}$ converges to a hypothesis j satisfying $h_j = L$.*

Clearly, Definition 6 can be directly modified to obtain *nonconstructive conservative and strong-monotonic learning.*

Looking at Definition 6 it should be noted that the IIM may need to know either an appropriate upper bound for n or even the precise value of n in order to exploit the fact that the target language L is from $\mathcal{C} \cap \{h_0, h_1, \ldots, h_n\}$.

To simplify notation, we make the following convention. Whenever we talk about nonconstructivity $\log n$, we assume that the logarithmic function to the base 2 is replaced by its integer valued counterpart $\lfloor \log n \rfloor + 1$, where $\log 0 =_{df} 1$.

Now we are ready to present our results. Note that some proofs have been influenced by ideas developed in the quite a different context, i.e., the paradigm of learning by erasing (also called co-learning). We do not explain it here but refer the reader to Jain *et al.* [14] as well as to Freivalds and Zeugmann [12].

3 Results

Already Gold [13] showed that $\mathcal{REG} \notin \mathit{LimTxt}$ and as mentioned in (1), even quite simple subclasses of $\mathcal{REG}$ are not in LimTxt. So, we start our investigations by asking for the *amount of nonconstructivity* needed to identify any indexable class in the limit from text with respect to any indexed hypothesis space $\mathcal{H}$.

3.1 Nonconstructive Learning of Indexable Classes

As we shall see, the needed amount of nonconstructivity is surprisingly small. To show this result, for every function $d \in \mathcal{R}_{mon}$ we define its *inverse* d_{inv} as follows $d_{inv}(n) = \mu y[d(y) \geq n]$ for all $n \in \mathbb{N}$. Recall that $\texttt{range}(d)$ is recursive for all $d \in \mathcal{R}_{mon}$. Thus, for all $d \in \mathcal{R}_{mon}$ we can conclude that $d_{inv}(n) \in \mathcal{R}$.

Theorem 1. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, let $d \in \mathcal{R}_{mon}$ be any function, and let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be identified with nonconstructivity $\log d_{inv}(n)$ in the limit from text with respect to $\mathcal{H}$.*

Proof. Assuming any help-word w of length precisely $\log d_{inv}(n)$, the IIM M creates a bitstring containing only 1s that has the same length as w. This bitstring is interpreted as a natural number k.

So, $k \geq d_{inv}(n)$, and thus

$$u_* =_{df} d(k) \geq d(d_{inv}(n)) \geq n \ . \tag{2}$$

We continue to define the IIM M in a way such that it will learn every language $L \in \mathcal{C} \cap \{L_0, \ldots, L_{u_*}\}$ from every of its texts. So, fix any such L, let t be any text for L, and let $m \in \mathbb{N}$.

Now, the idea to complete the proof is as follows. In the limit, the IIM M can determine the *number* ℓ of different languages enumerated in $L_0, \ldots, L_{u_*}$ as well as the *least indices* $j_1, \ldots, j_\ell$ of them and can then find the language among them which is equal to L. We assume the lexicographical ordering $\leq_{lo}$ of all strings from Σ^*, i.e., $s_i \leq_{lo} s_{i+1}$ for all $i \in \mathbb{N}$.

Using m, $t[m]$, and the decision function f for $\mathcal{H}$, the IIM M computes the least number r such that $m \leq r$ and $s \leq_{lo} s_r$ for all $s \in \mathit{content}(t[m])$. Next, M computes

$$\begin{aligned} L_0^r &= \{w \mid w \leq_{lo} s_r,\ f(0,w) = 1\} \\ L_1^r &= \{w \mid w \leq_{lo} s_r,\ f(1,w) = 1\} \\ &\vdots \\ L_{u_*}^r &= \{w \mid w \leq_{lo} s_r,\ f(u_*,w) = 1\}\ , \end{aligned}$$

and chooses the least indices $j_1, \ldots, j_{\ell_m}$ from $0, 1, \ldots, u_*$ of *all* the distinct languages in $L_0^r, \ldots, L_{u_*}^r$. From these languages $L_{j_z}^r$ all those are deleted for which $\mathit{content}(t[m]) \not\subseteq L_{j_z}^r$ (the inconsistent ones). From the remaining indices, the least index j is output such that $|L_j^r \setminus \mathit{content}(t[m])|$ is minimal.

Now, it easy to see that the sequence $(\ell_m)_{m \in \mathbb{N}}$ converges to ℓ, the number of the different languages enumerated in $L_0, \ldots, L_{u_*}$, and that the IIM M finds in the limit the least indices $j_1, \ldots, j_\ell$ for these pairwise different languages. From these languages $L_{j_1}, \ldots, L_{j_\ell}$ the ones satisfying $L \setminus L_{j_z} \neq \emptyset$ are deleted.

That leaves all those L_{j_z} with $L \subseteq L_{j_z}$. Now, by assumption there is a least $j \in \{0, \ldots, u_*\}$ with $L_j = L$. If $L \subset L_{j_z}$, then there is a string $s \in L_{j_z} \setminus L$, and as soon as this string appears in the competition, the index j wins. Thus, the sequence $(M(t[m], w))_{m \in \mathbb{N}}$ converges to j. □

So there is *no smallest* amount of nonconstructivity needed to learn $\mathcal{REG}$ and any subset thereof in the limit from text. But the amount of nonconstructivity cannot be zero, since then we would have $\mathcal{REG} \in \mathit{LimTxt}$. One can define a total function $t \in \mathfrak{T}$ such that $t(n) \geq d(n)$ for all $d \in \mathcal{R}_{mon}$ and all but finitely many n. Hence, $\log t_{inv}$ is then a lower bound for the amount of nonconstructivity needed to learn $\mathcal{REG}$ in the limit from text for the technique used to show Theorem 1.

We continue by asking what amount of nonconstructivity is needed to obtain *conservative* learning from text for any indexable class. Now, the situation is intuitively more complex, since $\mathit{ConsvTxt} \subset \mathit{LimTxt}$ (cf. [2, 20]). Also, it is easy to see that the IIM M given in the proof of Theorem 1 is in general *not* conservative. But the basic idea still works *mutatis mutandis* provided we know the number ℓ of different languages enumerated in $L_0, \ldots, L_n$.

Theorem 2. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, and let $\mathcal{H} = (L_j)_{j \in \mathbb{N}}$ be an indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be conservatively identified with nonconstructivity $\log n$ from text with respect to $\mathcal{H}$.*

Proof. Let $\mathcal{H} = (L_j)_{j \in \mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$, and let $n \in \mathbb{N}$. The help-word w is defined as follows. Since the IIM also needs to know a bound

on n, we always assume n to be a power of 2. Intuitively, we then add one bit and write the binary representation of the *exact number* ℓ of pairwise different languages enumerated in $L_0, \ldots, L_n$ behind the leading 1 including leading zeros. But of course we do not need the leading 1 in the help-word, since it can be added by the IIM M. So if the help-word w has length k, then the added leading 1 with $k-1$ zeros gives n and the bitstring w without the added leading 1 gives ℓ.

Given ℓ, the desired IIM M can find the least indices of these ℓ pairwise different languages by using the decision function f from the proof of Theorem 1 above, where r is large enough to detect ℓ different languages.

The rest is done inductively. The IIM M checks whether or not $t(0) \in L^r_{j_z}$, $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L^r_{j_z}$ with respect to set inclusion, and outputs the index of the minimal one with the smallest index. For $m > 0$, the IIM M then checks whether or not $\mathit{content}(t[m]) \subseteq L_{M(t[m-1])}$. If it is, it outputs $M(t[m-1])$.

Otherwise, it checks whether or not $\mathit{content}(t[m]) \subseteq L^r_{j_z}$, $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L^r_{j_z}$ with respect to set inclusion, and outputs the index of the minimal one with the smallest index. □

We also have the following lower bound.

Theorem 3. *There is a class $\mathcal{C} \in \mathcal{ID}$ and an indexed hypothesis space $\mathcal{H}$ for it such that for every IIM that learns $\mathcal{C}$ conservatively with respect to $\mathcal{H}$ less than $\log n - 2$ many bits of nonconstructivity are not enough.*

Next, we look at strong-monotonic learning. Again the situation is more complex, since $\mathit{SmonTxt} \subset \mathit{ConsvTxt}$ (cf. [20]). We add $L_0 = \emptyset$ to every hypothesis space allowed, i.e., we always consider class comprising hypothesis spaces.

Theorem 4. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, and let $\mathcal{H} = (L_j)_{j \in \mathbb{N}}$ be an indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified with nonconstructivity $2 \log n$ from text with respect to $\mathcal{H}$.*

Proof. The key observation is that it suffices to know the following *number* $p = |\{(i,j) \mid L_i \not\subseteq L_j,\ i,j = 0, \ldots, n\}|$. So, the help-word is just the binary encoding of p and n which is done *mutatis mutandis* as in the proof of Theorem 2. The rest is not too difficult, and thus omitted.

Again, the bound given in Theorem 4 cannot be improved substantially, since we have the following lower bound.

Theorem 5. *There is a class $\mathcal{C} \in \mathcal{ID}$ and an indexed hypothesis space $\mathcal{H}$ for it such that for every IIM that learns $\mathcal{C}$ strong-monotonically with respect to $\mathcal{H}$ less than $2 \log n - 4$ many bits of nonconstructivity are not enough.*

Having these general results, we can also ask what happens if we allow a suitably chosen hypothesis space for $\mathcal{REG}$ such as all DFAs. Then for all $i, j \in \mathbb{N}$ equality $L_i = L_j$ and subset $L_i \subseteq L_j$ are decidable, and thus we are in the setting described in the proof of Theorem 1. That is we have the following theorem.

Theorem 6. *Let $\mathcal{C} \subseteq \mathcal{REG}$ be arbitrarily fixed, let $d \in \mathcal{R}_{mon}$ be any function, and let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified with nonconstructivity $\log d_{inv}(n)$ from text with respect to $\mathcal{H}$.*

3.2 Nonconstructive Learning of Recursively Enumerable Classes

Next, we turn our attention to the amount of nonconstructivity needed to learn recursively enumerable classes of recursively enumerable languages.

Theorem 7. *Let $\psi \in \mathcal{P}^2$ be any numbering. Then there is always an IIM M learning the family $(W_i^\psi)_{i\in\mathbb{N}^+}$ in the limit from text with nonconstructivity $\log n$ with respect to $(W_i^\psi)_{i\in\mathbb{N}^+}$.*

Proof. The help-word w is essentially the same as in the proof of Theorem 2, i.e., it is a bitstring of b of length $\log n$ which is the binary representation of ℓ, the number of pairwise different languages enumerated in $W_1^\psi, \ldots, W_n^\psi$ plus n.

Let $L \in \mathcal{C} \cap \{W_1^\psi, \ldots, W_n^\psi\}$ and let t be any text for L. On input any $t[m]$ and the help-word w the desired IIM M executes the following.

(1) For all $0 < i \leq n$ enumerate W_i^ψ for m steps, that is, M tries to compute $\psi_i(0), \ldots, \psi_i(m)$ for at most m steps and enumerate those arguments x for which $\psi_i(x)$ turns out to be defined. Let $W_{i,m}^\psi$ be the resulting sets, $0 < i \leq n$.

(2) For all pairs (i,j) with $0 < i,j \leq n$ check whether or not $W_{i,m}^\psi \setminus W_{j,m}^\psi \neq \emptyset$. If it is, let $d(i,j)$ be the least element in $W_{i,m}^\psi \setminus W_{j,m}^\psi$. If there is no such element, we set $d(i,j) = \infty$.

(3) Using the numbers $d(i,j)$ then M checks whether or not there are ℓ pairwise different languages among $W_{1,m}^\psi, \ldots, W_{n,m}^\psi$. If not, then $M(t[m]) = 0$.
Otherwise, let $S = \{i \mid 0 < i \leq n,\ W_{j,m}^\psi \neq \emptyset\}$ and consider all sets $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| = \ell$. For each such set $\tilde{S} = \{j_1, \ldots, j_\ell\}$ compute the numbers $x_{j,k} =_{df} \min(W_{j,m}^\psi \triangle W_{k,m}^\psi)$ for all $j,k \in \tilde{S}$, where $j < k$ and let $s(\tilde{S})$ be the maximum of all those $x_{j,k}$. Furthermore, for each set $\tilde{S}$ we consider the ℓ-tuple $(j_1, \ldots, j_\ell)$, where $j_i < j_{i+1}$, $i = 1, \ldots, \ell - 1$. Using these tuples, we can order them lexicographically and then choose the first set $\tilde{S}$ in this order for which $s(\tilde{S})$ is minimized, i.e., $s(\tilde{S}) \leq s(\hat{S})$ for all $\hat{S}$ with $\hat{S} \subseteq S$ and $|\hat{S}| = \ell$. Let $i_1, \ldots, i_\ell$ be the elements of this set $\tilde{S}$ in their natural order.
Then M takes the languages $W_{i_1,m}^\psi, \ldots, W_{i_\ell,m}^\psi$ into consideration. From these candidate hypotheses $i_1, \ldots, i_\ell$ the least i is output for which $t[m]$ contains all finite $d(i,j)$, $j = i_1, \ldots, i_\ell$, and $t[m]$ does not contain any of the finite $d(j,i)$, $j = i_1, \ldots, i_\ell$. If there is no such i, then $M(t[m]) = 0$.

We have to show that M learns L in the limit from t. Note that the ℓ pairwise different languages are found in the limit, since the minimal element in the symmetric difference of the two languages tends to infinity if the two languages are equal (if any element is found at all). So, the set of candidate hypotheses stabilizes in the limit, and by construction M then outputs the correct i as soon as the initial segment is large enough. We omit details. □

The IIM defined in the proof of Theorem 7 even witnesses a much stronger result, i.e., it always converges to the minimum index i of the target language.

The following lower bound shows that Theorem 7 cannot be improved substantially.

Theorem 8. *There is a numbering $\psi \in \mathcal{P}^2$ such that no IIM M can learn the family $(W_i^{\psi})_{i\in\mathbb{N}^+}$ in the limit from text with nonconstructivity $\log n - 2$ with respect to $(W_i^{\psi})_{i\in\mathbb{N}^+}$.*

The situation considerably changes if we require conservative learning. In order to present this result, we need the following. A function $h\colon \mathbb{N} \to \mathbb{N}$ is said to be limiting recursive if there is a function $\tilde{h} \in \mathcal{R}^2$ such that $h(i) = \lim\limits_{n\to\infty} \tilde{h}(i,n)$.

Theorem 9. *For every limiting recursive function h there is a recursively enumerable family $(W_i^{\psi})_{i\in\mathbb{N}}$ of recursively enumerable languages such that no IIM with nonconstructivity at most h can learn $(W_i^{\psi})_{i\in\mathbb{N}}$ conservatively with respect to $(W_i^{\psi})_{i\in\mathbb{N}}$.*

4 Conclusions

We have presented a model for the inductive inference of formal languages from text that incorporates a certain amount of nonconstructivity. In our model, the amount of nonconstructivity needed to solve the learning problems considered has been used as a quantitative characterization of their difficulty.

We studied the problem of learning indexable classes under three postulates, i.e., *learning in the limit*, *conservative identification*, and *strong-monotonic inference*. As far as learning in the limit is concerned, the amount of nonconstructivity needed to learn any indexable class can be very small and there is no smallest amount that can be described in a computable way (cf. Theorem 1).

Moreover, we showed upper and lower bounds for conservative learning of indexable classes and for strong-monotonic inference roughly showing that the amount of nonconstructivity needed is $\log n$ for conservative learning and $2\log n$ for strong-monotonic inference.

However, if we allow canonical indexed hypothesis spaces for $\mathcal{REG}$ such that equality of languages is decidable, then the amount of nonconstructivity needed to learn $\mathcal{REG}$ even strong-monotonically can be made very small.

Finally, we studied the problem to learn recursively enumerable classes of recursively enumerable languages. In this setting, the amount of nonconstructivity needed to learn in the limit is $\log n$, while there is not even a limiting recursive bound for the amount of nonconstructivity to learn all recursively enumerable classes of recursively enumerable languages conservatively.

Acknowledgment. This research was performed partially while the third author was visiting the Institute of Mathematical Sciences at the National University of Singapore in September 2011. His visit was supported by the Institute. Sanjay Jain was supported in part by NUS grant numbers C252-000-087-001 and R252-000-420-112, and Frank Stephan was supported in part by NUS grant number R252-000-420-112.

References

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

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

[3] Barzdin, J.: Inductive inference of automata, functions and programs. In: Proc. of the 20-th International Congress of Mathematicians, Vancouver, Canada. pp. 455–460 (1974), (republished in *Amer. Math. Soc. Transl.* (2) 109, 1977, pages 107- 112)

[4] Bārzdiņš, J.M.: Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. Soviet Mathematics Doklady 9, 1251–1254 (1968)

[5] Beyersdorff, O., Köbler, J., Müller, S.: Proof systems that take advice. Information and Computation 209(3), 320–332 (2011)

[6] Cook, S., Krajiček, J.: Consequences of the provability of $NP \subseteq P/poly$. The Journal of Symbolic Logic 72(4), 1353–1371 (2007)

[7] Damm, C., Holzer, M.: Automata that take advice. In: Wiedermann, J., Hájek, P. (eds.) Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS '95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings. Lecture Notes in Computer Science, vol. 969, pp. 149–158. Springer, Berlin (1995)

[8] Erdős, P.: Some remarks on the theory of graphs. Bulletin of the American Mathematical Society 53(4), 292–294 (1947)

[9] Freivalds, R.: Amount of nonconstructivity in finite automata. In: Maneth, S. (ed.) Implementation and Application of Automata, 14th International Conference, CIAA 2009, Sydney, Australia, July 14-17, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5642, pp. 227–236. Springer, Berlin (2009)

[10] Freivalds, R.: Amount of nonconstructivity in deterministic finite automata. Theoretical Computer Science 411(38-39), 3436–3443 (2010)

[11] Freivalds, R., Zeugmann, T.: On the amount of nonconstructivity in learning recursive functions. In: Ogihara, M., Tarui, J. (eds.) Theory and Applications of Models of Computation, 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011, Proceedings. Lecture Notes in Computer Science, vol. 6648, pp. 332–343. Springer (2011)

[12] Freivalds, R., Zeugmann, T.: Co–learning of recursive languages from positive data. In: Bjørner, D., Broy, M., Potttosin, I.V. (eds.) Perspectives of System Informatics, Second International Andrei Ershov Memorial Conference, Akademgorodok, Novosibirsk, Russia, June 1996, Proceedings. Lecture Notes in Computer Science, vol. 1181, pp. 122–133. Springer (1996)

[13] Gold, E.M.: Language identification in the limit. Inform. Control 10(5), 447–474 (1967)

[14] Jain, S., Kinber, E., Lange, S., Wiehagen, R., Zeugmann, T.: Learning languages and functions by erasing. Theoretical Computer Science 241(1-2), 143–189 (2000)

[15] Jain, S., Osherson, D., Royer, J.S., Sharma, A.: Systems that Learn: An Introduction to Learning Theory, second edition. MIT Press, Cambridge, MA (1999)

[16] Jain, S., Stephan, F., Zeugmann, T.: On the amount of nonconstructivity in learning formal languages from text. Tech. Rep. TCS-TR-A-12-55, Division of Computer Science, Hokkaido University (2012)

[17] Jantke, K.P.: Monotonic and non-monotonic inductive inference. New Generation Computing 8(4), 349–360 (1991)
[18] Karp, R.M., Lipton, R.J.: Turing machines that take advice. L' Enseignement Mathématique 28, 191–209 (1982)
[19] Lange, S., Zeugmann, T.: Types of monotonic language learning and their characterization. In: Haussler, D. (ed.) Proc. 5th Annual ACM Workshop on Computational Learning Theory. pp. 377–390. ACM Press, New York, NY (1992)
[20] Lange, S., Zeugmann, T.: Language learning in dependence on the space of hypotheses. In: Pitt, L. (ed.) Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory. pp. 127–136. ACM Press, New York, NY (1993)
[21] Lange, S., Zeugmann, T., Zilles, S.: Learning indexed families of recursive languages from positive data: A survey. Theoretical Computer Science 397(1-3), 194–232 (2008)
[22] Rogers, Jr., H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill (1967), reprinted, MIT Press 1987.
[23] Zeugmann, T., Lange, S.: 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)
[24] Zeugmann, T., Minato, S., Okubo, Y.: Theory of Computation. Corona Publishing Co., LTD (2009)