On the Amount of Nonconstructivity in Learning Formal Languages from Text

Sanjay Jain[a,1], Frank Stephan[b,1], Thomas Zeugmann[c]

[a]*Department of Computer Science, National University of Singapore, Singapore 117417, Republic of Singapore*
[b]*Department of Mathematics, National University of Singapore, Singapore 119076, Republic of Singapore*
[c]*Division of Computer Science, Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan*

Abstract

Nonconstructive computations by various types of machines and automata have been considered by, for example, Karp and Lipton [21], and Freivalds [9, 10]. They allow to regard more complicated algorithms from the viewpoint of much 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. 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.

Key words: Inductive inference, learning in the limit, non-constructivity, formal languages.

1. Introduction

For millenia mathematicians have been aware of the method to show the existence of a mathematical object by constructing it, and the nonconstructive method of proving that it cannot fail to exist. On the other hand, before the end of the 19th century constructive proofs have been the dominant ones. And in the forties of the last century nonconstructive methods found their way to discrete mathematics (cf., for example, Erdős [8]).

Email addresses: sanjay@comp.nus.edu.sg (Sanjay Jain), fstephan@comp.nus.edu.sg (Frank Stephan), thomas@ist.hokudai.ac.jp (Thomas Zeugmann)

[1]This research has been supported by Ministry of Education Academic Research Fund grants R252-000-420-112, R146-000-234-112 (MOE2016-T2-1-019) and R146-000-304-112 (MOE2019-T2-2-121) to Frank Stephan (PI) and Sanjay Jain (Co-PI); furthermore, Sanjay Jain is supported in part by NUS grant C252-000-087-001.

If it is known that a mathematical object exists, one can also ask what does it take to construct such an object. In this context, the notion of complexity as well as the distinction between uniform and non-uniform models of computations comes into play. Allowing a uniform model of computation to use an arbitrary string as additional input converts it into a non-uniform model. Such strings are often called *advice*. An influential paper in this regard was Bārzdiņš [3], who introduced the notion of advice in the setting of Kolmogorov complexity of recursively enumerable sets.

Further examples comprise Karp and Lipton [21]. They studied the problem under what circumstances non-uniform upper bounds can be used to obtain uniform upper bounds. To achieve this goal the notion of a Turing machine that can take advice has been coined. Furthermore, Damm and Holzer [7] adapted the notion of advice to finite automata, and Cook and Krajiček [6] initiated the study of proof systems that use advice for the verification of proofs. Even more recently, Beyersdorff *et al.* [4] 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 [21] 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.

In the present paper we generalize the model of Freivalds and Zeugmann [11] to the inductive inference of formal languages. That is, 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, as usual, 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 elements of L and nothing else. In addition, the learner receives as a second input a bitstring of finite length which we call *help-word*. Given a correct / appropriate help-word, the learner can learn in the desired sense. Since there are infinitely many languages to learn, a parameterization is necessary, that is, 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 2.7). The difficulty of the learning problem is then measured by the length of the help-words needed, that is, 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 quite 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 [13, 15, 28]. 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, see [15, 25] and the references therein. Thus, it is only natural to consider some of these variations, too. In particular, we shall study *conservative* as well as *strong-monotonic* learning.

We prove upper and lower bounds for the amount of nonconstructivity in learning classes of formal 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 3.1), 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 3.2 and 3.3). 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 3.4 and 3.5).

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 3.7, 3.9 and 3.11).

2. Preliminaries

Any unspecified notation follows Rogers [26]. In addition to or in contrast with [26] 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 the set S. Let $\emptyset$, $\in$, $\subset$, $\subseteq$, $\supset$, and $\supseteq$ denote the empty set, element of, proper subset, subset, proper superset, and superset, respectively. Let S_1, S_2 be any sets. By $S_1 \setminus S_2$ we denote the difference of sets, and we write $S_1 \bigtriangleup S_2$ to denote the symmetric difference of S_1 and S_2, that is, $S_1 \bigtriangleup S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. 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 $\mathbb{N}$. For $n \in \mathbb{N}^+$, 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. 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.

Furthermore, let $f \in \mathcal{P}$, then we use $\mathrm{dom}(f)$ to denote the *domain* of the function f, that is, $\mathrm{dom}(f) = \{x \mid x \in \mathbb{N},\ f(x) \text{ is defined}\}$. By $\mathrm{range}(f)$ we denote the *range* of f, that is, $\mathrm{range}(f) = \{f(x) \mid x \in \mathrm{dom}(f)\}$.

It is technically most convenient to define recursively enumerable families of recursively enumerable languages as follows. Any function $\psi \in \mathcal{P}^2$ is called a numbering. Let $\psi \in \mathcal{P}^2$, then we write ψ_i instead of $\lambda x.\psi(i,x)$. We define $\mathcal{P}_\psi = \{\psi_i \mid i \in \mathbb{N}\}$. Furthermore, we set $W_i^\psi = \mathrm{dom}(\psi_i)$ and refer to it as the ith enumerated language. Clearly, the sets $W_i^\psi \subseteq \mathbb{N}$ are recursively enumerable. A numbering $\varphi \in \mathcal{P}^2$ is called a *Gödel numbering* for $\mathcal{P}$ [26] if $\mathcal{P}_\varphi = \mathcal{P}$, and for every numbering $\psi \in \mathcal{P}^2$, there is a *compiler* $c \in \mathcal{R}$ such that $\psi_i = \varphi_{c(i)}$ for all $i \in \mathbb{N}$. If $\varphi \in \mathcal{P}^2$ is any arbitrarily fixed Gödel numbering then we also use W_i as a shorthand for W_i^φ.

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 $\# \notin \Sigma^*$. We denote the empty string by λ and use $|w|$ to denote the length of any string $w \in \Sigma^*$. By $\mathcal{REG}$ we denote the class of all *regular* languages. Furthermore, we use $\mathcal{C}$ and $\mathcal{L}$ to denote any (infinite) class and family of languages, respectively.

Definition 2.1 (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* $(t(0), \ldots, t(n))$. We use $\mathrm{content}(t[n]) = \{t(0), \ldots, t(n)\} \setminus \{\#\}$ and $\mathrm{content}(t) = \{t(j) \mid j \in \mathbb{N}\} \setminus \{\#\}$ 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, that is, $M(t[n]) \in \mathbb{N}$. Then the sequence $(M(t[n]))_{n\in\mathbb{N}}$ has to *converge* (to some representation of the input), that is, 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 2.2 (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 refer to f as a* decision function.

Definition 2.3. *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 $\mathcal{REG}$ is an indexable class. Also, the class of all context-free languages and the class of all context-sensitive languages form an indexable class. Lange, Zeugmann and Zilles [25, 28] provide further information concerning indexable classes and their learnability.

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 2.4 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 2.4. *Let $\mathcal{C}$ be an indexable class. A family $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ is called 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 Lange and Zeugmann [24], 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 $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ to denote our hypothesis spaces, where we assume the interpretation given above.

Definition 2.5 (Gold [13]). *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ 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) *$L = L_j$.*

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.

Since, by the definition of convergence, only finitely many data of L were seen by the IIM up to the (unknown) point of convergence, whenever an IIM learns the possibly infinite language L, indeed some form of learning must have taken place. For this reason, hereinafter the terms *learn*, *infer* and *identify* are used interchangeably.

In the following modifications of Definition 2.5 additional requirements are introduced which the IIM has to satisfy.

Definition 2.6 (Angluin [1, 2], Jantke [20], Lange and Zeugmann [23]). *Let $\mathcal{C}$ be a class of languages, let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{C}$, and let $L \in \mathcal{C}$. An IIM M is said to* learn L with respect to $\mathcal{H}$

(A) conservatively*, and*
(B) strong-monotonically*,*

respectively, if M learns L in the limit from text with respect to $\mathcal{H}$ and for every text t for L the following conditions are satisfied:

(A) *for all* $n, m \in \mathbb{N}$, *if* $j = M(t[n]) \neq M(t[n{+}m])$ *then* $\mathrm{content}(t[n{+}m]) \not\subseteq L_j$*; and*

(B) *for all* $n, m \in \mathbb{N}$, *if* $j = M(t[n]) \neq M(t[n+m]) = k$ *then* $L_j \subseteq L_k$,

respectively. An IIM M learns $\mathcal{C}$ conservatively, or strong-monotonically with respect to $\mathcal{H}$ if M learns every $L \in \mathcal{C}$ conservatively, or strong-monotonically, respectively, with respect to $\mathcal{H}$.

We denote the resulting learning types by 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 [13, 28], even quite simple language classes cannot be learned from text, for example, the following class over the alphabet $\Sigma = \{a\}$:

$$\mathcal{C} = (L_j)_{j\in\mathbb{N}} \text{ where } L_0 = \{a^j \mid j \in \mathbb{N}^+\} \text{ and } L_j = \{a^\ell \mid 1 \leq \ell \leq j\}\ . \tag{1}$$

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 $L_j = L$ (for *LimTxt*) and M is conservative (strong-monotonic) for *ConsvTxt* (*SmonTxt*), respectively. More formally we have the following definition.

Definition 2.7. *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ 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 language $L \in \mathcal{C} \cap \{L_0, L_1, \ldots, L_n\}$ and every text t for L the sequence $(M(t[m], w))_{m\in\mathbb{N}}$ converges to a hypothesis j satisfying $L_j = L$.*

Clearly, Definition 2.7 can be directly modified to obtain *nonconstructive conservative and strong-monotonic learning*.

Looking at Definition 2.7 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 $\mathcal{C} \cap \{L_0, L_1, \ldots, L_n\}$.

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

Now we are ready to present our results. Note that the definitions developed in this paper were influenced by the corresponding definitions for function learning in [11]. The paper [11] also provided the proof idea for Theorem 3.1 and the idea about using the number of distinct functions/languages which is used in some of our proofs such as Theorem 3.4. Techniques from team learning (from papers such as [17, 27]) are used for Proposition 3.10. Another technique which has been useful is the concept of removing/erasing wrong grammars that is, the paradigm of learning by erasing (also called co-learning, see Jain et al. [14] and Freivalds and Zeugmann [12]). This concept is useful in the sense that we sometimes "eliminate" wrong grammars using the additional help-words provided.

3. Results

Already Gold [13] showed that $\mathcal{REG} \notin$ *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. In order to show this result, for every function $d \in \mathcal{R}_{mon}$ we define its *inverse* d_{inv} as follows. We set $d_{inv}(n) = \mu y[d(y) \geq n]$ for all $n \in \mathbb{N}$. Recall that $\mathrm{range}(d)$ is recursive for all $d \in \mathcal{R}_{mon}$. Thus, for all $d \in \mathcal{R}_{mon}$ we can conclude that $d_{inv} \in \mathcal{R}$.

Theorem 3.1. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$, and let $d \in \mathcal{R}_{mon}$ be any function. Then there is a computable IIM M such that the class $\mathcal{C}$ can be identified by M 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. Consequently, $k \geq d_{inv}(n)$. Let $u_* = d(k)$. Then we have

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

We define the IIM M in a way such that it will identify every language $L \in \mathcal{C} \cap \{L_0, \ldots, L_{u_*}\}$ from any 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 pairwise 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. This is achieved as follows. We assume any fixed length-lexicographical ordering $\leq_{lo}$ of all strings from Σ^*, that is, $s_0, s_1, \ldots$ such that $s_i \leq_{lo} s_{i+1}$ 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 $s \in \mathrm{content}(t[m])$. 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 the list $L^r_0, \ldots, L^r_{u_*}$.

From these languages $L^r_{j_z}$, IIM M deletes from consideration all those indices for which $\mathrm{content}(t[m]) \not\subseteq L^r_{j_z}$ (the inconsistent ones). From the remaining indices, M outputs the least index j such that $|L^r_j \setminus \mathrm{content}(t[m])|$ is minimal.

Now, it is easy to see that the sequence $(\ell_m)_{m\in\mathbb{N}}$ converges to ℓ, the number of the pairwise 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 from consideration by IIM M for large enough m — the m for which an element in $L \setminus L_{j_z}$ appears in $\mathrm{content}(t[m])$.

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 $s_p \in L_{j_z} \setminus L$, and as soon as $m \geq p$, the index j wins against j_z since $L = L_j \subseteq L_{j_z}$ and $m \geq p$ implies, $|L^r_j \setminus \mathrm{content}(t[m])| < |L^r_{j_z} \setminus \mathrm{content}(t[m])|$.

Thus, the sequence $(M(t[m], w))_{m\in\mathbb{N}}$ converges to j. □

The above result shows that for, indexed families, for any non-decreasing unbounded computable function f, the amount of nonconstructively needed to learn an indexed family in the limit can be bounded by f. The amount of nonconstructivity needed cannot be bounded by a constant, since otherwise we would have $\mathcal{REG} \in$ *LimTxt* (as one could fix the nonconstructive advice to be the one which appears as advice for learning $\{L_0, L_1, \ldots, L_n\}$, for infinitely many n). Thus, there is *no computable non-constant smallest* amount of nonconstructivity needed to learn $\mathcal{REG}$ in the limit from text. If one just considers the method of the above Theorem, then one can get a non-computable lower bound on the amount of nonconstructivity needed for this method as follows. 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. Consequently, $\log t_{inv}$ 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 3.1.

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 *ConsvTxt* $\subset$ *LimTxt* [2, 24]. Also, it is easy to see that the IIM M given in the proof of Theorem 3.1 is in general *not* conservative. But the basic idea still works *mutatis mutandis* provided we know the number ℓ of pairwise different languages enumerated in $L_0, \ldots, L_n$.

Theorem 3.2. *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 by M 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. Let $q \geq 1$ be minimal such that $n < 2^q$, and thus $\log n = q$. Let ℓ denote the number of different languages in $L_0, L_1, \ldots, L_{2^q-1}$. Then, the help-word w represents the number $\ell - 1$ using q bits. Note that ℓ is at least 1 and at most 2^q. Thus, $q = \log n$ bits are enough to represent $\ell - 1$. As an example, if

$n = 1 = 2^0$, then $q = 1$ and a help-word is of 1 bit size, which is enough to distinguish whether there are one or two different languages in L_0, L_1.

Given the help-word representing $\ell - 1$ using q bits, the desired IIM M can find the least indices of these ℓ pairwise different languages among $L_0, L_1, \ldots, L_{2^q-1}$ by using the decision function f for $(L_j)_{j\in\mathbb{N}}$ as in the proof of Theorem 3.1 above, where u_* is taken as $2^q - 1$ (recall that u_* in Theorem 3.1 was an upper bound on n), and r is additionally taken to be large enough to detect ℓ different languages. That is, it is assumed that r is large enough so that the list $L_0^r, L_1^r, \ldots, L_{2^q-1}^r$ contains at least ℓ different languages, besides the conditions that $r \geq m$, and $s \leq_{lo} s_r$ for all $s \in \mathrm{content}(T[m])$. Recall that $s_0, s_1, \ldots$ is again any fixed length-lexicographical ordering of all strings from Σ^*.

The rest is done inductively. Let t be a text for $L \in \mathcal{C} \cap \{L_0, L_1, \ldots, L_{2^q-1}\}$. On input $t[0]$, the IIM M checks whether or not $t(0) \in L_{j_z}^r$, where $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L_{j_z}^r$ with respect to set inclusion, and outputs the index of a minimal set among these with the smallest index. For $m > 0$, M checks whether or not $\mathrm{content}(t[m]) \subseteq L_{M(t[m-1])}$. If it is, it outputs $M(t[m-1])$. Otherwise, it checks whether or not $\mathrm{content}(t[m]) \subseteq L_{j_z}^r$, $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L_{j_z}^r$ with respect to set inclusion, and outputs the index of a minimal set among these with the smallest index. It is now easy to verify that M is a conservative learner and identifies $\mathcal{C}$. $\square$

We also have the following lower bound.

Theorem 3.3. *There is a class $\mathcal{C} \in \mathcal{ID}$ and an indexed hypothesis space $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ 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.*

Proof. This proof is done by diagonalization. We have to define a family $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ of uniformly recursive languages and then to show that it satisfies the assertion of the theorem using $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$. Let $\varphi \in \mathcal{P}^2$ be any fixed Gödel numbering of $\mathcal{P}$. We interpret every partial recursive function φ_i as an IIM, that is, we set $M_i = \varphi_i$. Below we shall use the alphabet $\Sigma = \{a, b\}$. Let *FINSEQ* denote the set of all finite sequences over Σ^*. We assume some fixed recursive encoding $\langle\cdot,\cdot\rangle$, from *FINSEQ* $\times \{0,1\}^*$ onto the natural numbers. To simplify notation we write $M(\sigma, \beta)$ instead of $M(\langle\sigma, \beta\rangle)$, where $\sigma \in$ *FINSEQ* and $\beta \in \{0,1\}^*$. Also, we write $M(\sigma, \beta)\downarrow = \ell$ provided the computation of $M(\sigma, \beta)$ stops in a finite number of steps and outputs ℓ.

Our construction is organized in blocks, where in Block i, $i \in \mathbb{N}^+$, we shall define languages L_{2^i+j} for all $j \in \{0, 1, \ldots, 2^i - 1\}$, that is, the languages $L_{2^i}, \ldots, L_{2^{i+1}-1}$ by diagonalizing against M_{i-1} and all possible help-words $\beta \in \{0,1\}^*$ with $0 \leq |\beta| < i$ including the empty one λ. Note that there are $2^i - 1$ many such β.

Part of the construction is based on an idea developed by Angluin [2]. For the sake of completeness we set $L_0 = \{b\}$ and define $L_1 = \{bb\}$.

For $i \geq 1$ we have to define 2^i many languages. In addition, we also define a text t_i incrementally. This is done in Block i as follows.

For $j = 0, \ldots, 2^i - 1$ execute the following.

Initialize $L_{2^i+j} = \{a^i\}$, set $C_i = \{\beta \mid \beta \in \{0,1\}^*,\ 0 \le |\beta| < i\}$, $D_i = \emptyset$, and initialize the text t_i by defining $t_i(0) = a^i$. Go to Stage 1.

Stage k. Execute the following instructions.

(A) Let $t_i(k) = a^i b^k$.

(B) Let $L_{2^i+j} = L_{2^i+j} \cup \{a^i b^k\}$ for all $j \in \{0, \ldots, 2^i - 1\} \setminus D_i$.

(C) For all $\beta \in C_i$ and all $\ell = 0, 1, \ldots, k$ check whether or not

$$M_{i-1}(t_i[\ell], \beta)\!\downarrow = 2^i + j \quad \text{for some } j \in \{0, \ldots, 2^i - 1\} \setminus D_i \tag{3}$$

can be verified within at most k steps of computation until the first such β, j and ℓ are found, if ever.

If β and j are found then

let $j_* = \min(\{0, \ldots, 2^i - 1\} \setminus (D_i \cup \{j\}))$; $C_i = C_i \setminus \{\beta\}$ and $D_i = D_i \cup \{j_*\}$.

Else do nothing.

Go to Stage $k + 1$.

End Stage k.

Now, it is easy to see that $(L_\ell)_{\ell \in \mathbb{N}}$ is a family of uniformly recursive languages.

Let $i \ge 1$ be arbitrarily fixed and suppose by way of contradiction that M_{i-1} conservatively learns all the languages L_{2^i+j}, where $0 \le j \le 2^i - 1$, defined in Block i by using help-words of size less than i. Note that $i - 1 = \lfloor \lg(2^{i+1} - 1) \rfloor + 1 - 2 = \log(2^{i+1} - 1) - 2$ by the convention made at the very end of Section 2.

Note that in Instruction (C), if β and j have been found, then exactly one element is added to D_i. Since there are only $2^i - 1$ possible help-words β, we directly conclude that the set $\{0, 1, 2, \ldots, 2^i - 1\} \setminus D_i$ is never empty. So, there is always a number j_* and by construction we have $j \neq j_*$.

Let β be any of the admissible help-words. We distinguish the following cases.

Case 1. The help-word β is not removed from C_i in any of the stages.

In this case M_{i-1} fails to identify L_{2^i+j} for all $j \in \{0, 1, 2, \ldots, 2^i - 1\} \setminus D_i$ when using the help-word β. Note that the D_i used here is the “eventual value of D_i.” Since by construction $\{0, 1, 2, \ldots, 2^i - 1\} \setminus D_i \neq \emptyset$, at least one language is not learned, a contradiction.

Case 2. The help-word β is removed from C_i in some Stage k.

Let j and j_* be as described in Instruction (C) above when executing this Stage k. Then by construction we know that $\mathrm{content}\,(t_i[k]) = L_{2^i+j_*}$, since j_* is added to D_i and so $L_{2^i+j_*}$ is not changed any further in Instruction (B). On the other hand, j is still not in D_i and thus $L_{2^i+j_*} \subset L_{2^i+j}$ after the execution of Stage $k+1$. Consequently, the IIM M_{i-1} cannot learn $L_{2^i+j_*}$ conservatively when using the help-word β, a contradiction.

Therefore, for every help-word, at least one of the languages in Block i is not learned by M_{i-1} or on which M_{i-1} is not conservative.

Finally, suppose that there is an IIM M that learns the indexable class $(L_\ell)_{\ell \in \mathbb{N}}$ with nonconstructivity $\log n - 2$ with respect to the hypothesis space $(L_\ell)_{\ell \in \mathbb{N}}$. Then there must be an $i \in \mathbb{N}$ such that $M = M_i$. Consequently, for $n = 2^{i+2} - 1$ there must be a help-word w of length less than $i + 1$ such that M_i conservatively learns all the languages $L_0, \dots, L_{2^{i+2}-1}$ from any of its texts and w.

Now, we consider in particular Block $i+1$, that is, the languages $L_{2^{i+1}}, \dots, L_{2^{i+2}-1}$. Since $|w| < i + 1$ there must be a β such that $w = \beta$ for some $\beta \in C_{i+1}$, for the eventual value of C_{i+1}. But as our discussion above showed, this is impossible. So, we have a contradiction and our supposition must be false. This proves the theorem. $\square$

Finally, we look at strong-monotonic learning. Again the situation is more complex, since *SmonTxt* $\subset$ *ConsvTxt*, see [24]. We also add $L_0 = \emptyset$ to every hypothesis space allowed, that is, we always consider class comprising hypothesis spaces.

Theorem 3.4. *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 $\mathcal{C}$, where $L_0 = \emptyset$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified by M with nonconstructivity $2 \log n$ from text with respect to $\mathcal{H}$.*

Proof. Suppose $q = \log n$. Thus, $n < 2^q$. The key observation is that along with q, it suffices to know the following *number*

$$p = |\{(i, j) \mid L_i \not\subseteq L_j,\ i, j \in \{0, \dots, 2^q - 1\}\}| \, .$$

So, the help-word is of size $2q$ encoding p. Note that $2q$ bits are enough to encode p in binary as $p < 2^{2q}$. Let $s_0, s_1, \dots$ be the length-lexicographical ordering of all strings from Σ^*. Let f be the decision function for $(L_j)_{j \in \mathbb{N}}$. Let $L_j^r = \{w \mid w \leq_{lo} s_r,\ f(j, w) = 1\}$. Given p and q, the IIM M can determine $j_1, j_2, \dots, j_\ell < 2^q$ such that (a) $L_{j_1}, L_{j_2}, \dots, L_{j_\ell}$ are pairwise different with $j_1 = 0$, and (b) for some r,

$$|\{(z, y) \mid L_{j_z}^r \not\subseteq L_{j_y}^r ; z, y \in \{1, 2, \dots, \ell\}\}| = p.$$

Note that by the hypothesis on p, all L_j, $j < 2^q$, belong to the set $\{L_{j_1}, L_{j_2}, \dots, L_{j_\ell}\}$, as otherwise L_j would be different from L_{j_1} and thus the value of p is not correct. Furthermore, $L_{j_z}^r \subseteq L_{j_y}^r$ iff $L_{j_z} \subseteq L_{j_y}$ as otherwise value of p would be incorrect. Here without loss of generality, one can assume that j_z is minimal index for L_{j_z}.

Initially, the IIM M outputs j_1 (recall that $j_1 = 0$, and thus the first output of M is for $\emptyset$). Thereafter, on input $t[m]$, a new conjecture is output only if there is a unique minimal language among $L_{j_1}, L_{j_2}, \dots, L_{j_\ell}$ containing $\mathrm{content}(t[m])$. In this case, the index j_z of this unique minimal language is output. Otherwise, the IIM M repeats its previous conjecture. Note that the IIM M can check and find the unique minimal language as above, as the ordering among languages based on subset can be determined based on $L_{j_z}^r$ as mentioned above. It follows that M is strong monotonic (as it always outputs the unique minimal index containing the input) and identifies $(L_j)_{j \in \mathbb{N}}$ with nonconstructive information $2 \log n$. $\square$

Again, the bound given in Theorem 3.4 cannot be improved substantially, since we have the following lower bound.

Theorem 3.5. *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.*

Proof. Again we use diagonalization to show the desired result. Let $\Sigma = \{a, b\}$. We construct a family $(L_j)_{j\in\mathbb{N}}$ of uniformly recursive languages. As in the proof of Theorem 3.3 let $M_i = \varphi_i$, where $\varphi \in \mathcal{P}^2$ is any fixed Gödel numbering of $\mathcal{P}$, and we use the same notations as there for IIMs. The construction is done in Blocks, where in Block i, $i \in \mathbb{N}^+$, we shall define languages L_{2^i+j} for all $j \in \{0, \dots, 2^i - 1\}$ by diagonalizing against M_{i-1} and the length-lexicographically first $2^{2(i-1)}$ possible help-words $\beta \in \{0, 1\}^*$. For the sake of completeness we set $L_0 = \emptyset$ and $L_1 = \{b\}$.

For every $i \geq 1$ we have to define 2^i many languages. We divide these languages in two groups of size 2^{i-1}, where the first group contains the languages L_{2^i+j} for $j = 0, \dots, 2^{i-1} - 1$. In order to simplify notation, we refer to the languages in the first group as L_{p_j}, $j = 0, \dots, 2^{i-1} - 1$, that is, $L_{p_j} = L_{2^i+j}$. The second group contains the remaining languages L_{2^i+j} for $j = 2^{i-1}, \dots 2^i - 1$. We refer to these languages as L_{q_k}, where $k = 0, \dots, 2^{i-1} - 1$, that is, $L_{q_k} = L_{2^i+2^{i-1}+k}$. Since the construction is a bit more complicated, for each L_{p_j} we incrementally define a text t_i^j, where $j = 0, \dots, 2^{i-1} - 1$.

In Block i do the following.

Initialize the languages $L_{p_j} = \{a^i\}$ for all $j = 0, \dots, 2^{i-1} - 1$ and $L_{q_k} = \{a^i\}$ for all $k = 0, \dots, 2^{i-1} - 1$, set $C_i = \{\beta \mid 0 \leq |\beta| < 2(i-1)\} \cup \{0^{2(i-1)}\}$, and let $t_i^j(0) = a^i$ for all $j = 0, \dots, 2^{i-1} - 1$. Furthermore, set $x = 0$ and set $D_j = \emptyset$ for $j = 0, \dots, 2^{i-1} - 1$. Go to Stage 1.

Stage s. Execute the following loop.

For $j = 0$ to $2^{i-1} - 1$ do
- $x = x + 1$;
- $L_{p_j} = L_{p_j} \cup \{a^i b^x\}$;
- $t_i^j(x) = a^i b^x$;
- $t_i^z(x) = t_i^z(x-1)$ for all $z \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$;
- $L_{q_z} = L_{q_z} \cup \{a^i b^x\}$ for all $z \in \{0, \dots, 2^{i-1} - 1\}$;
- **For** $k = 0$ to $2^{i-1} - 1$ do
  - If $k \in D_j$ then
    - do nothing in this iteration and continue with the next iteration of this For loop;
  - Else (that is, $k \notin D_j$) continue with the following.
  - For all $\beta \in C_i$ and $\ell = 0, \dots, x$ check whether or not

$$M_{i-1}(t_i^j[\ell], \beta)\!\downarrow = p_j \tag{4}$$

  - can be verified within at most s steps of computation.
  - Below we refer to (4) as Test (4).
  - If the Test (4) is satisfied for a $\beta \in C_i$ then
    - $C_i = C_i \setminus \{\beta\}$;
    - $x = x + 1$;
    - $L_{p_j} = L_{p_j} \cup \{a^i b^x\}$;
    - $t_i^j(x) = a^i b^x$;
    - $t_i^z(x) = t_i^z(x-1)$ for all $z \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$;

$L_{q_z} = L_{q_z} \cup \{a^i b^x\}$ for all $z \in \{0, \ldots, 2^i - 1\} \setminus \{k\}$;
$D_j = D_j \cup \{k\}$;
Else do nothing.
endfor
endfor
If $C_i \neq \emptyset$, Go to Stage $s + 1$.

End Stage s.

It should also be noted that we introduced the variable x above just to make sure that all texts have the same length and that any new element added to the corresponding languages has not been inserted in an earlier execution of the nested loops in the current stage nor did we insert the new element in a previous stage.

Now, it is easy to see that $(L_\ell)_{\ell \in \mathbb{N}}$ is a family of uniformly recursive languages.

Let $i \geq 1$ be arbitrarily fixed and suppose by way of contradiction that M_{i-1} strong-monotonically learns all the languages L_{2^i+j}, where $0 \leq j \leq 2^i - 1$, defined in Block i by using help-words of size less than $2(i-1)$ or the help-word $0^{2(i-1)}$. Note that

$$2(i-1) = 2(\lfloor \lg(2^{i+1} - 1) \rfloor + 1) - 4 = 2\log(2^{i+1} - 1) - 4$$

by the convention made at the very end of Section 2.

Furthermore, note that by construction for every j the set D_j can be updated at most 2^{i-1} many times. As soon as $D_j = \{0, \ldots, 2^{i-1} - 1\}$ the second loop does nothing for this j. Also note that for any change of some D_j a success in Test (4) is necessary. Thus, when $D_j = \{0, \ldots, 2^{i-1} - 1\}$ for all j, we must have also deleted all help-words.

Let β be any help-word in C_i as at the start of the construction. We distinguish the following cases.

Case 1. The help-word β is not removed from C_i in any of the stages.

In this case, there must be at least one j_* such that $D_{j_*} \subset \{0, \ldots, 2^{i-1} - 1\}$. So, if the Test (4) does never succeed for such a j_*, when using the help-word β then for $L_{p_{j_*}}$ we have a text $t_i^{j_*}$ on which the IIM M_{i-1} fails to learn $L_{p_{j_*}}$ when using β. So, the IIM M_{i-1} does not learn $L_{p_{j_*}}$ when using β as help-word for at least one text, a contradiction.

Case 2. The help-word β is removed from C_i in some stage.

Let s be the stage in which this happens. Then there must be a j_* and a k_* such that the variable j in the first loop takes the value j_* and the variable k in the second loop takes the value k_* and the Test (4) is satisfied for some ℓ. This also implies that $k_* \notin D_{j_*}$ at that point.

Now, by construction we know that $\mathrm{content}(t_i^{j_*}[x]) \subseteq L_{q_{k_*}}$ and $\mathrm{content}(t_i^{j_*}[x]) \subseteq L_{p_{j_*}}$, where x is as at the time when Test (4) is executed. However, now the language $L_{p_{j_*}}$ obtains the new element $a^i b^{x+1}$ while $L_{q_{k_*}}$ does *not*. Consequently, we have $L_{p_{j_*}} \not\subseteq L_{q_{k_*}}$. So, we have an initial segment of a text for $L_{q_{k_*}}$, that is, $t_i^{j_*}[x]$, such that the IIM M_{i-1}, using the help-word β, cannot strong-monotonically learn $L_{q_{k_*}}$ from any text extending $t_i^{j_*}[x]$, since it has output p_{j_*} on $t_i^{j_*}[\ell]$ for some $\ell \leq x$. This is a contradiction.

Consequently, for each help-word β there is at least one language in Block i that is not learned at all by M_{i-1} or for which M_{i-1} is not strong-monotonic.

The rest is done *mutatis mutandis* as in the proof of Theorem 3.3 and therefore omitted. □

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 from $d \in \mathcal{R}_{mon}$ we can find u_* as in the proof of Theorem 3.1, and then q such that $2^q > u_*$, and then p as in the proof of Theorem 3.4. This gives us the following theorem.

Theorem 3.6. *Let $\mathcal{C} \subseteq \mathcal{REG}$ be arbitrarily fixed, let $\mathcal{H} = (L_j)_{j \in \mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$ such that $\mathcal{H}$ has a decidable subset problem, and let $d \in \mathcal{R}_{mon}$ be any function. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified by M 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 3.7. *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 3.2, that is, for $q = \log n$, it is a bitstring w of length q which is the binary representation of $\ell - 1$, where ℓ is the number of pairwise different languages among $W_0^\psi, \ldots, W_{2^q-1}^\psi$. Recall that the help-word also provides q due to its length. Let $L \in \mathcal{C} \cap \{W_0^\psi, \ldots, W_{2^q-1}^\psi\}$ and let t be a text for L. On input $t[m]$ and the help-word w the desired IIM M executes the following.

(1) For all $i < 2^q$ 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, for $i < 2^q$.

(2) For all pairs (i, j), with $i, j < 2^q$, 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 $d(i, j) = \infty$. Let $x_{i,j} = \min(W_{i,m}^\psi \triangle W_{j,m}^\psi)$, for $i, j < 2^q$. Here if $W_{i,m}^\psi = W_{j,m}^\psi$, then $x_{i,j} = \infty$.

(3) Having the numbers $d(i, j)$ the IIM M checks whether or not there exist at least ℓ pairwise different languages among $W_{0,m}^\psi, \ldots, W_{2^q-1,m}^\psi$. If this is not the case, then $M(t[m]) = 0$.
For $S \subseteq \{i \mid i < 2^q\}$ of size ℓ, let $s(S) = \max\{x_{i,j} \mid i, j \in S, i \neq j\}$. Note that if there are at least ℓ pairwise different languages among $W_{0,m}^\psi, \ldots, W_{2^q-1,m}^\psi$,

then $s(S)$ is finite for some S of size ℓ (the S which contain ℓ different members of $W^{\psi}_{0,m}, \ldots, W^{\psi}_{2^q-1,m}$).

Let $\tilde{S}$ of size ℓ be such that $s(\tilde{S})$ is minimized (where if there are several such $\tilde{S}$, choose the lexicographically smallest one in some ordering). Suppose $\tilde{S} = \{i_1, i_2, \ldots, i_\ell\}$. Then M takes the languages $W^{\psi}_{i_1,m}, \ldots, W^{\psi}_{i_\ell,m}$ into consideration. From these candidate hypotheses $i_1, \ldots, i_\ell$ the least i is output for which $\mathrm{content}(t[m])$ contains all finite $d(i,j)$, $j = i_1, \ldots, i_\ell$, and $\mathrm{content}(t[m])$ does not contain any of the finite $d(j,i)$, $j = i_1, \ldots, i_\ell$. If there is no such i, then $M(t[m]) = 0$.

It remains to argue 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 $\tilde{S}$ and thus the sequence $i_1, i_2, \ldots, i_\ell$ and all finite $d(j,k)$, with $j,k \in \{i_1, i_2, \ldots, i_\ell\}$ stabilize in the limit. By construction M then outputs the correct i as soon as the initial segment $t[m]$ is large enough to contain all finite $d(i,j)$ for $j \in \{i_1, i_2, \ldots, i_\ell\} \setminus \{i\}$. $\square$

Note that the IIM defined in the proof of Theorem 3.7 even witnesses a much stronger result, that is, it always converges to the minimum index i of the target language.

The lower bound in Theorem 3.9 shows that Theorem 3.7 cannot be improved substantially. In order to show this result we need the following proposition (a more generalized version of this proposition is proved in Proposition 3.10).

Proposition 3.8. *Given a set $\mathcal{M} = \{M_1, M_2, M_3, \ldots, M_m\}$ of m IIMs, one can effectively find grammars $g^{\mathcal{M}}_0, g^{\mathcal{M}}_1, \ldots, g^{\mathcal{M}}_m$ such that none of $M_1, M_2, \ldots, M_m$ can learn in the limit each and every of the languages $W_{g^{\mathcal{M}}_0}, W_{g^{\mathcal{M}}_1}, \ldots, W_{g^{\mathcal{M}}_m}$.*

Theorem 3.9. *There is a numbering $\psi \in \mathcal{P}^2$ such that no IIM M can learn the family $(W^{\psi}_i)_{i\in\mathbb{N}^+}$ in the limit from text with nonconstructivity $\log n - 2$ with respect to $(W^{\psi}_i)_{i\in\mathbb{N}^+}$.*

Proof. Using Proposition 3.8, the theorem can be shown as follows. For any learner M, let M^{β} be the learner which behaves just like M with the advice β as additional input.

We define the numbering ψ as follows. Let φ be any fixed Gödel numbering of $\mathcal{P}$. We interpret every partial recursive function φ_i as an IIM M_i, that is, $M_i = \varphi_i$ for every $i \in \mathbb{N}$. Let M^{β}_i denote the machine M_i using the advice β.

The set W^{ψ}_j, for $j \in \{2^i, \ldots, 2^{i+1} - 1\}$, is defined to be $W_{g^{\mathcal{M}}_{j-2^i}}$, where $\mathcal{M} = \{M^{\beta}_i \mid |\beta| < i\}$. Note that there are $2^i - 1$ many help-words β with $0 \leq |\beta| < i$, and we thus have the needed number 2^i of grammars (cf. Proposition 3.8). Thus, for any i and any β of length at most $i-1$, the IIM M^{β}_i does not learn at least one of the languages W^{ψ}_j, $j \in \{2^i, \ldots, 2^{i+1} - 1\}$.

Now, suppose by way of contradiction that there is an IIM M that learns the whole family $(W^{\psi}_j)_{j\in\mathbb{N}^+}$ using nonconstructivity $\log n - 2$. Then there must be an $i \in \mathbb{N}$ such that $M = M_i$. So this IIM M_i learns the family $(W_j^\psi)_{j\in\mathbb{N}^+}$ using nonconstructivity $\log n - 2$. By using $n = 2^{i+1} - 1$, the IIM M_i^β, for some β with length at most $i-1$, learns all the languages W_j^ψ, for $j \in \{2^i, \ldots, 2^{i+1}-1\}$. A contradiction. $\square$

So, it remains to show Proposition 3.8. We shall show a bit stronger result.

Proposition 3.10. *Given a set $\mathcal{M} = \{M_1, M_2, M_3, \ldots, M_m\}$ of m IIMs, and two disjoint finite sets S_1 and S_2, where $S_1, S_2 \subseteq \mathbb{N}$, one can effectively (in $\mathcal{M}, S_1, S_2$) find grammars $g_0^{\mathcal{M},S_1,S_2}$, $g_1^{\mathcal{M},S_1,S_2}$, …, $g_m^{\mathcal{M},S_1,S_2}$ such that none of the IIMs $M_1, \ldots, M_m$ learns in the limit all the languages $W_{g_0^{\mathcal{M},S_1,S_2}}$, $W_{g_1^{\mathcal{M},S_1,S_2}}$, …, $W_{g_m^{\mathcal{M},S_1,S_2}}$, where we additionally have the properties that*

(a) $S_1 \subseteq W_{g_i^{\mathcal{M},S_1,S_2}}$, *for $i \le m$, and*
(b) $S_2 \cap W_{g_i^{\mathcal{M},S_1,S_2}} = \emptyset$, *for $i \le m$.*

Proof. We show below how to define $g_j^{\mathcal{M},S_1,S_2}$, for every j with $0 \le j \le |\mathcal{M}|$. These grammars are generated implicitly by using the operator recursion theorem [5]. We will define $W_{g_i^{\mathcal{M},S_1,S_2}}$ inductively on size of $\mathcal{M}$. If $\mathcal{M} = \emptyset$ then we let $W_{g_0^{\mathcal{M},S_1,S_2}} = S_1$.

Inductively, for $m > 0$, suppose $g_j^{\mathcal{M},S_1,S_2}$ has been defined for all $\mathcal{M}$ of size less than m, and all disjoint finite sets S_1, S_2.

Suppose $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ and let disjoint finite sets S_1, S_2 be given. The sets $W_{g_j^{\mathcal{M},S_1,S_2}}$, for $j \in \{1, \ldots, m\}$, are defined via a staging construction as follows. Note that the construction is different for each different input $\mathcal{M}, S_1, S_2$.

Fix such $\mathcal{M}, S_1, S_2$. For ease of notation below let h_j denote $g_j^{\mathcal{M},S_1,S_2}$.

Let σ_0 be a sequence with content S_1. Enumerate S_1 in each of W_{h_j}, $j \le m$.

At the beginning of Stage s, for $j \le m$, it will be the case that $\mathrm{content}(\sigma_s) = W_{h_j}$ enumerated before Stage s. Go to Stage 0.

Stage s. Dovetail Steps 1 and 2, until, if ever, Step 1 succeeds. If Step 1 succeeds, then stop Step 2 and go to Step 3.

***Step* 1.** Search for an extension σ of σ_s, where $\mathrm{content}(\sigma) \cap S_2 = \emptyset$, such that the condition $M_{(s \bmod m)+1}(\sigma_s) \ne M_{(s \bmod m)+1}(\sigma)$ is verified.

***Step* 2.** Let x be such that $x \notin \mathrm{content}(\sigma_s) \cup S_2$.
Enumerate x in W_{h_m}.
Let $S_2' = S_2 \cup \{x\}$. Let $S_1' = \mathrm{content}(\sigma_s)$.
Let $\mathcal{M}' = \mathcal{M} \setminus \{M_{(s \bmod m)+1}\}$.
Let W_{h_j}, for $j < m$, output more and more of $W_{g_j^{\mathcal{M}',S_1',S_2'}}$ until, if ever search in Step 1 succeeds (note that if and when search in Step 1 succeeds, W_{h_j} would have output only finitely many elements from $W_{g_j^{\mathcal{M}',S_1',S_2'}}$).

***Step* 3.** If and when Step 1 succeeds, let $X = \mathrm{content}\,(\sigma) \cup \bigcup_{j\le m} W_{h_j}$ enumerated up to now.
Enumerate X in W_{h_j}, for $j \le m$.
Let σ_{s+1} be an extension of σ such that $\mathrm{content}(\sigma_{s+1}) = X$.
Go to Stage $s+1$

End ***Stage*** $\boldsymbol{s}$**.**

We now claim that $g_j^{\mathcal{M},S_1,S_2}$ as defined above satisfy the proposition. This is by induction on size of $\mathcal{M}$. The proposition clearly holds when $|\mathcal{M}| = 0$ as $g_0^{\mathcal{M},S_1,S_2}$ is not learnt by any machine in $\mathcal{M}$ (as $\mathcal{M}$ is empty). For $m > 0$, suppose the proposition properties hold for all $\mathcal{M}$ of size less than m. Then, we show that it holds for $\mathcal{M}$ of size m.

Fix $\mathcal{M}$ (of size m) and $S_1,\ S_2$, and consider the following cases for the construction given above.

Case 1. There are infinitely many stages.

In this case, all W_{h_j}, $j \leq m$, are the same language given by $\mathrm{content}(\bigcup_{s\in\mathbb{N}} \sigma_s)$, and each of the learners $M_1, M_2, \ldots, M_m$ makes infinitely many mind changes on the text $\bigcup_{s\in\mathbb{N}} \sigma_s$.

Case 2. Stage s starts but does not finish.

In this case, clearly, $M_{(s \bmod m)+1}$ converges to $M_{(s \bmod m)+1}(\sigma_s)$ on all texts extending σ_s which do not contain any element from S_2. Thus, either $M_{(s \bmod m)+1}$ does not learn W_{h_m} (which contains x as defined in Stage s) or does not learn any of W_{h_j}, $j < m$, which do not contain x. Finally, by the induction hypothesis, none of the learners in $\mathcal{M}'$ learns all the languages W_{h_j}, for $j < m$. Thus, the proposition follows. $\square$

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 $h(i) = \lim_{t\to\infty} \tilde{h}(i,t)$ for all $i \in \mathbb{N}$.

Theorem 3.11. *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 $(W_i^{\psi})_{i\in\mathbb{N}}$.*

Proof. Let the limiting recursive function $h\colon \mathbb{N} \to \mathbb{N}$ be arbitrarily fixed. Furthermore, we assume any fixed function $\tilde{h} \in \mathcal{R}^2$ such that $h(i) = \lim_{n\to\infty} \tilde{h}(i,n)$ for all $i \in \mathbb{N}$.

Again, we fix any Gödel numbering φ of $\mathcal{P}$ and interpret each φ_i as an IIM, that is, we have $M_i = \varphi_i$ for all $i \in \mathbb{N}$.

For every number $i \in \mathbb{N}$ we construct two recursively enumerable sets W_{2i}^{ψ} and W_{2i+1}^{ψ} such that for each help-word $\beta \in \{0,1\}^*$ with $0 \leq |\beta| \leq h(2i+1)$, M_i fails to learn at least one of the languages W_{2i}^{ψ} and W_{2i+1}^{ψ} or M_i is not conservative. This is done as follows.

For each number $i \in \mathbb{N}$ do the following.

Begin

Set $t = 0$.

Initialize $W_{2i}^{\psi} = \{\langle i, 2x\rangle \mid x \in \mathbb{N}\}$ and $W_{2i+1}^{\psi} = \{\langle i, 2x+1\rangle \mid x \in \mathbb{N}\}$.

Go to (1).

(1) Compute $m_t = \tilde{h}(2i+1, t)$ and let $r_t = 2^{m_t+1} - 1$.
We define $C_t = \{1, \ldots, r_t\}$ and enumerate in lexicographic order all strings $\beta \in \{0,1\}^*$ with $0 \le |\beta| \le m_t$. We refer to these elements as β_j, where $j \in C_t$.
Dovetail the execution of (α) and (2).

(α) **Do** forever

$t = t + 1$;
`if` $\tilde{h}(2i+1, t-1) < \tilde{h}(2i+1, t)$ `then`
let $m_t = \tilde{h}(2i+1, t)$ and $r_t = 2^{m_t+1} - 1$;
Update $C_t = \{1, \ldots, r_t\}$ and update (A) by including the newly found $j \in C_t$.

enddo

(2) Start (A).

(A) For each $j \in C_t$ try to find, by dovetailing the following search, a finite sequence σ_{β_j} of elements from $\{\langle i, x\rangle \mid x \in \mathbb{N}\}$ such that the condition $M_i(\sigma_{\beta_j}, \beta_j)\!\downarrow = 2i$ is verified.
Once such a sequence σ_{β_j} is found, if ever, stop (A) for the j for which the search succeeded but continue for the remaining values of j, and update

$$W^{\psi}_{2i} = W^{\psi}_{2i} \cup \text{content}\,(\sigma_{\beta_j})\,, \tag{5}$$
$$W^{\psi}_{2i+1} = W^{\psi}_{2i+1} \cup \text{content}\,(\sigma_{\beta_j})\,. \tag{6}$$

Now start (B) for the j for which the search succeeded.

(B) Try to find a finite sequence τ_{β_j} of elements from $\{\langle i, x\rangle \mid x \in \mathbb{N}\}$ such that the condition $M_i(\sigma_{\beta_j} \diamond \tau_{\beta_j}, \beta_j)\!\downarrow = 2i+1$ is verified, where $\sigma_{\beta_j} \diamond \tau_{\beta_j}$ denotes the concatenation of the finite sequences σ_{β_j} and τ_{β_j}.
If such a τ_{β_j} is found, if ever, stop (B) for the j for which the search succeeded and update

$$W^{\psi}_{2i} = W^{\psi}_{2i} \cup \text{content}\,(\tau_{\beta_j})\,, \tag{7}$$
$$W^{\psi}_{2i+1} = W^{\psi}_{2i+1} \cup \text{content}\,(\tau_{\beta_j})\,. \tag{8}$$

Furthermore, set $C_t = C_t \setminus \{j\}$. If $C_t \neq \emptyset$ continue (A) or (B) for the remaining $j \in C_t$.
If $C_t = \emptyset$ then stop (A) and (B) until, if ever, in (α) the set C_t is updated.

End

Now, it is easy to see that $(W^{\psi}_{\ell})_{\ell\in\mathbb{N}}$ is a recursively enumerable family of recursively enumerable languages. It remains to show that no IIM M can learn $(W^{\psi}_{\ell})_{\ell\in\mathbb{N}}$ conservatively with nonconstructivity h with respect to $(W^{\psi}_{\ell})_{\ell\in\mathbb{N}}$.

Suppose the converse, that is, there is an IIM M that learns $(W^{\psi}_{\ell})_{\ell\in\mathbb{N}}$ conservatively with respect to $(W^{\psi}_{\ell})_{\ell\in\mathbb{N}}$. In order to obtain a contradiction, it suffices to show that there is some $n \in \mathbb{N}$ such that for each help-word w with $0 \le |w| \le h(n)$ there is a language W from our family such that $W \in \{W_0^\psi, \dots, W_n^\psi\}$ but W is not learned conservatively by M.

Since M is an IIM, there must be an $i \in \mathbb{N}$ such that $M = M_i$. Furthermore, since h is limiting recursive, there must be $t_0 \in \mathbb{N}$ such that $h(2i+1) = \tilde{h}(2i+1, t_0 + k)$ for all $k \in \mathbb{N}$. That is, as soon as our t in (α) satisfies $t \ge t_0$ no further update of C_t is occurring in (α). It should also be noted that we may have included more help-words β_j in our construction than actually necessary, since we only checked if $\tilde{h}(2i+1, t)$ does increase. But this does not matter as we shall see below.

We claim that for $n = 2i+1$ the desired contradiction can be obtained. Let us consider the sets W_{2i}^ψ and W_{2i+1}^ψ.

As said above, as soon as $t \ge t_0$ the set C_t does not change any further in the loop (α). So, let $m = h(2i+1)$ and $r = 2^{m+1} - 1$. It suffices to consider all help-words $\beta_1, \dots, \beta_r$, where $0 \le |\beta_j| \le m$ for all $j \in \{1, \dots, r\}$.

Let any such β_j be arbitrarily fixed. Then we try in (A) in particular any finite sequence σ with $\mathrm{content}(\sigma) \subseteq \{\langle i, x\rangle \mid x \in \mathbb{N}\}$ of elements of W_{2i}^ψ. Thus, if for all these finite sequences σ either $M_i(\sigma, \beta_j)$ remains undefined or turns out to be defined but $M_i(\sigma, \beta_j) \ne 2i$ then M_i fails to learn W_{2i}^ψ when using the help-word β_j, since there is no other index for this language in $(W_\ell^\psi)_{\ell \in \mathbb{N}}$. Here it should be noted that $W_{2i}^\psi \ne W_{2i+1}^\psi$, since the at most finitely often occurring updates in (5), (6), (7), and (8) do only add finitely often finitely many elements to W_{2i}^ψ and W_{2i+1}^ψ, respectively.

Otherwise, the search must eventually succeed, that is, we find a finite sequence σ_{β_j} such that $M_i(\sigma_{\beta_j}, \beta_j)\downarrow = 2i$.

Now, by construction we know that $\mathrm{content}\,(\sigma_{\beta_j}) \subseteq W_{2i+1}^\psi$, too (see Instruction (6)). So, in (B) we try in particular any finite sequence τ of elements of W_{2i+1}^ψ, and by construction also each sequence $\sigma_{\beta_j} \diamond \tau$ is an initial segment of a text for W_{2i+1}^ψ. Consequently, if we never verify $M_i(\sigma_{\beta_j} \diamond \tau, \beta_j)\downarrow = 2i+1$ in (B) then M_i fails to learn W_{2i+1}^ψ from every text starting with σ_{β_j}.

Otherwise, in (B) we find a τ_{β_j} such that $M_i(\sigma_{\beta_j} \diamond \tau_{\beta_j}, \beta_j)\downarrow = 2i+1$.

Taking (8) and (7), respectively, into account, by construction we know that

$$\mathrm{content}\,(\sigma_{\beta_j}) \cup \mathrm{content}\,(\tau_{\beta_j}) \subseteq W_{2i+1}^\psi$$

as well as

$$\mathrm{content}\,(\sigma_{\beta_j}) \cup \mathrm{content}\,(\tau_{\beta_j}) \subseteq W_{2i}^\psi\ .$$

Thus, $\sigma_{\beta_j} \diamond \tau_{\beta_j}$ is an initial segment of a text for both W_{2i}^ψ and W_{2i+1}^ψ. But since

$$M_i(\sigma_{\beta_j}, \beta_j)\downarrow = 2i \ne 2i+1 = M_i(\sigma_{\beta_j} \diamond \tau_{\beta_j}, \beta_j)\downarrow\ ,$$

we see that $M_i(\sigma_{\beta_j}, \beta_j) \ne M_i(\sigma_{\beta_j} \diamond \tau_{\beta_j}, \beta_j)$ even though $\mathrm{content}\,(\sigma_{\beta_j} \diamond \sigma_{\tau_j})$ is contained in W_{2i}^ψ. This is a contradiction to M_i (with help-word β_j) being conservative.

So, for every help-word β_j, $0 \le |\beta_j| \le m$ the IIM M_i either does not learn W_{2i}^ψ or W_{2i+1}^ψ or it is not conservative. Thus, the theorem is shown. $\square$

Since *SmonTxt* $\subseteq$ *ConsvTxt* (see [16, 18, 22]), Theorem 3.11 directly allows for the following corollary.

Corollary 3.12. *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 $(W_i^\psi)_{i\in\mathbb{N}}$ strong-monotonically with respect to the hypothesis space $(W_i^\psi)_{i\in\mathbb{N}}$.*

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, that is, *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 3.1).

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 pairwise containment of languages in the hypothesis space 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 or strong-monotonically.

Acknowledgment

We thank the anonymous referee for several helpful comments which improved the presentation of the paper. 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. Preliminary version of this paper appeared in TAMC 2012 [19].

References

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

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

[3] Jānis M. Bārzdiņš. 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.

[4] Olaf Beyersdorff, Johannes Köbler, and Sebastian Müller. Proof systems that take advice. *Information and Computation*, 209(3):320–332, 2011.

[5] John Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8(1):15–32, 1974.

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

[7] Carsten Damm and Markus Holzer. Automata that take advice. In *Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS '95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings*, volume 969 of *Lecture Notes in Computer Science*, pages 149–158, Berlin, 1995. Springer.

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

[9] Rūsiņš Freivalds. Amount of nonconstructivity in finite automata. In Sebastian Maneth, editor, *Implementation and Application of Automata, 14th International Conference, CIAA 2009, Sydney, Australia, July 14-17, 2009. Proceedings*, volume 5642 of *Lecture Notes in Computer Science*, pages 227–236, Berlin, 2009. Springer.

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

[11] Rūsiņš Freivalds and Thomas Zeugmann. On the amount of nonconstructivity in learning recursive functions. In Mitsunori Ogihara and Jun Tarui, editors, *Theory and Applications of Models of Computation, 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011, Proceedings*, volume 6648 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 2011.

[12] Rusins Freivalds and Thomas Zeugmann. Co–learning of recursive languages from positive data. In Dines Bjørner, Manfred Broy, and Igor V. Potttosin, editors, *Perspectives of System Informatics, Second International Andrei Ershov Memorial Conference, Akademgorodok, Novosibirsk, Russia, June 1996, Proceedings*, volume 1181 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 1996.

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

[14] Sanjay Jain, Efim Kinber, Steffen Lange, Rolf Wiehagen, and Thomas Zeugmann. Learning languages and functions by erasing. *Theoretical Computer Science*, 241(1-2):143–189, 2000. Special issue for ALT '96.

[15] Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. *Systems that Learn: An Introduction to Learning Theory, second edition*. MIT Press, Cambridge, Massachusetts, 1999.

[16] Sanjay Jain and Arun Sharma. On monotonic strategies for learning r.e. languages. In Setsuo Arikawa and Klaus P. Jantke, editors, *Algorithmic Learning Theory, 4th International Workshop on Analogical and Inductive Inference, AII '94, 5th International Workshop on Algorithmic Learning Theory, ALT '94, Reinhardsbrunn Castle, Germany, October 1994, Proceedings*, volume 872 of *Lecture Notes in Artificial Intelligence*, pages 349–364. Springer-Verlag, 1994.

[17] Sanjay Jain and Arun Sharma. Computational limits on team identification of languages. *Information and Computation*, 130:19–60, 1996.

[18] Sanjay Jain and Arun Sharma. Generalization and specialization strategies for learning r.e. languages. *Annals of Mathematics and Artificial Intelligence*, 23(1/2):1–26, 1998.

[19] Sanjay Jain, Frank Stephan, and Thomas Zeugmann. On the amount of nonconstructivity in learning formal languages from positive data. In Manindra Agrawal, S. Barry Cooper, and Ansheng Li, editors, *Theory and Applications of Models of Computation, 9th Annual Conference, TAMC 2012, Beijing, China, May 16-21, 2012, Proceedings*, volume 7287 of *Lecture Notes in Computer Science*, pages 423–434. Springer, 2012.

[20] Klaus P. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8(4):349–360, 1991.

[21] Richard M. Karp and Richard J. Lipton. Turing machines that take advice. *L' Enseignement Mathématique*, 28:191–209, 1982.

[22] Efim Kinber and Frank Stephan. Language learning from texts: Mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.

[23] Steffen Lange and Thomas Zeugmann. Types of monotonic language learning and their characterization. In David Haussler, editor, *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 377–390, New York, NY, 1992. ACM Press.

[24] Steffen Lange and Thomas Zeugmann. Language learning in dependence on the space of hypotheses. In Lenny Pitt, editor, *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 127–136, New York, NY, 1993. ACM Press.

[25] Steffen Lange, Thomas Zeugmann, and Sandra Zilles. Learning indexed families of recursive languages from positive data: A survey. *Theoretical Computer Science*, 397(1-3):194–232, 2008.

[26] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted, MIT Press 1987.

[27] Carl H. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29(4):1144–1165, 1982.

[28] Thomas Zeugmann and Steffen Lange. A guided tour across the boundaries of learning recursive languages. In *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer, 1995.