THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Computing 1, Singapore 117590

TRA1/09

Learnability of Automatic Classes

Sanjay Jain, Qinglong Luo and Frank Stephan

January 2009

T e c h n i c a l R e p o r t

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

OOI Beng Chin
Dean of School

Learnability of Automatic Classes

Sanjay Jain[⋆1], Qinglong Luo[1] and Frank Stephan[⋆⋆2]

[1] Department of Computer Science,
National University of Singapore, Singapore 117417, Republic of Singapore.
sanjay@comp.nus.edu.sg, luoqingl@comp.nus.edu.sg
[2] Department of Computer Science and Department of Mathematics,
National University of Singapore, Singapore 117543, Republic of Singapore.
fstephan@comp.nus.edu.sg

Abstract. The present work initiates the study of the learnability of automatic indexable classes which are classes of regular languages of a certain form. It is characterised which of these classes are explanatorily learnable. Furthermore, the notion of an automatic iterative learner is introduced and it is studied for automatic classes whether they are learnable by an automatic iterative learner, learnable by an automatic iterative learner with additional long-term memory or unlearnable by such a learner. The dependence of the learnability on the indexing is also investigated. This work brings together the fields of inductive inference and automatic structures.

1 Introduction

Consider the following scenario for learning. A learner is receiving, one piece at a time, data about a target concept. As the learner is receiving the data, it conjectures its hypothesis about what the target concept might be. The hypothesis may be modified/changed as more data is received. One can consider the learner to be successful if the sequence of hypotheses converges to a correct hypothesis which explains the target concept. This is essentially the model of explanatory learning proposed by Gold [11].

The concept classes of interest to us in this paper are the classes of languages (a language is a subset of Σ^*, for some finite alphabet Σ). The data provided to the learner then becomes a sequential presentation of all the elments of the target language, in arbitrary order, with repetition allowed (for technical reasons, to deal with empty set, we also allow a special symbol called $\Box$, representing no data, to be presented to the learner). Such a presentation of data is called a *text* for the language. Note that in a text, only positive data is presented to the learner, and negative data is not given (that is, the learner is not explicitly told that such and such elements do not belong to the language). If both positive and negative data are presented to the learner, then the mode of presentation is called *informant*. In this paper we will only be concentrating on learning from texts.

In most cases, one considers only recursive learners. The hypotheses produced by the learner

⋆ Supported in part by NUS grant number R252-000-308-112.
⋆⋆ Supported in part by NUS grant numbers R252-000-308-112 and R146-000-114-112.

describe the language to be learnt in some form, for example, they are grammars generating the language. The learner is said to **Ex**-learn the target language, if the sequence of hypotheses converges to one correct hypothesis describing the language to be learnt (**Ex** here stands for explanatory learning). Learning of one language L is not interesting, as the learner might ignore all inputs and always output the same hypothesis which is correct for L. Thus, what is considered is whether all languages from a class of languages is **Ex**-learnt by some particular learner. Finally, a class of languages is **Ex**-learnable if some learner **Ex**-learns it.

Since Gold, several other models of learning have been considered by the researchers. In **BC**-learning [4] the learner is not required to converge syntactically to one correct hypothesis, but it is just required that all hypotheses are correct from some time onwards. In other words, one requires only semantic convergence in this case. In **FEx**-learning [9], in addition to **BC**-learning one requires that the learner output only finitely many distinct hypotheses on any input text. That is, the learner eventually only vacillates between finitely many correct hypotheses.

Besides, the mode of convergence, researchers have also considered several properties of learner such as *consistency*, where the hypothesis of the learner is required to contain the elements in the input σ (see [5, 6]), *conservativeness*, where the learner is not allowed to change a hypothesis which is consistent with the data seen so far (see [1]) and *iterativeness*, where the new hypothesis of the learner depends only on the previous hypothesis and the latest datum (see [26, 27]). The formal definitions of the above criteria is given in Section 2 below.

Besides considering models of learning, there has also been interest in considering practical and concrete classes of learning like pattern languages [2, 10, 15, 18] and the class of regular languages [3]. As the class of all regular languages is not learnable from positive data [11], we consider in this work those natural subclasses where the membership problem is regular in the sense that one automaton accepts a combination (called "convolution") of an index and a word if and only if the word is in the language given by the index. This is formalized in the framework of automatic structures [7, 8, 12, 13, 16, 23, 24]. Such automatic classes can in some sense be considered to be simple. Here are some examples of automatic classes:

- The class of sets with up to k elements for a constant k;
- The class of all finite and cofinite subsets of $\{0\}^*$;
- The class of all intervals of an automatic linear order on a regular set;
- Given an automatic presentation of $(\mathbb{Z}, +, <)$ and a first-order formula $\Phi(x, a_1, \ldots, a_n)\}$ with parameters $a_1, \ldots, a_n \in \mathbb{Z}$, the class consisting of all sets $\{x \in \mathbb{Z} : \Phi(x, a_1, \ldots, a_n)\}$ with $a_1, \ldots, a_n \in \mathbb{Z}$.

It is known that the automatic relations are closed under first-order theory, as proven by Khoussainov and Nerode [16]. This makes several properties of such classes to be regular and thus decidable; it also makes it possible to define learners using first-order definitions.

A tell-tale set for a language L in a class $\mathcal{L}$ is a finite subset D of L such that, for every $L' \in \mathcal{L}$, $D \subseteq L' \subseteq L$ implies $L' = L$. A class $\mathcal{L}$ satisfies Angluin's Tell-Tale Condition iff every language L in $\mathcal{L}$ has a tell-tale set (with respect to $\mathcal{L}$). Angluin [1] showed that any class of languages which is learnable (even by a non-recursive learner, for which **Ex**-, **BC**- and **FEx**-learning are all the same) must satisfy Angluin's Tell-Tale Condition. We show in Theorem 11

that every automatic class, which satisfies the Angluin's Tell-Tale Condition, is **Ex**-learnable by a recursive learner which is additionally consistent and conservative. Additionally, it is decidable whether an automatic class satisfies the Angluin's Tell-Tale Condition, and thus whether it is **Ex**-learnable (see Corollary 12).

As we are considering learning of automatic structures, it is natural to also consider learners which are simpler than just being recursive. A natural idea would be to consider learners which are themselves described via automatic structures. In general, this does not lead to an interesting formulation as it is not clear how all the data may be presented to the learner, and how it accesses them. However, when one considers the special case of iterative learners, then there is a natural way to describe automatic iterative learners. An iterative learner [26, 27] can be considered as a mapping from previous hypothesis and new datum to a new hypothesis. An automatic iterative learner would then be a learner for which the above mapping is automatic. One can also additionally consider iterative learners with bounded long-term memory [17], where the learner maps its previous conjecture, its previous long-term memory and the new datum, to its new conjecture and new long-term memory.

As automatic structures are relatively simple to implement and analyze, it is interesting to explore the capabilities of such learners. In Section 3 we formally define automatic learners (iterative learners as well as iterative learners with long-term memory). Specifically we consider the following bounds on memory: memory bounded by a constant, memory bounded by the size of the hypothesis, memory bounded by the size of the largest word seen in input so far, besides the default cases of no memory (iterative learning) and the case where we do not put any specific bounds on memory except as implicit from the definition of automatic learners. Theorem 17 shows that there are automatic classes which are **Ex**-learnable (even iteratively) but not learnable by any automatic learners (even without any long-term memory bounds except implicitly due to definition of automatic learner).

In Section 3 we show the relationship between various iterative automatic learners and iterative automatic learners with long-term memory. For example, if long-term memory is not explicitly bounded, then automatic **Ex**-learning is same as automatic **BC**-learning (in contrast to the situation in learning of recursively enumerable languages by recursive learners, where there is a difference [4]). Additionally, for **BC**-learning, different bounds on long-term memory do not make a difference, as all automatically **BC**-learnable classes (with no explicit long-term memory bound) are iteratively automatically **BC**-learnable (See Theorem 18). Similarly, for **FEx**-learning, long-term memory bounds of size of the hypothesis collapses to no-long-term memory case (iterative learning). However, for both explanatory learning and vacillatory learning, there is a difference if one considers long-term memory bounded by hypothesis size, or whether long-term memory is bounded by the size of maximum word seen in the input so far (see Theorem 19). It is open at this point whether long-term memory size bounded by the longest word seen so far is equivalent to there being no explicit bound on long-term memory, for both **Ex** and **FEx**- automatic learning. For explanatory learning, it is additionally open whether constant memory size is equivalent to having hypothesis-size memory and whether maximum word-size

memory can simulate hypothesis-size memory.

In Section 4 we consider consistent learning by automatic learners. Unlike Theorem 11, where we saw that general learners for automatic classes can be made consistent, automatic learners cannot in general be made consistent. Theorem 23 shows that there is an automatic class $\mathcal{L}$ which is **Ex**-learnable by an automatic iterative learner but not **Ex**-learnable by a consistent automatic learner (with no constraints on long-term memory except implicit due to automatic learner). Theorem 24 shows that there is an automatic class $\mathcal{L}$, which is **Ex**-learnable by a consistent automatic learner or an iterative automatic learner, but not by a consistent iterative learner. Theorem 26 shows the existence of an automatic class $\mathcal{L}$ which is **Ex**-learnable by a consistent and iterative automatic learner using a class-comprising hypothesis space (i.e., using hypothesis from an automatic class which is a superset of the class $\mathcal{L}$), but not **Ex**-learnable by a consistent automatic learner (even using unconstrained long-term memory, except that implicit due to automatic learner) using a class preserving hypothesis space (i.e., using hypothesis space which contains languages only from $\mathcal{L}$).

One of the reasons for the difficulty of learning by iterative learners is that they forget past data. A way to overcome this is by requiring that every datum appears infinitely often in the text (such a text is called fat text [22]). In practice, this may be reasonable as every English word is potentially heard by a child infinitely often. It is natural to ask whether this overcomes the limitations of iterative learning. In Section 5 we consider such questions. In Theorem 29 we show that every automatic class which satisfies the Angluin's Tell-Tale Condition is **Ex**-learnable (using the automatic class itself as hypothesis space) from fat texts by an automatic learner with long-term memory bounded by the size of the largest word seen so far. If one allows class preserving hypothesis space, then one can even do **Ex**-learning in the above case by iterative automatic learners (that is, no long-term memory is needed).

In Theorem 32, we show the existence of automatic classes which are automatically iteratively learnable (even from normal texts) using a class preserving hypothesis space, but not conservatively iteratively learnable using a class preserving hypothesis space (even by arbitrary recursive learners) on a fat text.

Partial identification is a very general learning criterion, where one requires that some fixed correct hypothesis is output infinitely often by the learner while all other hypothesis are output only finitely often [22]. In Theorem 34 we show that every automatic class is partially learnable by an automatic iterative learner. This corresponds to the result by [22] that the whole class of recursively enumerable languages is partially learnable by some recursive learner.

2 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers. $\mathbb{Z}$ denotes the set of integers. $\emptyset$ denotes the empty set. $\subseteq, \supseteq, \subset, \supset$ respectively denote subset, superset, proper subset and proper superset. Furthermore, $\max S, \min S$ and $\operatorname{card} S$ respectively denote the maximum, minimum and cardinality of a set S. We take $\max \emptyset$ to be 0 and $\min \emptyset$ to be ∞.

Definition 1. We call any finite non-empty set an *alphabet*. Σ^* is the set of all strings (words) over the alphabet Σ. ϵ denotes the empty string. A string (word) of length n over some alphabet set Σ will be treated as a function from the set $\{0, 1, 2, \ldots, n-1\}$ to Σ. Thus, string x of length n is same as $x(0)x(1)x(2)\ldots x(n-1)$. We call a subset of Σ^* a *language*. We call a set of languages a *class*.

For every $L \subseteq \Sigma^*$, we define the characteristic function CF_L as an infinite string as follows. Suppose $z_0, z_1, \ldots$ is the ordering of all strings over Σ^* in length lexicographic order. Then, for every $n \in \mathbb{N}$,

$$\mathrm{CF}_L(n) = \begin{cases} 1 & \text{if } z_n \in L; \\ 0 & \text{otherwise.} \end{cases}$$

The relation $x <_{lex} y$ (respectively, $x <_{ll} y$) denotes that x is lexicographically (respectively, length lexicographically) before y. When one is considering sets of strings, we take $\min S$ to denote the length lexicographically least string in S. We let $\mathrm{succ}(x)$ denote the successor of x in the length lexicographic ordering of strings (over the corresponding alphabet). We let $\mathrm{prev}(x)$ denote the predecessor of x in the length lexicographic ordering of strings (over the corresponding alphabet).

In the present work we will only consider classes of regular sets. Furthermore, Σ will always refer to the alphabet on which languages and language classes are defined.

Definition 2. An *indexing* of a class $\mathcal{L}$ is a sequence of sets L_α with $\alpha \in I$, for some domain I, such that $\mathcal{L} = \{L_\alpha : \alpha \in I\}$.

Often we will refer to both, the class and the indexing, as $\{L_\alpha : \alpha \in I\}$, where the indexing is implicit. The I above is called the set of legal indices. We will always assume that the indices in I are taken as words over an alphabet and we usually denote this alphabet with the letter Γ.

Now we consider notions related to automatic structures. First, we consider the definition of a convolution of a tuple of strings. Intuitively, a convolution transforms rows of strings into a string of columns.

Definition 3 (Khoussainov, Nerode [16]). Let $n > 0$ and $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ be alphabets not containing $\#$. Let $x_1 \in \Sigma_1^*, x_2 \in \Sigma_2^*, \ldots, x_n \in \Sigma_n^*$ be given. Let $\ell = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$ and let $y_i = x_i \#^{\ell - |x_i|}$. Define z to be a string of length ℓ such that $z(j)$ is the symbol made up of the j-th symbols of the strings $y_1, y_2, \ldots, y_n$: $z(j) = (y_1(j), y_2(j), \ldots, y_n(j))$, where $z(j) \in (\Sigma_1 \cup \{\#\}) \times (\Sigma_2 \cup \{\#\}) \times \ldots \times (\Sigma_n \cup \{\#\})$. We call z the *convolution of* $x_1, x_2, \ldots, x_n$ and denote it as $\mathrm{conv}(x_1, x_2, \ldots, x_n)$.

Definition 4. Let $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ be alphabets not containing $\#$. Let $R \subseteq \Sigma_1^* \times \Sigma_2^* \times \ldots \times \Sigma_n^*$. We call the set S, which is defined as

$$S = \{\mathrm{conv}(x_1, x_2, \ldots, x_n) \ : \ (x_1, x_2, \ldots, x_n) \in R\},$$

the *convolution of* R. Furthermore, we say that R is *automatic* if and only if S is regular.

Next we recall the Fundamental Theorem of Automatic Structures which will be useful to define automatic learners and to decide the learnability of automatic classes.

Fact 5 (Blumensath, Grädel [8], Khoussainov, Nerode [16] — Fundamental Theorem of Automatic Structures). *Any relation that is first-order definable from existing automatic relations is automatic.*

We next define the notion of an automatic indexing.

Definition 6. An indexing $\{L_\alpha : \alpha \in I\}$ is *automatic* if and only if I is regular and $E = \{(\alpha, x) : x \in L_\alpha, \alpha \in I\}$ is automatic. A class is *automatic* if and only if it has an automatic indexing.

Next, we recall a few definitions of learning, followed by a result from Angluin in [1] that characterises learnable classes. For any alphabet Σ, Γ, we let

- $\Box$ be a special character not in Σ^* which is called the *pause symbol*;
- ? be a special character not in Γ^* which is called the *no-conjecture symbol*.

Let Σ be the alphabet over which languages are being considered. We use σ, τ to denote finite sequences over $\Sigma^* \cup \{\Box\}$, and T to denote infinite sequences over $\Sigma^* \cup \{\Box\}$. Length of a sequence σ is denoted by $|\sigma|$. $T[m]$ denotes the initial segment of T of length m. $\sigma \diamond \tau$ (respectively, $\sigma \diamond T$) denotes the concatenation of σ and τ (respectively, σ and T). For a sequence σ and string x, we often use $\sigma \diamond x$ to denote the concatenation of sequence σ with the sequence of length 1 consisting of string x. For ease of notation, when it is clear from context that concatenation of sequences is meant, we sometimes drop $\diamond$ symbol. Thus, $\sigma\tau$ means $\sigma \diamond \tau$. For every finite sequence σ over $\Sigma^* \cup \{\Box\}$, we define the content of σ, denoted as $\mathrm{cnt}(\sigma)$, as

$$\mathrm{cnt}(\sigma) = \{x \in \Sigma^* \ : \ \exists\, n < |\sigma|\, (\sigma(n) = x)\} \ .$$

Similarly, for every infinite sequence T over $\Sigma^* \cup \{\Box\}$, we define the content of T, denoted as $\mathrm{cnt}(T)$, as

$$\mathrm{cnt}(T) = \{x \in \Sigma^* \ : \ \exists\, n \in \mathbb{N}\, (T(n) = x)\} \ .$$

Furthermore, λ denotes the empty sequence over $\Sigma^* \cup \{\Box\}$. For every set L and every infinite sequence T over $\Sigma^* \cup \{\Box\}$ with $L = \mathrm{cnt}(T)$, we call T a *text for* L. For every $L \subseteq \Sigma^*$, let

$$\mathrm{txt}(L) = \{T \in (\Sigma^* \cup \{\Box\})^\omega \ : \ \mathrm{cnt}(T) = L\}$$
$$\mathrm{seq}(L) = \{\sigma \in (\Sigma^* \cup \{\Box\})^* \ : \ \mathrm{cnt}(\sigma) \subseteq L\} \ .$$

Given a class $\mathcal{L}$, a *hypothesis space* for $\mathcal{L}$ is a class $\{H_\alpha : \alpha \in J\} \supseteq \mathcal{L}$ with corresponding indexing, where J is the set of indices for the hypothesis space. We will only consider automatic hypothesis spaces.

A hypothesis space is *class-preserving* iff $\mathcal{L} = \{H_\alpha : \alpha \in J\}$. A hypothesis space is *class-comprising* iff $\mathcal{L} \subseteq \{H_\alpha : \alpha \in J\}$. A hypothesis space is *one-one* iff it is class-preserving and for every $L \in \mathcal{L}$ there is exactly one $\alpha \in J$ with $L = H_\alpha$.

A *learner* is a function $\mathbf{F} : (\Sigma^* \cup \{\Box\})^* \to J \cup \{?\}$. We use $\mathbf{M}$ and $\mathbf{N}$ for recursive learners,

F for learners which may not be recursive. We use **P** for iterative learners and **Q** for iterative learners with additional long-term memory. The learners **P** and **Q** are usually automatic. The definitions of iterative and automatic learners are given in Section 3 below.

Definition 7. Fix a class $\mathcal{L}$ and a hypothesis space $\{H_\alpha : \alpha \in J\}$ with J being the set of indices. Let **F** be a learner.

(a) [11] We say that **F Ex**-*learns* $\mathcal{L}$ if and only if for every $L \in \mathcal{L}$ and every $T \in \mathrm{txt}(L)$, there exists an $n \in \mathbb{N}$ and an $\alpha \in J$ with $H_\alpha = L$ such that, for every $m \geq n$, $\mathbf{F}(T[m]) = \alpha$.

(b) [4] We say that **F BC**-*learns* $\mathcal{L}$ if and only if for every $L \in \mathcal{L}$ and every $T \in \mathrm{txt}(L)$, there exists an $n \in \mathbb{N}$ such that for every $m \geq n$, $H_{\mathbf{F}(T[m])} = L$.

(c) [9] We say that **F FEx**-*learns* $\mathcal{L}$ if and only if **F BC**-learns $\mathcal{L}$ and for every $L \in \mathcal{L}$ and every $T \in \mathrm{txt}(L)$, the set $\{\mathbf{F}(T[n]) : n \in \mathbb{N}\}$ is finite.

(d) [22] We say that **F PId**-*learns* $\mathcal{L}$ if and only if for every $L \in \mathcal{L}$, for every text T for L, there exists an $\alpha \in J$ such that (i) $L_\alpha = L$, (ii) for every $n \in \mathbb{N}$, there exists a $k \geq n$ such that $\mathbf{F}(T[k]) = \alpha$, and (iii) for every $\beta \in J$ with $\beta \neq \alpha$, there exists an $n \in \mathbb{N}$ such that for every $k \geq n$, $\mathbf{F}(T[k]) \neq \beta$.

For **Ex**, **FEx**, **BC** and **PId** learning, one can assume without loss of generality that the learner never outputs ?. However, for some other criteria of learning, this may not be the case.

Definition 8. Let Σ and Γ be alphabets. Let $\{H_\alpha : \alpha \in J\}$ be a hypothesis space with some J being the set of indices. Let **F** be a learner.

(a) [5] We say that **F** is *consistent* on L if and only if for every $\sigma \in \mathrm{seq}(L)$, if $\mathbf{F}(\sigma) \in J$, then $\mathrm{cnt}(\sigma) \subseteq H_{\mathbf{F}(\sigma)}$. We say that **F** is consistent on $\mathcal{L}$, if it is consistent on each $L \in \mathcal{L}$.

(b) [1] We say that **F** is *conservative* on L if and only if for every $\sigma, \sigma' \in \mathrm{seq}(L)$, if $\mathbf{F}(\sigma) \in J$ and $\mathrm{cnt}(\sigma \diamond \sigma') \subseteq H_{\mathbf{F}(\sigma)}$ then $\mathbf{F}(\sigma \diamond \sigma') = \mathbf{F}(\sigma)$. We say that **F** is conservative on $\mathcal{L}$, if it is conservative on each $L \in \mathcal{L}$.

(c) [21, 25] We say that **F** is *set-driven* if and only if for every $\sigma_1, \sigma_2 \in (\Sigma^* \cup \{\Box\})^*$, if $\mathrm{cnt}(\sigma_1) = \mathrm{cnt}(\sigma_2)$, then $\mathbf{F}(\sigma_1) = \mathbf{F}(\sigma_2)$.

When we are considering learning consistently (conservatively, set-drivenly) a class $\mathcal{L}$, we mean learning of the class by a learner which is consistent (conservative, set-driven) on $\mathcal{L}$.

For each learning criterion **LC** such as **Ex**, **FEx**, **BC** and **PId**, we let **LC** also denote the collection of all classes which are **LC**-learned by a recursive learner using some class comprising hypothesis space.

Blum and Blum [6] introduced the notion of a *locking sequence* for a learner **F** on a set L learnt by **F**: locking sequence for learner **F** on L is any sequence $\sigma \in \mathrm{seq}(L)$ such that, for some fixed index e of L, $\mathbf{F}(\sigma\tau) = e$ for all $\tau \in \mathrm{seq}(L)$. Blum and Blum showed that a locking sequence always exists for languages learnt by **F** and this notion can be adapted for most learning criteria considered in this paper.

Using locking sequences, Angluin characterised classes that are **Ex**-learnable (by a not necessarily recursive learner). First, let us recall the definition of a tell-tale set, while introducing the definition of a tell-tale cut-off word.

Definition 9 (Angluin's Tell-Tale Condition [1]). Suppose $\mathcal{L}$ is a class of languages.

(a) For every $L \in \mathcal{L}$, we say that D is a *tell-tale set of* L (*in* $\mathcal{L}$) if and only if D is a finite subset of L and for every $L' \in \mathcal{L}$ with $D \subseteq L' \subseteq L$ we have $L' = L$.

(b) For every $L \in \mathcal{L}$ and $x \in \Sigma^*$, we say that x is a *tell-tale cut-off word of* L (*in* $\mathcal{L}$) if and only if $\{y \in L \ : \ y \leq_{ll} x\}$ is a tell-tale set of L.

(c) We say that $\mathcal{L}$ satisfies *Angluin's Tell-Tale Condition* if and only if every $L \in \mathcal{L}$ has a tell-tale set (in $\mathcal{L}$), or equivalently, a tell-tale cut-off word (in $\mathcal{L}$).

Fact 10 (Angluin [1]). *Let Σ be an alphabet. A class $\mathcal{L}$ of recursively enumerable languages is* **Ex***-learnable (by a not necessarily recursive learner) if and only if $\mathcal{L}$ satisfies Angluin's Tell-Tale Condition.*

Note that for non recursive learners, **Ex**, **BC** and **FEx** learning are equivalent. Given a uniformly recursive class $\{L_\alpha : \alpha \in J\}$, Angluin [1] proved that the learner can be chosen to be recursive iff there is a uniformly recursively enumerable class of sets E_α such that each E_α is a tell-tale set for L_α. Note that in general such recursive learners may not be consistent, conservative or set-driven. In particular it can be shown that there are classes of languages which can be recursively learnt, but cannot be consistently, conservatively or set-drivenly learnt (see respectively [5], [1] and [21, 25]).

Now using the Fundamental Theorem for automatic structures, we show that any automatic class satisfying Angluin's Tell-Tale Condition is **Ex**-learnable and the learner can be made to be recursive, consistent, conservative and set-driven.

Theorem 11. *Suppose $\mathcal{L}$ is automatic. Then, $\mathcal{L}$ is recursively, consistently, conservatively and set-drivenly* **Ex***-learnable if and only if $\mathcal{L}$ satisfies Angluin's Tell-Tale Condition.*

Proof. The necessity follows from Fact 10. For sufficiency suppose $\mathcal{L}$ is automatic and satisfies Angluin's Tell-Tale Condition. Let L_α, $\alpha \in I$, be an indexing of $\mathcal{L}$. Now consider the learner $\mathbf{M}$ such that $\mathbf{M}(\sigma)$ is defined as follows.

- If there exists an $\alpha \in I$ such that, for some $w \in \Sigma^*$,
  - (a) $\mathrm{cnt}(\sigma) \subseteq L_\alpha$,
  - (b) $\{x : x \leq_{ll} w, x \in L_\alpha\} \subseteq \mathrm{cnt}(\sigma)$,
  - (c) for all $\beta \in I$, $[\{x : x \leq_{ll} w, x \in L_\alpha\} \subseteq L_\beta \Rightarrow \neg[L_\beta \subset L_\alpha]]$,
  - (d) for all $\beta \leq_{ll} \alpha$, $[\beta \in I \wedge \mathrm{cnt}(\sigma) \subseteq L_\beta \Rightarrow L_\alpha \subseteq L_\beta]$,
- Then $\mathbf{M}(\sigma)$ is the length-lexicographically least such α
- Else $\mathbf{M}(\sigma) = ?$.

It is easy to verify that $\mathbf{M}$ is consistent, recursive and set driven. Furthermore, if $\mathbf{M}(\sigma) = \alpha$, and $\mathrm{cnt}(\tau) \subseteq L_\alpha$, then $\mathbf{M}(\sigma \diamond \tau) = \alpha$ also (as the conditions (a)–(d) above will be satisfied for $\sigma \diamond \tau$ also), and thus $\mathbf{M}$ is conservative. Using the Angluin's Tell-Tale condition property, it also follows that $\mathbf{M}$ **Ex**-learns $\mathcal{L}$. $\square$

As the tell-tale cut-off word version of Angluin's Tell-Tale Condition is first-order definable, we have the following corollary.

Corollary 12. *It is decidable whether an automatic family* $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ *is* **Ex***-learnable, where the input given to the decision-procedure is any DFA accepting the regular language* $\{\mathrm{conv}\,(\alpha, x) : x \in L_\alpha, \alpha \in I\}$.

Remark 13. One can obtain similar characterizations for other fundamental notions of learning.

(a) Recall that a class is *finitely learnable* [11] iff there is an **Ex**-learner which on every text T of a language in the class outputs exactly one index (plus perhaps the symbol ?) such that this index is correct. For automatic classes $\mathcal{L}$, finite learnability can be characterised as follows.

$\mathcal{L}$ is finitely learnable iff for every $L \in \mathcal{L}$ there is a finite set D_L such that $D_L \subseteq L$ and $D_L \not\subseteq L'$ for all $L' \in \mathcal{L} - \{L\}$.

The implication ($\Rightarrow$) follows directly from the work of Mukouchi [20]. For ($\Leftarrow$), suppose the the right hand side holds. Let L_α, $\alpha \in I$, be an automatic indexing of $\mathcal{L}$. Now the following learner $\mathbf{M}$ finitely learns $\mathcal{L}$. $\mathbf{M}(\sigma)$ is defined as follows. If there exists an $\alpha \in I$ such that: $\mathrm{cnt}(\sigma) \subseteq L_\alpha$ and for all $\alpha' \in I$, $\mathrm{cnt}(\sigma) \subseteq L_{\alpha'}$ implies $L_\alpha = L_{\alpha'}$, then $\mathbf{M}(\sigma)$ = length lexicographically least such α; otherwise $\mathbf{M}(\sigma) =?$. It is easy to verify that $\mathbf{M}$ finitely learns $\mathcal{L}$.

(b) A class is *strong monotonically learnable* [14] iff it has an **Ex**-learner $\mathbf{M}$ such that for any two subsequent hypotheses L_i, L_j of $\mathbf{M}$ it holds that $L_i \subseteq L_j$. Given an automatic class $\mathcal{L}$, one can again characterise whether $\mathcal{L}$ is strong monotonically learnable:

$\mathcal{L}$ is strong monotonically learnable iff for all $L \in \mathcal{L}$, there exists a finite set D_L such that $D_L \subseteq L$ and for all $L' \in \mathcal{L}$ if $D_L \subseteq L'$, then $L \subseteq L'$.

Lange, Zeugmann and Kapur [19] showed the direction ($\Rightarrow$). For the direction ($\Leftarrow$), assume that the right hand side holds. Let L_α, $\alpha \in I$, be an automatic indexing of $\mathcal{L}$. Now the following learner $\mathbf{M}$ strong monotonically learns $\mathcal{L}$. $\mathbf{M}(\lambda) =?$. For $\sigma \neq \lambda$, $\mathbf{M}(\sigma)$ is defined as follows. If there exists an $\alpha \in I$ such that: $\mathrm{cnt}(\sigma) \subseteq L_\alpha$ and for all $\alpha' \in I$, $\mathrm{cnt}(\sigma) \subseteq L_{\alpha'}$ implies $L_\alpha \subseteq L_{\alpha'}$, then $\mathbf{M}(\sigma)$ = length lexicographically least such α; otherwise $\mathbf{M}(\sigma)$ repeats its old conjecture (that is, $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$, where $\sigma = \tau \diamond x$, for some $x \in \mathbb{N} \cup \{\#\}$). It is easy to verify that $\mathbf{M}$ strong monotonically learns $\mathcal{L}$.

3 Automatic Learning of Automatic Classes

It was shown above that all automatic classes that satisfy Angluin's Tell-Tale Condition, can be learnt using a recursive learner. However, there are practical limitations to recursive learners. Learners that are able to "see" all available past data are not practical. Rather, most learners in the setting of artificial intelligence are iterative, in the sense that these learners conjecture incrementally as they are fed the input inductively, one word at a time [26, 27]. Iterative learners base their new conjectures only on their previous conjecture and the new datum (in other words they do not remember their past data, except as coded in the hypothesis).

In the realm of automatic structures, it is natural to consider automatic learners, where the learning function is in some way automatic. In the case of general recursive learners, there does not seem to be any natural correspondence which will lead to an interesting model. However, for iterative learners, there is a natural corresponding definition for automatic learners where the

the update function is automatic. Below we formally define automatic iterative learners and its variant, iterative learning with long-term memory.

Definition 14 (Wexler, Culicover [26], Wiehagen [27], Kinber, Stephan [17]). Let Σ, Γ and Δ be alphabets. Let $\mathcal{L}$ be a class (defined over alphabet Σ) and $\{H_\alpha : \alpha \in J\}$ be a hypothesis space with $J \subseteq \Gamma^*$. An *iterative learner* is any function

$$\mathbf{P} : (J \cup \{?\}) \times (\Sigma^* \cup \{\square\}) \to J \cup \{?\}$$

and an *iterative learner with long-term memory* is any function

$$\mathbf{Q} : ((J \cup \{?\}) \times \Delta^*) \times (\Sigma^* \cup \{\square\}) \to (J \cup \{?\}) \times \Delta^*$$

where Δ is a suitable alphabet for memory.

Given an iterative learner $\mathbf{P}$, we may extend the definition of $\mathbf{P}$ inductively to handle sequences of words instead of a single word. Indeed, given $\alpha \in J \cup \{?\}$, $\sigma \in (\Sigma^* \cup \{\square\})^*$ and $x \in \Sigma^* \cup \{\square\}$, let $\mathbf{P}(\alpha, \lambda) = \alpha$, and $\mathbf{P}(\alpha, \sigma \diamond x) = \mathbf{P}(\mathbf{P}(\alpha, \sigma), x)$. We can further extend this to take just the sequence of words as input. To do this, we write $\mathbf{P}(\sigma)$ to mean $\mathbf{P}(?, \sigma)$. In doing so, $\mathbf{P}$ becomes a learner in the syntactic sense. Thus we say that a learner is *iterative* if the learner can be identified in this way.

Similarly, for iterative learners with long-term memory, we define the following. For every $\alpha \in J \cup \{?\}$, $\mu \in \Delta^*$, $\sigma \in (\Sigma^* \cup \{\square\})^*$ and $x \in \Sigma^* \cup \{\square\}$, let $\mathbf{Q}((\alpha, \mu), \lambda) = (\alpha, \mu)$, $\mathbf{Q}((\alpha, \mu), \sigma \diamond x) = \mathbf{Q}(\mathbf{Q}((\alpha, \mu), \sigma), x)$ and $\mathbf{Q}(\sigma) = \mathbf{Q}((?, \varepsilon), \sigma)$. Here, for $\mathbf{Q}(\sigma) = (\alpha, \mu)$, we consider α as the conjecture and μ implicitly as its memory and not as its output. Thus we say that a learner is *iterative with long-term memory* if the learner can be identified in this way.

With these modifications, $\mathbf{P}$ and $\mathbf{Q}$ are seen as learners and the definitions of all the learning criteria carry over. Note that convergence of a learner $\mathbf{Q}$ is defined only with respect to the hypothesis and not the memory. For example, $\mathbf{Q}$ **Ex**-learns L on a text T iff the sequence of hypotheses converges syntactically to a correct one while there are no convergence constraints on the memory. Similarly one defines the other learning criteria only with respect to the sequence of hypotheses. Part (b) of the following definition is based on [17].

Definition 15. Suppose $\mathcal{L}$ is defined over alphabet Σ, and $\{H_\alpha : \alpha \in J\}$ is a hypothesis space. Suppose $\mathbf{P}$ is an iterative learner and $\mathbf{Q}$ is an iterative learner with long-term memory over some alphabet Δ.

(a) We say that $\mathbf{P}$ is *automatic* if and only if $\mathbf{P}$, as a relation over

$$(J \cup \{?\}) \times (\Sigma^* \cup \{\square\}) \times (J \cup \{?\}) \ ,$$

is automatic. We say that $\mathbf{Q}$ is *automatic* if and only if $\mathbf{Q}$, as a relation over

$$((J \cup \{?\}) \times \Delta^*) \times (\Sigma^* \cup \{\square\}) \times ((J \cup \{?\}) \times \Delta^*) \ ,$$

is automatic.

(b) We say that the long-term memory of $\mathbf{Q}$ is *bounded by the longest datum seen so far* if and only if there exists a constant $c \in \mathbb{N}$ such that for every $\sigma \in (\Sigma^* \cup \{\Box\})^*$, if $\mathbf{Q}(\sigma) = (\alpha, \mu)$, then $|\mu| \leq \max\{|x| \ : \ x \in \text{cnt}(\sigma)\} + c$.

We say that the long-term memory of $\mathbf{Q}$ is *bounded by the hypothesis size* if and only if there exists a constant $c \in \mathbb{N}$ such that for every $\sigma \in (\Sigma^* \cup \{\Box\})^*$, if $\mathbf{Q}(\sigma) = (\alpha, \mu)$, then $|\mu| \leq |\alpha| + c$.

We say that the long-term memory of $\mathbf{Q}$ is *bounded by a constant* if and only if there exists a constant $c \in \mathbb{N}$ such that for every $\sigma \in (\Sigma^* \cup \{\Box\})^*$, if $\mathbf{Q}(\sigma) = (\alpha, \mu)$, then $|\mu| \leq c$.

Automatic iterative learners with long-term memory are called automatic learners from now on.

Definition 16. For the following, hypothesis space is allowed to be any class comprising automatic family. Let **LC** be one of **Ex**, **BC**, **FEx** and **PId**. We let

(a) **AutoLC** be the set of all classes of languages that is **LC**-learned by some automatic learner with arbitrary long-term memory,

(b) **AutoWordLC** be the set of all classes of languages that is **LC**-learned by some automatic learner with long-term memory that is bounded by the longest datum seen so far,

(c) **AutoIndexLC** be the set of all classes of languages that is **LC**-learned by some automatic learner with long-term memory that is bounded by the hypothesis size,

(d) **AutoConstLC** be the set of all classes of languages that is **LC**-learned by some automatic learner with long-term memory that is bounded by a constant, and

(e) **AutoItLC** be the set of all classes of languages that is **LC**-learned by some automatic iterative learner.

We first show that automatic learners are not as powerful as general learners, even for learning automatic classes.

Theorem 17. *There exists an automatic $\mathcal{L}$ that is* **Ex** *learnable by some recursive iterative learner, but which is not* **AutoEx***-learnable.*

Proof. Any class of finite sets is easily seen to be learnable by a recursive iterative learner. However, the class $\mathcal{L}$ given by the indexing $L_\alpha = \{x : |x| = |\alpha|, x \neq \alpha\}$, $\alpha \in \{0,1\}^*$, is an automatic class but not in **AutoEx**. To see this, suppose $\mathbf{Q}$ **AutoEx** learns $\mathcal{L}$. Then, for large enough m, on some input σ, σ' where σ and σ' both consist of m elements of length m each (but $\text{cnt}(\sigma) \neq \text{cnt}(\sigma')$), we have that $\mathbf{Q}(\sigma) = \mathbf{Q}(\sigma')$ (as the size of the hypothesis and memory of $\mathbf{Q}$ after seeing such σ can be of length at most cm, for some constant c, though the number of possible sequences, with distinct content, containing m elements of length m is $\binom{2^m}{m}$). Let y, y' respectively be in $\text{cnt}(\sigma) - \text{cnt}(\sigma')$ and $\text{cnt}(\sigma') - \text{cnt}(\sigma)$. Let T be a text for $\{z : |z| = |y|, z \neq y, z \neq y'\}$. Then, $\mathbf{Q}$ on σT and $\sigma' T$ converges to the same index (or diverges on both). Thus, $\mathbf{Q}$ does not **AutoEx** learn $\mathcal{L}$. $\Box$

We now show the relationship between various long-term memory limitations for the three main criteria of learning: **Ex**, **BC** and **FEx**. Interestingly, if the memory is not explicitly constrained, then every automatic class which is **BC**-learnable can be **Ex**-learnt. For **BC**-learning, long-term

memory is not useful (for automatic learners), as such memory can be coded into the hypothesis itself, as long as one is allowed padding of the hypothesis.

Theorem 18. *The following equivalences and implications hold.*
(a) **AutoBC** = **AutoWordBC** = **AutoIndexBC** = **AutoConstBC** = **AutoItBC**.
(b) **AutoEx** = **AutoFEx** = **AutoBC**.
(c) **AutoIndexFEx** = **AutoConstFEx** = **AutoItFEx**.
(d) **AutoWordEx** ⊆ **AutoWordFEx**.
(e) **AutoIndexEx** = **AutoIndexFEx**.
(f) **AutoConstEx** = **AutoItEx**.

Proof. For the simulations below, we assume without loss of generality that the simulated learner does not output ?.

(a) It follows from the definition that **AutoItBC** ⊆ **AutoConstBC** ⊆ **AutoWordBC** ⊆ **AutoBC** and **AutoItBC** ⊆ **AutoIndexBC** ⊆ **AutoBC**. Thus it suffices to show that **AutoBC** ⊆ **AutoItBC**. Suppose **Q** **AutoBC**-learns $\mathcal{L}$, where the hypothesis space is H_α, $\alpha \in I$, and memory is over the alphabet Δ. Let $H'_{\alpha,\mu} = H_\alpha$, for $\alpha \in I, \mu \in \Delta^*$. If $\mathbf{Q}((\alpha,\mu),x) = (\alpha',\mu')$, then let $\mathbf{P}((\alpha,\mu),x) = (\alpha',\mu')$. It can easily be verified that **P** **AutoItBC** learns $\mathcal{L}$ using hypothesis space $H'_{\alpha,\mu}$, $\alpha \in I, \mu \in \Delta^*$.

(b) It suffices to show that **AutoBC** ⊆ **AutoEx**. Suppose **Q** **AutoBC**-learns $\mathcal{L}$, where the hypothesis space is H_α, $\alpha \in I \subseteq \Gamma^*$, and memory is over the alphabet Δ. Then consider the following $\mathbf{Q}'$. $\mathbf{Q}'$ uses the same hypothesis space H_α, but the memory is an element of $\Gamma^* \times \Delta^*$.

Suppose $\mathbf{Q}((\beta,\mu),x) = (\beta',\mu')$. Then $\mathbf{Q}'((\alpha,(\beta,\mu)),x) = (\alpha',(\beta',\mu'))$, where α' is the length lexicographically least member of I such that $L_{\alpha'} = L_{\beta'}$. It is easy to verify that above $\mathbf{Q}'$ **AutoEx**-learns $\mathcal{L}$.

(c) It suffices to show **AutoIndexFEx** ⊆ **AutoItFEx**. Suppose **Q** **AutoIndexFEx**-learns $\mathcal{L}$. Then the construction of part (a) witnesses that **P** **AutoItFEx**-learns $\mathcal{L}$, as the number of (α,μ) which are conjectured by Q will be finite in this case.

(d) This follows from definition.

(e) It suffices to show that **AutoIndexFEx** ⊆ **AutoIndexEx**. This can be proved similarly to part (b), except that instead of choosing the length lexicographically least equivalent index, one uses the padded version of the length lexicographically least equivalent index to make sure that the padded index is of length at least the length of the largest hypothesis output by Q so far (this is to make sure that the memory length is bounded by the size of the hypothesis plus a constant).

(f) It suffices to show that **AutoConstEx** ⊆ **AutoItEx**. Suppose **Q** **AutoConstEx**-learns $\mathcal{L}$ using hypothesis space H_α, $\alpha \in I$ and constant memory over alphabet Δ. Without loss of generality assume that memory size is always 1. Define $H'_{\alpha,w,S} = H_\alpha$, where $w \in \Delta$, $S \subseteq \Delta \times \Delta$ (we view S as a relation; note that as Δ is finite, one could consider S also as a member of the power set of $\Delta \times \Delta$).

Define $\mathbf{Q}'$, using hypothesis space given by H' above, as follows. Suppose $\mathbf{Q}((\alpha,w),x) = (\beta,y)$. If $\alpha \neq \beta$, then $\mathbf{Q}'((\alpha,w,S),x) = (\beta,y,\emptyset)$, else if $\alpha = \beta$ and (y,w) is in the transitive

closure of S viewed as a relation, then $\mathbf{Q}'((\alpha, w, S), x) = (\alpha, w, S \cup \{(w, y)\})$, else $\mathbf{Q}'((\alpha, w, S), x) = (\alpha, y, S \cup \{(w, y)\})$.

Note that, for any σ, if $\mathbf{Q}'(\sigma) = (\alpha, w, S)$, then for all $(y, y') \in S$, there exists an $x \in \text{cnt}(\sigma) \cup \{\Box\}$ such that $\mathbf{Q}((\alpha, y), x) = (\alpha, y')$. Thus, if (y, w) is in the transitive closure of S, then there exists a sequence τ, $\text{cnt}(\tau) \subseteq \text{cnt}(\sigma)$ such that $\mathbf{Q}((\alpha, y), \tau) = (\alpha, w)$. In other words, for every σ, there is a σ', which is obtained by replacing each symbol x in the sequence σ by a sequence $x \diamond \tau_x$ such that, if $\mathbf{Q}'(x_0 \diamond x_1 \diamond \ldots \diamond x_n) = (\alpha, w, S)$, then $\mathbf{Q}(x_0 \diamond \tau_{x_0} \diamond x_1 \diamond \tau_{x_1} \diamond \ldots x_n \diamond \tau_{x_n}) = (\alpha, w)$. Now fix a text T for $L \in \mathcal{L}$. Suppose $\mathbf{Q}'(T[n]) = (\alpha_n, w_n, S_n)$. Then, there exists a n_0 and an index α with $H_\alpha = L$ such that for all $n \geq n_0$, $\alpha_n = \alpha$ (by the analysis above, as for some modified text T' for L, the hypotheses of Q must converge to an index α for L). Further note that, if $\mathbf{Q}'((\alpha, w, S), x) = (\alpha, w', S')$, then $S \subseteq S'$, and (w, w') is in transitive closure of S'. It follows that $\lim_{n\to\infty} S_n$ converges, and $\lim_{n\to\infty} w_n$ converges, as all but finitely many w_n belong to the same equivalence class (with respect to the relation defined by $S = \lim_{n\to\infty} S_n$). It follows that $\mathbf{Q}'$ **Ex**-learns $\mathcal{L}$. $\Box$

Note that the above theorem (along with its proof) also holds if we require class preserving learning in all the cases.

The next theorem shows that, for **Ex** and **FEx** learning, there are classes which can be learnt by automatic learners having long-term memory bounded by longest word size seen so far while they cannot be learnt by automatic learners having long-term memory bounded by hypothesis size. (Note that **AutoIndexEx** = **AutoIndexFEx**, by Theorem 18).

The following theorem holds even when one considers class preserving hypothesis spaces (the diagonalization in part (c) can be done by using the diagonalizing class $\mathcal{L}$ itself as hypothesis space on the positive side and class comprising hypothesis space on the negative side).

Theorem 19. (a) **AutoItEx** $\subseteq$ **AutoWordEx** $\subseteq$ **AutoEx**.
(b) **AutoItEx** $\subseteq$ **AutoIndexEx** $\subseteq$ **AutoEx**.
(c) **AutoWordEx** $\not\subseteq$ **AutoIndexEx**.

Proof. The statements (a) and (b) follow from the definition.

For statement (c), consider the class $\mathcal{L} = \{L_\alpha : \alpha \in \{0,1\}^*\}$ with $L_\epsilon = 0^+$ and $L_\alpha = \{0^{i+1} : \alpha(i) = 1\} \cup \{\epsilon\}$ for all $\alpha \in \{0,1\}^+$.

To **AutoWordEx** learn $\mathcal{L}$, one uses memory over the alphabet $\{0,1\}^*$ and memorizes all strings in L_ϵ seen so far. Memory of the learner (on any input σ) is a word $z = z(0)z(1)\ldots z(n)$ such that $z(i) = 1$ iff $0^{i+1} \in \text{cnt}(\sigma)$, where n is the largest i such that $0^{i+1} \in \text{cnt}(\sigma)$ (here if σ does not contain any member of 0^+, then we take $z = 0$). Now the learner outputs index ϵ (with memory z as computed above) as long as it has not seen ϵ. Once it has seen ϵ, it outputs z as its conjecture (and has z also as its memory). It is easy to verify that the above learner witnesses that $\mathcal{L} \in$ **AutoWordEx**.

On the other hand suppose by way of contradiction that $\mathbf{Q}$ **AutoIndexEx**-identifies $\mathcal{L}$. Then, let σ be such that (i) $\text{cnt}(\sigma) \subseteq L_\epsilon$, and (ii) for all $\sigma' \supseteq \sigma$ such that $\text{cnt}(\sigma') \subseteq L_\epsilon$, if $\mathbf{Q}(\sigma') = (\alpha, \mu)$ and $\mathbf{Q}(\sigma) = (\alpha', \mu')$, then $\alpha = \alpha'$ (such a σ is called the locking sequence for $\mathbf{Q}$ on L_ϵ). Note that there exists such a sequence σ, as $\mathbf{Q}$ **Ex**-identifies $\mathcal{L}$. Now there exist τ, τ'

with $\mathrm{cnt}(\tau) \cup \mathrm{cnt}(\tau') \subseteq L_\epsilon$ such that $\mathrm{cnt}(\sigma\tau) \neq \mathrm{cnt}(\sigma\tau')$, and $\mathbf{Q}(\sigma\tau) = \mathbf{Q}(\sigma\tau')$ (this is so as the memory of $\mathbf{Q}$ has only finitely many possibilities, even though $\mathrm{cnt}(\sigma\tau)$ takes infinitely many possibilities).

Let $T_1 = \sigma\tau \diamond \epsilon^\infty$, and $T_2 = \sigma\tau' \diamond \epsilon^\infty$. It follows that $\mathbf{Q}$ would fail to **AutoIndexEx**-identify at least one of $\mathrm{cnt}(T_1)$ and $\mathrm{cnt}(T_2)$ respectively from the texts T_1 and T_2. $\square$

Note that the class $\mathcal{L}$ used in Theorem 19(c) is also not iteratively learnable by a recursive learner (essentially the same proof as used above shows this).

The following lists some of the open problems for automatic learners.

Open Problem 20. *The following problems are currently open:*
(a) *Is* **AutoEx** = **AutoWordEx***?*
(b) *Is* **AutoIndexEx** $\subseteq$ **AutoWordEx***?*
(c) *Is* **AutoIndexEx** $\subseteq$ **AutoItEx***?*
(d) *Is* **AutoWordFEx** = **AutoWordEx***?*

If the alphabet is unary, then every **AutoEx**-learner can be replaced by an **AutoWordEx**-learner which answers (a), (b) and (d) above in the affirmative for this special case. Note that the separation in Theorem 19 (c) is witnessed by a unary family and therefore carries over to this special case.

Theorem 21. *Suppose that* $\Sigma = \{0\}$ *and* $\mathcal{L} \subseteq \mathrm{powerset}(\Sigma^*)$ *is an automatic class. Then* $\mathcal{L}$ *is in* **AutoWordEx** *as witnessed by a conservative, consistent and set-driven learner if and only if* $\mathcal{L}$ *satisfies the Angluin's Tell-Tale Condition.*

Proof. The necessity follows from Fact 10. Suppose $\mathcal{L}$ is $\{L_\alpha : \alpha \in I\}$, where I is the set of indices. We show sufficiency below. We can code into memory (which uses alphabet $\{0,1\}$) all the strings seen so far by letting the memory z (after having seen input σ) be $z(0)z(1)\ldots z(n)$, where $z(i) = 1$ iff $0^i \in \mathrm{cnt}(\sigma)$. Here n would be the length of the largest string seen so far in the input text plus 1 (if no strings are seen, then we take $n = 0$). Thus, on any input σ, the learner can search for an α such that, for some $w \in \Sigma^*$,

(a) $\mathrm{cnt}(\sigma) \subseteq L_\alpha$,
(b) $\{x : x \leq_{ll} w, x \in L_\alpha\} \subseteq \mathrm{cnt}(\sigma)$,
(c) for all $\beta \in I$, $[\{x : x \leq_{ll} w, x \in L_\alpha\} \subseteq L_\beta \Rightarrow \neg[L_\beta \subset L_\alpha]]$,
(d) for all $\beta \leq_{ll} \alpha$, $[\beta \in I \wedge \mathrm{cnt}(\sigma) \subseteq L_\beta \Rightarrow L_\alpha \subseteq L_\beta]$,

It can then output length lexicographically least such α, if any. Note that the above learner is automatic, as $\mathrm{cnt}(\sigma)$ can be obtained using the memory and the new input element. The theorem follows. $\square$

Hence, for language classes over a unary alphabet, the learning criteria **AutoWordEx** and **AutoEx** coincide and properly contain the learning criterion **AutoIndexEx**.

Remark 22. In the case that one would not consider an automatic class but just a subclass $\mathcal{L}$ of an automatic class $\mathcal{K}$, one could solve some of the open problems mentioned above.

For example, there is a class $\mathcal{L} \subseteq \text{powerset}(\{0\}^*)$ which is a subclass of an automatic class and which has an automatic but neither a conservative nor a set-driven learner. Furthermore, there is no learnable automatic class $\mathcal{H}$ with $\mathcal{L} \subseteq \mathcal{H}$. Also, no automatic learner of this class can be a **AutoWordEx**-learner.

Here is a proof-sketch for this fact. Let $k(0), k(1), \ldots$ be a recursive one-one enumeration of K, the halting problem. The class consists of all sets $\{0^m : m \geq n\}$ for all n and all sets $L_{n,r} = \{0^m : n \leq m \leq n + r\}$ for which there exists a number $s > r$ with $k(s) = n$. Note that the set $L_{n,r}$ is added to the class iff $n \in K - \{k(0), k(1), \ldots, k(r)\}$.

It can be shown that $\mathcal{L}$ is automatically learnable using an automatic class comprising hypothesis space, given by $H_{n,0} = L_n$, and $H_{n,r+1} = L_{n,r}$. It can also be shown that the class is neither conservatively nor set-driven learnable nor **AutoWordEx**-learnable.

Furthermore, assume by way of contradiction that a learnable automatic class $\mathcal{H} \supseteq \mathcal{L}$ exists. Then no infinite set in $\mathcal{H}$ is the ascending union of finite sets in $\mathcal{H}$ (see [11]), hence there exists, for every n, a number $h(n) \geq n$ such that $\{0^m : n \leq m \leq h(n)\} \notin \mathcal{H}$. As $0^n \mapsto 0^{h(n)}$ is first-order definable from $\mathcal{H}$, h is recursive and $n \in K$ iff $n \in \{k(0), k(1), \ldots, k(h(n) - n)\}$, a contradiction.

4 Consistent Learning

Note that for general recursive learners, all learnable automatic classes have a consistent, conservative and set-driven recursive learner, see Theorem 11 above. Thus, on one hand, consistency, conservativeness and set-drivenness are not restrictive for learning automatic classes by recursive learners. On the other hand, in this section, we will show that consistency is a restriction when learning automatic classes by automatic learners. It will be interesting to explore similar results for conservativeness and set-drivenness.

The following theorem gives an automatic class which can be **Ex**-learnt by an iterative automatic learner but which cannot be **Ex**-learnt by any consistent automatic learner.

Theorem 23. *There exists an automatic $\mathcal{L}$ such that*

- *$\mathcal{L}$ is* **AutoItEx** *learnable using a class preserving hypothesis space.*
- *$\mathcal{L}$ is not consistently* **AutoEx** *learnable even using a class comprising hypothesis space.*

Proof. Let $\Sigma = \{0, 1, 2\}$. Let $\mathcal{L} = \{L_y : y \in \{0, 1\}^* \cup \{2\}\}$ where

- $L_\epsilon = \{0, 1\}^*$;
- $L_y = \{2^{|y|}\} \cup \{x \in \{0, 1\}^* : y \text{ is not a prefix of } x\}$ for $y \in \{0, 1\}^+$;
- $L_2 = \{0, 1, 2\}^*$.

We first show that $\mathcal{L}$ can be **AutoItEx**-learnt. We use the following hypothesis space: $H_\epsilon = L_\epsilon$, $H_2 = \{0, 1, 2\}^*$, and, for $y, z \in \{0, 1\}^+$ with $|y| = |z|$ and $y \leq_{ll} z$, $H_{y,z} = \{0, 1, 2\}^*$ and $H_{y,2} = L_y$.

We now define the iterative learner $\mathbf{P}$. If the learner ever sees the input 02 then it outputs H_2 and then never changes its mind. Besides the above case, the learner starts with a conjecture

for H_ϵ. If it ever sees 2^i, for some $i > 0$, in the input, then it continues with conjectures of form $H_{y,z}$, where $|y| = |z| = i$, and initially, $y = z = 0^i$ — conjecture of the form $H_{y,z}$ means that the learner has seen extensions (in $\{0,1\}^*$) for all $y' \leq_{ll} z$, with $y' \neq y$ and $|y'| = i$. If the learner later sees an extension of y, then it updates both y, z to succ(z) (here and in rest of the proof, succ and length lexicographic ordering is over the alphabet $\{0,1\}$). If the learner sees an extension of succ(z), then it will update z to succ(z). This continues, until the learner has seen extensions of all strings of length i, except for the one currently denoted by y. At this point, the learner can conclude that the input language must be L_y (unless it sees 02 in the input). Formally,

- $\mathbf{P}(\lambda) = \epsilon$.
- $\mathbf{P}(\epsilon, 02) = \mathbf{P}((y,2), 02) = \mathbf{P}((y,z), 02) = \mathbf{P}(2, w) = 2$, for all $w \in \{0,1,2\}^*$, $y, z \in \{0,1\}^+$, $|y| = |z|$.
- $\mathbf{P}(\epsilon, w) = \epsilon$, if $w \notin \{2^i : i > 0\} \cup \{02\}$.
- For $i > 0$, $\mathbf{P}(\epsilon, 2^i) = (0^i, 0^i)$.
- For $y, z \in \{0,1\}^+, |y| = |z|$, $\mathbf{P}((y,z), w) = (\text{succ}(z), \text{succ}(z))$, if $w \in \{0,1\}^*$ and w is an extension of y, and succ(z) is not the length lexicographically maximal string of length $|z|$.
- For $y, z \in \{0,1\}^+, |y| = |z|$, $\mathbf{P}((y,z), w) = (\text{succ}(z), 2)$, if $w \in \{0,1\}^*$ and w is an extension of y, and succ(z) is the length lexicographically maximal string of length $|z|$.
- For $y, z \in \{0,1\}^+, |y| = |z|$, $\mathbf{P}((y,z), w) = (y, \text{succ}(z))$, if $w \in \{0,1\}^*$, w is an extension of succ(z) and succ(z) is not the length lexicographically maximal string of length $|z|$.
- For $y, z \in \{0,1\}^+, |y| = |z|$, $\mathbf{P}((y,z), w) = (y, 2)$, if $w \in \{0,1\}^*$ and w is an extension of succ(z), and succ(z) is the length lexicographically maximal string of length $|z|$.
- For $y, z \in \{0,1\}^+, |y| = |z|$, $\mathbf{P}((y,z), w) = (y, z)$, if $w \neq 02$ and ($w \notin \{0,1\}^*$ or w is not an extension of either y or succ(z)).
- $\mathbf{P}((y,2), w) = (y, 2)$, for $w \neq 02$.

It is easy to verify that the above **P AutoItEx**-identifies $\mathcal{L}$.

To see that $\mathcal{L}$ is not consistently **AutoEx**-learnable, suppose by way of contradiction otherwise as witnessed by $\mathbf{Q}$ using hypothesis space $\{H_\alpha : \alpha \in J\}$.

Consider a locking sequence σ (conjecture wise) for $\mathbf{Q}$ on L_ϵ (that is, for some α such that $H_\alpha = L_\epsilon$: $\sigma \in \text{seq}(L_\epsilon)$, $\mathbf{Q}(\sigma) = (\alpha, \mu)$, and for all $\tau \in \text{seq}(L_\epsilon)$, $\mathbf{Q}(\sigma \diamond \tau) = (\alpha', \mu')$ implies $\alpha = \alpha'$). Let τ', τ'' and $m >$ maximum length string in cnt(σ) be such that (i) each of τ' and τ'' is of length m and contains m distinct strings from $\{0,1\}^m$, (ii) cnt$(\tau') \neq$ cnt(τ'') and (iii) $\mathbf{Q}(\sigma\tau') = \mathbf{Q}(\sigma\tau'')$. Note that there exist such τ', τ'' for large enough m as there are $\binom{2^m}{m}$ possibilities for the sequences of length m from $\{0,1\}^m$ with distinct content, but only c^{m+s} possibilities for $\mathbf{Q}(\sigma\tau'''')$, for some constant c where τ'''' is a sequence of length m from $\{0,1\}^m$ and s denotes the maximum of the size of the conjecture/memory of $\mathbf{Q}(\sigma)$. Suppose $y, y' \in \{0,1\}^m$ are such that $y \in$ cnt$(\tau') -$ cnt(τ'') and $y' \in$ cnt$(\tau'') -$ cnt(τ'). Let T be a text for L_y. Then, $\mathbf{Q}(\sigma\tau''T)$ must converge to an index for L_y. Let σ''' be a prefix of T such that $\mathbf{Q}(\sigma\tau''\sigma''') = (\alpha, \mu)$ where $H_\alpha = L_y$. But, then $\mathbf{Q}$ is not consistent on the text $\sigma\tau'\sigma'''T'$, where T' is a text for L_2. □

The following theorem gives an automatic class $\mathcal{L}$ which can be **Ex**-learnt by a consistent automatic learner or **Ex**-learnt by an iterative automatic learner but which cannot be **Ex**-learnt

by a consistent iterative automatic learner. Thus, requiring both consistency and iterativeness is more of a restriction compared to requiring only one of them.

Theorem 24. *There exists an automatic $\mathcal{L}$ such that*

- *$\mathcal{L}$ is consistently* **AutoEx** *learnable using $\mathcal{L}$ as hypothesis space;*
- *$\mathcal{L}$ is* **AutoItEx** *learnable using $\mathcal{L}$ as hypothesis space;*
- *$\mathcal{L}$ is not consistently* **AutoItEx** *learnable.*

Proof. Let $\Sigma = \{0,1,2\}$. Let $\mathcal{L} = \{L_\alpha : \alpha \in \{\epsilon\} \cup \{0,1\}^*1\}$ where $L_\epsilon = \{0,1\}^*$ and $L_\alpha = \{2\} \cup \{x : x \leq_{lex} \alpha 0^\infty\}$ for $\alpha \in \{0,1\}^*1$.

It is easy to verify that $\mathcal{L}$ is **AutoItEx** learnable, as one can initially output ϵ as the conjecture, and once 2 appears in the input, search for lexicographically largest α ending in a 1 such that some extension of α is in the remaining text (such extensions will appear infinitely often, for each α which has an extension in the input language).

To see that $\mathcal{L}$ is consistently **AutoEx**-learnable, note that one can memorize the lexicographically largest possible α ending in a 1 which is a prefix of some input string. Thus, we could essentially use the above algorithm to consistently **AutoEx**-learn $\mathcal{L}$.

To see that $\mathcal{L}$ cannot be consistently **AutoItEx**-learned, suppose by way of contradiction that $\mathbf{P}$ witnesses such learning.

Let σ be a locking sequence for $\mathbf{P}$ on L_ϵ. Then, let α be the lexicographically largest string ending in 1 which is a prefix of some string in σ (if there is no such string, then we take α to be 1). Then, $\mathbf{P}$ on text $\sigma \diamond T$, where T is a text for L_α, must converge to an index for L_α. Thus, $\mathbf{P}(\sigma\tau) =$ index for L_α, for some $\tau \in \text{seq}(L_\alpha)$. But, then $\mathbf{P}$ is not consistent on $\sigma \diamond \alpha 1 \diamond \tau$, as $\alpha 1 \notin L_\alpha$. $\square$

Remark 25. Note that the class from Theorem 24 is also not set-driven iteratively learnable: given an iterative learner $\mathbf{P}$, let σ be a locking sequence for L_ϵ and $\alpha = \mathbf{P}(\sigma \diamond 2)$. There is now a sequence τ of strings in L_ϵ such that $\mathbf{P}(\sigma \diamond 2 \diamond \tau)$ outputs some index for a proper superset of L_α. But from the locking sequence property of σ, it then follows that $\mathbf{P}(\sigma \diamond \tau \diamond 2) = \alpha$ and $\mathbf{P}$ is not set-driven.

One can extend this result and also show that an **AutoIndexEx**-learner $\mathbf{Q}$ of this class cannot be set-driven. The long-term memory of such a learner after having seen σ is bounded by a constant plus the hypothesis size and there are only finitely many different values which the long term memory can take after input of the form $\sigma \diamond \tau$, with τ being a sequence of data from L_ϵ. But there are infinitely many languages in $\mathcal{L}$ which contain $\text{cnt}(\sigma \diamond 2)$. Hence there are two sequences τ, τ' over L_ϵ such that $\mathbf{Q}(\sigma \diamond 2 \diamond \tau)$ and $\mathbf{Q}(\sigma \diamond 2 \diamond \tau')$ output different conjectures while the long term memory after $\sigma \diamond \tau$ and $\sigma \diamond \tau'$ is the same. It follows that the hypotheses issued by $\mathbf{Q}(\sigma \diamond \tau \diamond 2)$ and $\mathbf{Q}(\sigma \diamond \tau' \diamond 2)$ are the same while those issued by $\mathbf{Q}(\sigma \diamond 2 \diamond \tau)$ and $\mathbf{Q}(\sigma \diamond 2 \diamond \tau')$ are different; hence $\mathbf{Q}$ is not set-driven.

The following theorem shows the existence of an automatic class which can be **Ex**-learnt by a consistent automatic iterative learner using a class comprising hypothesis space, but cannot be **Ex**-learnt by a consistent automatic learner using a class preserving hypothesis space. Thus, in some cases having a larger hypothesis space can make consistency problem easier to handle.

Theorem 26. *There exists an automatic $\mathcal{L}$ such that*

- *the class $\mathcal{L}$ is* **AutoItEx***-learnable using a class preserving hypothesis space;*
- *the class $\mathcal{L}$ is consistently* **AutoItEx***-learnable using some class-comprising hypothesis space for $\mathcal{L}$;*
- *the class $\mathcal{L}$ is not consistently* **AutoEx***-learnable using any class-preserving hypothesis space for $\mathcal{L}$.*

Proof. Let $\Sigma = \{0,1,2\}$. Let $\mathcal{L} = \{L_\epsilon\} \cup \{L_y : y \in \{0,1\}^+\}$ where

- $L_\epsilon = \{0,1\}^*$;
- $L_y = \{2^{|y|}\} \cup \{x \in \{0,1\}^* : y \text{ is not a prefix of } x\}$ for all $y \in \{0,1\}^+$.

It is easy to verify that the learner $\mathbf{P}$ given in the proof of Theorem 23 **AutoItEx**-identifies $\mathcal{L}$ (using a class comprising hypothesis space). This learner is consistent on $\mathcal{L}$.

To see **AutoItEx** learnability using class preserving hypothesis space, one can use the learner $\mathbf{P}$ in the proof of Theorem 23, but for $y, z \in \{0,1\}^*$, $|y| = |z|$, we define H_2 and $H_{y,z}$ to be L_ϵ instead of $\{0,1,2\}^*$ (in particular, we do not need to use H_2).

To show that no learner using class preserving hypothesis space can consistently **AutoEx**-learn $\mathcal{L}$ we proceed as follows. Suppose by way of contradiction that $\mathbf{Q}$ consistently **AutoEx**-learns $\mathcal{L}$ using a class preserving hypothesis space H_α, $\alpha \in J$.

Consider the locking sequence σ (conjecture wise) for $\mathbf{Q}$ on L_ϵ (that is, for some α such that $H_\alpha = L_\epsilon$: $\sigma \in \text{seq}(L_\epsilon)$, $\mathbf{Q}(\sigma) = (\alpha, \mu)$, and for all $\tau \in \text{seq}(L_\epsilon)$, $\mathbf{Q}(\sigma \diamond \tau) = (\alpha', \mu')$ implies $\alpha = \alpha'$).

Let τ', τ'' and $m >$ maximum length string in $\text{cnt}(\sigma)$ be such that (i) each of τ', τ'' is of length m and contains m distinct strings from $\{0,1\}^m$, (ii) $\text{cnt}(\tau') \neq \text{cnt}(\tau'')$ and (iii) $\mathbf{Q}(\sigma\tau') = \mathbf{Q}(\sigma\tau'')$. Note that there exist such τ', τ'' for large enough m as there are $\binom{2^m}{m}$ possibilities for the sequences of length m from $\{0,1\}^m$ with distinct content, but only c^{m+s} possibilities for $\mathbf{Q}(\sigma\tau'''')$, for some constant c where τ'''' is a sequence of length m from $\{0,1\}^m$ and s denotes the maximum of the size of the conjecture/memory of $\mathbf{Q}(\sigma)$. Suppose $y, y' \in \{0,1\}^m$ are such that $y \in \text{cnt}(\tau') - \text{cnt}(\tau'')$ and $y' \in \text{cnt}(\tau'') - \text{cnt}(\tau')$. Let τ be a sequence which contains all elements of length m, except for y and y'. Then, $\mathbf{Q}(\sigma\tau'\tau \diamond 2^m) = \mathbf{Q}(\sigma\tau''\tau \diamond 2^m)$, but $\mathbf{Q}$ cannot be consistent on both $\sigma\tau'\tau \diamond 2^m$ and $\sigma\tau''\tau \diamond 2^m$. □

5 Automatic Learning from Fat Text

One of the reasons why iterative learning and its variations are restrictive is because the learners forget past data. So it is interesting to study the case when each datum appears infinitely often (such a text is called fat text). In the case of learning recursively enumerable sets, it has been shown that every explanatorily learnable class is also iteratively learnable from fat text [22]. In the following, it is investigated to which extent this result transfers to automatic learners.

Definition 27. [22] Let Σ be an alphabet. Let $T \in (\Sigma^* \cup \{\Box\})^\omega$. We say that T is *fat* if and only if for every $x \in \text{cnt}(T)$ and $n \in \mathbb{N}$, there exists a $k \geq n$ such that $T(k) = x$.

For every alphabet Σ and every $L \subseteq \Sigma^*$, we will let $\text{ftxt}(L) = \{T \in \text{txt}(L) : T \text{ is fat}\}$.

Definition 28. Let Σ be an alphabet. Let $\{H_\alpha : \alpha \in J\}$ be a hypothesis space with some J being the set of indices. Let $\mathbf{P}$ be an iterative learner. We say that $\mathbf{P}$ **Ex**-*learns* $\mathcal{L}$ *from fat text* if and only if for every $L \in \mathcal{L}$ and every $T \in \text{ftxt}(L)$, there exists an $n \in \mathbb{N}$ and an $\alpha \in J$ with $H_\alpha = L$ such that, for every $m \geq n$, $\mathbf{P}(T[m]) = \alpha$. Similarly one can define other criteria of learning from fat text.

Corollary 31 to the proof of the following theorem shows that fat texts allows one to iteratively automatically learn any class which is potentially learnable (that is, satisfies Angluin's Tell-Tale Condition).

Theorem 29. *Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ be an automatic class. Then $\mathcal{L}$ is* **AutoWordEx**-*learnable from fat text using the given hypothesis space $\{L_\alpha : \alpha \in I\}$ if and only if $\mathcal{L}$ satisfies Angluin's Tell-Tale Condition.*

Proof. The necessity follows from Fact 10. We will now show sufficiency.

Let I be the set of indices. We will assume below that α, β range over I.

We will construct the learner $\mathbf{Q}$ below. Without loss of generality we assume that if $\emptyset \in \mathcal{L}$, then $L_\epsilon = \emptyset$.

We let $L_\alpha[z]$ denote the set $\{y : y \leq_{ll} z\}$.

We will denote the conjecture/memory of the learner $\mathbf{Q}$ by $(\alpha, x, cons)$, where α is the conjecture, and $(x, cons)$ is the memory. Here $x \in \Sigma^*$ and $cons$ is just a consistency bit.

Suppose T is the input fat text for a language $L \in \mathcal{L}$. Then we will have the following four invariants, whenever α, α' below are not ?:

(I) If $\mathbf{Q}(T[m]) = (\alpha, x, cons)$, then $L_\alpha[x] \subseteq \text{cnt}(T[m])$. Furthermore, for any $m' < m$, if $\mathbf{Q}(T[m']) = (\alpha', x', cons')$, then $L_\alpha[x] \subseteq L_{\alpha'}[x'] \cup \{T(m'), T(m'+1), \ldots, T(m-1)\}$.

(II) If $\mathbf{Q}(T[m]) = (\alpha, x, cons)$ and $\mathbf{Q}(T[m+m']) = (\alpha', x', cons')$ then $\text{CF}_{L_\alpha[x]} \leq_{lex} \text{CF}_{L_{\alpha'}[x']} \leq_{lex} \text{CF}_L$.

(III) If $\mathbf{Q}(T[m]) = (\alpha, x, cons)$, then either $L_\alpha[x] = L_\alpha$, and no $\beta <_{ll} \alpha$ satisfies $L_\alpha[x] = L_\beta$ or $L_\alpha[x] \notin \mathcal{L}$ and no $\beta <_{ll} \alpha$ satisfies $L_\alpha[x] = L_\beta[x]$.

(IV) If $cons = 0$, then $L \not\subseteq L_\alpha$.

If L_α is infinite, then let $\text{ttcow}(\alpha)$ denote the length lexicographically least word w in L_α such that α is a tell tale cut off word for L_α and for all $\beta <_{ll} \alpha$ such that $L_\alpha \neq L_\beta$, $L_\beta[w] \neq L_\alpha[w]$. If L_α is finite, then let $\text{ttcow}(\alpha)$ denote $\max L_\alpha$. Note that $\text{ttcow}(\alpha)$ is automatic. We now define our learner $\mathbf{Q}$.

- If $\emptyset \in \mathcal{L}$, then $\mathbf{Q}(\lambda) = (\epsilon, \epsilon, 1)$.
- If $\emptyset \notin \mathcal{L}$, then $\mathbf{Q}(\lambda) =$?. In this case, $\mathbf{Q}$ continues to output ? until it receives an input y such that, for some α, $y =$ length lexicographically least element of L_α — at which point it outputs $(\alpha, y, 1)$, for length lexicographically least α such that $L_\alpha = \{y\}$, if there is such an α; otherwise it outputs $(\alpha, y, 1)$, for length lexicographically least α such that y is the length lexicographically least element of L_α.

- $\mathbf{Q}((\alpha, x, cons), \#) = (\alpha, x, cons)$.
- To define $\mathbf{Q}((\alpha, x, cons), y)$, for $y \neq \#$, use the first case below which applies.
  - Case 1: If $y \leq_{ll} x$ and $y \in L_\alpha$, then output $(\alpha, x, cons)$.
  - Case 2: If $y \leq_{ll} x$ and $y \notin L_\alpha$, and there exists a β such that $L_\beta[y] = L_\alpha[y] \cup \{y\}$, then
    if there exists a β such that $L_\beta = L_\alpha[y] \cup \{y\}$, then output $(\beta, y, 1)$ for length lexicographically least such β; otherwise output $(\beta, y, 1)$ for length lexicographically least β such that $L_\beta[y] = L_\alpha[y] \cup \{y\}$.
  - Case 3: If $y >_{ll} x$ and [$y \notin L_\alpha$ or $x < \text{ttcow}(\alpha)$ or $cons = 0$] and there exists a β such that $L_\beta[y] = L_\alpha[x] \cup \{y\}$, then
    if there exists a β such that $L_\beta = L_\alpha[x] \cup \{y\}$, then output $(\beta, y, 1)$ for length lexicographically least such β; otherwise output $(\beta, y, 1)$ for length lexicographically least β such that $L_\beta[y] = L_\alpha[x] \cup \{y\}$.
  - Case 4: Otherwise, let $cons' = cons \wedge y \in L_\alpha$ and output $(\alpha, x, cons')$.

It is easy to verify that the four invariants are satisfied. Also, clearly if $\emptyset \in \mathcal{L}$, then $\mathbf{Q}$ identifies $\emptyset$. Now, suppose T is a fat text for a language $L = L_\beta \in \mathcal{L}$, where β is length lexicographically minimized and $L \neq \emptyset$. Let $(\alpha_n, x_n, cons_n)$ denote $\mathbf{Q}(T[n])$. Note that by construction, except for initial period when $\mathbf{Q}$ conjectures ?, x_n always belongs to L.

Claim 30. *For all $y \in L$ with $y \leq \text{ttcow}(\beta)$, for all but finitely many n, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$.*

Proof of Claim: Note that, for $y \in L$, if $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$, then for all $n' \geq n$, $L[y] = L_{\alpha_{n'}}[y]$ and $y \leq x_{n'}$ (this follows from invariants I and II). Thus, it suffices to show that for all $y \leq \text{ttcow}(\beta)$, there exists an n such that $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$. To see this suppose by way of contradiction that y is length lexicographically least such that above does not hold. Clearly, y cannot be $\min L$, as for the least n such that $T(n-1) = \min L$ we will have $L[\min L] = L_{\alpha_n}[\min L]$ and $\min L \leq x_n$ (using invariant I, and either by first hypothesis of $\mathbf{Q}$ or by usage of Case 2 in definition of $\mathbf{Q}$ when it receives $T(n-1)$). Here note that "hypothesis" always means an output of the learner different from "?". Let $\text{prev}_L(y)$ be the length lexicographic predecessor of y in L. Thus, there exists an n_0 such that for all $n \geq n_0$, $L[\text{prev}_L(y)] = L_{\alpha_n}[\text{prev}_L(y)]$ and $x_n \geq_{ll} \text{prev}_L(y)$ (such an n_0 exists, as y above was the length lexicographically least violator of the claim).

Now we show that the y satisfies the claim. We first show that there exists an $n_3 > n_0$ such that $x_{n_3} \geq y$. So suppose by way of contradiction that such an n_3 does not exist. Then, for all $n \geq n_0$, we have that $x_n = \text{prev}_L(y)$, and $\alpha_n = \alpha_{n_0}$ (as Case 2 would not apply, and application of Case 3 would make $x_n \geq y$). Now, if $\text{prev}_L(y) < \text{ttcow}(\alpha_{n_0})$, then for the least $n_1 > n_0$ such that $T(n_1 - 1) = y$, we would have that x_{n_1} is made to be y (which is $> \text{prev}_L(y)$) by Case 3 (if it had not happened earlier). On the other hand, if $\text{prev}_L(y) \geq \text{ttcow}(\alpha_{n_0})$, then $L \not\subseteq L_{\alpha_{n_0}}$, by Angluin's Tell Tale Condition. Thus, for some $n_2 > n_0$, we have that $cons = 0$. It follows that, for least $n_3 > n_2$ such that $T(n_3 - 1) = y$, we would have that $x_{n_3} = y$, by Case 3.

Now, if $y \in L_{\alpha_{n_3}}$, then we are done. Otherwise, for the least $n_4 > n_3$, such that $T(n_4 - 1) = y$, we will have that $x_{n_4} = y$ and $y \in L_{\alpha_{n_4}}$, by Case 2 (all the intermediate steps between n_3 and

n_4 will not reduce x to below y, by Case 1, and y not appearing in between $T(n_3-1)$ and $T(n_4-1)$). This proves the claim.

It follows from the claim that for some n_5, for all $n \geq n_5$, $\mathrm{CF}_{L[\mathrm{ttcow}(\beta)]} \leq_{lex} \mathrm{CF}_{L_{\alpha_n}[x_n]} \leq_{lex} \mathrm{CF}_L$. If $\alpha_n = \beta$, for one such n, then the learner $\mathbf{Q}$ will not change its mind later, by construction of $\mathbf{Q}$. We show that such an n must exist. So suppose $\alpha_{n_5} \neq \beta$, it means that $L[x_{n_5}] \neq L_{\alpha_{n_5}}[x_{n_5}]$ (by invariant III, and the fact that $L_{\alpha_{n_5}}$ cannot be equal to $L[x_{n_5}]$, by Angluin's Tell-Tale Condition). Thus using invariants I and III we have that $L - L_\alpha[x_{n_5}]$ contains a length lexicographically least element x such that $\mathrm{ttcow}(\beta) < x \leq x_{n_5}$. It follows that, for the least $n' > n_5$, such that $T(n'-1) = x$, $\mathbf{Q}(T[n'])$ will be $(\beta, x_{n'}, \mathit{cons})$, by Case 2, and definition of ttcow.

It follows that $\mathbf{Q}$ **AutoWordEx**-identifies $\mathcal{L}$. $\square$

If one does not use the given hypothesis space $\{L_\alpha : \alpha \in I\}$ but instead the hypothesis space given by $H_{\mathrm{conv}(\alpha,x,\mathit{cons})} = L_\alpha$, then the above learning algorithm becomes an iterative learner using this hypothesis space which always conjectures $\mathrm{conv}(\alpha, x, \mathit{cons})$ in place of α and which has its full memory integrated into the hypothesis. Note that the update rules guarantee that the learner does not update its hypothesis if $L_\alpha[x]$ is a tell-tale set for L_α and L_α is the set to be learnt; hence the modified learner using the new hypothesis space does also converge syntactically on texts for sets to be learnt. This permits to formulate the following corollary.

Corollary 31. *Every automatic class satisfying Angluin's Tell-Tale Condition is* **AutoItEx***-learnable from fat text using a class-preserving hypothesis space.*

The next result shows that one cannot learn every given class iteratively from fat text using a one-one hypothesis space. So "padding", that is, the usage of the hypothesis as an auxiliary memory, is necessary for iterative learning from fat text. Furthermore, the following also shows constraints of iterative conservative automatic learners.

Theorem 32. *Let $\Sigma = \{0,1\}$ and for every $n \in \mathbb{Z}$ and $m \in \{0,1\}$, let $g(m,n) = 4n + 2m$ if $n \geq 0$ and $g(m,n) = -3 - 4n + 2m$ if $n < 0$. Then the class $\mathcal{L}$ defined by the indexing*

$$L_{0^{g(m,n)}} = \{0^{g(i,j)}1^k : i \in \{0,1\} \wedge j \in \mathbb{Z} \wedge k \in \mathbb{N} \wedge (i = m \Rightarrow j \leq n)\}$$

is automatic. This class is class-preservingly **AutoItEx***-learnable from normal text, class-comprisingly conservatively* **AutoItEx** *learnable from normal text, but neither conservatively iteratively learnable from fat text using a class preserving hypothesis space nor iteratively learnable from fat text using a one-one class preserving hypothesis space.*

Proof. For conservative **AutoItEx**-learning using a class-comprising hypothesis space, the hypothesis space used would be:

- $H_{h(0,n_0,n_1)} = L_{0^{g(0,n_0)}}$, for $n_0, n_1 \in \mathbb{Z}$;
- $H_{h(1,n_0,n_1)} = L_{0^{g(1,n_1)}}$, for $n_0, n_1 \in \mathbb{Z}$;
- $H_{h(\epsilon,n_0,n_1)} = \emptyset$,

where, for $n_0, n_1 \in \{\epsilon\} \cup \mathbb{Z}$, $h(a,b,c)$ is the convolution of a, b', and c' with $b' = 0^{g(0,b)}$ if $b \neq \epsilon$, $b' = 1$ if $b = \epsilon$; $c' = 0^{g(0,c)}$ if $c \neq \epsilon$ and $c' = 1$ if $c = \epsilon$. Learner $\mathbf{P}$ initially conjectures $h(\epsilon,\epsilon,\epsilon)$. We mention below the cases when $\mathbf{P}$ modifies its conjecture. In all other cases, the conjectures are not modified.

- $\mathbf{P}(h(\epsilon,\epsilon,\epsilon), 0^{g(0,j)}1^k) = h(\epsilon, j, \epsilon)$;
- $\mathbf{P}(h(\epsilon,\epsilon,\epsilon), 0^{g(1,j)}1^k) = h(\epsilon, \epsilon, j)$;
- $\mathbf{P}(h(\epsilon,j,\epsilon), 0^{g(1,j')}1^k) = h(0, j, j')$;
- $\mathbf{P}(h(\epsilon,\epsilon,j), 0^{g(0,j')}1^k) = h(0, j', j)$;
- $\mathbf{P}(h(0,j,j'), 0^{g(0,r)}1^k) = h(1, r, j')$, if $r > j$;
- $\mathbf{P}(h(1,j,j'), 0^{g(1,r)}1^k) = h(0, j, r)$, if $r > j'$.

Intuitively, conjectures of the form $h(\cdot, j, \cdot)$, (respectively $h(\cdot,\cdot,j)$) imply that a string of form $0^{g(0,j)}1^k$ (respectively, a string of form $0^{g(1,j)}1^k$) has been seen in the input. One can verify that $\mathbf{P}$ conservatively **AutoItEx**-identifies $\mathcal{L}$.

For class preserving **AutoItEx**-learning, one just modifies the above hypothesis space to have $H_{h(\epsilon,n_0,n_1)} = L_{0^{g(0,0)}}$ and the rest of the proof remains the same (note that the learner is no longer conservative).

To show that $\mathcal{L}$ is not conservatively learnable from fat text using a class preserving hypothesis space nor iteratively learnable from fat text using one-one class preserving hypothesis space, note the following: iterative learners using a one-one class preserving hypothesis space are conservative. So it suffices to show that no conservative learner using class preserving hypothesis space iteratively learns $\mathcal{L}$ from fat text.

Let $\mathbf{F}$ be any conservative iterative learner using class preserving hypothesis space $H_\alpha, \alpha \in J$. Let x be such that $\mathbf{F}(?, x) \neq ?$. Without loss of generality assume that the conjecture is for $L_{0^{g(0,n)}}$ (case of $L_{0^{g(1,n)}}$ is symmetric). If $x \in L_{0^{g(0,n-1)}}$, then $\mathbf{F}$ has overgeneralized and thus does not conservatively learn $\mathcal{L}$. Otherwise, if there is no σ (where $\mathrm{cnt}(\sigma) \subseteq 0^*1^*$) such that $\mathbf{F}(x \diamond \sigma)$ conjectures a hypothesis for $L_{0^{g(1,n')}}$, then we have that $\mathbf{F}$ does not learn $\mathcal{L}$. Otherwise, let $y_1, y_2, \ldots, y_k \in 0^*1^*$ be such that $H_{\mathbf{F}(x \diamond y_1 \diamond y_2 \diamond y_k)} = L_{0^{g(1,n')}}$, where $H_{\mathbf{F}(x \diamond y_1 \diamond y_2 \diamond y_r)} = L_{0^{g(0,s_r)}}$, for $r < k$ and some s_r. Then, we have that $y_1, y_2, \ldots, y_r$ must be of form $0^{g(0,\cdot)}1^k$, and thus $x, y_1, \ldots y_k$ belong to $L_{0^{g(1,n'-1)}}$ and thus $\mathbf{F}$ overgeneralizes and cannot conservatively learn $\mathcal{L}$. □

One might ask whether there are classes which can be learnt using some one-one hypothesis space but not be learnt using some other hypothesis space. The answer is "no". That is, if a class is **AutoItEx**-learnable using a one-one hypothesis space then it is also prescribed **AutoItEx**-learnable, that is, it can be learnt using any class-comprising automatic indexing. In the next result, the option "(from fat text)" has either to be taken at both places or at no place in the theorem.

Proposition 33. *If $\{L_\alpha : \alpha \in I\}, \{H_\beta : \beta \in J\}$ are automatic indexings, the mapping $\alpha \mapsto L_\alpha$ is one-one, every L_α is equal to some H_β and $\{L_\alpha : \alpha \in I\}$ is* **AutoItEx***-learnable (from fat text) using the hypothesis space $\{L_\alpha : \alpha \in I\}$, then it is also* **AutoItEx***-learnable (from fat text) using the hypothesis space $\{H_\beta : \beta \in J\}$.*

Proof. The proof of this proposition can be given by the straight-forward translation of the learner: Let $f(\alpha) = \min\{\beta : H_\beta = L_\alpha\}$ and g be the (partial) inverse with $g(\beta) = \min\{\alpha : L_\alpha = H_\beta\}$. Furthermore, let $f(?) = ?$ and $g(?) = ?$. The functions f, g are both first-order definable and hence automatic. Furthermore, g is defined on the range of f. Now one can replace the learner $\mathbf{Q}$ using the hypothesis space $\{L_\alpha : \alpha \in I\}$ by a new learner $\mathbf{Q}'$ mapping a hypothesis β and an input x to $f(\mathbf{Q}(g(\beta), x))$; note that, under the assumption that the initial value of the learner is ?, one can easily see by induction that all hypotheses output by $\mathbf{Q}'$ are in the range of f and hence in the domain of g. Thus $\mathbf{Q}'$ is well-defined (on valid inputs for the class being learnt). As automatic functions are closed under composition, the learner $\mathbf{Q}'$ is automatic. Furthermore, $\mathbf{Q}'$ converges to $f(\alpha)$ whenever $\mathbf{Q}$ converges to α. Hence the learner $\mathbf{Q}'$ is correct and uses the hypothesis space $\{H_\beta : \beta \in J\}$. Note that the type of text used (normal text or fat text) is for both learners the same. □

Next theorem shows that every automatic class (even those that may not satisfy Angluin's Tell-Tale Condition) are partially learnable by an automatic iterative learner. This corresponds to the result by [22] that the whole class of recursively enumerable languages is partially learnable by some recursive learner.

Theorem 34. *Every automatic $\mathcal{L}$ is* **AutoWordPId***-learnable from fat text.*

Proof. This is a modification of the proof of Theorem 29. In this case we do not need to keep track of *cons* and the memory x may grow unbounded.

Let Σ be the alphabet used for $\mathcal{L}$ and I be the set of indices. We will assume below that α, β range over I.

We will construct the learner $\mathbf{Q}$ below. Without loss of generality we assume that if $\emptyset \in \mathcal{L}$, then $L_\epsilon = \emptyset$.

We let $L_\alpha[z]$ denote the set $\{y : y \leq_{ll} z\}$.

We will denote the conjecture/memory of the learner $\mathbf{Q}$ by (α, x), where α is the conjecture, and x is the memory. Here $x \in \Sigma^*$.

Suppose T is the input fat text for a language $L \in \mathcal{L}$. Then we will have the following invariants, whenever α and α' below are not ?:

(I) If $\mathbf{Q}(T[m]) = (\alpha, x)$, then $L_\alpha[x] \subseteq \text{cnt}(T[m])$. Furthermore, for any $m' < m$, if $\mathbf{Q}(T[m']) = (\alpha', x')$, then $L_\alpha[x] \subseteq L_{\alpha'}[x'] \cup \{T(m'), T(m'+1), \ldots, T(m-1)\}$.
(II) If $\mathbf{Q}(T[m]) = (\alpha, x)$ and $\mathbf{Q}(T[m+m']) = (\alpha', x')$ then $\text{CF}_{L_\alpha[x]} \leq_{lex} \text{CF}_{L_{\alpha'}[x']} \leq_{lex} \text{CF}_L$.

We now describe the learner $\mathbf{Q}$.

- If $\emptyset \in \mathcal{L}$ then $\mathbf{Q}(\lambda) = (\epsilon, \epsilon)$.
- If $\emptyset \notin \mathcal{L}$ then $\mathbf{Q}(\lambda) = ?$. In this case, $\mathbf{Q}$ continues to output ? until it receives an input y such that, for some α, $y = \min L_\alpha$ — at which point it outputs (α, y), for the length lexicographically least such α.
- $\mathbf{Q}((\alpha, x), \#) = (\alpha, x)$.
- To define $\mathbf{Q}((\alpha, x), y)$, for $y \neq \#$, use the first case below which applies.

– Case 1: If $y >_{ll} x$ and there exists a β such that $L_\alpha[x] \cup \{y\} = L_\beta[y]$, then output (β, y), for length lexicographically least such β.

– Case 2: If $y \leq_{ll} x$ and $y \notin L_\alpha$, and there exists a β such that, $L_\beta[y] = L_\alpha[y] \cup \{y\}$, then output (β, y), for length lexicographically least such β.

– Case 3: If there exists a β such that $L_\beta = L_\alpha[x]$, then output (β, x), for length lexicographically least such β.

– Case 4: Otherwise output (α, x).

It is easy to verify that the invariants are satisfied. Clearly, $\mathbf{Q}$ learns $\emptyset$, if $\emptyset \in \mathcal{L}$. So suppose $L \neq \emptyset$, and $L = L_\beta \in \mathcal{L}$, where β is length lexicographically minimized. Suppose T is a fat text for L. Let (α_n, x_n) denote $\mathbf{Q}(T[n])$.

We claim that for all $y \in L$, for all but finitely many n, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$. Note that by invariants I and II, if $L[y] = L_{\alpha_n}[y]$, $y \leq x_n$, then for all $n' \geq n$, $L[y] = L_{\alpha_{n'}}[y]$ and $y \leq x_{n'}$. Thus, it suffices to show that for all $y \in L$, there exists an n such that $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$. To see this suppose by way of contradiction that above fails for some $y \in L$. Choose length lexicographically least such y. Clearly, y cannot be $\min L$ as for the least n such that $T(n-1) = \min L$, we will have $L[\min L] = L_{\alpha_n}[\min L]$ and $\min L \leq x_n$ (using invariant I, and either by first hypothesis of $\mathbf{Q}$ or by usage of Case 2 in definition of $\mathbf{Q}$ when it receives $T(n-1)$).

Now we show that the y satisfies the claim. Let $\text{prev}_L(y)$ denote the length lexicographic predecessor of y in L. Let n_0 be large enough such that for all $n \geq n_0$, $L[\text{prev}_L(y)] = L_{\alpha_n}[\text{prev}_L(y)]$, and $x_n \geq_{ll} \text{prev}_L(y)$ (by y being length lexicographically least violator of the claim, such an n_0 exists). Let $n'' > n$ be such that $T(n''-1) = y$. Then, $L_{\alpha_{n''}}[y] = L[y]$ and $x_{n''} \geq_{ll} y$ (Case 1 and 2 both will ensure this, if it is not already true).

It follows that $\text{CF}_{L_{\alpha_n}[x]}$ converges to CF_L from below. If L is finite, then let n_1 be such that $\text{CF}_{L_{\alpha_{n_1}}} = \text{CF}_L$ and $x_{n_1} \geq \max L$. Let $n_2 > n_1$ be such that $T(n_2 - 1) \neq \#$. Then, by Case 3, it follows that, for all $n \geq n_2$, $\alpha_n = \beta$. Thus, $\mathbf{Q}$ **AutoItPId**-identifies all the finite sets in $\mathcal{L}$.

So suppose L is infinite. As $\text{CF}_{L_{\alpha_n}[x_n]}$ converges to CF_L from below, it follows that no α with $L_\alpha \neq L$ would be output infinitely often. Furthermore, no non-length lexicographically minimal index for any language is ever output. So it suffices to show that β is output infinitely often. Let x be large enough so that $x \in L$, $L[x] \neq L_\alpha[x]$, for any $\alpha <_{ll} \beta$. Now, as $\text{CF}_{L_{\alpha_n}[x_n]}$ converges to CF_L from below, for large enough n, for all $n' \geq n$, $x_{n'} \geq_{ll} x$ and $\text{CF}_{L[x]} = \text{CF}_{L_{\alpha_{n'}}[x]}$. Now consider any $n' \geq n$. Note that either $\alpha_{n'} = \beta$ or $L - L_{\alpha_{n'}} \neq \emptyset$, as either $L_{\alpha_{n'}}$ is finite or $L[x] = L_{\alpha_{n'}}[x] \subseteq L_{\alpha_{n'}}[x_{n'}] \subseteq L$ and $\alpha_{n'}$ is the length lexicographically least index α which satisfied $L_\alpha[x_{n'}] = L_{\alpha_{n'}}[x_{n'}]$ (by definition of $\mathbf{Q}$). Thus, by Case 1, using invariant 1, for any $n'' > n'$ such that $T(n''-1) =$ the length lexicographically least element in $L - L_{\alpha_{n'}}[x]$, we have $\alpha_{n''} = \beta$, unless $\alpha_{n'''} = \beta$ for some n''' with $n' \leq n''' \leq n''$. The theorem follows. □

Acknowledgments. We would like to thank John Case, Henning Fernau, Pavel Semukhin and Thomas Zeugmann for discussions about the subject of learning classes with automatic indexings.

References

1. Dana Angluin. Inductive Inference of Formal Languages from Positive Data. *Information and Control* 45:117–135, 1980.
2. Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
3. Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.
4. Janis Bārzdiņš. Two Theorems on the Limiting Synthesis of Functions *Theorems of Algorithms and Programs* 1:82–88, 1974.
5. Janis Bārzdiņš. Inductive inference of automata, functions and programs. In *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–560, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107–112, 1977.
6. Leonore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
7. Achim Blumensath. *Automatic Structures*. Diploma thesis, RWTH Aachen, 1999.
8. Achim Blumensath and Erich Grädel. Automatic Structures. *15th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 51–62, IEEE Computer Society, 2000.
9. John Case. The Power of Vacillation in Language Learning. *SIAM Journal of Computing*, 28(6):1941–1969, 1999.
10. Thomas Erlebach, Peter Rossmanith, Hans Stadtherr, Angelika Steger and Thomas Zeugmann. Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. *Theoretical Computer Science*, 261(1):119-156, 2001.
11. E. Mark Gold. Language Identification in the Limit. *Information and Control* 10:447–474, 1967.
12. Bernard R. Hodgson. *Théories décidables par automate fini*. Ph.D. thesis, University of Montréal, 1976.
13. Bernard R. Hodgson. On direct products of automaton decidable theories. *Theoretical Computer Science*, 19:331–335, 1982.
14. Klaus P. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.
15. Michael Kearns and Leonard Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. *Proceedings of the Second Annual ACM Workshop on Computational Learning Theory*, pages 57–71, Morgan Kaufmann Publishers Inc., 1989.
16. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity*, International Workshop LCC 1994, Springer LNCS 960:367–392, 1995.
17. Efim Kinber and Frank Stephan. Language Learning from Texts: Mind Changes, Limited Memory and Monotonicity. *Information and Computation* 123(2):224–241, 1995.
18. Steffen Lange and Rolf Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.

19. Steffen Lange, Thomas Zeugmann and Shyam Kapur. Characterizations of Monotonic and Dual Monotonic Language Learning. *Information and Computation*, 120:155–173, 1995.
20. Yasuhito Mukouchi. Characterization of finite identification. *Proceedings of the Third International Workshop on Analogical and Inductive Inference* (AII), *Springer Lecture Notes in Artificial Intelligence* 642:260–267, 1992.
21. Daniel Osherson, Michael Stob and Scott Weinstein, Learning Strategies. *Information and Control*, 53:32–51, 1982.
22. Daniel Osherson, Michael Stob and Scott Weinstein, *Systems That Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists*. Bradford — The MIT Press, Cambridge, Massachusetts, 1986.
23. Sasha Rubin. *Automatic Structures*. Ph.D. Thesis, University of Auckland, 2004.
24. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
25. Gisela Schäfer-Richter. *Uber Eingabeabhangigkeit und Komplexitat von Inferenzstrategien.* PhD thesis, RWTH Aachen, 1984.
26. Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition*. MIT Press, 1980.
27. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik*, 12:93–99, 1976.