Learnability of Automatic Classes

Sanjay Jain[a,1], Qinglong Luo[b], Frank Stephan[c,1]

[a]Department of Computer Science, National University of Singapore, Singapore 117417
[b]DSO National Laboratories, 20 Science Park Drive, Singapore 118230
[c]Department of Mathematics and Department of Computer Science, National University of Singapore, Singapore 119076

Abstract

The present work initiates the study of the learnability of automatic indexable classes which are classes of regular languages of a certain form. Angluin's tell-tale condition characterises when these classes are explanatorily learnable. Therefore, the more interesting question is when learnability holds for learners with complexity bounds, formulated in the automata-theoretic setting. The learners in question work iteratively, in some cases with an additional long-term memory, where the update function of the learner mapping old hypothesis, old memory and current datum to new hypothesis and new memory is automatic. Furthermore, the dependence of the learnability on the indexing is also investigated. This work brings together the fields of inductive inference and automatic structures.

Key words: Inductive Inference, Automatic Structures

1. Introduction

The present work studies inductive inference [17] within the framework of automata theory and, in particular, automatic structures. The basic scenario of inductive inference is that a learner is receiving, one piece at a time, data about a target concept. As the learner is receiving the data, it conjectures a hypothesis about what the target concept might be. The hypothesis may be modified or 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.

The concept classes of interest to us in this paper are classes of regular languages. (A regular language is a subset of Σ^*, for some finite alphabet Σ, which is recognised by a finite automaton.) The data provided to the learner then

Email addresses: `sanjay@comp.nus.edu.sg` (Sanjay Jain), `lqinglon@dso.org.sg` (Qinglong Luo), `fstephan@comp.nus.edu.sg` (Frank Stephan)

[1]Sanjay Jain was supported in part by NUS grants R252-000-308-112 and C252-000-087-001. Frank Stephan was supported in part by NUS grants R146-000-114-112 and R252-000-308-112.

becomes a sequential presentation of all the elements of the target language, in arbitrary order, with repetition allowed. To deal with the empty language, we also allow a special symbol $\square$ to be presented to the learner. This symbol represents no data. Such a presentation of data is called a *text* for the language. Note that a text presents only positive data to the learner, and not negative data, that is, the learner is not explicitly told which 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 concerned with learning from texts.

In many cases, one considers only recursive learners. The hypotheses produced by the learner describe the language to be learnt in some form. For example, they might be grammars generating the language. The learner is said to **Ex**-learn the target language iff the sequence of hypotheses converges to one correct hypothesis describing the language to be learnt. Here "**Ex**-learn" stands for explanatory learning. Learning of one language L is not interesting, as the learner might ignore all inputs and always output a hypothesis which is correct for L. Thus, what is considered is whether all languages from a class of languages are **Ex**-learnt by some learner. When such a learner exists for a class, that class is said to be **Ex**-learnable.

Since [17], several other models of learning have been considered by the researchers. For example, in behaviourally correct learning (**BC**-learning) [5] the learner is not required to converge syntactically to one correct hypothesis; rather, 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 vacillatory learning (**FEx**-learning) [10], the learner eventually vacillates between only finitely many hypothesis, all of which are correct.

Besides the mode of convergence, researchers have also considered several properties of learners such as

- *consistency*, where the hypothesis of the learner is required to contain the elements seen in the input so far (see [6, 7]),
- *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 [36, 37]). Iterative learning is often also called incremental learning.

The formal definitions of the above criteria are given in Section 2 below.

Besides considering models of learning, there has also been interest in considering learning of practical and concrete classes such as the pattern languages [2, 14, 23, 27], elementary formal systems [35] and the regular languages [4]. As the class of all regular languages is not learnable from positive data [17], Angluin [3] initiated the study of the learnability of certain subclasses of the regular languages from positive data. In particular, she showed the learnability of the class of k-reversible languages. These studies were later extended [13, 15, 18]. The classes considered in these studies were all superclasses of the class of all 0-reversible languages, which is not automatic; for example, every language $\{0,1\}^*\{2\}^n\{0,1\}^*$ is 0-reversible but the class of these languages is not automatic. Also many other subclasses of regular languages which have been considered in the literature are not automatic. Furthermore, learnability of regular languages from counterexamples and queries has also been studied (for example by Angluin [4] and Ibarra and Jiang [21]) but we will not be concerned with these learning models in this paper.

In this work, we consider those subclasses of regular languages where the membership problem is regular in the sense that one automaton accepts a combination (called a *convolution*) of an index and a word iff the word is in the language given by the index. This is formalised in the framework of automatic structures [8, 9, 19, 20, 24, 32, 33]. 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 [24]. This makes several properties of such classes regular and thus decidable; it also makes it possible to define learners using first-order definitions. Studies in automatic structures have connections to model checking and Boolean algebra [9, 24].

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 9 that every automatic class that satisfies 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 Angluin's tell-tale condition and thus whether it is **Ex**-learnable (see Corollary 10).

As we are considering learning of automatic classes, 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. This would put both, the learners and the classes to be learnt into a unified framework. Furthermore, the automatic learners are linear time computable [11] and additional constraints on the memory can be satisfied. This approach is justified by the observation that a learner might observe much more data than it can remember and therefore it is not realistic to assume that the whole learning history can be remembered. To model the above, we consider variants of iterative learners [36, 37] and learners with bounded long-term memory [16, 25]. The basic idea is that the learner reads in each round a datum and updates the long term memory and the hypothesis based on this datum; for automatic learners, this update function is then required to be automatic.

As automatic structures are relatively simple to implement and analyse, 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 the 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 16 shows that there are automatic classes which are **Ex**-learnable (even iteratively) but not learnable by any automatic learners.

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 the same as automatic **BC**-learning, in contrast to the situation in learning of recursively enumerable languages by recursive learners, where there is a difference [5]. 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 17). Similarly, for **FEx**-learning, all automatically **FEx**-learnable classes with long-term memory bounded by size of the hypothesis are iteratively **FEx**-learnable. 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 the largest word seen in the input so far (see Theorem 18). For both **Ex** and **FEx**-learning, it is open at this point whether bounding the size of the long-term memory by the size of the longest word seen so far is equivalent to there being no explicit bound on the size of the long-term memory. For explanatory learning, it is additionally open whether constant size memory 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 9, where we show that general learners for automatic classes can be made consistent, automatic learners cannot in general be made consistent. Theorem 22 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 those implicit due to the learner being automatic. Theorem 23 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 25 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 hypotheses from an automatic class which is a superset of the class $\mathcal{L}$), but not **Ex**-learnable by a consistent automatic learner (with no constraints on long-term memory, except those implicit due to the learner being automatic) using a class preserving hypothesis space, i.e., using a 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. An attempt to overcome this is to require that every datum appears infinitely often in the text — such a text is called a *fat text* [31]. Fat texts are quite frequently studied in learning theory. In Section 5 we investigate the natural question of whether requiring fat texts permits the limitations of iterative learning and related criteria to be overcome. In Theorem 28 we show that every automatic class that satisfies Angluin's tell-tale condition is **Ex**-learnable (using the automatic class itself as the 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 an arbitrary class preserving hypothesis space, then one can even do **Ex**-learning in the above case by iterative automatic learners and no additional long-term memory is needed.

In Theorem 31, 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 one-one class preserving hypothesis space (even by arbitrary recursive learners) on fat texts.

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 hypotheses are output only finitely often [31]. In Theorem 35 we show that every automatic class is partially learnable by an automatic iterative learner. This corresponds to the result by Osherson, Stob and Weinstein [31] that the whole class of all recursively enumerable languages is partially learnable by some recursive learner.

2. Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers. Let $\mathbb{Z}$ denote the set of integers. The symbol $\emptyset$ denotes the empty set. Symbols $\subseteq, \supseteq, \subset, \supset$, 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, where $\max \emptyset = 0$ and $\min \emptyset = \infty$.

An alphabet Σ is any non-empty finite set and Σ^* is the set of all strings (words) over the alphabet Σ. The symbol ε denotes the empty string. A string of length n over 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 the same as $x(0)x(1)x(2)\ldots x(n-1)$. A language is a subset of Σ^* and a class is a set of languages.

The relation $x <_{lex} y$ denotes that x is lexicographically (that is, in dictionary order) before y. The relation $x <_{ll} y$ denotes that x is length-lexicographically before y, that is, either $|x| < |y|$, or $|x| = |y|$ and $x <_{lex} y$. Note that, for any given alphabet, the reflexive closure of $<_{ll}$ is a linear order over the strings of that alphabet. When we consider sets of strings, we take $\min S$ to denote the length-lexicographically least string in S. Let $\operatorname{succ}_L(x)$ denote the length-lexicographically least y, if any, in L such that $x <_{ll} y$.

For a language $L \subseteq \Sigma^*$, 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, $\mathrm{CF}_L(n) = 1$, if $z_n \in L$; $\mathrm{CF}_L(n) = 0$, otherwise. For any language L, we let $L[y]$ denote the set $\{x \in L : x \leq_{ll} y\}$.

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 1. 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 2 (Khoussainov, Nerode [24]). 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 $z(j)$ is a symbol in the alphabet $(\Sigma_1 \cup \{\#\}) \times (\Sigma_2 \cup \{\#\}) \times \ldots \times (\Sigma_n \cup \{\#\})$. We call z the *convolution of* $x_1, x_2, \ldots, x_n$ and denote it as $\mathrm{conv}(x_1, x_2, \ldots, x_n)$. Let $R \subseteq \Sigma_1^* \times \Sigma_2^* \times \ldots \times \Sigma_n^*$. We call the set $S = \{\mathrm{conv}(x_1, x_2, \ldots, x_n) : (x_1, x_2, \ldots, x_n) \in R\}$, the *convolution of* R. Furthermore, we say that R is *automatic* iff the convolution of R is regular.

For ease of notation, we often write just $(x_1, x_2, \ldots, x_n)$ instead of $\mathrm{conv}(x_1, x_2, \ldots, x_n)$ and $L_{x_1,\ldots,x_n}$ or $H_{x_1,\ldots,x_n}$ in place of $L_{\mathrm{conv}(x_1,\ldots,x_n)}$ or $H_{x_1,\ldots,x_n}$ respectively. We next define the notion of an automatic indexing.

Definition 3. An indexing $\{L_\alpha : \alpha \in I\}$ is *automatic* iff I is regular and $E = \{(\alpha, x) : x \in L_\alpha, \alpha \in I\}$ is automatic. A class is *automatic* iff it has an automatic indexing.

Khoussainov and Nerode [24] found the following fundamental result on automatic structures which is useful to define automatic learners and to decide the learnability of automatic classes.

Fact 4 (Blumensath, Grädel [9], Khoussainov, Nerode [24]). *Any relation that is first-order definable from existing automatic relations is automatic.*

Next, we recall a few learning relevant definitions, followed by a result from Angluin [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 $\Sigma^* \cup \{\Box\}$. Furthermore, λ denotes the empty sequence. The length of a sequence σ is denoted by $|\sigma|$. $T[m]$ denotes the initial segment of T of length m. We let $\sigma \diamond \tau$ (respectively, $\sigma \diamond T$) denote 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 the context that concatenation of sequences is meant, we sometimes drop the symbol $\diamond$. Thus, $\sigma\tau$ means $\sigma \diamond \tau$. For a finite sequence σ over $\Sigma^* \cup \{\Box\}$, content of σ, denoted by $\mathrm{cnt}(\sigma)$, is defined as $\mathrm{cnt}(\sigma) = \{x \in \Sigma^* : \exists\, n < |\sigma|\, (\sigma(n) = x)\}$. Similarly, for every infinite sequence T over $\Sigma^* \cup \{\Box\}$, content of T, denoted by $\mathrm{cnt}(T)$, is defined as $\mathrm{cnt}(T) = \{x \in \Sigma^* : \exists\, n \in \mathbb{N}\, (T(n) = x)\}$. 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 $\mathrm{txt}(L) = \{T \in (\Sigma^* \cup \{\Box\})^\omega : \mathrm{cnt}(T) = L\}$ and $\mathrm{seq}(L) = \{\sigma \in (\Sigma^* \cup \{\Box\})^* : \mathrm{cnt}(\sigma) \subseteq L\}$.

Given a class $\mathcal{L}$, a *hypothesis space* for $\mathcal{L}$ is an indexing $\{H_\alpha : \alpha \in J\} \supseteq \mathcal{L}$, where J is the set of indices for the hypothesis space. We will only consider automatic hypothesis spaces. A hypothesis space is *class preserving* with respect to $\mathcal{L}$ iff $\mathcal{L} = \{H_\alpha : \alpha \in J\}$. A hypothesis space is *class comprising* with respect to $\mathcal{L}$ iff $\mathcal{L} \subseteq \{H_\alpha : \alpha \in J\}$. A hypothesis space is *one-one class preserving* with respect to $\mathcal{L}$ iff it is class preserving and, for every $L \in \mathcal{L}$, there is exactly one $\alpha \in J$ with $L = H_\alpha$. In usage of above definitions, we often drop "with respect to $\mathcal{L}$", if the class $\mathcal{L}$ is clear from context.

A *learner* is a function $\mathbf{F} : (\Sigma^* \cup \{\Box\})^* \to J \cup \{?\}$. We use $\mathbf{M}$ and $\mathbf{N}$ for recursive learners, and $\mathbf{F}$ for learners which may not be recursive. We use $\mathbf{P}$ for iterative learners and $\mathbf{Q}$ for iterative learners with additional long-term memory. The learners $\mathbf{P}$ and $\mathbf{Q}$ are usually automatic. Iterative and automatic learners are defined in Section 3 below.

Definition 5. Fix a class $\mathcal{L}$ and a hypothesis space $\{H_\alpha : \alpha \in J\}$ with J being the set of indices. Let $\mathbf{F}$ be a learner.
(a) [17] We say that $\mathbf{F}$ **Ex**-*learns* $\mathcal{L}$ iff 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) [5] We say that $\mathbf{F}$ **BC**-*learns* $\mathcal{L}$ iff 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) [10] We say that $\mathbf{F}$ **FEx**-*learns* $\mathcal{L}$ iff $\mathbf{F}$ **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) [31] We say that $\mathbf{F}$ **Part**-*learns* $\mathcal{L}$ iff for every $L \in \mathcal{L}$ and every $T \in \mathrm{txt}(L)$, 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 **Part** learning, one can assume without loss of generality that the learner never outputs ?. However, for some other criteria of learning, this is not necessarily the case.

Definition 6. Let Σ and Γ be alphabets. Let $\{H_\alpha : \alpha \in J\}$ be a hypothesis space with some $J \subseteq \Gamma^*$ being the set of indices. Let $\mathbf{F}$ be a learner.
(a) [6] We say that $\mathbf{F}$ is *consistent* on $L \subseteq \Sigma^*$ iff 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 $\mathbf{F}$ is consistent on $\mathcal{L} \subseteq \mathrm{powerset}(\Sigma^*)$ iff it is consistent on each $L \in \mathcal{L}$.
(b) [1] We say that $\mathbf{F}$ is *conservative* on $L \subseteq \Sigma^*$ iff 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 $\mathbf{F}$ is conservative on $\mathcal{L} \subseteq \mathrm{powerset}(\Sigma^*)$ iff it is conservative on each $L \in \mathcal{L}$.
(c) [30, 34] We say that $\mathbf{F}$ is *set-driven* iff 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 **Part**, 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 [7] introduced the notion of a *locking sequence* for a learner $\mathbf{F}$ on a set L learnt by $\mathbf{F}$: a locking sequence for a learner $\mathbf{F}$ on L is any sequence $\sigma \in \mathrm{seq}(L)$ such that, for some fixed index e for 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 **Ex**-learnt by $\mathbf{F}$ and this notion can be adapted for most learning criteria considered in this paper.

Using locking sequences, techniques of Angluin [1] can be used to characterise 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 7 (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* $\mathcal{L}$) iff 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* $\mathcal{L}$) iff $\{y \in L : y \leq_{ll} x\}$ is a tell-tale set of L (in $\mathcal{L}$).
(c) We say that $\mathcal{L}$ satisfies *Angluin's tell-tale condition* iff 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 8 (Based on Angluin [1]). *Let Σ be an alphabet. A class $\mathcal{L}$ of recursively enumerable languages is* **Ex***-learnable* (*by a not necessarily recursive learner*) *iff $\mathcal{L}$ satisfies Angluin's tell-tale condition.*

Note that for non-recursive learners, **Ex**, **FEx** and **BC** 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, $\{E_\alpha : \alpha \in J\}$, 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 [6], [1] and [30, 34]).

Using the Fundamental Theorem for automatic structures, the following theorem shows 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 9. *Suppose $\mathcal{L}$ is automatic. Then, there is a learner which recursively, consistently, conservatively and set-drivenly* **Ex***-learns $\mathcal{L}$ iff $\mathcal{L}$ satisfies Angluin's tell-tale condition.*

Proof. ($\Leftarrow$) This follows from Fact 8.

($\Rightarrow$) Suppose $\mathcal{L}$ is automatic and satisfies Angluin's tell-tale condition. Let $\{L_\alpha : \alpha \in I\}$, be an indexing of $\mathcal{L}$. Now consider the learner $\mathbf{M}$ (which uses $\{L_\alpha : \alpha \in I\}$ as the hypothesis space) 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 $\beta \in I$, $[\beta \leq_{ll} \alpha \wedge \mathrm{cnt}(\sigma) \subseteq L_\beta \Rightarrow L_\alpha \subseteq L_\beta]$,

- 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. Also, 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.

Consider now any text T for a language $L \in \mathcal{L}$ and let α be its minimal index. If n is sufficiently large, then it follows from Angluin's tell-tale condition that (i) $\mathrm{cnt}(T[n])$ is a tell-tale set for L and (ii) for all $\beta <_{ll} \alpha$, either $\mathrm{cnt}(T[n]) \not\subseteq L_\beta$ or $L_\alpha \subseteq L_\beta$. So all large enough n satisfy $\mathbf{M}(T[n]) = \alpha$. It follows that $\mathbf{M}$ **Ex**-learns $\mathcal{L}$. ∎

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

Corollary 10. *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 are descriptions of Σ, Γ (where, $L_\alpha \subseteq \Sigma^*$ and $I \subseteq \Gamma^*$) and finite automata recognising the regular languages I and $\{(\alpha, x) : x \in L_\alpha, \alpha \in I\}$.*

Remark 11. One can obtain similar characterisations for other fundamental notions of learning.

(a) Recall that a class is *finitely learnable* [17] 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 ?) and 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 [29]. For ($\Leftarrow$), suppose the right hand side holds. Let $\{L_\alpha : \alpha \in I\}$, be an automatic indexing of $\mathcal{L}$. Now the following learner $\mathbf{M}$ finitely learns $\mathcal{L}$. On input σ, $\mathbf{M}(\sigma)$ conjectures the length-lexicographically least $\alpha \in I$ such that

- $\mathrm{cnt}(\sigma) \subseteq L_\alpha$ and
- for all $\beta \in I$, $\mathrm{cnt}(\sigma) \subseteq L_\beta$ implies $L_\alpha = L_\beta$.

If such an α does not exist then $\mathbf{M}(\sigma) =?$. It is easy to verify that $\mathbf{M}$ finitely learns $\mathcal{L}$.

(b) A class is *strong monotonically learnable* [22] iff there exists an **Ex**-learner $\mathbf{M}$ for the class such that for any two subsequent hypotheses α, β of $\mathbf{M}$ on a text, with $\alpha \neq ?$ and $\beta \neq ?$, it holds that $L_\alpha \subseteq L_\beta$. 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 [28] showed the direction ($\Rightarrow$). For the direction ($\Leftarrow$), assume that the right hand side holds. Let $\{L_\alpha : \alpha \in I\}$, be an automatic indexing of $\mathcal{L}$. Now the following learner $\mathbf{M}$ strong monotonically learns $\mathcal{L}$. On input σ, $\mathbf{M}(\sigma)$ conjectures the length-lexicographically least α such that, $\mathrm{cnt}(\sigma) \subseteq L_\alpha$ and for all $\beta \in I$, if $\mathrm{cnt}(\sigma) \subseteq L_\beta$ then $\mathrm{cnt}(\sigma) \subseteq L_\alpha \subseteq L_\beta$. In the case that there is no such α then $\mathbf{M}(\sigma) =?$. 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 memorise all 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, one word at a time [36, 37]. An iterative learner bases its new conjecture only on its previous conjecture and the new datum. In other words, such a learner does not remember its 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 would lead to an interesting model. However, for iterative learners, there is a natural corresponding definition for automatic learners where the update function is automatic. Below we formally define automatic iterative learning and its variant, iterative learning with long-term memory.

Definition 12 (Wexler and Culicover [36], Wiehagen [37]). Let the alphabets Σ, Γ and Δ be given. 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

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

An *iterative learner with long-term memory* is any function

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

where the strings in Δ^* represent the memory of the learner.

Given an iterative learner $\mathbf{P}$, we now write $\mathbf{P}(w_0 w_1 \ldots w_n)$ as a short hand for the expression $\mathbf{P}(\ldots \mathbf{P}(\mathbf{P}(?, w_0), w_1), \ldots, w_n)$. Similarly, for an iterative learner $\mathbf{Q}$ with long term memory, we write $\mathbf{Q}(w_0 w_1 \ldots w_n)$ as a short hand for the expression $\mathbf{Q}(\ldots \mathbf{Q}(\mathbf{Q}((?, \varepsilon), w_0), w_1), \ldots, w_n)$. Here, for $\mathbf{Q}(\sigma) = (\alpha, \mu)$, we consider α as the conjecture and μ implicitly as its memory and not as its output. 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. Parts (b)–(d) of the following definition is based on [16, 25].

Definition 13. Suppose $\mathcal{L}$ is defined over alphabet Σ, and $\{H_\alpha : \alpha \in J\}$, $J \subseteq \Gamma^*$, 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* iff the relation

$$\{(\alpha, w, \beta) : \alpha, \beta \in J \cup \{?\}\,, w \in \Sigma^* \cup \{\Box\} \text{ and } \mathbf{P}(\alpha, w) = \beta\}$$

is automatic. We say that $\mathbf{Q}$ is *automatic* iff the relation

$$\{(\alpha, \mu, w, \beta, \nu) : \alpha, \beta \in J \cup \{?\}\,, \mu, \nu \in \Delta^*, w \in \Sigma^* \cup \{\Box\} \text{ and } \mathbf{Q}((\alpha, \mu), w) = (\beta, \nu)\}$$

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

(α, μ), then $|\mu| \leq |\alpha| + c$.
(d) We say that the long-term memory of $\mathbf{Q}$ is *bounded by a constant* iff there exists a constant $c \in \mathbb{N}$ such that, for every $\sigma \in (\Sigma^* \cup \{\Box\})^*$, if $\mathbf{Q}(\sigma) = (\alpha, \mu)$, then $|\mu| \leq c$. [2]

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

Definition 14. For the following, the hypothesis space is allowed to be any class comprising automatic family. Let **LC** be one of **Ex**, **FEx**, **BC** and **Part**. We let
(a) **AutoLC** be the set of all classes of languages that are **LC**-learned by some automatic learner with arbitrary long-term memory,
(b) **AutoWordLC** be the set of all classes of languages that are **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 are **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 are **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 are **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. The following proposition is useful:

Proposition 15. *Suppose* $\mathbf{Q}$ *is an automatic iterative learner with long-term memory. Then, for some constant* c*, for all* $\sigma \in (\Sigma^* \cup \{\Box\})^*$*, if* $\mathbf{Q}(\sigma) = (\alpha, \mu)$*, then* $|\mu| \leq c * |\sigma| + \max\{|w| : w \in \mathrm{cnt}(\sigma)\}$.

Proof. The proposition follows using the fact that, for some constant c, if $Q((\alpha, \mu'), x) = (\alpha', \mu'')$, then $\max\{|\mu''|, |\alpha'|\} \leq \max\{|\mu'|, |\alpha|, |x|\} + c$. ∎

We will implicitly use the above proposition in several of our proofs.

Theorem 16. *There exists an automatic* $\mathcal{L}$ *that is* **Ex** *learnable by some recursive iterative learner, but which is not* **AutoEx***-learnable.*

[2] Note that in subsequent work [12], a more restrictive version of constant memory was considered. Here the learner $\mathbf{Q}$ only memorises μ and forms each hypothesis as a function of μ and the current datum. To address the issue of learner wishing to repeat its previous hypothesis (which is not stored), a slight modification of learning definition is done: the learner is said to learn iff eventually it conjectures a correct hypothesis α, and from then onwards always outputs either ? or α. This restrictive learnability notion is not implied by iterative learnability as the class of all finite subsets of $\{0\}^*$ is iteratively learnable but not with constant memory in the just described setting.

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 **Q** **AutoEx** learns $\mathcal{L}$. Then, for large enough m, there exist σ, σ' such that (i) each of σ, σ' is of length m and contains m distinct strings from $\{0,1\}^m$, (ii) $\text{cnt}(\sigma) \neq \text{cnt}(\sigma')$ and (iii) $\mathbf{Q}(\sigma) = \mathbf{Q}(\sigma')$. Note that there exist such σ, σ' for large enough m as there are $\binom{2^m}{m}$ possibilities for the sequences of length m (with distinct content) containing exactly m elements from $\{0,1\}^m$, but the size of the hypothesis and memory of **Q** after seeing such sequences can be of length at most cm, for some constant c (see Proposition 15). 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, **Q** on σT and $\sigma' T$ converges to the same index or diverges on both. Thus, **Q** does not **AutoEx** learn $\mathcal{L}$. ∎

We now consider the relationship between various long-term memory limitations for the 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 17. *The following equivalences and containments 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 definitions that **AutoItBC** $\subseteq$ **AutoConstBC** $\subseteq$ **AutoWordBC** $\subseteq$ **AutoBC** and **AutoItBC** $\subseteq$ **AutoIndexBC** $\subseteq$ **AutoBC**. Thus it suffices to show that **AutoBC** $\subseteq$ **AutoItBC**. Suppose **Q** **AutoBC**-learns $\mathcal{L}$, where the hypothesis space is $\{H_\alpha : \alpha \in I\}$, and the 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 the hypothesis space $\{H'_{\alpha,\mu} : \alpha \in I, \mu \in \Delta^*\}$.

(b) It suffices to show that **AutoBC** $\subseteq$ **AutoEx**. Suppose **Q** **AutoBC**-learns $\mathcal{L}$, where the hypothesis space is $\{H_\alpha : \alpha \in I\}$, $I \subseteq \Gamma^*$, and the memory is over the alphabet Δ. Then consider the following **Q**$'$. **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 **Q**$'$ **AutoEx**-learns $\mathcal{L}$.

(c) It suffices to show **AutoIndexFEx** $\subseteq$ **AutoItFEx**. Suppose that **Q**

AutoIndexFEx-learns $\mathcal{L}$. Then the construction of part (a) witnesses that **P** **AutoItFEx**-learns $\mathcal{L}$, as the number of distinct (α, μ) which are output by **Q** on a given text for a language learnt by **Q** will be finite.

(d) The direction **AutoWordEx** $\subseteq$ **AutoWordFEx** follows from the definition. For the converse direction, one can use the same proof as under (b); but one has to note explicitly that the sizes of β and μ are always bounded by a constant plus the size of the longest datum seen so far; as α is the length-lexicographically first index with $L_\alpha = L_\beta$, (hypothesis, memory) of the new learner $\mathbf{Q}'$ given as $(\alpha, (\beta, \mu))$ satisfies the same length-bound.

(e) It suffices to show that **AutoIndexFEx** $\subseteq$ **AutoIndexEx**. This can be proved similarly to part (b), except that instead of simply choosing the length-lexicographically least equivalent index, one additionally pads the index so that its length is 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** $\subseteq$ **AutoItEx**. Suppose **Q** **AutoConstEx**-learns $\mathcal{L}$ using the hypothesis space $\{H_\alpha : \alpha \in I\}$ 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 $w \in \Delta$, $S \subseteq \Delta \times \Delta$.

Define $\mathbf{Q}'$, using the hypothesis space given by $\{H'_{\alpha,w,S} : \alpha \in I, w \in \Delta, S \subseteq \Delta \times \Delta\}$ 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 reflexive and 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 $x \in \mathrm{cnt}(\sigma) \cup \{\square\}$ such that $\mathbf{Q}((\alpha, y), x) = (\alpha, y')$. Thus, if (y, w) is in the reflexive and transitive closure of S, then there exists a sequence τ, with $\mathrm{cnt}(\tau) \subseteq \mathrm{cnt}(\sigma)$, 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 an n_0 and an index α with $H_\alpha = L$ such that, for all $n \geq n_0$, $\alpha_n = \alpha$. This holds because, by the previous analysis, there exists a suitably modified text for L on which **Q** converges 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 the reflexive and 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 $\mathcal{L}$. ∎

Note that the above theorem (along with its proof) also holds if we require class preserving learning in all the cases, that is, if the hypothesis space used by the learners is class preserving.

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 17.

The following theorem holds even when one considers class preserving hypothesis spaces. The diagonalisation in part (c) can be done by using the given indexing as the hypothesis space on the positive side, and any class comprising hypothesis space on the negative side.

Theorem 18. (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 definitions.

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

To **AutoWordEx** learn $\mathcal{L}$, one uses memory over the alphabet $\{0,1\}^*$ and memorises all strings in L_ε seen so far. The 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 $0^{i+1} \in \mathrm{cnt}(\sigma)$, where $n = \max(\{i : 0^{i+1} \in \mathrm{cnt}(\sigma)\} \cup \{0\})$. 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 **Q** **AutoIndexEx**-learns $\mathcal{L}$. Then, let σ be such that (i) $\mathrm{cnt}(\sigma) \subseteq L_\varepsilon$ and (ii) for all $\sigma' \supseteq \sigma$ such that $\mathrm{cnt}(\sigma') \subseteq L_\varepsilon$, if $\mathbf{Q}(\sigma') = (\alpha, \mu)$ and $\mathbf{Q}(\sigma) = (\alpha', \mu')$, then $\alpha = \alpha'$. Such a σ is called the locking sequence for **Q** on L_ε. Note that there exists such a sequence σ, as **Q** **Ex**-learns $\mathcal{L}$. Now there exist τ, τ' with $\mathrm{cnt}(\tau) \cup \mathrm{cnt}(\tau') \subseteq L_\varepsilon$ such that $\mathrm{cnt}(\sigma\tau) \neq \mathrm{cnt}(\sigma\tau')$, and $\mathbf{Q}(\sigma\tau) = \mathbf{Q}(\sigma\tau')$. The existence of such τ and τ' follows from the fact that the memory of $\mathbf{Q}(\sigma\tau)$ has only finitely many possibilities, even though $\mathrm{cnt}(\sigma\tau)$ takes infinitely many possibilities.

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

Note that the class $\mathcal{L}$ used in Theorem 18(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 19. *The following problems are currently open:*
(a) *Is* **AutoEx** = **AutoWordEx***?*
(b) *Is* **AutoIndexEx** $\subseteq$ **AutoWordEx***?*
(c) *Is* **AutoIndexEx** $\subseteq$ **AutoItEx***?*

If the alphabet is unary, then every **AutoEx**-learner can be replaced by an **AutoWordEx**-learner which answers (a) and (b) above in the affirmative for this special case. Also, note that the separation in Theorem 18 (c) is witnessed by a family of languages defined over unary alphabet.

Theorem 20. *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 iff* $\mathcal{L}$ *satisfies Angluin's tell-tale condition.*

Proof. ($\Leftarrow$) This follows from Fact 8.

($\Rightarrow$) This proof is similar to the proof of Theorem 9. Suppose $\mathcal{L}$ is $\{L_\alpha : \alpha \in I\}$, where I is the set of indices. The learner codes into memory, using alphabet $\{0, 1\}$, all the strings seen so far. The memory of the learner after having seen input σ is a word $z = z(0)z(1)\dots z(n)$ such that $z(i) = 1$ iff $0^i \in \mathrm{cnt}(\sigma)$, where $n = \max(\{|w| + 1 : w \in \mathrm{cnt}(\sigma)\} \cup \{0\})$. Then, on any input σ, the learner searches 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 $\beta \in I$, $[\beta \leq_{ll} \alpha \wedge \mathrm{cnt}(\sigma) \subseteq L_\beta \Rightarrow L_\alpha \subseteq L_\beta]$.

The learner then outputs length-lexicographically least such α, if any; otherwise, the learner outputs ?. Note that the above learner is automatic, as $\mathrm{cnt}(\sigma)$ can be obtained using the memory and the new input element. Furthermore, the size of α as above is bounded by the size of the largest element in σ plus a constant: The reason is that the memory is not longer than the longest word seen so far and that the hypothesis is computed by an automatic function from the memory and the current datum. Now, similarly to the proof of Theorem 9, it can be shown that the above learner **AutoWordEx**-learns $\mathcal{L}$. The theorem follows. ∎

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

Remark 21. If one were to consider not an automatic class, but just a subclass $\mathcal{L}$ of an automatic class $\mathcal{K}$, then one could solve some of the open problems mentioned above.

For example, there is a class $\mathcal{L} \subseteq \mathrm{powerset}(\{0\}^*)$ 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 $\mathcal{L}$ can be an **AutoWordEx**-learner.

Here is a proof-sketch of this fact. Let $k(0), k(1), \dots$ be a recursive one-one enumeration of K, the halting problem. The class consists of all sets $L_n = \{0^m : m \geq n\}$ 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 $n \in K - \{k(0), k(1), \dots, k(r)\}$.

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 [17]. 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 $n \in \{k(0), k(1), \dots, k(h(n) - n)\}$, 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 9 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 questions 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 22. *There exists an automatic $\mathcal{L}$ such that*
(a) *$\mathcal{L}$ is* **AutoItEx** *learnable using a class preserving hypothesis space;*
(b) *$\mathcal{L}$ 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

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

We first show that $\mathcal{L}$ can be **AutoItEx**-learnt. We use the following hypothesis space:

- $H_{\varepsilon,\varepsilon} = L_\varepsilon$,
- $H_{2,2} = \{0,1,2\}^*$,
- for $y, z \in \{0,1\}^+$ with $|y| = |z|$ and $y \leq_{ll} z$, $H_{y,z} = \{0,1,2\}^*$ and
- for $y \in \{0,1\}^+$, $H_{y,2} = L_y$.

Thus, the hypothesis space used is $\{H_\alpha : \alpha \in J\}$, where $J = \{(\varepsilon,\varepsilon),(2,2)\} \cup \{(y,2),(y,z) : y,z \in \{0,1\}^+, y \leq_{ll} z, |y| = |z|\}$. Below, let $\mathrm{succ}(w)$ denote $\mathrm{succ}_{\{0,1\}^*}(w)$, the length-lexicographic least string w' in $\{0,1\}^*$ such that $w <_{ll} w'$. We now define the iterative learner $\mathbf{P}$. If the learner ever sees the input 02 then it outputs $(2,2)$ and never changes its mind thereafter. Besides the above case, the learner starts with the conjecture $(\varepsilon,\varepsilon)$. If it ever sees 2^i, for some $i > 0$, in the input, then it continues with conjectures of the form (y,z), where $|y| = |z| = i$ and initially $y = z = 0^i$. Intuitively, a conjecture of the form (y,z) (with $|y| = |z| > 0$) 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 $\mathrm{succ}(z)$. If the learner sees an extension of $\mathrm{succ}(z)$, then it will update z to $\mathrm{succ}(z)$. 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,

- $\mathbf{P}(\lambda) = (\varepsilon, \varepsilon)$.
- $\mathbf{P}((\varepsilon, \varepsilon), 02) = \mathbf{P}((y, 2), 02) = \mathbf{P}((y, z), 02) = \mathbf{P}((2, 2), w) = (2, 2)$, for all $w \in \{0, 1, 2\}^*$, $y, z \in \{0, 1\}^+$, $|y| = |z|$ and $y \leq_{ll} z$.
- $\mathbf{P}((\varepsilon, \varepsilon), w) = (\varepsilon, \varepsilon)$, if $w \notin \{2^i : i > 0\} \cup \{02\}$.
- For $i > 0$, $\mathbf{P}((\varepsilon, \varepsilon), 2^i) = (0^i, 0^i)$.
- 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 $\text{succ}(z)$ 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 $\text{succ}(z)$ 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 $w \in \{0, 1\}^*$ and w is an extension of $\text{succ}(z)$ and $\text{succ}(z)$ 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 $\text{succ}(z)$ and $\text{succ}(z)$ 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 $\text{succ}(z)$).
- $\mathbf{P}((y, 2), w) = (y, 2)$, for $w \neq 02$.

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

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

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

(i) m is greater than the length of any string in $\text{cnt}(\sigma)$;
(ii) each of τ' and τ'' is of length m and contains m distinct strings from $\{0, 1\}^m$,
(iii) $\text{cnt}(\tau') \neq \text{cnt}(\tau'')$ and
(iv) $\mathbf{Q}(\sigma\tau') = \mathbf{Q}(\sigma\tau'')$.

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^{2m+s} possibilities for $\mathbf{Q}(\sigma\tau'''')$, for some constant c where τ'''' is a sequence of length m from $\{0, 1\}^m$ and $s = |\sigma|$ (see Proposition 15). 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 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 restrictive than requiring only one of them.

Theorem 23. *There exists an automatic $\mathcal{L}$ such that*
(a) *$\mathcal{L}$ is consistently* **AutoEx** *learnable using $\mathcal{L}$ as the hypothesis space;*
(b) *$\mathcal{L}$ is* **AutoItEx** *learnable using $\mathcal{L}$ as the hypothesis space;*
(c) *$\mathcal{L}$ is not consistently* **AutoItEx** *learnable.*

Proof. Let $\Sigma = \{0, 1, 2\}$. Let $\mathcal{L} = \{L_\alpha : \alpha \in \{\varepsilon\} \cup \{0, 1\}^*1\}$ where $L_\varepsilon = \{0, 1\}^*$ 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 memorise 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, $\mathbf{P}(\sigma\tau)$ is an index for L_α, for some $\tau \in \mathrm{seq}(L_\alpha)$. But, then $\mathbf{P}$ is not consistent on $\sigma \diamond \alpha 1 \diamond \tau$, as $\alpha 1 \notin L_\alpha$. ∎

Remark 24. Note that the class from Theorem 23 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 $\mathbf{P}(\sigma \diamond 2 \diamond \tau) \neq \mathbf{P}(\sigma \diamond 2)$. But from the locking sequence property of σ, it 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 $\mathrm{cnt}(\sigma \diamond 2)$. 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 makes the consistency problem easier to handle. Similar phenomenon for monotonic learning (for recursive learners) has been observed by Lange and Zeugmann [26].

Theorem 25. *There exists an automatic class $\mathcal{L}$ such that*
(a) *$\mathcal{L}$ is* **AutoItEx**-*learnable using a class preserving hypothesis space;*
(b) *$\mathcal{L}$ is consistently* **AutoItEx**-*learnable using some class comprising hypothesis space for $\mathcal{L}$;*
(c) *$\mathcal{L}$ is not consistently* **AutoEx**-*learnable using any class preserving hypothesis space for $\mathcal{L}$.*

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

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

One can verify that the learner $\mathbf{P}$ given in the proof of Theorem 22 **AutoItEx**-learns $\mathcal{L}$ using a class comprising hypothesis space. This learner is consistent on $\mathcal{L}$.

To see **AutoItEx** learnability using a class preserving hypothesis space, one can use the learner $\mathbf{P}$ in the proof of Theorem 22, but for $y, z \in \{0,1\}^*$, $|y| = |z|$, we define $H_{2,2}$ and $H_{y,z}$ to be L_ε instead of $\{0,1,2\}^*$ (in particular, we do not need to use $H_{2,2}$).

To show that no learner using a 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 $\{H_\alpha : \alpha \in J\}$.

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

Let τ', τ'' and m be such that (i) m is greater than the length of any string in $\mathrm{cnt}(\sigma)$, (ii) each of τ', τ'' is of length m and contains m distinct strings from $\{0,1\}^m$, (iii) $\mathrm{cnt}(\tau') \neq \mathrm{cnt}(\tau'')$ and (iv) $\mathbf{Q}(\sigma\tau') = \mathbf{Q}(\sigma\tau'')$. 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^{2m+s} possibilities for $\mathbf{Q}(\sigma\tau'''')$, for some constant c where τ'''' is a sequence of length m from $\{0,1\}^m$ and $s = |\sigma|$ (see Proposition 15). Suppose $y, y' \in \{0,1\}^m$ are such that $y \in \mathrm{cnt}(\tau') - \mathrm{cnt}(\tau'')$ and $y' \in \mathrm{cnt}(\tau'') - \mathrm{cnt}(\tau')$. 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 texts [31]. In the following, it is investigated to which extent this result transfers to automatic learners.

Definition 26. [31] Let Σ be an alphabet. Let $T \in (\Sigma^* \cup \{\square\})^\omega$. We say that T is *fat* iff for every $x \in \mathrm{cnt}(T)$ and $n \in \mathbb{N}$, there exists a $k \geq n$ such that $T(k) = x$. For $L \subseteq \Sigma^*$, we let $\mathrm{ftxt}(L) = \{T \in \mathrm{txt}(L) \ : \ T \text{ is fat}\}$.

Definition 27. 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 texts* iff for every $L \in \mathcal{L}$ and every $T \in \mathrm{ftxt}(L)$, there exists an $n \in \mathbb{N}$ and an $\alpha \in J$ with $H_\alpha = L$ such that, for every $m \geq n$, $\mathbf{P}(T[m]) = \alpha$. The other learning criteria considered in this paper are similarly adapted to fat texts.

Corollary 30 to the proof of the following theorem shows that fat texts allow one to iteratively automatically learn any class which is potentially learnable, that is, which satisfies Angluin's tell-tale condition.

Theorem 28. *Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ be an automatic class. Then $\mathcal{L}$ is* **AutoWordEx**-*learnable from fat texts using the given hypothesis space $\{L_\alpha : \alpha \in I\}$ iff $\mathcal{L}$ satisfies Angluin's tell-tale condition.*

Proof. ($\Leftarrow$) This follows from Fact 8, as for learning by recursive learners without memory constraints, **Ex**-learnability from fat texts is same as **Ex**-learnability from normal texts.

($\Rightarrow$) 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. Without loss of generality we assume that if $\emptyset \in \mathcal{L}$, then $\varepsilon \in I$ and $L_\varepsilon = \emptyset$.

We will now construct the learner $\mathbf{Q}$. We 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 $L_\alpha[x] \subseteq \mathrm{cnt}(T[m])$. 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 $\mathrm{CF}_{L_\alpha[x]} \leq_{lex} \mathrm{CF}_{L_{\alpha'}[x']} \leq_{lex} \mathrm{CF}_L$.

(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 $\mathrm{ttcow}(\alpha)$ denote the length-lexicographically least word w in L_α such that w 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 $\mathrm{ttcow}(\alpha)$ denote $\max L_\alpha$. Note that $\mathrm{ttcow}(\alpha)$ is automatic. We now define our learner $\mathbf{Q}$.

- If $\emptyset \in \mathcal{L}$, then $\mathbf{Q}(\lambda) = (\varepsilon, \varepsilon, 1)$.
- 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 is the length-lexicographically least element of L_α — at which point it outputs $(\alpha, y, 1)$, for the 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.
- $\mathbf{Q}((\alpha, x, cons), \Box) = (\alpha, x, cons)$.
- To define $\mathbf{Q}((\alpha, x, cons), y)$, for $y \neq \Box$, 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,
    else output $(\beta, y, 1)$ for length-lexicographically least β with $L_\beta[y] = L_\alpha[y] \cup \{y\}$.
  - Case 3: If $y >_{ll} x$ and $[y \notin L_\alpha$ or $x < \mathrm{ttcow}(\alpha)$ 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,
    else output $(\beta, y, 1)$ for length-lexicographically least β with $L_\beta[y] = L_\alpha[x] \cup \{y\}$.
  - Case 4: Otherwise, let $cons' = (cons \wedge y \in L_\alpha)$ and output $(\alpha, x, cons')$.

Note that the size of new hypothesis β in Case 2 and Case 3 above, if any, is bounded by the size of y plus a constant. Furthermore, it is easy to verify that the four invariants are satisfied. Also, clearly if $\emptyset \in \mathcal{L}$, then $\mathbf{Q}$ learns $\emptyset$. Now, suppose T is a fat text for a language $L = L_\beta \in \mathcal{L}$, where β is length-lexicographically minimised and $L \neq \emptyset$. Let $(\alpha_n, x_n, cons_n)$ denote $\mathbf{Q}(T[n])$. Note that by construction, except for an initial period where $\mathbf{Q}$ conjectures $\emptyset$ or ?, x_n always belongs to L. Furthermore, $\mathrm{ttcow}(\beta) \in L$.

Claim 29. *(a) 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 $y \leq x_{n'}$.*
(b) For $y = \min L$, for all but finitely many n, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$.
(c) Suppose $y < \mathrm{ttcow}(\beta)$, and for all $n \geq n_0$, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$. Then, there exists an $n_3 \geq n_0$ such that $x_{n_3} \geq \mathrm{succ}_L(y)$.
(d) For all $y \in L$ with $y \leq \mathrm{ttcow}(\beta)$, for all but finitely many n, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$.

Proof of Claim: (a) This follows from the invariants (I) and (II).

(b) Let n be least such that $T(n-1) = \min L$. By using invariant (I) and either by the first non ? hypothesis of $\mathbf{Q}$ or by the usage of Case 2 or 3 in the definition of $\mathbf{Q}$ when it receives $T(n-1)$, we have that $L[\min L] = L_{\alpha_n}[\min L]$ and $\min L \leq x_n$. Part (b) now follows from part (a).

(c) Suppose by way of contradiction that such an n_3 does not exist. Then, for all $n > n_0$, we have that $x_n = y$, and $\alpha_n = \alpha_{n_0}$, as Case 2 would not apply and an application of Case 3 would make $x_n > y$. Now, if $y < \mathrm{ttcow}(\alpha_{n_0})$, then for the least $n_1 > n_0$ such that $T(n_1 - 1) = \mathrm{succ}_L(y)$, we would have that x_{n_1} is made to be $\mathrm{succ}_L(y)$ by Case 3. On the other hand, if $y \geq \mathrm{ttcow}(\alpha_{n_0})$, 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 the least $n_3 > n_2$ such that $T(n_3 - 1) = \mathrm{succ}_L(y)$, we would have that $x_{n_3} = \mathrm{succ}_L(y)$, by Case 3.

(d) We show the statement by induction on length-lexicographic ordering of $y \in L$ with $y \leq \mathrm{ttcow}(\beta)$. By part (b), the statement holds for $y = \min L$. Suppose, the statement holds for some $y < \mathrm{ttcow}(\beta)$, $y \in L$. Then, we show it for $\mathrm{succ}_L(y)$. Let n_0 be such that, for all $n \geq n_0$, $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$. By part (c), there exists an n_3 such that $x_{n_3} \geq \mathrm{succ}_L(y)$. Now, if $\mathrm{succ}_L(y) \in L_{\alpha_{n_3}}$, then we are done by part (a), invariants (I), (II) and induction. Otherwise, for the least $n_4 > n_3$, such that $T(n_4 - 1) = \mathrm{succ}_L(y)$, we will have that $x_{n_4} = \mathrm{succ}_L(y)$ and $\mathrm{succ}_L(y) \in L_{\alpha_{n_4}}$, by Case 2 (all the intermediate steps between n_3 and n_4 will not reduce x to below $\mathrm{succ}_L(y)$, as $\mathrm{succ}_L(y)$ does not appear in between $T(n_3 - 1)$ and $T(n_4 - 1)$, and Case 2 is the only case which can reduce x_n). This proves the statement for $\mathrm{succ}_L(y)$ and completes the proof of the claim.

It follows from part (d) of the claim that for some number 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 the construction of $\mathbf{Q}$. We show that such an n must exist. So suppose $\alpha_{n_5} \neq \beta$. This means that $L[x_{n_5}] \neq L_{\alpha_{n_5}}[x_{n_5}]$ (by invariant (III), definition of ttcow 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 (using part (a)) that, for the least $n' > n_5$ such that $T(n'-1) = x$, $\mathbf{Q}(T[n'])$ will be $(\beta, x_{n'}, cons)$, by Case 2 and the definition of ttcow.

It follows that $\mathbf{Q}$ **AutoWordEx**-learns $\mathcal{L}$. ∎

Suppose instead of using the given hypothesis space $\{L_\alpha : \alpha \in I\}$ one uses the hypothesis space $\{H_{\alpha,x,cons} : \alpha \in I, x \in \Sigma^*, cons \in \{0,1\}\}$, where $H_{\alpha,x,cons} =$

L_α. Then the above learning algorithm **Q** becomes an iterative learner using this hypothesis space. It uses conjectures $(\alpha, x, cons)$ instead of conjecture α and memory $(x, cons)$. Note that the update rules guarantee that the learner **Q** does not update its hypothesis if $x \geq \text{ttcow}(\alpha)$, 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 yields the following corollary.

Corollary 30. *Every automatic class satisfying Angluin's tell-tale condition is* **AutoItEx**-*learnable from fat texts using a class preserving hypothesis space.*

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

Theorem 31. *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 texts, class comprisingly conservatively* **AutoItEx** *learnable from normal texts, but neither conservatively iteratively learnable from fat texts using a one-one class preserving hypothesis space nor iteratively learnable from fat texts using a one-one class preserving hypothesis space.*

Proof. Intuitively, $g(0, \cdot)$ and $g(1, \cdot)$ are 1–1 computable functions such that $\{g(0, n) : n \in \mathbb{Z}\}$ and $\{g(1, n) : n \in \mathbb{Z}\}$ partition the set of natural numbers. Furthermore, from $0^{g(i,j)}$ and $0^{g(i',j')}$ one can automatically determine whether $i = 0$ or $i = 1$, whether $i' = 0$ or $i' = 1$, whether $j < j'$ and whether $j' < j$ (this later property is needed for **P** below to be automatic).

For conservative **AutoItEx**-learning using a class comprising hypothesis space, the hypothesis space used is:

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

where, for $n_0, n_1 \in \{\varepsilon\} \cup \mathbb{Z}$, $h(a, b, c)$ is the convolution of a, b' and c' with $b' = 0^{g(0,b)}$ if $b \neq \varepsilon$, $b' = 1$ if $b = \varepsilon$; $c' = 0^{g(0,c)}$ if $c \neq \varepsilon$ and $c' = 1$ if $c = \varepsilon$. The learner **P** initially conjectures $h(\varepsilon, \varepsilon, \varepsilon)$. We mention below the cases when **P** modifies its conjecture. In all other cases, the conjectures are not modified. Intuitively, conjectures of the form $h(\cdot, j, \cdot)$, (respectively $h(\cdot, \cdot, j)$) imply that a string of the form $0^{g(0,j)}1^k$ (respectively, a string of the form $0^{g(1,j)}1^k$) has been seen in the input.

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

One can verify that $\mathbf{P}$ conservatively **AutoItEx**-learns $\mathcal{L}$.

For class preserving **AutoItEx**-learning, one just modifies the above hypothesis space to have $H_{h(\varepsilon,n_0,n_1)} = L_{0^{g(0,0)}}$ 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 texts using a one-one class preserving hypothesis space nor iteratively learnable from fat texts using a one-one class preserving hypothesis space, note the following: an iterative learner that uses a one-one class preserving hypothesis space is conservative. So it suffices to show that no conservative learner that uses a one-one class preserving hypothesis space iteratively learns $\mathcal{L}$ from fat texts.

Let $\mathbf{F}$ be any conservative iterative learner that uses a one-one 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 overgeneralised 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 $H_{\mathbf{F}(x \diamond y_1 \diamond y_2 \diamond \ldots \diamond y_k)} = L_{0^{g(1,n')}}$, where $H_{\mathbf{F}(x \diamond y_1 \diamond y_2 \diamond \ldots \diamond y_r)} = L_{0^{g(0,s_r)}}$, for $r < k$ and some s_r, where s_r's are distinct and different from n. Then, we have that $y_1, y_2, \ldots, y_k$ must be of form $0^{g(0,\cdot)}1^*$, since the learner is conservative and uses a one-one class preserving hypothesis space. Also, note that x is of the form $0^{g(0,\cdot)}1^*$ as $x \notin L_{0^{g(0,n-1)}}$. Thus $x, y_1, \ldots, y_k$ belong to $L_{0^{g(1,n'-1)}}$ and thus $\mathbf{F}$ overgeneralises and cannot conservatively learn $\mathcal{L}$. ∎

Theorem 32. *Suppose an automatic iterative learner* **Ex**-*identifies* $\mathcal{L}$ *using a class preserving hypothesis space. Then, there is an automatic conservative and iterative learner* $\mathbf{M}'$ *which identifies* $\mathcal{L}$ *from fat texts.*

Proof. Suppose $\mathbf{M}$ is an automatic iterative learner which **Ex**-identifies $\{L_\alpha : \alpha \in I\}$ from fat texts using the hypothesis space $\{L_\alpha : \alpha \in I\}$. Without loss of generality assume that initial conjecture of $\mathbf{M}$ is ?.

Let $S = \bigcap_{\alpha \in I} L_\alpha$. For $\alpha \in I \cup \{?\}$, let $mc(\alpha) = 1$ if $\alpha = ?$ or there exists an $x \in L_\alpha$ such that $\mathbf{M}(\alpha, x) \neq \alpha$; otherwise $mc(\alpha) = 0$.

If $S \in \mathcal{L}$, then let $e_0 = 0$ and $H_{e_0} = S$, else let $e_0 = ?$.

For $\alpha \in I \cup \{?\}$ and $w \in \Sigma^* - S$, let $H_{\alpha,w} = L_\alpha$, if $mc(\alpha) = 0$; otherwise, let $H_{\alpha,w} = L_{\alpha'}$, where α' is length-lexicographically least such that $w \notin L_{\alpha'}$.

Define $\mathbf{M}'$ as follows, where $\mathbf{M}'$ uses hypothesis space $\{H_e : e \in J\}$, where $J = \{(\alpha, w) : \alpha \in I \cup \{?\}, w \in \Sigma^* - S\} \cup \{e_0\} - \{?\}$.

Initially, $\mathbf{M}'(\lambda) = e_0$.
$\mathbf{M}'(e_0, x) = e_0$, for $x \in S \cup \{\#\}$.
$\mathbf{M}'(e_0, x) = (\mathbf{M}(?, x), x)$, for $x \notin S \cup \{\#\}$.
$\mathbf{M}'((\alpha, w), x) = (\mathbf{M}(\alpha, x), w)$.

Now, suppose T is the input fat text for a language $L \in \mathcal{L}$. If $L = S$, then clearly, $\mathbf{M}'$ identifies L. Otherwise, let n be least such that $T(n) \notin S$. Let T' be obtained from T by deleting $T[n]$, that is $T'(m) = T(n+m)$. Let $w = T'(0) = T(n)$. Then, it is easy to see that T' is still a fat text for L. Furthermore, for all $m \geq 0$, $\mathbf{M}'(T[n+m+1]) = (\mathbf{M}(T'[m+1]), w)$. Also, note that $\mathbf{M}$ converges on T' to α such that $mc(\alpha) = 0$ (otherwise, due to T' being fat text, either $\mathbf{M}$ does not identify L or makes a further mind change on T'). Thus, $\mathbf{M}'$ converges on T to (α, w) and $H_{\alpha,w} = L_\alpha$. Furthermore, $\mathbf{M}'$ is conservative as on previous conjecture (α', w) and input x, if $\mathbf{M}'((\alpha', w), x) \neq (\alpha', w)$, then either $mc(\alpha') = 1$ (and thus $w \notin H_{\alpha',w}$, but w belongs to the input seen so far) or $mc(\alpha') = 0$ (and thus $x \notin L_{\alpha'} = H_{\alpha',w}$). ∎

Remark 33. *Suppose one uses the following modified definition of conservativeness:* $\mathbf{M}$ *is conservative if for any* σ *and* x*, if* $x \in H_{\mathbf{M}(\sigma)}$*, then* $\mathbf{M}(\sigma x) = \mathbf{M}(\sigma)$*. Then the class used in Theorem 31 cannot be learnt by any conservative and iterative learner from fat texts using a class preserving hypothesis space; the diagonalisation proof as given in Theorem 31, works for this case also.*

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

Proposition 34. *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 texts) using the hypothesis space* $\{L_\alpha : \alpha \in I\}$*, then* $\{L_\alpha : \alpha \in I\}$ *is also* **AutoItEx***-learnable (from fat texts) 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. ∎

The next theorem shows that every automatic class (even those that may not satisfy Angluin's tell-tale condition) is partially learnable from fat texts by an automatic iterative learner. This corresponds to the result by [31] that the whole class of all recursively enumerable languages is partially learnable by some recursive learner.

Theorem 35. *Every automatic $\mathcal{L}$ is* **AutoWordPart***-learnable from fat texts.*

Proof. This is a modification of the proof of Theorem 28. 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. Without loss of generality we assume that if $\emptyset \in \mathcal{L}$, then $\epsilon \in I$ and $L_\varepsilon = \emptyset$.

We now construct the learner $\mathbf{Q}$. 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 $L_\alpha[x] \subseteq \mathrm{cnt}(T[m])$. 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 $\mathrm{CF}_{L_\alpha[x]} \leq_{lex} \mathrm{CF}_{L_{\alpha'}[x']} \leq_{lex} \mathrm{CF}_L$.

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

- If $\emptyset \in \mathcal{L}$ then $\mathbf{Q}(\lambda) = (\varepsilon, \varepsilon)$.
- 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.
- $\mathbf{Q}((\alpha, x), \square) = (\alpha, x)$.
- To define $\mathbf{Q}((\alpha, x), y)$, for $y \neq \square$, 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).

Note that the size of new hypothesis β in Cases 1 to 3 above, if any, is bounded by the size of y plus a constant. Furthermore, it is easy to verify that the invariants are satisfied. Clearly, if $\emptyset \in \mathcal{L}$, then $\mathbf{Q}$ learns $\emptyset$. So suppose $L \neq \emptyset$ and $L = L_\beta \in \mathcal{L}$, where β is length-lexicographically minimised. Suppose T is a fat text for L. Let (α_n, x_n) denote $\mathbf{Q}(T[n])$.

Claim 36. *(a) For all n, 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 $y \leq x_{n'}$.*

(b) For all $y \in L$, there exists an n such that $L[y] = L_{\alpha_n}[y]$ and $y \leq x_n$.

Proof of claim. Part (a): This follows using invariants (I) and (II).

Part (b): Clearly, $y = \min L$ satisfies part (b), 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 the usage of Case 1 or 2 in definition of $\mathbf{Q}$ when it receives $T(n-1)$).

Now suppose part (b) holds for some $y \in L$. Then we show that it holds for $\mathrm{succ}_L(y)$. Let n_0 be large enough such that for all $n \geq n_0$, $L[y] = L_{\alpha_n}[y]$ and $x_n \geq_{ll} y$. Let $n'' > n_0$ be such that $T(n''-1) = \mathrm{succ}_L(y)$. Then, $L_{\alpha_{n''}}[\mathrm{succ}_L(y)] = L[\mathrm{succ}_L(y)]$ and $x_{n''} \geq_{ll} \mathrm{succ}_L(y)$ (Case 1 and 2 both will ensure this, if it is not already true). This completes the proof of the claim.

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

Now suppose L is infinite. As $\mathrm{CF}_{L_{\alpha_n}[x_n]}$ converges to CF_L from below, it follows that no α with $L_\alpha \neq L$ would be output infinitely often. Furthermore, no index which is not length-lexicographically minimal index for some 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, using the claim above and as $\mathrm{CF}_{L_{\alpha_n}[x_n]}$ converges to CF_L from below, for large enough n, for all $n' \geq n$, $x_{n'} \geq_{ll} x$ and $\mathrm{CF}_{L[x]} = \mathrm{CF}_{L_{\alpha_{n'}}[x]}$. 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 the definition of $\mathbf{Q}$). Thus, by Case 1, using invariant (I), for any $n'' > n'$ such that $T(n''-1)$ is 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. ∎

6. Conclusion

The present work initiates the investigations of the learnability of automatic classes and also the notion of automatic learners. Such learners are restrictive when they have to learn from all texts; only if they are fed with fat texts where each data item-occurs infinitely often they can explanatorily learn all automatic

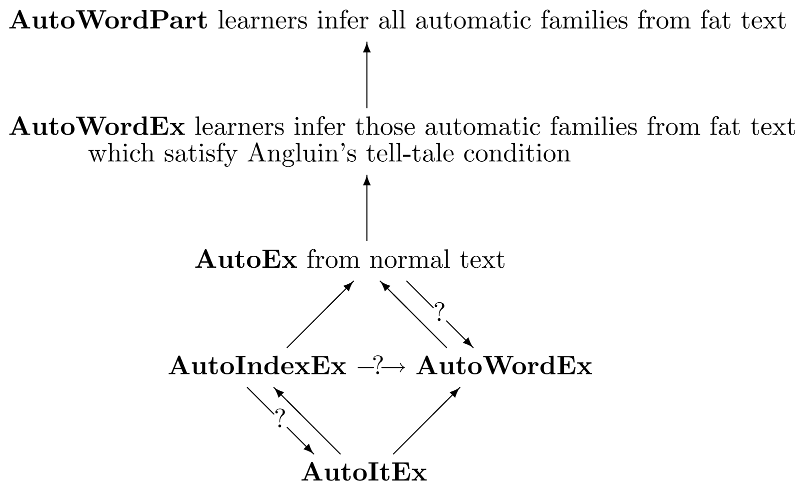

Figure 1: Major Inclusions. Solid arrows denote inclusion. Arrows with a question mark denote that the inclusion is open. If an inclusion does not follow by using reflexive and transitive closure of any of these two types of arrows, then it does not hold.

classes which satisfy Angluin's tell-tale criterion. Furthermore, partial automatic learners can infer all automatic families from fat text. Figure 1 gives the most important inclusions found — note that all notions except for the two topmost ones learn from normal texts. Several implications linked to memory are neither proven nor disproven. For example, is there a class which can be learnt by an automatic learner using hypothesis sized memory which cannot be learnt by an iterative automatic learner? Furthermore, is restricting the memory to the size of the longest word seen so far a real restriction in automatic learning? Besides these fundamental questions, we also studied the amount of restrictions given by consistency and conservativeness. While in standard inductive inference, the undecidability of membership problem with respect to the hypotheses is the main reason for inconsistent learners being more powerful, in automatic learning, the main reason that inconsistent learners might be more powerful than consistent ones are the implicit and explicit memory restrictions during the learning process which make it impossible to keep track of all the data observed so far.

Acknowledgments. We would like to thank John Case, Henning Fernau, Pavel Semukhin, Trong Dao Le and Thomas Zeugmann for discussions about the subject of learning classes with automatic indexings. We thank Trong Dao Le for pointing out an error in an earlier version of Theorem 31. We also thank the anonymous referees for several helpful comments.

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. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.

[4] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[5] Janis Bārzdiņš. Two theorems on the limiting synthesis of functions. *Theory of Algorithms and Programs* 1:82–88, 1974.

[6] Janis Bārzdiņš. Inductive inference of automata, functions and programs. *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–460, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107–112, 1977.

[7] Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[8] Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.

[9] Achim Blumensath and Erich Grädel. Automatic structures. *15th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 51–62, IEEE Computer Society, 2000.

[10] John Case. The power of vacillation in language learning. *SIAM Journal of Computing*, 28:1941–1969, 1999.

[11] John Case, Sanjay Jain, Trong Dao Le, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic Learning of Subclasses of Pattern Languages. *5th International Conference on Language and Automata Theory and Applications* (LATA 2011). To appear.

[12] John Case, Sanjay Jain, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic learners with feedback queries. *Models of Computation in Context, Seventh Conference on Computability in Europe* (CiE), Sofia University, Bulgaria, 2011. To appear.

[13] François Denis, Aurélien Lemay and Alain Terlutte. Some classes of regular languages identifiable in the limit from positive data. *Grammatical Inference: Algorithms and Applications, Sixth International Colloquium*, ICGI 2002. Springer LNCS 2484:63–76, 2002.

[14] 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*, 261:119-156, 2001.

[15] Henning Fernau. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711, 2003.

[16] Rusins Freivalds, Efim Kinber, Carl Smith. On the impact of forgetting on learning machines. *Journal of the ACM*, 42:1146–1168, 1995.

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

[18] Tom Head, Satoshi Kobayashi and Takashi Yokomori. Locality, reversibility, and beyond: learning languages from positive data. *Algorithmic Learning Theory, Ninth International Conference*, (ALT). Springer LNAI 1501:191–204, 1998.

[19] Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.

[20] Bernard R. Hodgson. On direct products of automaton decidable theories. *Theoretical Computer Science*, 19:331–335, 1982.

[21] Oscar H. Ibarra and Tao Jiang. Learning regular languages from counterexamples. *Proceedings of the first annual workshop on Computational Learning Theory* (COLT), pages: 371–385. Morgan Kaufmann Publishers Inc, USA, 1988.

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

[23] Michael Kearns and Leonard Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 57–71, Morgan Kaufmann Publishers Inc., 1989.

[24] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity*, (International Workshop LCC 1994). Springer LNCS 960:367–392, 1995.

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

[26] Steffen Lange and Thomas Zeugmann. Language learning in dependence on the space of hypotheses. *Proceedings of the Sixth Annual Conference on Computational Learning Theory* (COLT), pages 127–136. ACM Press, 1993.

[27] Steffen Lange and Rolf Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.

[28] Steffen Lange, Thomas Zeugmann and Shyam Kapur. Characterizations of monotonic and dual monotonic language learning. *Information and Computation*, 120:155–173, 1995.

[29] Yasuhito Mukouchi. Characterization of finite identification. *Proceedings of the Third International Workshop on Analogical and Inductive Inference* (AII), Springer LNAI, 642:260–267, 1992.

[30] Daniel Osherson, Michael Stob and Scott Weinstein, Learning strategies. *Information and Control*, 53:32–51, 1982.

[31] 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.

[32] Sasha Rubin. *Automatic Structures.* Ph.D. Thesis, University of Auckland, 2004.

[33] Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.

[34] Gisela Schäfer-Richter. *Uber Eingabeabhangigkeit und Komplexitat von Inferenzstrategien.* PhD thesis, RWTH Aachen, 1984.

[35] Takeshi Shinohara. Rich Classes inferable from positive data: length–bounded elementary formal systems. *Information and Computation*, 108:175–186, 1994.

[36] Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition.* MIT Press, 1980.

[37] Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik* (Journal of Information Processing and Cybernetics), 12:93–99, 1976.