Automatic Learners with Feedback Queries

John Case[1], Sanjay Jain[⋆,2], Yuh Shin Ong[2], Pavel Semukhin[⋆⋆,3] and Frank Stephan[⋆⋆⋆,2,4]

[1] Department of Computer and Information Sciences,
University of Delaware, Newark, DE 19716-2586, USA.
case@cis.udel.edu
[2] Department of Computer Science, National University of Singapore,
Singapore 117417, Republic of Singapore.
sanjay@comp.nus.edu.sg and yuhshin@gmail.com
[3] Department of Computer Science, University of Regina,
Canada
pavel@semukhin.name
[4] Department of Mathematics, National University of Singapore,
Singapore 119076, Republic of Singapore.
fstephan@comp.nus.edu.sg

Abstract. Automatic classes are classes of languages for which a finite automaton can decide whether a given element is in a set given by its index. The present work studies the learnability of automatic families by automatic learners which, in each round, output a hypothesis and update a long term memory, depending on the input datum, via an automatic function, that is, via a function whose graph is recognised by a finite automaton. Many variants of automatic learners are investigated: where the long term memory is restricted to be the just prior hypothesis whenever this exists, cannot be of size larger than the size of the longest example or has to consist of a constant number of examples seen so far. Furthermore, learnability is also studied with respect to queries which reveal information about past data or past computation history; the number of queries per round is bounded by a constant. These models are generalisations of the model of feedback queries, given by Lange, Wiehagen and Zeugmann.

1 Introduction

The present work carries on recent investigations of learnability properties in connection with automatic structures and automatic families [9, 18, 19]. An advantage of an automatic family over general indexed families [1, 24, 26] is that the first-order theory of automatic families, as well as of automatic structures in general, is decidable [15, 16, 22]. Here in the first-order theory, the predicates (relations) and functions (mappings) allowed are automatic. Furthermore, relations and functions that are first-order defined from other automatic relations and functions are automatic again [15, 16, 22]. Also, automatic functions are linear time computable [9]. These nice

⋆ Supported in part by NUS grant number C-252-000-087-001 and R252-000-420-112.
⋆⋆ Supported in part by NUS grant number R146-000-114-112.
⋆⋆⋆ Supported in part by NUS grant numbers R146-000-114-112 and R252-000-420-112.

properties of automatic structures make them not only a useful tool in learning theory but also in other areas such as model checking and Boolean algebras [6, 7, 22, 30, 31]. Common examples of automatic predicates from the prior literature are predicates to compare the length of strings, the lexicographic order (denoted by $\leq_{lex}$) and the length-lexicographic order (denoted by $\leq_{ll}$). Here x is *length-lexicographically less than* y iff either $|x| < |y|$ or $|x| = |y|$ and $x <_{lex} y$, where $|x|$ denotes the length of string x.

Furthermore, although the class of all regular languages is learnable using queries [4], this is not true for the case of inductive inference from positive data [1, 13]. Hence, it is worth investigating more closely which classes of regular languages are learnable from positive data and which are not. For example, Angluin [3] considered learnability of the class of k-reversible languages. These studies were later extended [11, 14]. In this context, it is useful to consider which automatic families are learnable and which are not. As noted by Jain, Luo and Stephan [18], even the class of 0-reversible languages is not automatic. However, some very nice subclasses of pattern languages [2, 9] are automatic families and learnable automatically, that is, by learners which are given using finite automata.

The underlying model of learnability we consider is inductive inference [1, 13, 21, 28] and the main changes to the standard model of inductive inference are the following two: (1) the target class of languages for learning is an automatic family [15–17, 19, 20, 22], that is, membership for the class to be learnt is recognised by a finite automaton in a uniform way; (2) the learner itself has to be automatic [18]. These learners will then be given by a function, where in each stage, the learner outputs a hypothesis and updates its long term memory based on a current input and its previous long term memory; this function has to be recognised by a finite automaton. Such learners satisfy much more realistic complexity bounds than learners which have access to the full history of all past data and computations. A further motivation for studying learners which are automatic is that in some situations (such as space exploration by robots), it may be more reasonable to have finite automata as a model rather than Turing machines. Another motivation for the work goes back to the programme of Khoussainov and Nerode [22] to find which results from computable model theory can be transferred to model theory based on finite automata.

The notion of learners with explicit bounds on the long term memory had already been studied previously in the setting of algorithmic learners [12, 23]. Such memory restrictions were considered as too restrictive. This led to enrichment of the learners by allowing feedback queries and other instruments to access some, but not all information about the past [8, 25, 33]. The present work investigates these notions for the case of automatic learners learning automatic families.

Outline of the paper. Section 2 gives the basic notation and definitions. Section 3 provides some examples to give some insight into the above definitions and notions. Section 4 provides the main results on learning with feedback queries. Theorem 10 shows that every automatic family satisfying Angluin's tell-tale condition has a feedback learner with only one query per round and with an additional long term memory permitted to be as large as the longest datum seen so far. Theorem 11 shows that there is a class which has a learner employing only one feedback query per round and without any long term memory but which does not have an automatic learner relying on long term memory only. Theorems 12 and 15 relate various memory types with feedback queries and investigate the hierarchies which result from counting the size of the bounded example memory and the number of feedback queries per round. Section 5 deals with learners using a marked memory space (see Section 2 for definition). Theorem 16 shows that the more general marked memory space of type 2 permits to learn any learnable class with a long term memory bounded by the size of the hypothesis. In contrast to this, Theorem 17 shows that such a result is not possible for a marked memory space of type 1. It is an open problem whether a learner using long term memory bounded by hypothesis size can be replaced by an iterative class-preserving learner [18]. Theorem 18 gives a partial answer: this is possible if the iterative learner has additionally access to a marked memory space of type 1.

2 Learning with Feedback and Memory Limitations

The symbol $\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. Given two strings $x = x(0)x(1) \ldots x(n-1)$ and $y = y(0)y(1) \ldots y(m-1)$, over the alphabet Σ, we define the convolution [22], *conv*(x, y), over the alphabet $(\Sigma \cup \{\diamond\})^2$ as follows (where $\diamond \notin \Sigma$). Let $p = \max\{n, m\}$, and $x' = x\diamond^{p-n}$, and $y' = y\diamond^{p-m}$. Then, $conv(x, y) = (x'(0), y'(0))(x'(1), y'(1)) \ldots (x'(p-1), y'(p-1))$. Similarly, one can define *conv* on multiple arguments. A relation (predicate) R or a function f is called *automatic* if the set $\{\mathrm{conv}(x_1, x_2, \ldots, x_n) : R(x_1, x_2, \ldots, x_n)\}$ and $\{\mathrm{conv}(x_1, x_2, \ldots, x_m, y) : f(x_1, x_2, \ldots, x_m) = y\}$, respectively, are regular.

A family of languages, $\{L_\alpha : \alpha \in I\}$ is said to be *automatic* [22] iff (a) I (called the index domain) is regular, (b) there is a regular set D (called the domain) such that each $L_\alpha \subseteq D$ and (c) the set $\{\mathrm{conv}(\alpha, x) : \alpha \in I \wedge x \in D \wedge x \in L_\alpha\}$ is regular. Here D and I are sets of strings over some finite alphabet.

Automatic structures are structures given by finitely many automatic relations and functions — where these structures can also be considered in a more general sense, when they are just isomorphic to a collection of finitely many automatic predicates and functions with corresponding regular domains.

Fix a domain $D \subseteq \Sigma^*$, where Σ is a finite alphabet. Let $\# \notin \Sigma$. A text is a mapping from $\mathbb{N}$ to $D \cup \{\#\}$. We let $T[n]$ denote $T(0)T(1) \ldots T(n-1)$. Content of a text T, denoted *content*(T), is $\{T(i) : i \in \mathbb{N}\} - \{\#\}$. Sequences are initial segments of texts. Content of a sequence $\sigma = T[n]$, denoted *content*(σ), is $\{T(i) : i < n\} - \{\#\}$. Intuitively, # denotes pauses in the presentation of data. A text T is for a language L iff *content*$(T) = L$.

We will be considering learning of an automatic family $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ by a learner using hypothesis space $\mathcal{H} = \{H_\beta : \beta \in J\}$, where $\mathcal{L}$ and $\mathcal{H}$ are automatic families, with I and J being regular sets and languages L_α, H_β being subsets of a regular set $D \subseteq \Sigma^*$, for Σ being a finite alphabet.

A learner is an algorithmic device mapping $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to $\Gamma^* \times (J \cup \{?\})$, where Γ is a finite alphabet. Intuitively, members of Γ^* represent the long term memory of the learner. Furthermore, ? represents that the learner repeats its previous hypothesis.

The basic model of learning [13] is given as follows. Fix an input text T for a target language L. The learner has initial memory $mem_0 \in \Gamma^*$ and initial hypothesis $\beta_0 \in J \cup \{?\}$. In stage n, the learner receives the input $T(n)$, updates its previous memory mem_n to mem_{n+1} and outputs a hypothesis β_{n+1}. For general learners as studied by Gold [13], there is no restriction on the learners except for the mapping $(mem_n, T(n)) \mapsto (mem_{n+1}, \beta_{n+1})$ being computable (here note that the learner does not know n, unless it stores it in its memory). The learner learns [13] a language L iff for all texts T for L, for β_n as defined above, there is an n such that (i) $H_{\beta_n} = L$, and (ii) for all $m \geq n$, $\beta_m \in \{\beta_n, ?\}$. The learner learns a class $\mathcal{L}$ of languages iff it learns each $L \in \mathcal{L}$. This model of learning is also referred to as **TxtEx**-learning.

For learning automatic families of languages a characterization based on Angluin's condition [1] determines when an automatic family is learnable (where the learner can even be made consistent, conservative and set driven, for the positive side [18]).

Proposition 1 (Based on Angluin [1]). *An automatic family $\{L_\alpha : \alpha \in I\}$ is learnable by an algorithmic learner iff, for every $\alpha \in I$, there is a bound b_α such that, for all $\beta \in I$, the implication*

$$\{x \in L_\alpha : |x| \leq b_\alpha\} \subseteq L_\beta \subseteq L_\alpha \Rightarrow L_\beta = L_\alpha$$

holds. We call the set $\{x \in L_\alpha : |x| \leq b_\alpha\}$ a tell-tale set for L_α. This condition is called Angluin's tell-tale condition*. Note that we can take $b_\alpha = |\alpha| + c$ for a suitable constant c independent of α.*

Therefore, the challenge is to study learnability by more restrictive learners. In the setting of automatic structures, it is natural that such learners are automatic [18], that is, for which the mapping $(mem_n, T(n)) \mapsto (mem_{n+1}, \beta_{n+1})$ is automatic. Hypothesis and updated memory of such learners can be computed in time *linear in their previous memory and current datum* [9]. The price paid is that the learner can no longer access the full past history of the data observed. In general, the requirement of a learner to be automatic is a real restriction [18].

Jain, Luo and Stephan [18] had considered various ways in which the size of the long term memory of the automatic learners is bounded in length. The length-restrictions considered are as follows. For the following, T is an arbitrary input text, mem_n and β_n denotes the long term memory and hypothesis of the learner just before getting input $T(n)$.

(a) the size of the hypothesis plus a constant; that is, for some constant c (independent of T), $|mem_n| \leq |\beta_n| + c$.

(b) the size of the longest datum observed so far plus a constant;, that is, for some constant c (independent of T), $|mem_n| \leq \max\{|T(i)| : i < n\} + c$.

(c) just constant size; that is, $|mem_n| \leq c$, for some constant c.

For the ease of notation, the "plus a constant" is omitted in the notations below. Note that the learner is not constrained regarding which alphabet it uses for its memory; therefore, it might, for example, store the convolution of up to k number of examples in the case that the size of the longest datum seen so far is the memory bound (here k is some a priori fixed constant).

Note that, in the case that memory bounded by the hypothesis size is allowed, the learner can memorise the most recent hypothesis output, and, thus, abstain from outputting ?, outputting instead the stored most recent hypothesis.

As such memory limited learners are quite restrictive, it is natural to consider mechanisms which provide some access to past data, besides what can be remembered in the memory by automatic learners. If one considers *fat texts*, see [28], where every data item is repeated infinitely often in the text, then Jain, Luo and Stephan [18] showed that automatic learners are able to learn all automatic families satisfying Angluin's tell-tale condition. Hence, the present work looks at criteria which are more powerful than just limited memory structure but less powerful than fat texts. These methods are based on active strategies of the learner, such as making feedback queries about whether some data item has already been seen in the past. While the mechanisms presented here are well-studied in the case of algorithmic learners, the combination of such mechanisms with automatic learners is novel. Also novel is the generalised model of memory space, which subsumes feedback learning and related criteria.

For the following definitions, $\mathcal{L} = \{L_\beta : \beta \in I\}$ refers to the language class being learnt. L_α refers to the target language, and a text T for L_α is the input given to the learner. $\mathcal{H} = \{H_\beta : \beta \in J\}$ refers to the family used by the learner as hypothesis space. Furthermore, mem_n and β_n denote the long term memory and hypothesis of the learner just before receiving input $T(n)$. We sometimes consider memory of the learner as a set. For this, $conv(x_1, x_2, \ldots, x_r)$ represents the set $\{x_1, x_2, \ldots, x_r\}$. When we consider memory as a set, we assume that it was in sorted order: that is, $x_1 \leq_{ll} x_2 \leq_{ll} \ldots \leq_{ll} x_r$. This allows for automatic updating and testing of elements in a set. For ease of notation, we will often refer directly to the sets in these cases, rather than the representation.

Definition 2. (a) An automatic learner is called *iterative* iff, for all n, $mem_n = \beta_n$.

(b) An automatic learner is a *learner with k-bounded example memory* iff, for all n, $mem_n \subseteq content(T[n])$, number of elements in mem_n is at most k, and $mem_{n+1} \subseteq mem_n \cup \{T(n)\}$ (note that the memory of the learner here is interpreted as a set).

If k is not specified, we call the learner a bounded example memory learner.

The following notion of feedback learning for general inductive inference was first studied by Wiehagen [33] and Lange and Zeugmann [25].

Definition 3. An *automatic learner* using *k-feedback* (also called a learner which uses k feedback queries) is a learner, with an associated automatic query function Q asking k questions per round, defined as follows on input text T.

Initial memory of the learner is mem_0, initial hypothesis of the learner is β_0. Furthermore, mem_n, and β_n denote the long term memory and hypothesis of the learner just before receiving input $T(n)$.

(a) Q is an automatic mapping from $(\Gamma^*, \Sigma^* \cup \{\#\})$ to a subset of Σ^* of size k.

(b) Suppose, $Q(mem_n, T(n)) = S_n = \{y_1, y_2, \ldots, y_k\}$. Let $b_i = 1$ iff $y_i \in content(T[n])$. Then, the mapping, $(mem_n, T(n), b_1, b_2, \ldots, b_k) \mapsto (mem_{n+1}, \beta_{n+1})$ is automatic.

For the following definition, we provide an automatic learner with a different kind of memory (called marked memory space), which is a set of strings over a finite alphabet Δ. This marked memory will only grow (set inclusion wise) as the learner gets more data. Marked memory can be considered as a generalization of feedback learning as feedback learning can be simulated by Type 1 memory.

Definition 4. An *automatic learner* using *a marked memory space* is a learner, with an associated marked memory space, an automatic query function Q (asking k questions per round, for some constant k), and a marked memory space updater F defined as follows on input text T.

Initial memory of the learner is mem_0, initial hypothesis of the learner is β_0 and initial marked memory space is $Z_0 = \emptyset$. Furthermore, mem_n, Z_n and β_n denote the long term memory, the marked memory and hypothesis of the learner just before receiving input $T(n)$.

(a) Q is an automatic mapping from $(\Gamma^*, \Sigma^* \cup \{\#\})$ to a subset of Δ^* of size k.

(b) Suppose, $Q(mem_n, T(n)) = S_n = \{y_1, y_2, \ldots, y_k\}$. Let $b_i = 1$ iff $y_i \in Z_n$. Then, the mapping, $(mem_n, T(n), b_1, b_2, \ldots, b_k) \mapsto (mem_{n+1}, \beta_{n+1})$ is automatic. Furthermore, $Z_{n+1} = Z_n \cup X_{n+1}$, where

(i) for Type 1 memory space, there is an automatic function F such that $F(mem_n, T(n), b_1, b_2, \ldots, b_k) = X_{n+1}$, and

(ii) for Type 2 memory space, there is an automatic function F such that for all $w \in \Delta^*$, $F(mem_n, T(n), b_1, b_2, \ldots, b_k, w) = 1$ iff $w \in X_{n+1}$.

Note that in case of Type 1 memory space, X_n is necessarily finite (with cardinality bounded by some fixed constant), whereas for Type 2 memory, X_n may be infinite.

Remark 5. An *automatic feedback learner* is a special case of an automatic learner using a marked memory space of Type 1, for which, in Definition 4, the domain D is used for memory instead of Δ^*, and $X_{n+1} = \{T(n)\}$.

Definition 6. An *automatic learner using hypothesis queries* is a special case of a learner using a marked memory space, for which, in Definition 4, the index set J of the hypothesis space is used for memory instead of Δ^* and $X_{n+1} = \{\beta_{n+1}\}$. This allows a learner to check whether it had earlier issued a particular hypothesis.

For ease of notation, when describing hypothesis query learners, we just give the queries made by the learner at each input, rather than giving details of X_n and Z_n.

Notation. For a learner M, let $M(\sigma)$ denote the hypothesis of the learner M after having seen input segment σ. For input segments σ and τ, let $\sigma \circ \tau$ denote the concatenation of σ and τ. Furthermore, let $\sigma \circ x$ denote concatenation of σ and the sequence consisting of just the element x. For a string x, $x(i)$ denotes the $(i+1)$-th element in the string x. Thus, string x is same as $x(0)x(1)\ldots x(|x|-1)$.

Many of these learning notions had been defined earlier without requiring that the learners are automatic. The general notion of learning which is underlying the notion of an automatic learner is due to Gold [13] and is called explanatory learning. The variant with an explicit long term memory as used here was introduced by Freivalds, Kinber and Smith [12]. The special case of iterative learning is quite popular and predates the definition of general memory limitations, it was introduced by Wiehagen [33] and later by Wexler and Culicover [32]. Bounded example memory was considered by Osherson, Stob and Weinstein [28]; Lange and Zeugmann [25] extended this study. Wiehagen [33] and Lange and Zeugmann [25] introduced and studied feedback learning; Case, Jain, Lange and Zeugmann [8] quantified the amount of feedback queries per round.

For many of these types of automatic learners for automatic families, one can choose the hypothesis space $\mathcal{H}$ to be equal to $\mathcal{L}$; this may sometimes cause a restriction, for example, when the amount of the memory allowed to the learner depends on the size of the hypothesis or when the long term memory of the learner has to be the most recent hypothesis, as in the case of iterative learning. The main reason for hypothesis space not to be critical in many cases is that one can automatically convert the indices from one automatic family to another for the languages which are common to both automatic families. This stands in a contrast to the corresponding results for indexed families of recursive languages [25, 26]. A result in the present work which depends on the choice of the hypothesis space is Theorem 18. In the case that the hypothesis space does not matter, often, for the ease of notation, the languages are given in place of the indices as conjectures of the learner.

3 Some Illustrative Examples

We now provide some examples to give insight into the learning criteria considered and their properties. Example 7 shows that learnability by automatic learners cannot be characterised from the inclusion structure of a family alone, as the inclusion structure in the class of co-singleton sets is independent of the alphabet size.

Example 7. *The family of all co-singleton sets* $\{0,1\}^* - \{x\}$*, with* $x \in \{0,1\}^*$*, is automatic. It does not have an automatic learner, as such a learner cannot memorise all the data observed* [18]*. However, it can be learnt by an automatic feedback learner (using one query per round), which converges to a hypothesis for* $\{0,1\}^* - \{x\}$*, for the length-lexicographically least member* x *of* $\{0,1\}^*$ *for which the feedback query answer remains negative forever.*

In contrast, the family of all sets $\{0\}^* - \{x\}$*, with* $x \in \{0\}^*$*, has an automatic learner using memory bounded by the size of the longest datum seen so far. This is so because, automatic families defined over unary alphabet can be learnt by an automatic learner whenever they satisfy Angluin's tell-tale condition* [18]*.*

Example 8 deals with intervals of the lexicographic ordering; one could formulate similar results also with other automatic linear orderings. For the case of the lexicographic order, there is a difference between closed and open intervals.

Example 8. *The family of the closed intervals* $L_{\mathrm{conv}(x,y)} = \{z \in \{0,1\}^* : x \leq_{lex} z \leq_{lex} y\}$ *is automatic and can also be learnt by an automatic learner with* 2*-bounded example memory. Furthermore, it has an automatic iterative learner. Both learners memorise, either explicitly or implicitly by padding into the hypothesis, the lexicographically least and greatest data seen so far.*

The family of the open intervals $L_{\mathrm{conv}(x,y)} = \{z \in \{0,1\}^* : x <_{lex} z <_{lex} y\}$ *is also automatic. However, it cannot be learnt as it violates Angluin's tell-tale condition. The open interval*

$L_{\mathrm{conv}(000,1)}$ *is the ascending union of the open intervals* $L_{\mathrm{conv}(0^3,01^m)}$*; already Gold* [13] *observed that classes of this form cannot be learnt from positive data.*

The following examples show the limitations and possibilities for learners which use a marked memory space but do not have access to any long term memory. In case (a) the marked memory space can be made more concrete by just marking off which hypotheses have been issued and which data-items have been observed. Note that the family in (a) is the same as in Example 7.

Example 9. (a) *An automatic learner without any long term memory, but using at most one feedback and at most two hypothesis queries per round, can learn the class of all co-singleton subsets of* $\{0,1\}^*$.

(b) *An automatic learner, using a marked memory space of type 1 but no other memory, cannot learn the class* $\{S \subseteq \{0,1\}^* : S$ *has at most two elements*$\}$.

Proof. (a) The automatic learner for the class is the following. For $y \in \{0,1\}^*$, let $L_y = \{0,1\}^* - \{y\}$. Let $succ(x)$ be the length-lexicographic successor of x within the given domain. Assume that the input is of the form $x01^n$ for some n. Then the learner computes the y with $succ(y) = x$, if any. If y exists and queries determine that hypothesis y was conjectured previously, hypothesis x was not conjectured previously and y has been observed in the input data, then the learner conjectures hypothesis x. If y as above does not exist and queries determine that hypothesis x was not conjectured previously, then the learner conjectures x. In all other cases, the learner conjectures ? in order to signal that there is no new conjecture (that is, it repeats the previous conjecture). When learning L_x, it is easy to see that, for all $y <_{ll} x$, eventually y is conjectured, y is observed in the input and, then, $succ(y)$ is conjectured. Hence, x is eventually conjectured. The learner never conjectures $succ(x)$ as x never shows up in the input; indeed, the learner will output ? after the time it conjectures x.

(b) Suppose an automatic learner using marked memory space of type 1 (and no long term memory) learns the class $\{\{x,y\} : x <_{ll} y\}$. Now consider the text $y \circ x \circ x \circ x \circ \ldots$ being presented to the learner. Note that the learner, when reading x, can ask only constantly many questions. Thus, there are two strings $y, z >_{ll} x$ such that, among the strings queried by x, the same strings are marked on input y and input z. It follows that the learner either converges on the texts $y \circ x \circ x \circ x \circ \ldots$ and $z \circ x \circ x \circ x \circ \ldots$ to the same conjecture, or abstains from taking x into account for making its conjecture, which is thus based on y or z only, respectively. Hence, the learner fails to learn one of the languages $\{y\}, \{z\}, \{x,y\}, \{x,z\}$. $\square$

Further examples on learnable automatic families can be found in the prior literature on automatic learners [18, 19].

4 Learning with Feedback Queries

The following result shows that feedback queries together with a quite liberal long term memory permit to reach the full learning power, that is, every family satisfying Angluin's tell-tale condition can be learnt this way. Equivalently one can also say that the automatic learners with feedback are as powerful as algorithmic learners without memory limitations.

Theorem 10. *If automatic family $\mathcal{L}$ satisfies Angluin's tell-tale condition, then $\mathcal{L}$ can be learnt by an automatic learner using one-feedback query per round and a long term memory bounded by the longest word seen so far plus a constant.*

Proof. Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic family satisfying Angluin's tell-tale condition. Without loss of generality assume that for $\alpha, \alpha' \in I$ with $\alpha \neq \alpha'$, $L_\alpha \neq L_{\alpha'}$, By results from [19], there exists a constant c such that, (i) for each index α, $\{w : |w| \leq |\alpha| + c\}$ is a tell-tale set for L_α and (ii) for each $L \in \mathcal{L}$, if L is finite, then there exists an index α such that $L = L_\alpha$ and $|\alpha| \leq c + d$ where d is the length of the longest string in L. Fix such a constant c. The goal of the learner is to find an α such that

(a) the input contains the above tell-tale set for L_α and
(b) every element in the input is contained in L_α.

Intuitively, for each potential conjecture α, the learner checks if the above tell-tale set for L_α is contained in the input. If so, then it checks if every string in the input language is contained in L_α. If any of these are violated, then the learner tries the next possible conjecture α. However, potential obstacles for doing this are

- the learner may not yet have seen the tell-tale set for L_α, but these elements appear later in input;
- the learner may already have seen an element outside L_α, forgotten this fact, and the future elements to be seen are all in L_α.

To address the first problem above, each potential α will be tested more and more times until the algorithm finds the correct hypothesis. To address the second problem, feedback queries are used to check, for all strings length-lexicographically at most the largest datum seen before current conjecture α was tried, if any of them has been seen earlier but not in L_α.

The memory of the learner is of the form $\text{conv}(z, \alpha, y, b)$. Here z is the length-lexicographically largest datum seen so far, α is the current conjecture (which will be of length at most a constant plus the length of z), $b \in \{0, 1\}$ is used to remember whether the algorithm is testing clause (a) or (b) above, and y is used to remember which current string (relevant to (a) and (b) above) is being tested (the length of y will be at most the maximum of (a constant plus the length of α, the length of z)). Note that for (b), the algorithm need only use feedback for strings length-lexicographically smaller than the largest string seen up to the point at which the algorithm started testing for (b) above; rest of the strings are tested as they arrive.

Without loss of generality assume that ε is the length-lexicographically least string in D (the domain for the languages) as well as in I, the set of indices.

Initial memory of the learner is $(\varepsilon, \alpha, \varepsilon, 0)$, where α is the length-lexicographically largest index which is of size at most c. Let $succ(y), pred(y)$ denote the length-lexicographic successor and predecessor of y in domain D, and $pred(\alpha)$ denote the length-lexicographic predecessor of α in the index set I (here $pred(\varepsilon) = \varepsilon$). On input x and previous memory (z, α, y, b) the algorithm follows that of the following three cases which applies.

Case $b = 0$: (* this step checks whether the learner has already seen the tell-tale set for L_α *)

If $y \in L_\alpha$ and y has not been seen in the input so far

Then the new memory of the learner is $(z', \alpha', \varepsilon, 0)$, where z' is the length-lexicographically largest datum seen so far; if $\alpha \neq \varepsilon$, then α' is length-lexicographic predecessor of α, otherwise α' is the length-lexicographically largest index of length at most $|z'|+c$; conjecture of the learner is irrelevant in this case. (* Here the learner has not seen the tell-tale set for L_α, and thus tries the next possible α. *)

Else If y is the length-lexicographically largest string of length at most $|\alpha| + c$

Then new memory of the learner is $(z', \alpha, z', 1)$, where z' is the length-lexicographically largest string seen so far. The conjecture of the learner is L_α. (* Here the learner has seen the tell-tale set for L_α, and thus goes on to test if the input language is a subset of L_α. *)

Else let new memory of the learner be $(z', \alpha, succ(y), 0)$ and the conjecture of the learner be L_α, where z' is the length-lexicographically largest string seen so far. (* Here the learner continues checking whether the tell-tale set for L_α has been seen or not. *)

Case $b = 1$ and $y \notin L_\alpha$ but y has been seen in the input so far or $x \neq \#$ and $x \notin L_\alpha$: The new memory of the learner is $(z', \alpha', \varepsilon, 0)$, where z' is the length-lexicographically largest datum seen so far; if $\alpha \neq \varepsilon$, then α' is length-lexicographic predecessor of α, otherwise α' is the length-lexicographically largest index of length at most $|z'| + c$; conjecture of the learner is irrelevant in this case. (* Here the learner has seen a datum not belonging to L_α, and thus tries the next possible α. *)

Neither of the two cases above: The new memory of the learner is $(z', \alpha, pred(y), 1)$, where z' is the the length-lexicographically largest datum seen so far; conjecture of the learner is L_α.

Suppose the input text is for a target language L_γ. If $L_\alpha \neq L_\gamma$, then any memory of the form $(\cdot, \alpha, \cdot, \cdot)$, will eventually be updated with changed value of α as either the input does not contain the tell-tale set for L_α or the input contains an element not in L_α. Also, as length of γ is at most $|z|+c$, for the longest element z in the input, eventually it will be the case that the learner has a memory of the form $(z, \alpha, \varepsilon, 0)$, where $L_\alpha = L_\gamma$ and the input seen already contains all the elements in $\{x \in L_\alpha : |x| \leq |\alpha| + c\}$. But this implies that the learner will eventually have memory $(z', \alpha, y = z', 1)$, where z' is the longest datum seen up to that time (this happens when the learner has verified that all the elements of the above tell-tale set for L_α are present in the input). From then on the value of y decreases (until its value reaches ε) in each stage, and the learner always conjectures L_α. Thus, the learner learns the target language L_γ. □

Thus, the learners considered in the above result are as general as recursive learners (see Proposition 1). Therefore, the next results compare various more restrictive models of learning with feedback and limitations on the long term memory. First, it is shown that there are cases where feedback queries are more important than any form of long term memory.

Theorem 11. *There is a class $\mathcal{L}$ satisfying the following two statements:*

(a) *An automatic learner without any long term memory, but using one-feedback query per round, can learn $\mathcal{L}$;*

(b) *No automatic learner learns $\mathcal{L}$.*

Proof. Let $L_\varepsilon = \{0,1\}^+$ and, for all $x \in \{0,1\}^+$, let $L_x = \{y \in \{0,1\}^* : |y| \le |x|, y \ne x\} \cup \{y2^n : |y| = |x|, n > 0\}$. Let $\mathcal{L} = \{L_y : y \in \{0,1\}^*\}$.

(a) $\mathcal{L}$ can be learnt by an automatic learner using 1 feedback query. On inputs from $\{0,1\}^*$, query if ε belongs to the input seen — if not, then output L_ε; otherwise, output ?. On inputs of the form $x2^n$, $n > 0$, query whether x belongs to the input seen — if not, then output L_x else output ?. It is easy to verify that the above learner learns $\mathcal{L}$.

(b) Suppose by way of contradiction that M is automatic and learns $\mathcal{L}$. Without loss of generality assume that M always outputs a hypothesis.

Let m be so large that there are two sequences σ_1 and σ_2, each containing m elements of $\{0,1\}^m$, such that $\mathit{content}(\sigma_1) \ne \mathit{content}(\sigma_2)$ and M has the same memory and hypothesis after processing either σ_1 or σ_2. As M is automatic, such m, σ_1 and σ_2 exist (as the memory of M can be of size at most a constant times m after seeing such σ_1 or σ_2, and there are $\binom{2^m}{m}$ different subsets of $\{0,1\}^m$ of size m). Let x_1 be in $\mathit{content}(\sigma_1) - \mathit{content}(\sigma_2)$ and x_2 be in $\mathit{content}(\sigma_2) - \mathit{content}(\sigma_1)$. Let T be a text for $\{y : |y| \le m, y \ne x_1, y \ne x_2\} \cup \{y2^n : |y| = m\}$. Then, M on $\sigma_1 \circ T$ and $\sigma_2 \circ T$ converges to the same hypothesis, though these are respectively texts for L_{x_2} and L_{x_1}. $\square$

Theorem 12. *There is a class $\mathcal{L}$ with the following properties:*

(a) *$\mathcal{L}$ can be learnt by an automatic learner with* 1*-bounded example memory;*
(b) *$\mathcal{L}$ can be learnt by an automatic learner using two feedback queries per round, along with a long term memory bounded by the size of the hypothesis;*
(c) *$\mathcal{L}$ cannot be learnt by an automatic iterative learner using feedback queries.*

Proof. Let $\mathcal{L}$ consist of $L_0 = \{0\}^+$ and $L_{\mathrm{conv}(x,y)} = \{x, y\} \cup \{0^{n+1} : x(n) = 1\}$, where $x \in \{0,1\}^* \cdot \{1\}$ and $y \in \{0\}^{|x|} \cdot \{0\}^*$.

(a) The learner using the bounded example memory, on every input z and long term memory A, takes the first case below which applies:

1. $z = \#$: The learner outputs ? to signal the absence of a new conjecture and A remains unchanged.
2. $A = \emptyset$ and $z \in 0^*$: The learner conjectures L_0 and A is updated to $\{z\}$.
3. $A = \{0^n\}$ and $z = y$ for some $y \in \{0\}^*$: The learner conjectures L_0 and A is updated to $\{0^{\max\{n,|y|\}}\}$.
4. $A = \emptyset$ and $z = x$ for some $x \in \{0,1\}^* \cdot \{1\}$: The learner conjectures $L_{x,0^{|x|}}$ and A is updated to $\{x\}$.
5. $A = \{0^n\}$ and $z = x$ for some $x \in \{0,1\}^* \cdot \{1\}$ and $n < |x|$: The learner conjectures $L_{\mathrm{conv}(x,0^{|x|})}$ and A is updated to $\{x\}$.
6. $A = \{0^n\}$ and $z = x$ for some $x \in \{0,1\}^* \cdot \{1\}$ and $n \ge |x|$: The learner conjectures $L_{\mathrm{conv}(x,0^n)}$ and A is updated to $\{x\}$.
7. $A = \{x\}$ with $x \in \{0,1\}^* \cdot \{1\}$ and $z = y$ for $y \in \{0\}^*$ with $|y| > |x|$: The learner conjectures $L_{\mathrm{conv}(x,y)}$ and A remains unchanged.
8. $A = \{x\}$ with $x \in \{0,1\}^* \cdot \{1\}$ and ($z = x$ or $z = y$ for $y \in \{0\}^*$ with $|y| \le |x|$): The learner outputs ? to signal that there is no new conjecture and A remains unchanged.

It is easy to see that the learner above is automatic. Clearly, the learner succeeds to learn L_0, as the learner conjectures L_0 when the first datum appears in the input and then never changes its mind. Furthermore, when learning $L_{\mathrm{conv}(x,0^{|x|})}$, the learner will issue this hypothesis on receipt of the datum x and from then on abstain from conjecturing a new hypotheses. When learning $L_{\mathrm{conv}(x,y)}$ with $y \in \{0\}^*$ and $|y| > |x|$, there are two cases: If the learner receives the datum y at least one time after receiving x, then the learner will conjecture $\mathrm{conv}(x, y)$ on the receipt of this y; as y is the only word in $\{0\}^* \cap L_{\mathrm{conv}(x,y)}$ which is longer than x, this is the only case when such a hypothesis is issued; thus the learner learns $L_{\mathrm{conv}(x,y)}$. Otherwise the learner receives y before x and thus $A = \{y\}$ when x is received by the learner as input for the first time — at which point the learner will issue the hypothesis $L_{\mathrm{conv}(x,y)}$ and output ? from then onwards. Thus the learner learns $L_{\mathrm{conv}(x,y)}$ as well.

(b) The automatic learner using feedback queries and hypothesis size memory first conjectures L_0 and keeps one bit of information which tells whether an even or an odd number of distinct data items that have been seen so far — this can be determined by making a feedback query on the current datum.

If and when the learner sees $x \notin 0^*$ in the input, the parity bit is used in order to determine whether the parity of the number of examples observed coincides with the number of the elements in $L_{\mathrm{conv}(x,0^{|x|})}$. If so, then the learner conjectures $L_{\mathrm{conv}(x,0^{|x|})}$. Otherwise, the learner searches, using its second feedback query in each round, for an $n > |x|$ such that 0^n appears in the input (this can be done by using hypotheses (which can be stored in memory) of the form $\mathrm{conv}(x, 0^r)$, and correspondingly querying 0^r in the next round, where $r = |x| + 1, |x| + 2, |x| + 3, \ldots$).

Note that, during the above process, if a new datum, not previously seen, is observed by the learner, then the parity of the number of observed data becomes consistent with $L_{\mathrm{conv}(x,0^{|x|})}$ and the algorithm above switches back to outputting $L_{\mathrm{conv}(x,0^{|x|})}$.

(c) For an automatic iterative learner M using feedback queries, there is a locking sequence for M on L_ε, that is, there exists a σ such that the conjecture of M does not change beyond σ on any text for L_ε which starts with σ (see [5, 28]). Without loss of generality assume that σ contains at least one string. Let x be such that $L_{x,0^{|x|}} = \{x\} \cup \mathit{content}(\sigma)$. While processing the text $T = \sigma \circ x \circ x \circ x \ldots$, M asks only finitely many different feedback queries. Let n be such that $n > |x|$, n is greater than length of any element in σ and 0^n is not queried by M on the text T. Then M converges on the texts $\sigma \circ 0^n \circ x \circ x \circ x \circ x \ldots$ and $\sigma \circ x \circ x \circ x \circ x \circ x \ldots$ to the same hypothesis although these two texts are for two different languages, namely $L_{x,0^{|x|}}$ and $L_{x,0^n}$. $\square$

The following theorem gives the advantages of bounded example memory over feedback.

Theorem 13. *Let $i, j \geq 1$. Let $\mathcal{L} = \{F : |F| = i + 1\}$. Then,*

(a) Some automatic learner can learn $\mathcal{L}$ using i-bounded example memory.

(b) No learner with $i - 1$-bounded example memory and j-feedback can learn $\mathcal{L}$.

Proof. (a) The automatic learner memorises the first i distinct elements in the input. If the learner has already memorised i elements and sees a datum not in the memory, then it conjectures the corresponding set of $i+1$ elements, else it outputs ?. It is easy to verify that the above learner learns $\mathcal{L}$.

(b) Suppose by way of contradiction a learner M learns $\mathcal{L}$ using $(i-1)$-bounded example memory and j-feedback. Then, consider $x_0, x_1, \ldots, x_i \in \{0,1\}^k$ with $k > i$ such that these strings are distinct and in lexicographic ascending order and $C(x_0 x_1 \ldots x_i) \geq (i+1) \cdot k - c$ where C is the (plain) Kolmogorov complexity [27] and c is a constant depending on i but not on k such that $\binom{2^k}{i+1} \geq 2^{(i+1)\cdot k - c}$ for all $k > i$; note that there are at least $\binom{2^k}{i+1}$ many ascendingly ordered tuples of distinct $i+1$ binary strings of length k whenever $k > i$. At each time where M makes a correct conjecture on a text for $\{x_0, x_1, \ldots, x_i\}$, one can compute $x_0 x_1 \ldots x_i$ from the hypothesis of the learner, which depends only on M's current memory and M's current datum and the answers to the feedback queries. Up to an additive constant independent of k: the memory has Kolmogorov complexity at most $(i-1)\cdot k$; the current datum has Kolmogorov complexity at most k; the feedback answers have the Kolmogorov complexity at most j. It follows that $x_0 x_1 \ldots x_i$ has Kolmogorov complexity at most $i \cdot k + j + d$ where d is a constant which depends on i, j but not on k; this implies $(i+1)\cdot k - c \leq i \cdot k + j + d$ and $k \leq j + c + d$, a contradiction to the fact that k can be arbitrary large. $\square$

The following theorem gives the advantages of feedback over bounded example memory.

Theorem 14. *For $w \in \{0,1\}^n$, let $L_w = \{\{0,1\}^n \cdot 2 \cdot 0^*\} - \{w \cdot 2\}$. Let $\mathcal{L} = \{L_w : w \in \{0,1\}^+\}$. Then, $\mathcal{L}$ can be learnt using 1 feedback query. However, $\mathcal{L}$ cannot be learnt by any automatic learner.*

Proof. (a) Consider a learner which works as follows:

– On all inputs not of the form $w20^+$, the learner outputs ?;
– On input $w20^+$, the learner queries $w2$ — if the answer is no, then the learner conjectures L_w, else the learner outputs ?.

When learning L_w the learner will see infinitely often an input of the form $w20^+$ and conjecture L_w infinitely often. For other strings $v \in \{0,1\}^{|w|}$, the learner conjectures L_v only finitely often, as eventually the datum $v2$ is observed in the input and from then on the corresponding feedback query is answered negatively. Furthermore, if $|v| \neq |w|$ then no input of the form $v20^+$ is observed and therefore L_v is never conjectured. Hence there are only finitely many wrong conjectures and infinitely many correct conjectures and thus the learner learns $\mathcal{L}$ using one feedback query.

(b) Suppose by way of contradiction that M is an automatic learner which 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 $\{0,1\}^m 2$, (ii) $content(\sigma) \neq content(\sigma')$ and (iii) $M(\sigma) = M(\sigma')$. 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 M after seeing such sequences can be of length at most cm, for some constant c [18]. Let $y2$ and $y'2$ respectively be in $content(\sigma) - content(\sigma')$ and $content(\sigma') - content(\sigma)$. Let T be a text $L_y \cap L_{y'}$. Then, M on σT and $\sigma' T$ converges to the same index or diverges on both. Thus, M does not learn $\mathcal{L}$. $\square$

The following gives a class which can be learnt using either bounded example memory or feedback of size k but not of smaller size.

Theorem 15. *Let $k \geq 1$. Let $\mathcal{L} = \{F : \exists n\, [\emptyset \subseteq F \subseteq \{0^m : (k+1)n \leq m < (k+1)(n+1)\}]\}$. Then the following statements hold:*

(a) *Some automatic learner can learn $\mathcal{L}$ using k-bounded example memory;*
(b) *Some automatic learner without any long term memory can learn $\mathcal{L}$ using k feedback queries;*
(c) *$\mathcal{L}$ cannot be learnt by any automatic learner using only $k-1$ bounded example memory (where the learner does not have any memory besides the examples memorised);*
(d) *An automatic learner without any long term memory cannot learn $\mathcal{L}$ using only $k-1$ feedback queries.*

Proof. (a) The automatic learner memorises its inputs as long as it sees at most k elements. If the input element is in the memory or #, then the learner outputs ?. Otherwise (the input element is not in the memory), the learner outputs a conjecture for the set of memorised elements plus the new element seen. It is easy to verify that the above learner learns $\mathcal{L}$.

(b) On input 0^m such that $(k+1)n \leq m < (k+1)(n+1)$, the automatic learner can query the rest of the elements in $\{0^\ell : (k+1)n \leq \ell < (k+1)(n+1), \ell \neq m\}$, and output the conjecture corresponding to the elements found to be present in the input.

(c) Suppose by way of contradiction that the automatic learner learns $\mathcal{L}$ using at most $k-1$ memory elements. Let F be of minimal cardinality such that, for some input segment σ with $content(\sigma) = F$, the memory of the learner after seeing σ is a proper subset of F, where $F \in \mathcal{L}$. Suppose S_F is the memory. Let x be such that $x \notin F$ and $F \cup \{x\} \in \mathcal{L}$. Now, the learner cannot distinguish between input being $\sigma \circ x \circ x \circ x \circ x \ldots$ and $\tau \circ x \circ x \circ x \circ x \ldots$, where τ is some segment with $content(\tau) = S_F$.

(d) Suppose by way of contradiction otherwise. Let $x \in \{0\}^*$ be such that the learner on empty input does not conjecture a language containing x. Suppose n is such that $(k+1)n \leq |x| < (k+1)(n+1)$. Let y be such that, $(k+1)n \leq |y| < (k+1)(n+1)$, $y \neq x$ and the learner does not query y on input x. Then, both $\{x\}$ and $\{x, y\}$ are in $\mathcal{L}$, but the learner fails on at least one of the texts $y \circ x \circ x \circ x \circ x \circ \ldots$ and $x \circ x \circ x \circ x \circ \ldots$ to learn the corresponding language. $\square$

Parts (a) and (b) of the above theorem can be generalised to show that the class $\mathcal{L}$ can be learnt by an automatic learner which uses, for given $r \in \{0, 1, \ldots, k\}$, a bounded example memory of size r and $k - r$ feedback queries.

5 Learning using a Marked Memory Space

An automatic learner using a marked memory space of type 1 is a generalization of a learner using feedback queries. Hence it follows from Theorem 10 that every class satisfying Angluin's tell-tale condition can be learnt by a learner with marked memory space of type 1 and a long term memory bounded by the size of the largest example seen so far plus a constant. Hence, the explorations in this section target at more restricted limitations of the long term memory combined with the usage of a marked memory space.

Theorem 16. *Every automatic family satisfying Angluin's tell-tale condition has an automatic learner, with long term memory bounded by hypothesis size plus a constant, using in addition a marked memory space of type 2.*

Proof. The aim of the learner is to search for an α such that a tell-tale set for L_α is contained in the input, and the input language is contained in L_α. The idea of the proof is similar to that of Theorem 10. However a marked memory space is used instead of feedback queries. Furthermore, the long term memory needs will be less than that used in Theorem 10.

In the algorithm for the learner, one uses a marked memory space consisting of entries of the form (hypothesis, 0), (hypothesis, 1) and (datum, 2). The algorithm has two variables, α ranging over the indices and y ranging over possible words. Let $succ(\alpha)$ denote the length-lexicographic successor of index α and $succ(y)$ denote the length-lexicographic successor of datum y within the respective domains. Without loss of generality, suppose ε is the length-lexicographically least possible values for both indices and data.

The long term memory holds the hypothesis α and the pointer y. Note that y is only used to test whether all elements of a tell-tale set are there in the input seen so far. Recall from Proposition 1 that there is a constant c such that the strings in L_α of length at most $|\alpha| + c$ forms a tell-tale set for L_α. Hence, the size of the long term memory $\mathrm{conv}(\alpha, y)$ is bounded by the size of the hypothesis α plus a constant. Furthermore, x refers to the current datum, but x will not be memorised in the long term memory and discarded when reading the next datum. The initial values of α and y are ε. Memory of the form $(z, 2)$ is used for just marking the data seen so far. Marking of $(\alpha, 1)$ is used to denote that L_α does not contain some input string seen so far (thus the main remaining job of the algorithm is to check whether the tell-tale set for L_α is contained in the input seen so far). Marking of $(\alpha, 0)$ is used to denote that hypothesis length-lexicographically smaller than α have been considered earlier. This allows us to consider each possible hypothesis arbitrarily often until a correct hypothesis is found. In each stage of the algorithm the following is done:

1. Beginning of the stage.
   Current input is x and current memory is $\mathrm{conv}(\alpha, y)$.
   Query whether $(\alpha, 0)$, $(\alpha, 1)$ and $(y, 2)$ are marked in the memory space.
2. If $(\alpha, 0)$ is not marked,
   then mark $(\alpha, 0)$, let $\alpha = \varepsilon$, let $y = \varepsilon$ and go to step 6.
3. If $(\alpha, 1)$ is marked or $x \neq \#$ and $x \notin L_\alpha$,
   then let $\alpha = succ(\alpha)$, let $y = \varepsilon$ and go to step 6.
4. If $y \in L_\alpha$ and $(y, 2)$ is not yet marked and $y \neq x$,
   then let $\alpha = succ(\alpha)$, let $y = \varepsilon$ and go to step 6.
5. If $|succ(y)| \leq |\alpha| + c$,
   then let $y = succ(y)$ and go to step 6.
6. Mark $(x, 2)$. If $x \neq \#$, then mark $(\beta, 1)$ for all β where $x \notin L_\beta$.
7. Output the hypothesis α.
8. The new memory is the new value of $\mathrm{conv}(\alpha, y)$.
   End of the stage.

Note that the above steps are done in each stage when processing a new datum x, starting at step 1 and going up to step 8. All markings are made after the queries on old markings had been done and the updates might depend on the old markings. The algorithm can be realised by automatic functions using the answers to the queries given and the queries can also be determined by automatic functions.

Now it remains to show that the algorithm learns the class $\mathcal{L}$. For this, assume, without loss of generality, that every language in the class has exactly one index. Suppose that L_γ is the language to be learnt. Further suppose all elements of L_γ which are of length at most $|\gamma| + c$ have already shown up in the input. If the current index α is not equal to γ, then either during the inference process some $x \notin L_\alpha$ shows up, causing $(\alpha, 1)$ to be marked, and thus causing α to be updated to $succ(\alpha)$ in step 3, or there exists a $z \in L_\alpha - L_\gamma$ with $|z| \leq |\alpha| + c$ (due to tell-tale set property). In the later case, eventually the variable y will take the value z and then α will be updated to $succ(\alpha)$ by step 4, as z will never be observed and, therefore, $(z, 2)$ will never be marked. Furthermore, when α takes a new value not taken before, step 2 will reset α to ε and α will cycle to all the values again until it eventually reaches the value of γ. Then steps 2 and 3 will not apply. Furthermore, as all the members of L_γ of length at most $|\gamma| + c$ have already been observed, step 4 will also not apply. Therefore, the value of α will no longer change and y will eventually stabilise. So the algorithm would converge to $\alpha = \gamma$ and will conjecture L_γ from that point of time onwards. Thus, L_γ is learnt by the algorithm. $\Box$

One might ask whether it is necessary to have a marked memory space of type 2 in the above result. The next result shows that in some cases this is indeed needed and a marked memory space of type 1 is not enough.

Theorem 17. *Let $\mathcal{L}$ be the class consisting of $\{0\}^+$ and all finite sets F with $\{\varepsilon\} \subseteq F \subseteq \{0\}^*$. $\mathcal{L}$ has an automatic learner. However, if an automatic learner is permitted to use only long term memory bounded by the size of its current hypothesis plus a marked memory space of type 1, then $\mathcal{L}$ is not learnable.*

Proof. Jain, Luo and Stephan [18] showed that the class $\mathcal{L}$ can be learnt by an automatic learner, as it is a class over the unary alphabet and satisfies Angluin's tell-tale condition.

Now consider any automatic learner where the memory is bounded by the hypothesis size. Using standard locking sequence methods [5, 28], one can show that, when learning the language $\{0\}^+$, there is a sequence σ and a constant c such that the learner, on any text for $\{0\}^+$ starting with σ, does not change the conjecture after having processed σ and never takes a long term memory value of length longer than c. Furthermore, there is a constant c' such that, when reading an input 0^n, the learner marks or queries only strings of length up to c' or of length between $n - c'$ and $n + c'$.

Now consider the learner on input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$, where $n_{i+1} > n_i + 2c'$ and $n_1 > a + 2c' + m$ for a being the length of the longest string in σ and m being the length of the longest string marked by the learner while reading σ. Note that, on input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$, there is a long term memory value which is repeated infinitely often in the above run of the learner. Let this long term memory value be mem. Let k be large enough such that memory of the learner after seeing $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k}$ is mem, and any string of length at most c' which is ever marked (on input text $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$), gets marked by the time the learner sees $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k}$. Now consider the behaviour of the learner on input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ \varepsilon^\infty$. As this input is in $\mathcal{L}$, the learner converges on this input text to some conjecture (say β) and does not query or mark any string of length larger than a number d (as the learner receives only finitely many strings in input).

Let r and $r' > r$ be such that, $n_r > n_k + d + 2c'$ and memory of the learner is mem after having seen $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ 0^{n_{k+1}} \circ \ldots \circ 0^{n_{r-1}}$ and also after having seen $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ 0^{n_{k+1}} \circ \ldots \circ 0^{n_{r-1}} \circ 0^{n_r} \circ \ldots \circ 0^{n_{r'}}$. Note that there exist such r and r', as the learner has memory mem infinitely often on the input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$; thus, the memory of the learner is also mem after having seen $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ 0^{n_r} \circ 0^{n_{r+1}} \circ \ldots \circ 0^{n_{r'}}$ as the learner's outputs only depend on its memory and the answer to the queries asked.

Thus, the learner also converges to β on the input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ 0^{n_r} \circ 0^{n_{r+1}} \circ \ldots \circ 0^{n_{r'}} \circ \varepsilon^\infty$. It follows that the learner converges to the same conjecture on the two different languages given by the texts $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ \varepsilon^\infty$ and $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ 0^{n_r} \circ 0^{n_{r+1}} \circ \ldots \circ 0^{n_{r'}} \circ \varepsilon^\infty$. Thus, the learner fails to learn $\mathcal{L}$. $\square$

It is open whether an automatic learner, with the memory bounded by the hypothesis size, can be made iterative [18]. Theorem 18 deals with the counterpart of this problem when a marked memory space of type 1 is also permitted.

Theorem 18. *If a class $\mathcal{L}$ has an automatic learner with its long term memory bounded by hypothesis size and not using any marked memory space, then $\mathcal{L}$ also has an automatic iterative learner using hypothesis queries and a new underlying automatic family as hypothesis space.*

Proof. Assume that automatic learner M is learning $\mathcal{L} = \{L_i : i \in I\}$ using a memory over Γ^*. Without loss of generality assume that initial hypothesis of M is not ?. Let $H_{\mathrm{conv}(i,g)} = L_i$ for all $i \in I$ and $g \in \Gamma^*$. For a current datum x and memory g and hypothesis i, let

$$F(\mathrm{conv}(i,g),x) = \begin{cases} \mathrm{conv}(i,h), & \text{if } M \text{ on input } x \text{ and memory } g \\ & \text{conjectures ? and takes new memory } h; \\ \mathrm{conv}(j,h), & \text{if } M \text{ on input } x \text{ and memory } g \\ & \text{conjectures } j \text{ and takes new memory } h. \end{cases}$$

The function F is automatic. Note that F can be considered as a learner which starts with initial hypothesis $\mathrm{conv}(\mathit{inithyp}, \mathit{initmem})$, where $\mathit{initmem}$ is the initial memory of M and $\mathit{inithyp}$ is the initial hypothesis of M. Then, for any input segment σ, $F(\sigma) = \mathrm{conv}(i,g)$, where i is the most recent hypothesis of M after reading input σ and g is the memory of M after reading input σ.

Now one builds a new automatic iterative learner N, which uses hypothesis space $\{H_{\mathrm{conv}(i,g)} : i \in I, g \in \Gamma^*\}$ and which tries to follow M and F as closely as possible, but which does not return to a hypotheses it has once abandoned. N starts with the same initial hypothesis as F that is with conv($inithyp, initmem$). Then the update function of N is the following one:

$$N(\mathrm{conv}(i,g),x) = \begin{cases} F(\mathrm{conv}(i,g),x), & \text{if } F(\mathrm{conv}(i,g),x) \text{ had not been} \\ & \text{conjectured previously;} \\ \mathrm{conv}(i,g), & \text{if } F(\mathrm{conv}(i,g),x) \text{ had been} \\ & \text{conjectured previously.} \end{cases}$$

Note that N can find out using a hypothesis query whether $F(\mathrm{conv}(i,g),x)$ had been conjectured previously. As F is automatic, so is N. Furthermore, assume that $N(x_0 \circ x_1 \circ \ldots \circ x_n) = \mathrm{conv}(i_n, g_n)$ for some text $x_0 \circ x_1 \circ \ldots$ for some language in $\mathcal{L}$. Then there are sequences $\tau_0, \tau_1, \ldots$ (with τ_n containing elements from $\{x_0, x_1, \ldots, x_n\}$) such that F after processing $x_0 \circ \tau_0 \circ x_1 \circ \tau_1 \circ \ldots \circ x_n \circ \tau_n$ has the hypothesis (i_n, g_n). One can show this by induction on n. It is clearly true for $n = 0$, as one can choose $\tau_0 = \varepsilon$. Assume $n > 0$ and assume the statement to be true for values smaller than n. If $N(\mathrm{conv}(i_{n-1}, g_{n-1}), x_n)$ is defined via the first clause in its definition, then one can choose τ_n to be empty sequence. If $N(\mathrm{conv}(i_{n-1}, g_{n-1}), x_n)$ is defined via the second clause in its definition, then suppose $F(conv(i_{n-1}, g_{n-1}), x_n) = \mathrm{conv}(i_m, g_m)$, which was conjectured by N after $x_0 \circ x_1 \circ \ldots \circ x_m$ (here m could be -1 and $F(conv(i_{n-1}, g_{n-1}), x_n)$ is then the initial hypothesis of N). One then chooses τ_n to be $x_{m+1} \circ \tau_{m+1} \circ x_{m+2} \circ \tau_{m+2} \circ \ldots \circ x_{n-1} \circ \tau_{n-1}$. It is easy to verify that $N(x_0 \circ x_1 \circ \ldots \circ x_n) = F(x_0 \circ \tau_0 \circ x_1 \circ \tau_1 \circ \ldots \circ x_n \circ \tau_n)$.

Suppose that the sequence of hypothesis of M converges on the text $x_0 \circ \tau_0 \circ x_1 \circ \tau_1 \circ \ldots$ to i. Then, for almost all n, i_n as defined above is i. Furthermore, $|g_n| \leq |i| + c$ for some constant c and all n. As N does not return to old abandoned hypotheses, the sequence of (i_n, g_n) also converges to a pair conv(i, g); here the first component must be i as almost all i_n are i. It follows from $H_{\mathrm{conv}(i,g)} = L_i$ that N learns the input language. □

Acknowledgements

Preliminary version of this paper appeared in CiE 2011 [10]. We would like to thank the anonymous referees of the conference for several useful 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. Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
6. Achim Blumensath. *Automatic structures.* Diploma thesis, RWTH Aachen, 1999.

7. Achim Blumensath and Erich Grädel. Automatic structures. *15th Annual IEEE Symposium on Logic in Computer Science* (LICS), pages 51–62, IEEE Computer Society, 2000.
8. John Case, Sanjay Jain, Steffen Lange and Thomas Zeugmann. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110, 1999.
9. John Case, Sanjay Jain, Trong Dao Le, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic Learning of Subclasses of Pattern Languages. *Language and Automata Theory and Applications,* LATA 2011, Taragona, Spain, May 2011, Proceedings. Springer LNCS 6638:192–203, 2011.
10. John Case, Sanjay Jain, Yuh Shin Ong, Pavel Semukhin and Frank Stephan. Automatic Learners with Feedback Queries. Seventh conference on *Computability in Europe*, CiE 2011, Sofia, Bulgaria, June/July 2011, Proceedings. Springer LNCS 6735:31–40, 2011.
11. Henning Fernau. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711, 2003.
12. Rusins Freivalds, Efim Kinber and Carl H. Smith. On the impact of forgetting on learning machines. *Journal of the Association of Computing Machinery*, 42:1146–1168, 1995.
13. E. Mark Gold. Language identification in the limit. *Information and Control* 10:447–474, 1967.
14. 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.
15. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.
16. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
17. Sanjay Jain, Qinglong Luo, Pavel Semukhin and Frank Stephan. Uncountable automatic classes and learning. *Theoretical Computer Science*, 412:1805–1820, 2011. Special Issue on Algorithmic Learning Theory, 2009.
18. Sanjay Jain, Qinglong Luo and Frank Stephan. Learnability of automatic classes. *Language and Automata Theory and Applications*, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings. Springer LNCS 6031:321-332, 2010. Also as Technical Report TRA1/09, School of Computing, National University of Singapore, 2009.
19. Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank Stephan. On automatic families. TRB1/10, School of Computing, National University of Singapore, 2010.
20. Sanjay Jain, Eric Martin and Frank Stephan. Robust learning of automatic classes of languages. To appear in, *Algorithmic Learning Theory, 22nd International Conference*, ALT 2011.
21. Sanjay Jain, Daniel N. Osherson, James S. Royer and Arun Sharma. *Systems That Learn.* MIT Press, 2nd Edition, 1999.
22. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity.* Springer LNCS 960:367–392, 1995.
23. Efim Kinber and Frank Stephan. Language learning from texts: mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.

24. 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.
25. Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.
26. Steffen Lange, Thomas Zeugmann and Sandra Zilles. Learning indexed families of recursive languages from positive data: a survey. *Theoretical Computer Science*, 397:194–232, 2008.
27. Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Third Edition, Springer, 2008.
28. 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.
29. Lenny Pitt. Inductive inference, DFAs, and computational complexity. *Analogical and Inductive Inference, Proceedings of the Second International Workshop (AII'89).* Springer LNAI 397:18–44, 1989.
30. Sasha Rubin. *Automatic Structures.* Ph.D. Thesis, The University of Auckland, 2004.
31. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
32. Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition.* Cambridge, Massachusetts, The MIT Press, 1980.
33. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik* (EIK) 12:93–99, 1976.