# 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 current hypothesis whenever this exists, cannot be of length larger than the length of the longest datum seen, 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 basic model of learning in inductive inference may be described as follows. A learner is receiving data, one piece at a time, about a target concept. As it is receiving data, it conjectures hypothesis about what the target concept might be. The learner may update its hypothesis as it receives more and more data. The learner is said to identify or learn the target concept if its sequence of hypotheses stabilises on a correct hypothesis. This is basically the model of explanatory learning considered by Gold [15]. Over time several other models of learning have

⋆ 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.

been investigated (see [23] for some of these models).

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, 15]. 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, 13, 16].

An automatic family of languages is a family for which membership question can be uniformly decided by a finite automaton. An advantage of an automatic family over general indexed families [1, 26, 28] is that the first-order theory of automatic families, as well as of automatic structures in general, is decidable [17, 18, 24]. Here the predicates (relations) and functions (mappings) allowed in the first-order theory are automatic. Furthermore, relations and functions that are first-order defined from other automatic relations and functions are automatic again [17, 18, 24]. Also, automatic functions are linear time computable [9]. These nice properties of automatic structures make them a useful tool not only in learning theory but also in other areas such as model checking and group theory [6, 12, 24, 30, 31, 34, 35]. 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 $x$.

Jain, Luo and Stephan [20] (see also, [9, 19, 22]) considered learnability of automatic families [21]. They showed that any automatic family can be explanatorily learnt by a recursive learner iff it satisfies a condition known as Angluin's tell-tale condition (see Proposition 2 below). Thus, it was more interesting to consider some restricted form of learners. One natural restriction, as considered by [20], on learners is that they are automatic, i.e., their graph is accepted by a finite automaton. For this purpose, the learner is modeled as working in rounds and having a long-term memory. In each round learner receives a new datum, and based on this datum and its previous long-term memory, updates its long-term memory and outputs a new hypothesis. The learner learns the target language if its sequence of hypotheses converges to an index for the target language (where the index is interpreted in some hypothesis space, which is an automatic family). If we do not restrict the long-term memory or the computational power of the learner in any way, then this model of learning is equivalent to the above mentioned model of explanatory learning. For automatic learner, we require that the mapping (previous memory, new datum) $\mapsto$ (new memory, hypothesis) is an automatic function, that is, its graph is 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 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 [24] to find which results from computable model theory can be transferred to model theory based on finite automata. Some very nice sub-classes of pattern languages [2] are automatic families and learnable by automatic learners [9]. However, Jain, Luo and Stephan [20] showed that not every learnable automatic family can be learnt by an automatic learner (see Example 14 and Theorem 17 below).

Jain, Luo and Stephan [20] considered some natural restrictions on long-term memory of the learner: the long-term memory of the learner is its current hypothesis (iterative learning), the long-term memory is bounded in length by the length of the current hypothesis plus a constant (we refer to this as hypothesis-length long-term memory), and the long-term memory is bounded in length by the length of the longest datum seen so far plus a constant (we refer to this as datum-length long-term memory). It was shown that some automatic classes can be learnt by an automatic learner using datum-length long-term memory but not by any automatic learner using hypothesis-length long-term memory. However it was open whether automatic learners using datum-length long-term memory are as powerful as general automatic learners or whether any automatic learner using hypothesis-length long-term memory can be simulated by an iterative automatic learner or by an automatic learner using datum-length long-term memory. Jain, Luo and Stephan [20] showed that if one considers *fat* texts, i.e., the inputs where every element of the target language appears infinitely often, then automatic learners using datum-length long-term memory can learn all automatic families which satisfy Angluin's tell-tale condition. Since [20] several papers [9, 19, 22] have considered learning of automatic classes by automatic learners. The present work carries on investigations into learnability of automatic families by automatic learners.

The notion of learners with explicit bounds on the long-term memory had already been studied previously in the setting of algorithmic learners [14, 25]. 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, 27, 38]. The present work investigates these notions for the case of automatic learners learning automatic families. We do not investigate all possible combinations of memory models, but look at some of the more typical and natural combinations. In particular, we will look at what additional features allow an automatic learner to learn the class of all automatic families satisfying Angluin's tell-tale condition.

**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 16 shows that every automatic family satisfying Angluin's tell-tale condition has an automatic feedback learner with only one query per round and using datum-length long-term memory. Theorem 17 shows that there is a class which has an automatic 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 18 and 20 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 automatic learners using a marked memory space (see Section 2 for definition). Theorem 21 shows that the more general marked memory space of type 2 permits to learn any learnable class by an automatic learner using hypothesis-length long-term memory. In contrast to this, Theorem 22 shows that such a result is not possible for a marked memory space of type 1. It is an open problem whether an automatic learner using hypothesis-length long-term memory can be replaced by an iterative automatic learner [20]. Theorem 23 gives a partial answer: this is possible if the iterative automatic learner has additionally access to a marked memory space of type 1. The following tables summarise the main inclusions and non-inclusions; all learners considered in these tables are automatic:

| Inclusion | Given Criterion | Implied Criterion |
|---|---|---|
| Theorem 16 | Angluin's tell-tale criterion | one feedback query plus datum-length memory |
| Theorem 21 | Angluin's tell-tale criterion | hypothesis-length memory plus type 2 marked memory space |
| Theorem 23 | hypothesis-length memory | iterative plus hypothesis queries |

| Non-Inclusion | Satisfied Criterion | Diagonalised Criterion |
|---|---|---|
| Theorem 17 | memoryless one feedback query | automatic learners |
| Theorem 18 | 1-bounded example memory | iterative plus feedback |
| Theorem 18 | hypothesis-length memory plus feedback query | iterative plus feedback |
| Theorem 19 | $k$-bounded example memory | $(k-1)$-bounded example memory plus feedback queries |
| Theorem 20 | $k$-bounded example memory / memoryless $k$-feedback queries / constantly bounded memory | $(k-1)$-bounded example memory / memoryless $(k-1)$-feedback queries |
| Theorem 22 | datum-length memory | hypothesis-length memory plus type 1 marked memory space |

## 2 Notations, learning with feedback queries and memory limitations

The symbol $\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. The length of the string $x$ is denoted by $|x|$. Empty string is denoted by $\varepsilon$. For a string $x$, $x(i)$ denotes the $(i+1)$-th element in the string $x$; thus, string $x$ is same as $x(0)x(1)\ldots x(|x|-1)$. Let $x <_{lex} y$ denote that $x$ is lexicographically before $y$ (where we assume some fixed ordering of letters of the alphabet $\Sigma$ of the strings). Let $x \leq_{lex} y$ iff $x <_{lex} y$ or $x = y$, let $x >_{lex} y$ iff $y <_{lex} x$, and let $x \geq_{lex} y$ iff $y \leq_{lex} x$. We say that $x$ is *length-lexicographically less than* $y$ (written $x <_{ll} y$) iff either $|x| < |y|$ or $|x| = |y|$ and $x <_{lex} y$. Let $x \leq_{ll} y$ iff $x = y$ or $x <_{ll} y$, let $x >_{ll} y$ iff $y <_{ll} x$, and let $x \geq_{ll} y$ iff $y \leq_{ll} x$.

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 $\Sigma$, we define the convolution [24], $conv(x, y)$, 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 $\{\mathrm{conv}(x_1, x_2, \ldots, x_n) : R(x_1, x_2, \ldots, x_n)\}$ or $\{\mathrm{conv}(x_1, x_2, \ldots, x_m, y) : f(x_1, x_2, \ldots, x_m) = y\}$, respectively, is regular [24].

A family of languages $\{L_\alpha : \alpha \in I\}$ is said to be *automatic* [20, 21] iff (a) $I$ (called the index domain) is a regular set of strings over some finite alphabet, (b) there is a finite alphabet $\Sigma$ such that each $L_\alpha \subseteq \Sigma^*$ and (c) the set $\{\mathrm{conv}(\alpha, x) : \alpha \in I \wedge x \in L_\alpha\}$ is regular.

Automatic structures are structures given by finitely many automatic relations and functions. Historically, much work has been dedicated to the question of which natural structures are isomorphic to the automatic ones, for example $(\mathbb{N}, +, <)$ is isomorphic to an automatic one while, $(Q, +)$ is not [36].

Fix a finite alphabet $\Sigma$ for the languages. Let $\# \notin \Sigma^*$. A *text* is a mapping from $\mathbb{N}$ to $\Sigma^* \cup \{\#\}$. Traditionally, $\#$ is used to denote pauses in the presentation of data and is also needed as the only text for empty language is $\#^\infty$; one could come around the use of $\#$ by requiring that all languages to be learnt are non-empty, but this turned out to be notationally awkward in various cases. The *content* of a text $T$, denoted $\mathit{content}(T)$, is $\{T(i) : i \in \mathbb{N}\} - \{\#\}$. We say that $T$ is a text for $L$ iff $\mathit{content}(T) = L$. *Sequences* are initial segments of texts. We let $T[n]$ denote $T(0)T(1)\ldots T(n-1)$. We let $\Lambda$ denote the empty sequence. The content of a sequence $\sigma = T[n]$, denoted $\mathit{content}(\sigma)$, is $\{T(i) : i < n\} - \{\#\}$. Let $|\sigma|$ denote the length of the sequence $\sigma$. For sequences $\sigma$ and $\tau$, let $\sigma \circ \tau$ denote the concatenation of $\sigma$ and $\tau$. Furthermore, let $\sigma \circ x$ denote concatenation of $\sigma$ and the sequence consisting of just the element $x$. Note that $xx$ is a string obtained by concatenating $x$ with itself while $x \circ x$ is a data sequence where the datum $x$ is presented twice as input.

We will be considering learning of an automatic family $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ by a learner using as hypothesis space an automatic family $\mathcal{H} = \{H_\beta : \beta \in J\}$.

**Definition 1.** (Based on Gold [15]) Suppose $\Sigma$ is a finite alphabet for languages, and $I, J$ are regular index sets over some finite alphabet. Suppose $? \notin J$ is a special symbol (denoting repeat of previous conjecture by the learner).

Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ be the class to be learnt and $\mathcal{H} = \{H_\beta : \beta \in J\}$ be a hypothesis space, where both are automatic families, with languages being subset of $\Sigma^*$. Suppose $\Gamma$ is a finite alphabet used for storing memory by learners.

(a) A *learner* is an algorithmic mapping from $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to $\Gamma^* \times (J \cup \{?\})$.
A learner has an initial long-term memory $\mathit{mem}_0 \in \Gamma^*$ and initial hypothesis $\mathit{hyp}_0 \in J \cup \{?\}$.

(b) Suppose a learner $M$ with initial long-term memory $\mathit{mem}_0$ and initial hypothesis $\mathit{hyp}_0$ and a text $T$ for a language $L$ is given.
  (i) Let $\mathit{mem}_0^T = \mathit{mem}_0, \mathit{hyp}_0^T = \mathit{hyp}_0$.
  (ii) For $k > 0$, let $(\mathit{mem}_k^T, \mathit{hyp}_k^T) = M(\mathit{mem}_{k-1}^T, T(k-1))$.
  Intuitively, $\mathit{mem}_k^T$ and $\mathit{hyp}_k^T$ are the long-term memory and hypothesis of the learner $M$ after having seen the input $T[k]$.
  (iii) $M$ *converges* on text $T$ to $\beta$ iff there exists a $t$ such that $\mathit{hyp}_t^T = \beta$ and, for all $t' \geq t$, $\mathit{hyp}_{t'}^T \in \{\beta, ?\}$.
  (iv) $M$ *learns* the language $L$ (using hypothesis space $\mathcal{H}$) from the text $T$ iff $M$ converges on text $T$ to a hypothesis $\beta$ such that $H_\beta = L$.

(c) $M$ *learns* a language $L$ (using hypothesis space $\mathcal{H}$) iff $M$ learns $L$ from all texts for the language $L$ (using hypothesis space $\mathcal{H}$).

(d) $M$ *learns* $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) iff $M$ learns all languages in $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).
(e) $\mathcal{L}$ is said to be *learnable* iff some learner $M$ learns $\mathcal{L}$ using some hypothesis space $\mathcal{H}'$.

For ease of notation, when $T$ is implicit, we will drop $T$ from the superscript of $mem_n^T$ and $hyp_n^T$.

For a learner $M$, let $M(\sigma)$ denote the hypothesis of the learner $M$ after having seen input segment $\sigma$. This notation applies even for other kinds of learners defined below.

Intuitively, one can view the learner as operating in stages (rounds) on an input text $T$ for a target language $L$. The learner has initial long-term memory $mem_0 \in \Gamma^*$ and initial hypothesis $\beta_0 \in J \cup \{?\}$. In stage (round) $n$, the learner receives the input $T(n)$, updates its previous long-term memory $mem_n$ to $mem_{n+1}$ and outputs a hypothesis $\beta_{n+1}$. For general learners as studied by Gold [15], 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 long-term memory). The learner learns [15] a language $L$ iff for all texts $T$ for $L$, for $\beta_n$ as defined above, the sequence $\beta_0, \beta_1, \ldots$ converges to an index (in $J$ for the hypothesis space $\mathcal{H}$) for $L$. This model of learning is also referred to as explanatory learning from text or **TxtEx**-learning. We used the symbol ? to denote that either the learner does not change its previous hypothesis (this is useful for some long-term memory limited models of learner) or the learner has not yet seen enough data for its initial conjecture.

For learning automatic families of languages a characterisation based on Angluin's condition [1] determines when an automatic family is learnable [20].

**Proposition 2 (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 $b_\alpha$ 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 $L_\alpha$. The condition "$\{x \in L_\alpha : |x| \leq b_\alpha\} \subseteq L_\beta \subseteq L_\alpha \Rightarrow L_\beta = L_\alpha$" is called* Angluin's tell-tale condition*. Note that we can take $b_\alpha = |\alpha| + c$ for a suitable constant $c$ independent of $\alpha$.*

Therefore, the challenge is to study learnability by more restrictive learners. In the setting of automatic structures, it is natural to require that the learners are also *automatic* [20], that is, the mapping $(mem_n, T(n)) \mapsto (mem_{n+1}, \beta_{n+1})$ for the learners is automatic. The hypothesis and the updated long-term memory of such a learner can be computed in time *linear in their previous long-term memory and current datum* [9]. The price paid is that the learner can no longer access the full past history of the data observed, as the automaticity of the mapping $(mem_n, T(n)) \mapsto (mem_{n+1}, \beta_{n+1})$, forces the learner to forget some past data. In general, the requirement of a learner to be automatic is a real restriction [20] (for examples of a learnable automatic class which cannot be learnt by an automatic learner, see Example 14 and Theorem 17). However, for unary alphabets, automatic learners are as powerful as non-automatic ones.

**Theorem 3 (Jain, Luo and Stephan [20]).** *Suppose $\{L_\alpha : \alpha \in I\}$ is an automatic family over a unary alphabet. Then, $\{L_\alpha : \alpha \in I\}$ is learnable by an automatic learner (which uses long-*

*term memory bounded in length by the length of the longest datum seen so far) iff the family satisfies Angluin's tell-tale condition.*

Jain, Luo and Stephan [20] had considered various ways in which the long-term memory of the automatic learners can be bounded in length. The length-restrictions considered are as follows. For the following, $T$ is an arbitrary input text, $mem_n$ and $\beta_n$ denotes the long-term memory and hypothesis of the learner just before getting input $T(n)$.

(a) the length of the hypothesis plus a constant; that is, for some constant $c$ (independent of $T$ and $n$), $|mem_n| \leq |\beta_n| + c$; we refer to this as hypothesis-length memory.
(b) the length of the longest datum seen so far plus a constant; that is, for some constant $c$ (independent of $T$ and $n$), $|mem_n| \leq \max\{|T(i)| : i < n\}+c$; we refer to this as datum-length memory.
(c) just constant length; that is, $|mem_n| \leq c$, for some constant $c$ (independent of $T$ and $n$); we refer to this as constant-length memory.

For 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 long-term memory; therefore, it might, for example, store the convolution of up to $k$ many examples in the case that it uses datum-length memory (here $k$ is some constant). Note that, in the case of hypothesis-length (or better) long-term memory, the learner can memorise the most recent hypothesis output, and, thus, abstain from outputting ? (since it can output the stored most recent hypothesis).

Furthermore, it should be noted that for most of our results, the restriction in (c) is strengthened to not to use any memory at all. There are, however, classes where constant-length long-term memory helps, for example when learning all sets of the form $\{x\}$ or $\{x0, x1\}$; the long-term memory keeps track of whether strings ending with 0 have been seen so far and whether strings ending with 1 have been seen so far. Without that constant long-term memory, convergence to a correct hypothesis would not be possible. The class given in Theorem 20 can also be learnt with a constant amount of long-term memory as defined under (c) and without queries of any type.

Also note that, as one can choose the alphabet size of the long-term memory, the usage of the "plus some constant" could be replaced by using $\max\{d, 1\}$ where $d$ is the length of the longest datum seen so far or the length of the current hypothesis. Therefore often the literature on long-term memory bounds [14, 25] takes into account the number of binary bits one has to use to represent it. In the field of automatic learning, however, using datum-length or hypothesis-length memory is more adequate; the reason is that these bounds come out naturally from the framework: first, the length of minimal indices is quite invariant over hypotheses spaces used; second, the smallest indices of a finite set in an automatic family corresponds, up to a constant depending on the family and not on the set, with the length of its longest element; third, the automatic functions doing the updates of the long-term memory and computing the hypotheses and queries cannot make their values longer than the maximum of the long-term memory length and the length of the current datum plus a constant [20, 21].

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 long-term memory by automatic learners. If one considers *fat texts* [32], where every data item is repeated infinitely often in the text, then Jain, Luo and Stephan [20] showed that automatic learners (without any long-term memory restrictions except due to the definition of automatic learners, but not using any marked memory space as considered below) 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 marked memory space, which subsumes feedback learning and related criteria.

For the following definitions, $\Sigma$ is the alphabet for the languages, $\Gamma$ is the alphabet used by the learner for its long-term memory. $\mathcal{L} = \{L_\beta : \beta \in I\}$ refers to the language class being learnt; $L_\alpha$ refers to the target language, and a text $T$ for $L_\alpha$ 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 $\beta_n$ denote the long-term memory and hypothesis of the learner just before receiving input $T(n)$. We sometimes consider the long-term 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 long-term memory as a set, we assume that it was in sorted order: that is, $x_1 <_{ll} x_2 <_{ll} \ldots <_{ll} x_r$. 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 4 (Based on [37, 38]).** An automatic learner is called *iterative* iff, for all $n$, $mem_n = \beta_n$.

**Definition 5 (Based on [27, 32]).** 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 long-term 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 [38] and Lange and Zeugmann [27].

**Definition 6 (Based on [8, 27, 38]).** 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. The initial long-term memory of the learner is $mem_0$ and the initial hypothesis of the learner is $\beta_0$. Given an input text $T$, let $mem_n$, and $\beta_n$ denote the long-term memory and hypothesis of the learner after having seen $T[n]$ (that is just before receiving input $T(n)$).

(a) $Q$ is an automatic mapping from $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to the set of subsets of $\Sigma^*$ 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 $\Delta$. This marked memory will only grow (set inclusion wise) as the learner gets more data. Marked memory can be considered as a generalisation of feedback learning since feedback learning can be simulated by Type 1 memory.

**Definition 7.** 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 long-term memory of the learner is $mem_0$, initial hypothesis of the learner is $\beta_0$ and initial marked memory space is $Z_0 = \emptyset$. Furthermore, $mem_n$, $Z_n$ and $\beta_n$ denote the long-term memory, the marked memory and hypothesis of the learner after having seen the input $T[n]$ (that is just before receiving input $T(n)$).

(a) $Q$ is an automatic mapping from $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to the set of subsets of $\Delta^*$ 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 the case of Type 1 memory space, $X_{n+1}$ is technically represented as a convolution of the elements of the set, and is thus necessarily finite with cardinality bounded by some constant. For Type 2 memory, $X_{n+1}$ may be infinite.

**Remark 8.** An *automatic feedback learner* is a special case of an automatic learner using a marked memory space of Type 1, for which, in Definition 7, $\Sigma^*$ is used for memory instead of $\Delta^*$, and $X_{n+1} = \{T(n)\} \cap \Sigma^*$.

**Definition 9.** An *automatic learner using $k$-hypothesis queries* is a special case of an automatic learner using a marked memory space of Type 1 for which, in Definition 7, the index set $J$ of the hypothesis space is used for memory instead of $\Delta^*$, $X_{n+1} = \{\beta_{n+1}\} - \{?\}$ and $Q$ is an automatic mapping from $\Gamma^* \times (\Sigma^* \cup \{\#\})$ to the set of subsets of $J$ of size $k$. 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 $Z_n$.

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 [15] and is called explanatory learning. The variant with an explicit long-term memory as used here was introduced by Freivalds, Kinber and Smith [14]. The special case of iterative learning is quite popular and predates the definition of general memory limitations, it was introduced by Wiehagen [38] and later by Wexler and Culicover [37]. Bounded example memory was considered by Osherson, Stob and Weinstein [32]; Lange and Zeugmann [27] extended this study. Wiehagen [38] and Lange and Zeugmann [27] introduced and studied feedback learning; Case, Jain, Lange and Zeugmann [8] quantified the number of feedback queries per round.

The following definition and lemma by L. Blum and M. Blum [5] is useful to prove some of our results.

**Definition 10 (Blum and Blum [5]).** A sequence $\sigma$ is said to be a *locking sequence* for $M$ on $L$, if (i) $content(\sigma) \subseteq L$, (ii) $M(\sigma)$ is an index for $L$ (in the hypothesis space used by $M$), and (iii) for all $\tau$ such that $content(\tau) \subseteq L$, $M(\sigma) = M(\sigma \circ \tau)$, that is $M$ does not change its hypothesis beyond $\sigma$ on any text for $L$ starting with $\sigma$.

**Lemma 11 (Blum and Blum [5]).** *Suppose a learner $M$ learns $L$. Then there exists a locking sequence for $M$ on $L$.*

For some of the restrictions/variations 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, in the case of hypothesis-length long-term memory or iterative learning. The main reason for hypothesis space not to be crucial 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 [27, 28]. A result in the present work which depends on the choice of the hypothesis space is Theorem 23. 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.

The following lemma is useful to bound the length of the long-term memory of automatic learners.

**Lemma 12 (Jain, Luo and Stephan [20]).** *Let $M$ be an automatic learner. For some constant $c$, for all $\sigma$, if $mem$ and $hyp$ are the long-term memory and hypothesis of $M$ after having seen input $\sigma$, then* $\max\{|mem|, |hyp|\} \leq c + c * |\sigma| + \max\{|w| : w \in content(\sigma)\}$.

## 3 Some illustrative examples

We now provide some examples to give insight into the learning criteria considered and their properties. Example 13 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 13.** *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* $L_{\mathrm{conv}(000,01^m)}$*; already Gold* [15] *observed that classes of this form cannot be learnt from positive data.*

Example 14 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 14.** *The family of all co-singleton sets* $\{0,1\}^* - \{x\}$*, with* $x \in \{0,1\}^*$*, is automatic and satisfies Angluin's tell-tale condition. It does not have an automatic learner, as such a learner cannot memorise all the data observed* [20]*. 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 datum-length long-term memory. This follows from Theorem 3.*

Example 15 shows 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 consists of the hypotheses that have been issued and the data-items that have been observed. Note that the family in (a) is the same as in Example 14, though in contrast the learner here does not have any long-term memory and thus does not remember its previous hypothesis; instead it reconstructs part of the hypotheses history using the current datum and hypothesis queries.

**Example 15.** (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 long-term 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 $x$ in $\{0,1\}^*$. 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 (including the case of the input being of the form $1^n$ for some $n$), 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 $\{\{x,y\} : x <_{ll} y\}$ (using some hypothesis space $\{H_\alpha : \alpha \in J\}$). Now consider the text $y \circ x \circ x \circ x \circ \ldots$ being presented to the learner. Note that the learner, when it receives input $x$, 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 placed in the marked memory space by the learner on input $y$ or on 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 [20, 21].

## 4 Learning with feedback queries

The following result shows that feedback queries, together with datum-length long-term memory, allows automatic learners to learn every automatic family satisfying Angluin's tell-tale condition. Equivalently one can also say that the automatic learners with feedback are as powerful as algorithmic learners without memory limitations. Later (Theorem 22) we will show that hypothesis-length long-term memory is not enough to get such a result.

**Theorem 16.** *If an 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 datum-length long-term memory.*

**Proof.** Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic family satisfying Angluin's tell-tale condition. As one can decide equivalence of indices in an automatic family [21], without loss of generality assume that every distinct $\alpha, \alpha' \in I$ satisfy $L_\alpha \neq L_{\alpha'}$ (otherwise, we can ignore non-minimal indices). By results from [21], there exists a constant $c$ such that (i) for each $\alpha \in I$, $\{w : |w| \leq |\alpha| + c\} \cap L_\alpha$ is a tell-tale set for $L_\alpha$ (below, tell-tale set for $L_\alpha$ will refer to this set) and (ii) for each $L \in \mathcal{L}$, if $L$ is finite, then there exists an index $\alpha$ such that $L = L_\alpha$ and $|\alpha| \leq c + \max\{|x| : x \in L\}$. Fix such a constant $c$. The goal of the learner is to find an $\alpha$ such that

(a) the input contains $\{w : |w| \leq |\alpha| + c\} \cap L_\alpha$, (which is a tell-tale set for $L_\alpha$) and
(b) every element in the input is contained in $L_\alpha$.

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

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

To address the first problem above, each potential $\alpha$ 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 length-lexicographically largest datum seen before current conjecture $\alpha$ was tried, if any of them has been seen earlier but not in $L_\alpha$.

The long-term memory of the learner is of the form $\mathrm{conv}(z, \alpha, y, b)$. Here $z$ is the length-lexicographically largest datum seen so far ($z$ is $\varepsilon$ for the case that the input does not contain any datum), $\alpha$ is the current conjecture (which will be of length at most $|z| + c$), $b \in \{0, 1\}$ 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 $\max\{|z|, |\alpha|\} + c$). Note that for (b), the algorithm need only use feedback for strings length-lexicographically smaller than the length-lexicographically largest string seen up to the point at which the algorithm started testing for (b) above; the rest of the strings are tested as they arrive.

Without loss of generality assume that $\varepsilon$ is the length-lexicographically least string in $\Sigma^*$ (the domain for the languages) as well as in $I$, the set of indices.

Initial long-term memory of the learner is $(\varepsilon, \alpha, \varepsilon, 0)$, where $\alpha$ is the length-lexicographically largest index which is of length at most $c$. Let $succ(y), pred(y)$ denote the length-lexicographic successor and predecessor of $y$ in $\Sigma^*$, and $pred(\alpha)$ denote the length-lexicographic predecessor of $\alpha$ in the index set $I$ (here $pred(\varepsilon) = \varepsilon$). On input $x$ and previous long-term memory $(z, \alpha, y, b)$ the algorithm follows the first of the three cases below which applies. Here the feedback query asked by the learner is $y$.

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

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

Then the new long-term 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 $\alpha'$ is length-lexicographic predecessor of $\alpha$, otherwise $\alpha'$ 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 above tell-tale set for $L_\alpha$, and thus tries the next possible $\alpha$. *)

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

Then new long-term 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 $L_\alpha$. (* Here the learner has seen the above tell-tale set for $L_\alpha$, and thus goes on to test if the input language is a subset of $L_\alpha$. *)

Else let new long-term memory of the learner be $(z', \alpha, succ(y), 0)$ and the conjecture of the learner be $L_\alpha$, where $z'$ is the length-lexicographically largest string seen so far. (* Here the learner continues checking whether the above tell-tale set for $L_\alpha$ 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 long-term 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 $\alpha'$ is the length-lexicographic predecessor of $\alpha$, otherwise $\alpha'$ 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 $L_\alpha$, and thus tries the next possible $\alpha$. *)

Neither of the two cases above: The new long-term 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 $L_\alpha$.

Suppose the input text is for a target language $L_\gamma$. If $L_\alpha \neq L_\gamma$, then any long-term memory of the form $(\cdot, \alpha, \cdot, \cdot)$, will eventually be updated with changed value of $\alpha$ as either the input does not contain the above tell-tale set for $L_\alpha$ or the input contains an element not in $L_\alpha$. Also, as length of $\gamma$ is at most $|z|+c$, for the longest datum $z$ in the input, eventually it will be the case that the learner has a long-term 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 long-term 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 $L_\alpha$ are present in the input). From then on the value of $y$ decreases (until its value reaches $\varepsilon$) in each stage, and the learner always conjectures $L_\alpha$. Thus, the learner learns the target language $L_\gamma$. $\square$

Thus, the learners considered in the above result are as general as recursive learners (see Proposition 2). 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 (including bounded example memory). Note that Example 14 also gives a class which can be learnt by an automatic learner using one feedback query per round but cannot be learnt by an automatic learner without any marked space memory. However, the learner in Example 14 needs hypothesis-length long-term memory or datum-length long-term memory. The feedback learner in the following theorem does not use any long-term memory.

**Theorem 17.** *Let $\Sigma = \{0,1,2\}$. Let $L_\varepsilon = \{0,1\}^+$. For all $x \in \{0,1\}^+$, let $L_x = \{y \in \{0,1\}^* : |y| \leq |x|, y \neq x\} \cup \{y2^n : |y| = |x|, n > 0\}$. Let $\mathcal{L} = \{L_x : x \in \{0,1\}^*\}$. Then, the following two statements hold:*

(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.** (a) $\mathcal{L}$ can be learnt by an automatic learner (without any long-term memory) using 1 feedback query as follows. On inputs from $\{0,1\}^*$, query if $\varepsilon$ belongs to the input seen — if not, then output $L_\varepsilon$; 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 $\sigma_1$ and $\sigma_2$, each containing $m$ elements of $\{0,1\}^m$, such that $content(\sigma_1) \neq content(\sigma_2)$ and $M$ has the same long-term memory and hypothesis after processing either $\sigma_1$ or $\sigma_2$. As $M$ is automatic, such $m$, $\sigma_1$ and $\sigma_2$ exist (as, by Lemma 12, the long-term memory of $M$ can be of length at most a constant times $m$ after seeing such $\sigma_1$ or $\sigma_2$, and there are $\binom{2^m}{m}$ different subsets of $\{0,1\}^m$ of size $m$). Let $x_1$ be in $content(\sigma_1) - content(\sigma_2)$ and $x_2$ be in $content(\sigma_2) - content(\sigma_1)$. Let $T$ be a text for $\{y : |y| \leq m, y \neq x_1, y \neq x_2\} \cup \{y2^n : |y| = m \text{ and } n > 0\}$. Then, $M$ on $\sigma_1 \circ T$ and $\sigma_2 \circ T$ converges to the same hypothesis or diverges on both, though these are respectively texts for $L_{x_2}$ and $L_{x_1}$. Thus, $M$ does not learn $\mathcal{L}$. $\square$

The next two results give advantages of having bounded example memory over just feedback queries (without any long-term memory).

**Theorem 18.** *Let $\Sigma = \{0,1\}$. 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 $y \in \{0\}^{|x|} \cdot \{0\}^*$. Then the following three statements hold:*

(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 hypothesis-length long-term memory;*
(c) *$\mathcal{L}$ cannot be learnt by an automatic iterative learner using feedback queries.*

**Proof.** (a) Intuitively, the learner keeps in long-term memory the longest string seen, while the input is contained in $L_0$. Once the learner sees an element $x$ not in $0^*$, it memorises $x$. This long-term memory, along with the new input element, determines the conjecture of the learner.

The learner using the bounded example memory, on every input $z$ and long-term memory $A$, updates its memory and outputs a hypothesis based on the following table.

| old memory $A$ | input datum $z$ | new memory $A'$ | hypothesis |
|---|---|---|---|
| — | # | $A' = A$ | ? |
| $A = \emptyset$ | $z \in 0^+$ | $A' = \{z\}$ | $L_0$ |
| $A = \emptyset$ | $z \in \{0,1\}^*1$ | $A' = \{z\}$ | $L_{\mathrm{conv}(z,0^{\lvert z\rvert})}$ |
| $A = \{0^n\}$ | $z \in 0^+$ | $A' = \{0^{\max\{n,\lvert z\rvert\}}\}$ | $L_0$ |
| $A = \{0^n\}$ | $z \in \{0,1\}^*1$ | $A' = \{z\}$ | $L_{\mathrm{conv}(z,0^{\max\{n,\lvert z\rvert\}})}$ |
| $A = \{x\} \subseteq \{0,1\}^*1$ | $\lvert z\rvert > \lvert x\rvert$ | $A' = A$ | $L_{\mathrm{conv}(x,z)}$ |
| $A = \{x\} \subseteq \{0,1\}^*1$ | $\lvert z\rvert \leq \lvert x\rvert$ | $A' = A$ | ? |

It is easy to see that the learner above is automatic. We now show that the learner learns $\mathcal{L}$. 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 member of $\{0\}^+ \cap L_{\mathrm{conv}(x,y)}$ 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-length long-term memory initially conjectures hypothesis for $L_0$. The learner keeps in its long-term memory the string $x$, if any, in the input such that $x \notin 0^*$ along with one bit of information which tells whether an even or an odd number of distinct data items from $0^*$ have been seen so far — this parity bit can be updated if a not previously seen datum from $0^*$ is observed in the input (whether a datum is new or old can be determined by making one feedback query). Along with these, the learner also keeps in memory some other information as needed below.

If the learner has seen $x \notin 0^*$ in the input, then the parity bit is used in order to determine whether the parity of the number of strings from $0^*$ already seen in the input 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 long-term memory) of the form $\mathrm{conv}(x, 0^r)$, and correspondingly querying $0^r$ in the next round, where $r = |x|+1, |x|+2, |x|+3, \ldots$).

Here, note that, during the above searching 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 given above switches back to outputting $L_{\mathrm{conv}(x,0^{|x|})}$.

(c) Suppose by way of contradiction that an automatic iterative learner $M$ using feedback queries learns $\mathcal{L}$. Thus, by Lemma 11, there is a locking sequence for $M$ on $L_0$, that is, there exists a $\sigma$ such that the conjecture of $M$ does not change beyond $\sigma$ on any text for $L_0$ which starts with $\sigma$ (see [5, 32]). Without loss of generality assume that $\sigma$ contains at least one string. Let $x$ be such that $L_{\mathrm{conv}(x,0^{|x|})} = \{x\} \cup \mathit{content}(\sigma)$. 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 $\mathit{content}(\sigma)$ 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 $L_{\mathrm{conv}(x,0^{|x|})}$ and $L_{\mathrm{conv}(x,0^n)}$. $\square$

Let $C(x)$ denote the plain Kolmogorov complexity [29], that is, the length of the smallest program $p$ such that $U(p) = x$, where $U$ is some fixed Universal Turing machine.

**Theorem 19.** *Let $i, j \geq 1$. Let the alphabet for the languages be $\Sigma = \{0,1\}$. Let $\mathcal{L} = \{F \subseteq \Sigma^* : |F| = i+1\}$. Then,*

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

*(b) No learner with $(i-1)$-bounded example memory and using $j$-feedback queries per round 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 long-term 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 queries. 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 [29] 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 long-term memory and $M$'s current datum and the answers to the feedback queries. Up to an additive constant independent of $k$: the long-term memory has Kolmogorov complexity at most $(i-1) \cdot k$; the current datum has Kolmogorov complexity at most $k$; the feedback answers have 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 arbitrarily large. □

Note that the proof of part (b) of Theorem 19 also shows that the class $\mathcal{L}$ cannot be learnt using feedback queries and constant-length long-term memory.

The following gives a class which can be learnt using either bounded example memory of size $k$ or using at most $k$ feedback queries, but not using smaller bounded example memory size or smaller number of feedback queries. Although Theorem 20 also uses cardinality techniques, the overall proof method is quite different and the resulting class can be learnt with constant-length long-term memory.

**Theorem 20.** *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.*
(e) *An automatic learner using a constant amount of long-term memory can learn $\mathcal{L}$ without queries of any type.*

**Proof.** Note that $\mathcal{L}$ is an automatic family, as one can use the index set $\{\text{conv}(x_0, x_1, \ldots, x_k) : (\exists n)(\forall i \leq k)\, [x_i = \# \text{ or } x_i \in 0^* \text{ and } [(k+1)n \leq |x_i| < (k+1)(n+1)]]\}$, which is clearly a regular set. Then, $\mathcal{L}$ is the family of languages $L_{\text{conv}(x_0, x_1, \ldots, x_k)} = \{x_i : i \leq k, x_i \neq \#\}$.

(a) The initial conjecture and long-term memory of the learner are $\emptyset$. The automatic learner memorises its inputs as long as it sees at most $k$ elements. If the input element is # or is in the long-term memory, then the learner outputs ?. Otherwise (the input element is not in the long-term 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) The initial conjecture of the learner is $\emptyset$. On input $\#$, the learner outputs ?. On input $0^m$, the learner first finds an $n$ such that $(k+1)n \leq m < (k+1)(n+1)$. Note that this can be done automatically. The automatic learner then queries the elements in $\{0^\ell : (k+1)n \leq \ell < (k+1)(n+1), \ell \neq m\}$, and outputs the conjecture corresponding to the set $S \cup \{0^m\}$, where $S$ is the set of queries answered positively.

(c) Suppose by way of contradiction that an automatic learner learns $\mathcal{L}$ using at most $(k-1)$ memory elements. Let $F \in \mathcal{L}$ be of minimal cardinality such that, for some input segment $\sigma$ with *content*$(\sigma) = F$, the long-term memory of the learner after seeing $\sigma$ is a proper subset of $F$. Suppose $S_F$ is the long-term memory. Thus, $S_F$ is a proper subset of $F$ and for every input sequence $\sigma'$ with content $S_F$, the long-term memory of the learner after seeing $\sigma'$ is $S_F$. 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 $\tau$ is some segment with *content*$(\tau) = S_F$.

(d) Suppose by way of contradiction otherwise. Let $\sigma$ be an initial sequence of $\#^\infty$ such that the feedback learner outputs the conjecture for $\emptyset$ on $\sigma$. Let $x$ be any member of $\{0\}^*$. 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 to learn the input text on at least one of the texts $\sigma \circ y \circ x \circ x \circ x \circ x \circ \ldots$ and $\sigma \circ x \circ x \circ x \circ x \circ \ldots$, as the learner cannot remember whether it has seen $y$.

(e) The learner starts with a hypothesis for $\emptyset$, and in its long-term memory keeps track, for each number $\ell < k+1$, whether it has seen a string of length $(k+1) \cdot n + \ell$ for some $n$. Whenever a datum of the form $0^{(k+1)n+m}$ is observed, the learner updates the long-term memory accordingly, determines the number $n$ and conjectures the (least index for the) set $\{0^{(k+1)n+\ell} : \ell < k+1$ and $\ell$ is in the long-term memory$\}$. The learner outputs ? in the case that it observes $\#$. Eventually the learner sees the last of the strings $0^{(k+1)n+m}$ on the input and due to the long-term memory it conjectures the right hypothesis; from now on, all hypotheses of the learner are either the least index for the correct set in the hypothesis space or ?. Hence the learner learns $\mathcal{L}$. $\square$

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. We do not yet know if this is optimal.

## 5 Learning using a marked memory space

An automatic learner using a marked memory space of type 1 is a generalisation of a learner using feedback queries. Hence it follows from Theorem 16 that every automatic class satisfying Angluin's tell-tale condition can be learnt by an automatic learner with marked memory space of type 1 and datum-length long-term memory. 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 21.** *Every automatic family satisfying Angluin's tell-tale condition has an automatic learner using hypothesis-length long-term memory and a marked memory space of type 2.*

**Proof.** Suppose $\Sigma$ is the alphabet for the languages, and $\{L_\alpha : \alpha \in I\}$ is an automatic family which satisfies Angluin's tell-tale condition. Without loss of generality assume that for all distinct $\alpha, \alpha'$ in $I$, $L_\alpha \neq L_{\alpha'}$.

The aim of the learner is to search for an $\alpha$ such that a tell-tale set for $L_\alpha$ is contained in the input, and the input language is contained in $L_\alpha$. The idea of the proof is similar to that of Theorem 16. 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 16.

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, $\alpha$ ranging over the indices and $y$ ranging over $\Sigma^*$. Let $succ(\alpha)$ denote the length-lexicographic successor of index $\alpha$ (in $I$) and $succ(y)$ denote the length-lexicographic successor of datum $y$ (in $\Sigma^*$). Without loss of generality, suppose $\varepsilon$ is the length-lexicographically least element of $I$.

The long-term memory holds the hypothesis $\alpha$ and the pointer $y$. Note that $y$ is only used to test whether all elements of a tell-tale set are in the input seen so far. Recall from Proposition 2 that there is a constant $c$ such that the strings in $L_\alpha$ of length at most $|\alpha| + c$ forms a tell-tale set for $L_\alpha$. Hence, the length of the long-term memory $\mathrm{conv}(\alpha, y)$ is bounded by the length of the hypothesis $\alpha$ 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 $\alpha$ and $y$ are $\varepsilon$. In the marked memory space, an element of the form $(z, 2)$ is used for remembering the data seen so far; an element of the form $(\alpha, 1)$ is used to denote that $L_\alpha$ 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 $L_\alpha$ is contained in the input seen so far); an element of the form $(\alpha, 0)$ is used to denote that hypotheses length-lexicographically smaller than $\alpha$ have been considered earlier. This allows us to consider each possible hypothesis arbitrarily often until a correct hypothesis is found. In each round of the algorithm the following is done:

**Beginning of the round.**

1. Current long-term memory is $\mathrm{conv}(\alpha, y)$ and the new input is $x$.
   Query whether $(\alpha, 0)$, $(\alpha, 1)$ and $(y, 2)$ are in the marked memory space.
2. If $(\alpha, 0)$ is not in marked memory space,
   then place $(\alpha, 0)$ in the marked memory space, let $\alpha = \varepsilon$, let $y = \varepsilon$ and go to step 6.
3. If $(\alpha, 1)$ is in the marked memory space 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 in the marked memory space 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. Place $(x, 2)$ in the marked memory space. If $x \neq \#$, then place $(\beta, 1)$ in the marked memory space for all $\beta$ where $x \notin L_\beta$.

7. Output the hypothesis $\alpha$.
8. The new long-term memory is the new value of $\mathrm{conv}(\alpha, y)$.

**End of the round.**

Placing of elements in the marked memory space is done after the queries on the old marked memory space has been done and the updates depend on the old marked memory space.

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 $L_\gamma$ is the language to be learnt. Further suppose all elements of $L_\gamma$ which are of length at most $|\gamma| + c$ have already shown up in the input. If the current index $\alpha$ is not equal to $\gamma$, then either during the inference process some $x \notin L_\alpha$ shows up, causing $(\alpha, 1)$ to be placed in the marked memory space, and thus causing $\alpha$ 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 $\alpha$ will be updated to $succ(\alpha)$ by step 4, as $z$ will never be observed and, therefore, $(z, 2)$ will never be placed in the marked memory space. Furthermore, when $\alpha$ takes a new value not taken before, step 2 will reset $\alpha$ to $\varepsilon$ and $\alpha$ will cycle to all the values again until it eventually reaches the value of $\gamma$. Then steps 2 and 3 will not apply. Furthermore, as all the members of $L_\gamma$ of length at most $|\gamma| + c$ have already been observed, step 4 will also not apply. Therefore, the value of $\alpha$ will no longer change and $y$ will eventually stabilise. So the algorithm would converge to $\alpha = \gamma$ and will conjecture $L_\gamma$ from that point of time onwards. Thus, $L_\gamma$ is learnt by the algorithm. □

Note that the learner in the above theorem can be made iterative, if one uses arbitrary hypothesis space rather than as given by the class being learnt. For this, one chooses the hypothesis space $H_{\mathrm{conv}(\alpha,y)} = L_\alpha$. Note here that the long-term memory of the learner in the above proof also converges, and thus if one takes the hypothesis of the learner as its long-term memory, then the learner becomes iterative.

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. This result also shows that just hypothesis-length long-term memory (even with feedback queries) is not enough to learn all automatic families. Contrast this with Theorem 16, where datum-length long-term memory along with one feedback query every round was enough to learn all the automatic families satisfying Angluin's tell-tale condition.

The following theorem uses the class used in [20] to separate automatic learnability using datum-length long-term memory from automatic learnability using hypothesis-length long-term memory. The diagonalisation method used here is a generalisation of the technique used in [20].

**Theorem 22.** *Let $\mathcal{L}$ be the class consisting of $L_0 = \{0\}^+$ and all finite sets $L_\alpha = \{0^i : \alpha(i) = 1\}$, for $\alpha \in \{1\} \cdot \{0,1\}^*$.*

*Then, $\mathcal{L}$ has an automatic learner using datum-length long-term memory. However, an automatic learner using hypothesis-length long-term memory plus a marked memory space of type 1, cannot learn $\mathcal{L}$.*

**Proof.** Note that $\varepsilon \in L_\alpha$, for all $\alpha \in \{1\} \cdot \{0,1\}^*$. As $\mathcal{L}$ satisfies Angluin's tell-tale condition, learnability of $\mathcal{L}$ by an automatic learner using datum-length long-term memory follows from Theorem 3.

Now consider any automatic learner with type 1 marked memory space, but with only hypothesis-length long-term memory. Using standard locking sequence methods [5, 32], one can show that, when learning the language $\{0\}^+$, there is a sequence $\sigma$ and a constant $c$ such that the learner, on any text for $\{0\}^+$ starting with $\sigma$, does not change the conjecture after having processed $\sigma$ and never takes a long-term memory value of length longer than $c$ (this holds as the long-term memory of the learner is bounded in length by the length of its hypothesis (plus a constant), which does not change beyond $\sigma$). Furthermore, there is a constant $c'$ such that, when reading an input $0^n$, the learner queries or places in the marked memory 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 datum in $\sigma$ and $m$ being the length of the longest string in the marked memory space after the learner has seen $\sigma$. 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 (because the number of possible long-term memory values on this text is finite, as they are bounded in length by $c$). Let this long-term memory value be $mem$. Let $k$ be large enough such that long-term 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 in marked memory space (on input text $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$), gets in the marked memory space 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 $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots \circ 0^{n_k} \circ \varepsilon \circ \varepsilon \circ \varepsilon \cdots$. As this input is in $\mathcal{L}$, the learner converges on this input text to some conjecture (say $\beta$) and does not query or places in marked memory space 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 long-term 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 long-term memory $mem$ infinitely often on the input $\sigma \circ 0^{n_1} \circ 0^{n_2} \circ \ldots$; thus, the long-term 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 long-term memory and the answer to the queries asked.

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

It is open whether an automatic learner using hypothesis-length long-term memory can be made iterative [20]. Theorem 23 deals with the counterpart of this problem when we permit a marked memory space of type 1 to the simulator.

**Theorem 23.** *If a class $\mathcal{L}$ has an automatic learner using hypothesis-length long-term memory 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 long-term memory over $\Gamma^*$. Without loss of generality assume that the initial hypothesis of $M$ is not ?. Let $H_{\text{conv}(i,g)} = L_i$ for all $i \in I$ and $g \in \Gamma^*$. For a current datum $x$ and long-term memory $g$ and hypothesis $i$, let

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

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

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

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

Note that $N$ can find out using a hypothesis query whether $F(\text{conv}(i,g),x)$ had been conjectured previously. As $F$ is automatic, so is $N$. Furthermore, assume that $N(x_0 \circ x_1 \circ \ldots \circ x_n) = \text{conv}(i_n, g_n)$ 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 $\tau_n$ 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 $(i_n, g_n)$. We now 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 $N(\text{conv}(i_{n-1}, g_{n-1}), x_n)$ is defined via the first clause in its definition, then one can choose $\tau_n$ to be empty sequence. If $N(\text{conv}(i_{n-1}, g_{n-1}), x_n)$ is defined via the second clause in its definition, then suppose $F(\mathit{conv}(i_{n-1}, g_{n-1}), x_n) = \text{conv}(i_m, g_m)$, which was conjectured by $N$ after $x_0 \circ x_1 \circ \ldots \circ x_m$ (here $m$ could be $-1$ and $F(\mathit{conv}(i_{n-1}, g_{n-1}), x_n)$ is then the initial hypothesis of $N$). One then chooses $\tau_n$ 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 hypotheses 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 $\mathrm{conv}(i_n, g_n)$ also converges to a pair $\mathrm{conv}(i, g)$; 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. $\square$


## Acknowledgements

A preliminary version of this paper appeared in CiE 2011 [10]. We would like to thank the anonymous referees of the conference and this journal 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 2000, 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. Accepted for *Information and Computation.* Preliminary version appeared in *Language and Automata Theory and Applications,* 5th International Conference, LATA 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, Proceedings. Springer LNCS 6735:31–40, 2011.
11. François Denis, Aurélien Lemay and Alain Terlutte. Some classes of regular languages identifiable in the limit from positive data. *Sixth International Colloquium on Grammatical Inference: Algorithms and Applications* (ICGI), pages 63–76, Springer LNCS 2484, 2002.
12. David Epstein, James Cannon, Derek Holt, Silvio Levy, Michael Paterson and William Thurston. *Word Processing in Groups.* Jones and Bartlett Publishers, Boston, Massachusetts, 1992.

13. Henning Fernau. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711, 2003.
14. Rusins Freivalds, Efim Kinber and Carl H. Smith. On the impact of forgetting on learning machines. *Journal of the ACM*, 42:1146–1168, 1995.
15. E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
16. Tom Head, Satoshi Kobayashi and Takashi Yokomori. Locality, reversibility, and beyond: learning languages from positive data. *Algorithmic Learning Theory*, Ninth International Conference, ALT 1998, Proceedings. Springer LNAI 1501:191–204, 1998.
17. Bernard R. Hodgson. *Théories décidables par automate fini.* Ph.D. thesis, University of Montréal, 1976.
18. Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.
19. 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.
20. Sanjay Jain, Qinglong Luo and Frank Stephan. Learnability of automatic classes. To appear in Journal of Computer and System Sciences. Preliminary version appeared in *Language and Automata Theory and Applications*, 4th International Conference, LATA 2010, Proceedings. Springer LNCS 6031:321-332, 2010.
21. Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank Stephan. On automatic families. *Proceedings of the 11th Asian Logic Conference*, ALC 2009, in Honor of Professor Chong Chitat's 60th birthday, pages 94–113. World Scientific, 2011.
22. Sanjay Jain, Eric Martin and Frank Stephan. Robust learning of automatic classes of languages. *Algorithmic Learning Theory*, 22nd International Conference, ALT 2011, Proceedings. Springer LNAI 6925: 55–69, 2011.
23. Sanjay Jain, Daniel N. Osherson, James S. Royer and Arun Sharma. *Systems That Learn.* MIT Press, 2nd Edition, 1999.
24. Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity.* 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 1993, pages 127–136. ACM Press, 1993.
27. Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.
28. 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.
29. Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Third Edition, Springer, 2008.
30. André Nies. Describing groups. *Bulletin of Symbolic Logic*, 13(3):305–339, 2007.

31. André Nies and Pavel Semukhin. Finite automata presentable Abelian groups. *Annals of Pure and Applied Logic*, 161:458–467, 2009.
32. 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.
33. Lenny Pitt. Inductive inference, DFAs, and computational complexity. *Analogical and Inductive Inference, Proceedings of the Second International Workshop*, AII 1989. Springer LNAI 397:18–44, 1989.
34. Sasha Rubin. *Automatic Structures.* Ph.D. Thesis, The University of Auckland, 2004.
35. Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.
36. Todor Tsankov. The additive group of the rationals does not have an automatic presentation. *The Journal of Symbolic Logic*, 76(4):1341–1351, 2011.
37. Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition.* Cambridge, Massachusetts, The MIT Press, 1980.
38. Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik* (EIK), 12:93–99, 1976.