Automatic Learning from Positive Data and Negative Counterexamples

Sanjay Jain[⋆1] and Efim Kinber[2]

[1] School of Computing, National University of Singapore, Singapore 117543. Email: sanjay@comp.nus.edu.sg

[2] Department of Computer Science, Sacred Heart University, Fairfield, CT 06432-1000, U.S.A. Email: kinbere@sacredheart.edu

Abstract. We introduce and study a model for learning in the limit by finite automata from positive data and negative counterexamples. The focus is on learning classes of languages with a membership problem computable by finite automata (so-called automatic classes). We show that, within the framework of our model, finite automata (automatic learners) can learn all automatic classes when memory of a learner is restricted by the size of the longest datum seen so far. We also study capabilities of automatic learners in our model with other restrictions on the memory and how the choice of negative counterexamples (arbitrary, or least, or the ones whose size is bounded by the longest positive datum seen so far) can impact automatic learnability.

1 Introduction

In the paper [JLS10], the authors introduced an "automatic" variant of the well-known Gold's model for learning in the limit from positive data: the family of target languages is computable by a finite automaton (automatic family), and a learner is a finite automaton itself (automatic learning). More specifically, a family of target languages is defined by a regular index set, and the membership problem in these languages is regular in the sense that one finite automaton recognizes a combination (so called "convolution") of an index and a word if and only if the word is in the language defined by the index. They also considered three different natural types of limits on the size of the (long-term) memory available to the learner before outputting the next conjecture: (a) memory is bounded by the size of the longest positive input datum seen so far (plus a constant); (b) memory is bounded by the size of the current hypothesis (plus a constant); (c) the learner can store in the memory the last hypothesis only.

The authors of [JLS10] established that automatic learners are much weaker than unrestricted recursive learners — even when learning automatic classes. In particular, not every automatic class is automatically learnable. Moreover, they showed the following modification of D. Angluin's result from [Ang80]: An automatic class is learnable by a recursive learner iff it satisfies Angluin's

⋆ Supported in part by NUS grant number C-252-000-087-001.

tell-tale condition. The authors of [JLS10] also obtained a number of interesting results showing differences between automatic learners on automatic classes with different limitations on the memory mentioned above, as well as impact on the learners of the requirement of *consistency* — when every conjecture must be consistent with the input seen so far.

Since automatic learners are not able to learn many automatic classes from positive data alone, it is natural to ask: under which conditions *all* automatic classes of languages are automatically learnable? In [JK08], the authors introduced and motivated a notion of learning languages in the limit from full positive data and a finite number of negative counterexamples provided to the learner whenever it's hypothesis contains data that is not a part of the target language. This approach to learning in the limit arguably is more natural than learning just from positive examples — for instance, children learning languages get corrected when using wrong words [HPTS84] (yet, as is probably the case in natural learning processes, in this model of learning, the learner does not get *all* negative examples). In a sense, this model combines two different and popular approaches to learning in the limit — learning languages from positive data and learning concepts from subset queries and counterexamples ([Ang88]), whereas none of these two approaches by itself adequately represents the process of language acquisition. In [JK08], the authors considered three different types of negative counterexamples provided to the learner: (a) arbitrary, (b) least, and (c) bounded by the size of the largest positive datum seen so far (the latter type is motivated by possible computational limitations on the "teacher" providing counterexamples). In this paper, we adapt the notion of learning with negative counterexamples to automatic learning of automatic classes. Our major result (Theorem 8) is that such automatic learners, even when required to be *iterative* (that is, whose memory stores just the last hypothesis), can learn *every* automatic class! On the other hand, interestingly, we have not been able to make such learners consistent with data seen so far. Yet, Theorem 9 shows that consistency can be achieved if the learners always receive the least negative counterexamples. On the other hand, as it follows from a result in [JLS10], there are automatic classes that cannot be learned even by non-automatic learners if the size of counterexamples is bounded by the size of the longest positive input datum seen so far (Theorem 11). Still, with this bound on the size of counterexamples, automatic learners with memory limited by the size of the longest positive input datum seen so far can learn automatic classes consisting only of infinite languages (Theorem 10).

We also show that some automatic classes cannot be learned automatically using bounded negative counterexamples with memory limited by the size of the current hypothesis (and, thus, when only the last hypothesis can be stored in the memory) — see Theorem 13, but can be learned automatically with memory limited by size of the longest positive datum seen so far even without negative counterexamples.

Theorem 14 shows the advantage of negative counterexamples, even for automatic iterative learners, compared to not having negative counterexamples, even

for unrestricted recursive learners. Our last result (Theorem 15) shows that not every automatic class can be learned automatically and *monotonically* — that is, when every next conjecture includes the positive data covered by the previous one — even if the least negative counterexamples are provided.

A number of related problems remain open. In particular, we do not know whether every automatic family can be consistently learnt using arbitrary negative counterexamples. We also do not know whether iterative automatic learners or automatic learners with memory bounded by hypothesis size, receiving bounded negative counterexamples, can learn all automatic classes having only infinite languages. Some relations between various memory bounds on automatic learners using bounded negative counterexamples are also open.

2 Preliminaries

The set of natural numbers is denoted by N. Let Σ denote a finite alphabet. The set of all strings over the alphabet Σ is denoted by Σ^*. The empty string is denoted by ϵ. A string of length n is treated as a function from $\{0, 1, \ldots, n-1\}$ to Σ. Thus, $x = x(0)x(1), \ldots, x(n-1)$, where x is a string of length n. The length of a string x is denoted by $|x|$. We say that a string w is length-lexicographically smaller than string w' (written $w <_{ll} w'$) iff $|w| < |w'|$ or $|w| = |w'|$ and w is lexicographically below w' (where we assume some canonical ordering of elements of Σ). We let $w \leq_{ll} w'$ denote that either $w = w'$ or $w <_{ll} w'$. We let $\mathrm{succ}_S(w)$ denote the least w' such that $w <_{ll} w'$, and $w' \in S$ (if there is no such string, then we let $\mathrm{succ}_S(w)$ to be undefined).

We let $\emptyset, \subseteq$ and $\subset$ respectively denote emptyset, subset and proper subset. We let $\mathrm{card}(S)$ denote the cardinality of set S. When considering sets of natural numbers, we let $\max(S), \min(S)$ respectively denote the maximum and minimum of a set S, where $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. When we are considering sets of strings S, we let $\max(S)$ and $\min(S)$ be the length-lexicographically largest and smallest string in S respectively, where if $S = \emptyset$, then we take $\max(S) = \#$ (where $\#$ is a special pause symbol, see below).

Convolution of two strings $x = x(0)x(1)\ldots x(n-1)$ and $y = y(0)y(1)\ldots y(m-1)$, denoted $conv(x, y)$, is defined as follows. Let x', y' be strings such that $x'(i) = x(i)$ for $i < n$, $x'(i) = \#$ for $n \leq i < \max(\{m, n\})$, $y'(i) = y(i)$ for $i < m$, and $y'(i) = \#$ for $m \leq i < \max(\{m, n\})$, where $\# \notin \Sigma^*$ is a special padding symbol. Thus, x', y' are obtained from x, y by padding the smaller string with $\#$'s. Then, $conv(x, y) = z$, where $|z| = \max(\{m, n\})$ and $z(i) = (x'(i), y'(i))$, for $i < \max(\{m, n\})$. Note that z is a string over the alphabet $(\Sigma \cup \{\#\}) \times (\Sigma \cup \{\#\})$. Similarly, one can define convolution of more than two strings. Intuitively, giving a convolution of two strings to a machine means giving two strings in parallel, with the shorter string being padded with $\#$s.

We say that an n-ary relation R is *automatic*, if $\{conv(x_1, x_2, \ldots, x_n) : (x_1, x_2, \ldots, x_n) \in R\}$ is regular. Similarly, an n-ary function f is automatic if $\{conv(x_1, x_2, \ldots, x_n, y) : f(x_1, x_2, \ldots, x_n) = y\}$ is regular.

A family of languages, $(L_\alpha)_{\alpha\in I}$ is said to be an *automatic family* if the index set I is regular and the set $\{conv(\alpha, x) : \alpha \in I, x \in L_\alpha\}$ is regular. Here the sets L_α are sets of strings over some finite alphabet. We often identify an automatic family $(L_\alpha)_{\alpha\in I}$ with the class $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where the indexing is implicit. We say that the automatic family $(L_\alpha)_{\alpha\in I}$ is 1–1 (or the indexing is 1–1), if for all $\alpha, \beta \in I$, $L_\alpha = L_\beta$ implies $\alpha = \beta$.

It can be shown that any family, relation or function that is first order definable using other automatic relations or functions is itself automatic.

Lemma 1 ([BG00], [KN95]) *Any relation that is first-order definable from existing automatic relations is automatic.*

We often implicitly use the above fact in our proofs. The present work considers learnability of automatic families in the presence of counterexamples. For this, let us consider a definition of a learner. This definition is given in a form slightly different from the one traditional in inductive inference. When there are no memory restrictions, this definition turns out to be essentially the same as the traditional definition. We use a different form to make it easier to consider automatic learners.

A *text* T is a mapping from N to $\Sigma^*\cup\{\#\}$. Here $\# \notin \Sigma^*$ denotes pauses in the presentation of data. We let $T[n]$ denote the initial sequence of T of length n, that is, $T[n] = T(0)T(1)\ldots T(n-1)$. The content of a text T, denoted content(T), is $\{T(i) : i \in N\} - \{\#\}$. Similarly, content$(T[n]) = \{T(i) : i < n\} - \{\#\}$. We let σ range over initial sequences of texts. We let Λ denote the empty sequence. We let $SEQ(S)$ denote the set of all finite sequences σ such that content$(\sigma) \subseteq S$. We let $\sigma\diamond\tau$ denote the concatenation of two sequences σ and τ. By abusing notation, for $x \in \Sigma^*\cup\{\#\}$, we use $\sigma\diamond x$ to denote the concatenation of σ with the sequence containing just one element x.

Definition 2 Suppose Σ, Δ are finite alphabets used for languages and memory of learners respectively, where $\# \notin \Sigma^*$. Suppose J is a regular index set (over some finite alphabet) for the hypothesis space used by the learner. Below ? is a special symbol not in J, which stands for "repeat the previous conjecture."

(a) A *learner* is a mapping from $\Delta^* \times (\Sigma^* \cup \{\#\})$ to $\Delta^* \times (J \cup \{?\})$. A learner has an initial memory $mem_0 \in \Delta^*$, and initial hypothesis $hyp_0 \in J \cup \{?\}$.
(b) Suppose a learner $\mathbf{M}$ with initial memory mem_0 and initial hypothesis hyp_0 is given. Below, σ is a sequence over $\Sigma^*\cup\{\#\}$ and $x \in \Sigma^*\cup\{\#\}$. We extend the definition of $\mathbf{M}$ to sequences by inductively defining
$\mathbf{M}(\Lambda) = (mem_0, hyp_0)$;
$\mathbf{M}(\sigma\diamond x) = \mathbf{M}(mem, x)$, where $\mathbf{M}(\sigma) = (mem, hyp)$, for some $hyp \in J\cup\{?\}$. Additionally, for $|\sigma| \geq 1$, we inductively define $\mathbf{M}(mem, \sigma\diamond x) = \mathbf{M}(mem', x)$, where $\mathbf{M}(mem, \sigma) = (mem', hyp')$, for some $hyp' \in J \cup \{?\}$.
(c) We say that $\mathbf{M}$ converges on a text T to a hypothesis β (written: $\mathbf{M}(T)\downarrow_{hyp} = \beta$) iff there exists a t such that,
(i) $\mathbf{M}(T[t]) \in \Delta^* \times \{\beta\}$, and
(ii) for all $t' \geq t$, $\mathbf{M}(T[t]) \in \Delta^* \times \{\beta, ?\}$.

Intuitively, $\mathbf{M}(\sigma) = (mem, hyp)$ means that the memory and hypothesis of the learner $\mathbf{M}$ after having seen the sequence σ are mem and hyp respectively. We can think of a learner as receiving a text T for the language L, one element at a time. At each input, the learner updates its previous memory, and outputs a new conjecture (hypothesis), where ? denotes repeating the previous hypothesis. If the sequence of hypotheses converges to a grammar for L, then we say that the learner **TxtEx**-learns the language L from the text T (here **Ex** denotes "explains", and **Txt** denotes learning from text). Now we define learnability formally.

Definition 3 (Based on Gold [Gol67])
Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a target class, and $\mathcal{H} = \{H_\beta : \beta \in J\}$ is a hypothesis space, where both $\mathcal{L}$ and $\mathcal{H}$ are automatic families of languages.

(a) We say that $\mathbf{M}$ **TxtEx**-learns the language L (using hypothesis space $\mathcal{H}$) from a text T iff $\mathbf{M}(T)\downarrow_{hyp} = \beta$ such that $H_\beta = L$.
(b) We say that $\mathbf{M}$ **TxtEx**-learns a language L (using hypothesis space $\mathcal{H}$) iff $\mathbf{M}$ **TxtEx**-learns L from all texts for the language L (using hypothesis space $\mathcal{H}$).
(c) We say that $\mathbf{M}$ **TxtEx**-learns $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) iff $\mathbf{M}$ **TxtEx**-learns all languages in $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).
(d) **TxtEx** $= \{\mathcal{L} : (\exists \mathbf{M})[\mathbf{M}$ **TxtEx**-learns $\mathcal{L}$ using some hypothesis space$]\}$.

We drop the reference to "using hypothesis space $\mathcal{H}$", when the hypothesis space is clear from the context. A hypothesis space $\mathcal{H}$ is said to be *class preserving* [LZ93] for learning a class $\mathcal{L}$ if $\mathcal{L} = \mathcal{H}$. A hypothesis space $\mathcal{H}$ is said to be *class comprising* [LZ93] for learning a class $\mathcal{L}$ if $\mathcal{L} \subseteq \mathcal{H}$.

Definition 4 Suppose a learner $\mathbf{M}$ using an automatic family $\mathcal{H} = \{H_\beta : \beta \in J\}$ as the hypothesis space is given.

(a) [JLS10] A learner $\mathbf{M}$ is called an *automatic learner* iff its graph is automatic. That is, $\{conv(mem, x, mem', hyp') : \mathbf{M}(mem, x) = (mem', hyp')\}$ is regular.
(b) [Wie76] $\mathbf{M}$ is said to be *iterative* iff, for all finite sequences σ, $\mathbf{M}(\sigma) = (mem, hyp)$ implies $mem = hyp$.
$\mathbf{M}$ is said to be *word-size* memory bounded iff there exists a constant c such that for all finite sequences σ, $\mathbf{M}(\sigma) = (mem, hyp)$ implies $|mem| \leq \max(\{|w| : w \in \text{content}(\sigma)\}) + c$.
$\mathbf{M}$ is said to be *hypothesis-size* memory bounded iff there exists a constant c such that for all finite sequences σ, $\mathbf{M}(\sigma) = (mem, hyp)$ implies $|mem| \leq |hyp| + c$.
(Note that if a learner is iterative then its memory is hypothesis-size bounded, but hypothesis-size bound on the memory does not imply that a learner is iterative.)
(c) [Bār74] $\mathbf{M}$ is said to be *consistent* iff, for all finite sequences σ, if $\mathbf{M}(\sigma) = (mem, hyp)$ with $hyp \neq ?$, then $\text{content}(\sigma) \subseteq H_{hyp}$.

(d) [Wie90] $\mathbf{M}$ is said to be *monotonic* iff for all texts T, for all $t < t'$, if $\mathbf{M}(T[t]) = (mem, hyp), \mathbf{M}(T[t']) = (mem', hyp')$, $hyp \neq ?$, $hyp' \neq ?$, then $\text{content}(T) \cap H_{hyp} \subseteq \text{content}(T) \cap H_{hyp'}$.

Note that the above constraints are required even on texts for languages outside the class $\mathcal{L}$. Note that when a learner gets positive data only, then a learner's conjecture may contain data that is not in the target language. In this situation, the learner may not be able to know that it went "beyond" the target language, as it does not receive any negative data. To address this issue, Jain and Kinber [JK08] considered the notion of learning with negative counterexamples. In this, for every hypothesis, a learner receives as input a negative counterexample, if there exists any. Thus, intuitively, the learner gets two input texts: one for positive data as above, and another for negative counterexamples.

Definition 5 (Based on [JK08]) Suppose Σ, Δ are finite alphabets used for languages and memory of learners respectively, where $\# \notin \Sigma^*$. Suppose J is a regular index set for the hypothesis space used by the learner.

(a) A *learner* learning using negative examples is a mapping from $\Delta^* \times (\Sigma^* \cup \{\#\}) \times (\Sigma^* \cup \{\#\})$ to $\Delta^* \times (J \cup \{?\})$.
A learner has an initial memory $mem_0 \in \Delta^*$, and initial hypothesis $hyp_0 \in J \cup \{?\}$.
(b) Suppose a learner $\mathbf{M}$ with initial memory mem_0 and initial hypothesis hyp_0 is given. We extend the definition of $\mathbf{M}$ to sequences as follows. Below, σ, τ are sequences over $\Sigma^* \cup \{\#\}$ with $|\sigma| = |\tau|$, and $x, y \in \Sigma^* \cup \{\#\}$.
$\mathbf{M}(\Lambda) = (mem_0, hyp_0)$;
$\mathbf{M}(\sigma \diamond x, \tau \diamond y) = \mathbf{M}(mem, x, y)$, where $\mathbf{M}(\sigma, \tau) = (mem, hyp)$, for some $hyp \in J \cup \{?\}$.
Additionally, for $|\sigma| = |\tau| \geq 1$, we inductively define $\mathbf{M}(mem, \sigma \diamond x, \tau \diamond y) = \mathbf{M}(mem', x, y)$, where $\mathbf{M}(mem, \sigma, \tau) = (mem', hyp')$, for some $hyp' \in J \cup \{?\}$.
(c) We say that $\mathbf{M}$ converges on text T with negative counterexample text T' to a hypothesis β (written: $\mathbf{M}(T, T')\downarrow_{hyp} = \beta$) iff there exists a t such that
(i) $\mathbf{M}(T[t], T'[t]) \in \Delta^* \times \{\beta\}$, and
(ii) for all $t' \geq t$, $\mathbf{M}(T[t], T'[t]) \in \Delta^* \times \{\beta, ?\}$.

Intuitively, $\mathbf{M}(\sigma, \tau) = (mem, hyp)$ means that the memory and the hypothesis of the learner $\mathbf{M}$ after having seen the sequence σ and the negative counterexample sequence τ is mem and hyp, respectively. Below, **NC** in the criteria names denotes learning from negative counterexample. **B** and **L** in $\mathbf{BNC}, \mathbf{LNC}$, denote "bounded" and "least".

Definition 6 (Based on [JK08]) Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is a target class, and $\mathcal{H} = \{H_\beta : \beta \in J\}$ is a hypothesis space, where both $\mathcal{L}$ and $\mathcal{H}$ are automatic families of languages over an alphabet Σ. Below, for ease of notation, we take $H_? = \emptyset$.

(a) (i) We say that T' is a *counterexample text* for $\mathbf{M}$ on an input text T for a language L iff for all n, where $\mathbf{M}(T[n], T'[n]) = (mem, hyp)$,
if $H_{hyp} \subseteq L$, then $T'(n) = \#$, and
if $H_{hyp} \not\subseteq L$, then $T'(n) \in H_{hyp} - L$.

(ii) We say that T' is a *least-counterexample text* for $\mathbf{M}$ on an input text T for a language L iff for all n, where $\mathbf{M}(T[n], T'[n]) = (mem, hyp)$,
if $H_{hyp} \subseteq L$, then $T'(n) = \#$, and
if $H_{hyp} \not\subseteq L$, then $T'(n) = \min(H_{hyp} - L)$.

(iii) We say that T' is a *bounded counterexample text* for $\mathbf{M}$ on an input text T for a language L iff for all n, where $\mathbf{M}(T[n], T'[n]) = (mem, hyp)$,
if $H_{hyp} \cap \{x \in \Sigma^* : x \leq \max(\text{content}(T[n]))\} \subseteq L$, then $T'(n) = \#$, and
if $H_{hyp} \cap \{x \in \Sigma^* : x \leq \max(\text{content}(T[n]))\} \not\subseteq L$, then $T'(n) \in H_{hyp} \cap \{x \in \Sigma^* : x \leq \max(\text{content}(T[n]))\} - L$.
(That is, the size of a counterexample is bounded by the size of the longest positive datum seen so far; consequently, if the size of the least counterexample to the current conjecture exceeds this bound, no counterexample is provided.)

(b) We say that $\mathbf{M}$ **NCEx**-learns the language L (using hypothesis space $\mathcal{H}$) iff for all texts T for L, for all counterexample texts T' for $\mathbf{M}$ on input text T, $\mathbf{M}(T, T')\downarrow_{hyp} = \beta$ such that $H_\beta = L$.

(c) We say that $\mathbf{M}$ **NCEx**-learns $\mathcal{L}$ (using hypothesis space $\mathcal{H}$) if it **NCEx**-learns all languages in $\mathcal{L}$ (using hypothesis space $\mathcal{H}$).

(d) **NCEx** $= \{\mathcal{L} : (\exists \mathbf{M})[\mathbf{M}$ **NCEx**-learns $\mathcal{L}$ using some hypothesis space $]\}$.
One can similarly define learnability criteria **LNCEx** and **BNCEx** for learning from least-counterexample or bounded counterexamples.

Furthermore, automatic, consistent, monotonic learning and various memory restricted learning criteria can be similarly defined for learning from counterexamples. Here for word-size memory constraint, we bound the memory by the largest word seen in either the text (for positive data) or the counterexample text. Also, for consistency we require that the learner is consistent with positive examples as well as negative counterexamples, that is, for any text T and corresponding negative counterexample text T', if $\mathbf{M}(T[t], T'[t]) = (mem, hyp)$ with $hyp \neq ?$, then $\text{content}(T[t]) \subseteq H_{hyp}$ and $\text{content}(T'[t]) \cap H_{hyp} = \emptyset$.

We use "**Auto**" in the name of the learning criteria to denote that we require the learners to be automatic. For example, **AutoTxtEx** denotes **TxtEx**-learning by an automatic learner. Similarly, we use **Cons** and **Mon** in the name of the learning criteria to denote that the learners are consistent and monotonic, respectively. Similarly, we use **Word** and **Hyp** in the name of the learning criteria to denote that the memory of the learners is appropriately bounded. For **It** memory restriction, as is common in the literature, we replace the term "**Ex**" in the name of the criterion by "**It**".

For example, **AutoWordNCEx** denotes **NCEx** learnability by a learner which is automatic and word memory size bounded. **AutoMonNCIt** denotes **NCEx** learnability by a learner which is automatic, monotonic and iterative.

3 Results

We begin with an easily provable useful technical proposition.

Proposition 7 *Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic family, where the indexing is 1–1. Then, there exists a constant c such that the following hold.*

(a) [JOPS11] For all $\alpha \in I$ such that L_α is finite, $|\alpha| \leq c + \max(\{|x| : x \in L_\alpha\})$.

(b) For all $\alpha \in I, u \in \Sigma^*$, let $ProbExt(\alpha, u) = \{\beta : L_\alpha \subset L_\beta \subseteq L_\alpha \cup \{x : x \leq_{ll} u\}\}$. Then, for all $\alpha \in I, u \in \Sigma^*$ and $\beta \in ProbExt(\alpha, u)$, $|\beta| \leq \max(\{|\alpha|, |u|\}) + c$.

(c) For all $\alpha \in I$, for all $u \in \Sigma^*$, there exists a $\beta \in I$ such that $|\beta| \leq |u| + c$ and $L_\beta \cap \{x : |x| \leq |u|\} = L_\alpha \cap \{x : |x| \leq |u|\}$.

Intuitively, part (a) of the above proposition says that the indices for finite sets are not too big in an automatic family. Part (b) of the proposition says that if L_α and L_β differ only on strings $\leq_{ll} u$, then the index for β is not much bigger than $\max(\{|\alpha|, |u|\})$. Part (c) of the above proposition says that for any index α and string u, there exists a short β such that L_β is consistent with L_α for strings below u.

Our first major result shows that automatic **NCEx**-learners with word-size memory limit can learn any automatic class.

Theorem 8 *Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ be an automatic family. Then,*

*(a) $\mathcal{L} \in$ **AutoNCIt**. The learner uses a class preserving hypothesis space.*

*(b) $\mathcal{L} \in$ **AutoWordNCEx**. The learner uses the hypothesis space $(H_\alpha)_{\alpha \in I}$, where $H_\alpha = L_\alpha$.*

Proof. Due to space restrictions, we only show part (a). Part (b) can be proven in a way similar to Theorem 10.

Without loss of generality assume that the indexing $(L_\alpha)_{\alpha \in I}$ is 1–1 (otherwise, we can ignore the non-minimal indices of I, which can be automatically determined as (non) minimal indices can be expressed as a first order formula using automatic relations (see Lemma 1)). Furthermore, assume that I is infinite (otherwise, the theorem is trivial). Let c be as in Proposition 7 (for $\mathcal{L}$).

Let i_0 be a special symbol which we take to be length-lexicographically smaller than all members of I. This is for ease of presentation of the proof.

Suppose L is the target language. The aim of the learner $\mathbf{M}$ is to find an α such that $L_\alpha \subseteq L$ and $L \subseteq L_\alpha$. The learner can check if $L_\alpha \subseteq L$ using the counterexamples. However, the learner may not easily be able to check if $L \subseteq L_\alpha$, as it may have forgotten some past data. To overcome this problem is the main aim of the construction.

The learner keeps memory of the form $conv(\alpha, u, \beta, b)$, where $\alpha \in I \cup \{i_0\}$, $\beta \in I$, $u \in \Sigma^* \cup \{\#\}$, and $b \in \{0, 1\}$. In case $\alpha = i_0$ in the memory, then we will have $b = 1$ (that is, the memory will never be of the form $conv(i_0, u, \beta, 0)$).

The hypothesis of the learner is directly linked to its memory: If, $[b = 1$ or $[b = 0$ and $|\beta| \leq \max(\{|\alpha|, |u|\}) + c]]$, then $H_{conv(\alpha,u,\beta,b)} = L_\beta$; otherwise,

$H_{\mathtt{C}conv(\alpha,u,\beta,b)} = L_\alpha$ (note that the above implicitly gives the hypothesis space used by the learner, which is class preserving as $\alpha = i_0$ implies $b = 1$).

Intuitively, α (when $\alpha \neq i_0$) is the index for which the learner is currently testing (or last tested) whether $L_\alpha = L$ (in case the learner finds $L_\alpha \neq L$, it may continue to keep the same α for some time until it finds an appropriate replacement for it). The α used by the learner will always have the property that $L_\alpha \subseteq L$. Thus the learner then needs to check if $L \subseteq L_\alpha$. Though the learner can check for any future elements seen in the input whether they belong to L_α (this is kept track of by using the parameter b in the memory), the learner may not be able to check whether the past data belonged to L_α, as it may have forgotten them. For this purpose, learner keeps track of a parameter u which length-lexicographically bounds any elements in the past which may be in $L - L_\alpha$ (how the learner keeps track of u will be clearer later). The learner uses the parameter β to search for any potential index such that $L_\alpha \subset L_\beta \subseteq L$. If such a β exists, then the learner replaces α above by β, and continues the process. In case such a β does not exist, then α would be the only possible index for L. The learner uses Proposition 7(b) to bound the search for such β in case the learner, since it has started testing for L_α, has not seen an element in $L - L_\alpha$.

We now proceed formally. Let T be a text for the input language L and T' be a sequence of counterexamples. Suppose $\mathbf{M}(T[n], T'[n]) = (mem_n, hyp_n)$, where $mem_n = conv(\alpha_n, u_n, \beta_n, b_n)$. We will always have $hyp_n = mem_n$. Thus, the learner is iterative. The invariants maintained by the learner related to the memory are as follows. For ease of notation below, we take $L_{i_0} = \emptyset$. For all n:

(I1) $\alpha_n \leq_{ll} \alpha_{n+1} \leq_{ll} \beta_n \leq_{ll} \beta_{n+1}$.
(I2) $L_{\alpha_n} \subseteq L_{\alpha_{n+1}} \subseteq L$.
(I3) For all $\alpha' <_{ll} \beta_n$ such that $\alpha' \neq \alpha_n$ and $\alpha' \in I$, $L_{\alpha'} \neq L$.
(I4) $\max(\text{content}(T[n]) - L_{\alpha_n}) \leq_{ll} u_n$. Furthermore $u_n \leq_{ll} u_{n+1}$.

Let m be the least number such that $\alpha_m = \alpha_n$.
(I5) $b_n = 0$ iff $\alpha_n \neq i_0$ and $\{T(s) : m \leq s < n\} \subseteq L_{\alpha_n}$.
(I6) If $b_n = 0$, then $u_n = u_m$; otherwise $u_n = \max(\{u_m\} \cup \{T(s) : m \leq s < n\} - L_{\alpha_n})$.
(I7) If $m < n$ then, $\beta_{n-1} = \beta_n$ iff $[b_{n-1} = 0$ and $|\beta_{n-1}| > \max(\{|\alpha_n|, |u_n|\}) + c]$.

We now specify how the learner computes $\alpha_n, u_n, \beta_n, b_n$. Initially, $\alpha_0 = i_0$, β_0 is the length-lexicographically least element of I, $b_0 = 1$ and u_0 is the length-lexicographically least element of Σ^*. We now describe how the memory of the learner is updated after receiving input $T(n), T'(n)$ (where the previous memory is $(\alpha_n, u_n, \beta_n, b_n)$).

Let $u_{n+1} = u_n$, if $T(n) = \#$ or $[\alpha_n \neq i_0$ and $T(n) \in L_{\alpha_n}]$; otherwise, $u_{n+1} = \max(\{u_n, T(n)\})$. For defining, $\alpha_{n+1}, \beta_{n+1}, b_{n+1}$, consider the following cases.

Case 1: $[b_n = 1$ or $[b_n = 0$ and $|\beta_n| \leq \max(\{|\alpha_n|, |u_n|\}) + c]]$.

In this case $H_{hyp_n} = L_{\beta_n}$.

Case 1a: $[\alpha_n = i_0$ or $L_{\alpha_n} \subset L_{\beta_n}]$ and $T'(n) = \#$.
In this case, $L_{\beta_n} \subseteq L$ and either $\alpha_n = i_0$ or $L_{\alpha_n} \subset L_{\beta_n}$. Thus, α_n is not a correct index for L (note that $i_0 \notin I$).
Let $\alpha_{n+1} = \beta_{n+1} = \beta_n$ and $b_{n+1} = 0$.

Case 1b: $[\alpha_n \neq i_0$ and $\neg[L_{\alpha_n} \subset L_{\beta_n}]]$ or $T'(n) \neq \#$.
In this case, either $\alpha_n = \beta_n$ or $L_{\beta_n} \neq L$ (note that $(L_\alpha)_{\alpha \in I}$ is 1–1).
Let $\alpha_{n+1} = \alpha_n$, $\beta_{n+1} = \mathrm{succ}_I(\beta_n)$.
Let $b_{n+1} = 1$, if $b_n = 1$ or $T(n) \notin L_{\alpha_n}$; otherwise $b_{n+1} = 0$.

Case 2: Not Case 1.
In this case $H_{hyp_n} = L_{\alpha_n}$.
Let $\alpha_{n+1} = \alpha_n$, $\beta_{n+1} = \beta_n$.
Let $b_{n+1} = 1$, if $b_n = 1$ or $T(n) \notin L_{\alpha_n}$; otherwise $b_{n+1} = 0$.

It is now easy to verify that the learner is automatic and word size memory bounded. Definition of $\alpha_0, \beta_0, b_0, u_0$ clearly maintain the invariants. We now show that the construction maintains the invariants while defining $\alpha_{n+1}, \beta_{n+1}, u_{n+1}$, b_{n+1}. Note that in Case 1a, $\alpha_n \neq \beta_n = \alpha_{n+1}$, which is the only case which changes value of α_n. (I1) is clearly maintained by both cases ($\beta_{n+1} \geq_{ll} \beta_n$ in both cases, and α_{n+1} is either α_n or β_n). (I2) is maintained as the only time $\alpha_{n+1} \neq \alpha_n$ is via Case 1a, where $L_{\alpha_n} \subseteq L_{\beta_n} \subseteq L$ holds. (I3) is maintained as in Case 1a, $L_{\beta_n} \subseteq L$ and either $\alpha_n = i_0$ or $L_{\alpha_n} \subset L_{\beta_n}$; in Case 1b, $L_{\beta_n} \neq L$ or $\beta_n = \alpha_n$, and in Case 2 $\alpha_{n+1} = \alpha_n$ and $\beta_{n+1} = \beta_n$. (I4), (I5) and (I6) are also maintained by definition of u_{n+1} and b_{n+1} in both cases. Note that in Case 1a, $L_{\alpha_n} \subseteq L_{\beta_n} = L_{\alpha_{n+1}}$. (I7) is trivially maintained by Case 1a; Case 1b and Case 2 also maintain (I7) as Case 1b makes $\beta_{n+1} \neq \beta_n$ and Case 2 makes $\beta_{n+1} = \beta_n$ (note the conditions for Cases 1 and 2).

Now, suppose $L \in \mathcal{L}$. By invariants (I1) and (I3), $\alpha_n \leq_{ll} \alpha'$, for α' such that $L_{\alpha'} = L$. It follows using (I1) that $\lim_{n\to\infty} \alpha_n$ converges, to say α. Here, note that $\alpha \neq i_0$, as eventually by Case 1b, a β_n would be chosen such that $L_{\beta_n} \subseteq L$, making $\alpha_{n+1} = \beta_n$ via Case 1a. If $L_\alpha = L$, then clearly by (I5), $\lim_{n\to\infty} b_n$ also converges; If $L_\alpha \neq L$, then by (I1) and (I3), $\lim_{n\to\infty} \beta_n$ converges, (since β_n is then bounded by the index for L) and thus by (I7) $\lim_{n\to\infty} b_n$ converges. Thus, in either case $\lim_{n\to\infty} b_n$ converges, to say b. If $L_\alpha \neq L$, then using (I1) and (I3), we have that $\lim_{n\to\infty} \beta_n$ converges; if $L_{\alpha_n} = L$, then by (I5) $\lim_{n\to\infty} b_n = 0$, and thus by (I6) $\lim_{n\to\infty} u_n$ converges, and thus by (I7) $\lim_{n\to\infty} \beta_n$ converges. Hence, in both cases we have that $\lim_{n\to\infty} \beta_n$ converges, to say β. Thus, by (I4), (I7) we have that $\lim_{n\to\infty} u_n$ converges, to say u. Thus, the memory of the learner converges to $conv(\alpha, u, \beta, b)$. By (I2) we have that $L_\alpha \subseteq L$. By, (I7) we have that $|\beta| > \max(\{|\alpha|, |u|\}) + c$ and $b = 0$. Thus, $H_{\mathbf{C}conv(\alpha,u,\beta,b)} = L_\alpha$. Furthermore, using the invariants (I3), (I4) and Proposition 7(b), we have that $L_\alpha = L$.

Thus, $\mathbf{M}$ $\mathbf{NCIt}$-learns $\mathcal{L}$. ∎

Hypotheses of the learner in the above theorem are not consistent with the data seen so far. We can make the learner consistent if it receives least counterexamples whenever it's hypothesis contains data that is not a part of the target language.

Theorem 9 (Frank Stephan, personal communication) *Let $\mathcal{L}$ be an automatic class. Then, $\mathcal{L} \in$* **AutoWordConsLNCEx** *as witnessed by a learner using a class comprising hypothesis space.*

Now we turn to automatic **BNCEx**-learning with word-size memory limit.

Theorem 10 *Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ be an automatic family which consists only of infinite languages. Then, $\mathcal{L} \in$* **AutoWordBNCEx**. *The learner uses the hypothesis space $(H_\alpha)_{\alpha \in I}$, where $H_\alpha = L_\alpha$.*

Proof. Without loss of generality assume that the indexing $(L_\alpha)_{\alpha \in I}$ is 1–1 (otherwise, we can ignore the non-minimal indices of I, which can be automatically determined). Furthermore, assume that I is infinite (otherwise, the theorem is trivial). Let c be as in Proposition 7 (for $\mathcal{L}$).

For ease of presentation, the size of the memory of the learner is word size bounded only for the case when the input language is in the class $\mathcal{L}$. One can easily convert such a learner to always having word-size memory bound by remembering the length-lexicographically largest word seen in the text/counterexample text, and if the memory tries to exceed the appropriate bound (relevant constant plus the size of the remembered largest word), then abandoning the learning process.

The learner $\mathbf{M}$ has memory of the form: (α, w, u, β), where $\alpha, \beta \in I$, $w, u \in \Sigma^* \cup \{\#\}$. Let T be a text for the input language L and T' be a sequence of counterexamples. Suppose $\mathbf{M}(T[n], T'[n]) = (mem_n, hyp_n)$, where $mem_n = (\alpha_n, w_n, u_n, \beta_n)$.

Intuitively, α_n is the index for which the learner is currently testing if $L_{\alpha_n} = L$. The length-lexicographically largest element seen in the input $T[n]$ is denoted by w_n. The length-lexicographically largest element seen in the text T before the learner starts testing for α_n is denoted by u_n.

If $L_{\alpha_n} \not\subseteq L$, $L \in \mathcal{L}$ and $\mathbf{M}$ conjectures L_{α_n} infinitely often then the learner will eventually get a counterexample for it as every language in $\mathcal{L}$ is infinite. For the elements received after the learner starts testing for α_n, the learner can check if they belong to L_{α_n} as the elements are received. However, the learner may have forgotten the elements it had seen before it starts testing for L_{α_n} (note that all the forgotten elements would be $\leq_{ll} u_n$, though we do not exactly know which). For testing whether such elements are in $L - L_{\alpha_n}$, the learner checks if there is some $\beta \in I$ which satisfies: $L_{\alpha_n} \subset L_\beta \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}$ and $L_\beta \cap \{x : x \leq_{ll} u_n\} \subseteq L$. Such β's (satisfying $L_{\alpha_n} \subset L_\beta \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}$) are finite in number and can be determined using Proposition 7(b).

We proceed formally now. The invariants maintained by the learner related to the memory are as follows. For all n:

(I1) $w_n = \max(\text{content}(T[n]))$.
(I2) $\alpha_n \leq_{ll} \alpha_{n+1}$.
(I3) For all $\alpha' <_{ll} \alpha_n$ with $\alpha' \in I$, $L_{\alpha'} \neq L$, where $\text{content}(T'[n]) \cap L_{\alpha'} \neq \emptyset$ or there exists an $x \leq_{ll} w_n$, $x \in L - L_{\alpha'}$.

Let m be the least number such that $\alpha_m = \alpha_n$.

(I4) $u_n = \max(\text{content}(T[m]))$.
(I5) If $n = m$, then $\beta_n = \alpha_n$.
(I6) **(i)** $\text{content}(T[n]) - \{x : x \leq_{ll} u_n\} \subseteq L_{\alpha_n}$,
(ii) for all β such that $\alpha_n <_{ll} \beta <_{ll} \beta_n$, if $L_{\alpha_n} \subset L_\beta \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}$, then $L_\beta \not\subseteq L$, and
(iii) if $m < n$ and $|\beta_{n-1}| \leq \max(\{|\alpha_n|, |u_n|\}) + c$, then $\beta_{n-1} <_{ll} \beta_n$.
(iv) if $m < n$ and $|\beta_{n-1}| > \max(\{|\alpha_n|, |u_n|\}) + c$, then $\beta_{n-1} = \beta_n$.

The hypothesis of the learner is directly obtainable from memory as follows. If $|\beta_n| \leq \max(\{|\alpha_n|, |u_n|\}) + c$, then $hyp_n = \beta_n$; otherwise, $hyp_n = \alpha_n$. Thus, it is enough to specify how the learner computes $\alpha_n, w_n, u_n, \beta_n$.

Initially, $\alpha_0 = \beta_0 = <_{ll}$-least element of I, $w_0 = u_0 = \#$. We now describe how memory of the learner is updated after receiving input $T(n), T'(n)$ (where the previous memory is $(\alpha_n, w_n, u_n, \beta_n)$).

Case 1: $T(n) \notin L_{\alpha_n}$ or $T'(n) \in L_{\alpha_n}$ or $[T'(n) = \#, |\beta_n| \leq \max(\{|\alpha_n|, |u_n|\}) + c$ and $L_{\alpha_n} \subset L_{\beta_n} \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}]$.
This case implies that $L_{\alpha_n} \neq L$ as either $T(n) \in L - L_{\alpha_n}$ or $T'(n) \in L_{\alpha_n} - L$ or $[L_{\beta_n} \cap \{x : x \leq_{ll} u_n\} \subseteq L_{\beta_n} \cap \{x : x \leq_{ll} w_n\} \subseteq L$ and $L_{\beta_n} \cap \{x : x \leq_{ll} u_n\} \not\subseteq L_{\alpha_n}]$. Furthemore, note that either $L_{\alpha_n} \cap \text{content}(T'[n+1]) \neq \emptyset$, or there exists a $x \leq_{ll} w_{n+1}$ such that $x \in L - L_{\alpha_n}$.
Let $\alpha_{n+1} = \beta_{n+1} = \text{succ}_I(\alpha_n)$. Let $w_{n+1} = u_{n+1} = \max(\text{content}(T[n+1]))$.
Note that w_{n+1}, u_{n+1} can be computed using w_n and $T(n)$.

Case 2: Not Case 1 and $|\beta_n| \leq \max(\{|\alpha_n|, |u_n|\}) + c$
Note that in this case $hyp_n = \beta_n$. Furthermore, either $L_{\beta_n} \not\subseteq L$ (when, $T'(n) \neq \#$) or $\neg[L_{\alpha_n} \subset L_{\beta_n} \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}]$ (as Case 1 does not hold).
Let $\alpha_{n+1} = \alpha_n$, $u_{n+1} = u_n$, $w_{n+1} = \max(\text{content}(T[n+1]))$, $\beta_{n+1} = \text{succ}_I(\beta_n)$.

Case 3: Not Case 1 and $|\beta_n| > \max(\{|\alpha_n|, |u_n|\}) + c$
Note that in this case $hyp_n = \alpha_n$. Furthermore, $T(n) \in L_{\alpha_n}$ and $T'(n) = \#$.
Let $\alpha_{n+1} = \alpha_n$, $\beta_{n+1} = \beta_n$, $u_{n+1} = u_n$, and $w_{n+1} = \max(\text{content}(T[n+1]))$.

Clearly, the learner $\mathbf{M}$ is automatic.

The invariants (I1), (I2), (I4), (I5), (I6)(iii), (iv) are clearly maintained by the construction. For (I3) note that Case 1 is the only case where $\alpha_{n+1} \neq \alpha_n$, and in this case $L_{\alpha_n} \neq L$. For (I6)(i), note that if $T(n)$ is not in L_{α_n}, then by Case 1, $\alpha_{n+1} \neq \alpha_n$; thus, using (I4), (I6)(i) holds. For (I6)(ii), note that in Case 1, $\beta_{n+1} = \alpha_{n+1}$, in Case 3 $\beta_{n+1} = \beta_n$ and in Case 2, $L_{\beta_n} \not\subseteq L$ or $\neg[L_{\alpha_n} \subset L_{\beta_n} \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}]$; thus, (I6)(ii) is maintained in all the cases.

By (I5), (I6)(iii), (iv), we have that length of β is at most a constant more than $\max(\{|\alpha_n|, |w_n|\})$. Furthermore, by (I1), (I3) and Proposition 7(c), we have that $|\alpha_n|$ is bounded in length by a constant plus $\max(\text{content}(T[n]) \cup \text{content}(T'[n]))$. Thus, $\mathbf{M}$ is word-size memory bounded.

Now, for $L = L_{\alpha'}$, $\alpha' \in I$, by invariant (I3), $\alpha_n \leq_{ll} \alpha'$. Thus, by (I2) $\lim_{n\to\infty} \alpha_n$ converges, to say α. Thus, by (I4), $\lim_{n\to\infty} u_n$ converges, to say

u. Furthermore, using the invariants (I5) and (I6)(iii), we have that, for all but finitely many n, $|\beta_n| > \max(\{|\alpha_n|, |u_n|\}) + c$. Thus, by (I3), (I6)(i), (I6)(ii), and Proposition 7(b), we have that either, $L_\alpha \not\subseteq L$ or $L_\alpha = L$. Furthermore, by definition of hyp_n, for all but finitely many n, $hyp_n = \alpha$. Thus, if $L_\alpha \not\subseteq L$, then by cardinality of L being infinite, we must have $T'(n) \in L_\alpha$ (and thus Case 1 holding) for large enough n, a contradiction.

It follows that $L_\alpha = L$. Thus, the learner **M NCEx**-learns $\mathcal{L}$. ∎

Yet a result from [JK08] can be used to show that some automatic classes cannot be **BNCEx**-learned (even by a non-automatic learner).

Theorem 11 *[JK08] Let $\mathcal{L} = \{\Sigma^*\} \cup \{L_x : x \in \Sigma^*\}$, where $L_x = \{y : y \leq_{ll} x\}$. Then, $\mathcal{L}$ is an automatic family and $\mathcal{L} \notin$* **BNCEx**.

The following corollary shows that, for the unrestricted automatic learnability, as well as automatic learnability with all types of memory restrictions, there are automatic classes that are **NCEx**-learnable, but not **BNCEx**-learnable.

Corollary 12 *(a)* **AutoNCIt** − **AutoBNCIt** $\neq \emptyset$.
(b) **AutoHypNCEx** − **AutoHypBNCEx** $\neq \emptyset$.
(c) **AutoWordNCEx** − **AutoWordBNCEx** $\neq \emptyset$.
(d) **AutoNCEx** − **AutoBNCEx** $\neq \emptyset$.

Our next result shows that some automatic class, while not **HypBNCEx**-learnable, can be automatically learned with word-size memory without negative counterexamples.

Theorem 13 **AutoWordTxtEx** − **HypBNCEx** $\neq \emptyset$.

Let $\Sigma = \{0\}$. Let $L_0 = \{0^{2n} : n \geq 0\}$.
Let $L_{1^i} = \{0^{2n} : n \leq i\} \cup \{0^{2i+1}\}$. Let $L_{(2^i,3^j)} = L_{1^i} \cup \{0^{2j}\}$.
Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $I = \{0, 1^i, (2^i, 3^j) : i, j \in N\}$.
Then $\mathcal{L}$ witnesses Theorem 13. We omit the detailed proof.

The next theorem shows that automatic iterative learners using negative counterexamples still can sometimes learn automatic classes that cannot be learned using positive data alone.

Theorem 14 (**AutoWordNCEx** ∩ **AutoWordBNCEx** ∩ **AutoNCIt** ∩ **AutoBNCIt**) − **TxtEx** $\neq \emptyset$.

Let $\Sigma = \{a\}$. Let $L_\epsilon = a^*$, and $L_w = L_0 - \{w\}$, for $w \in a^+$. Let $\mathcal{L} = \{L_w : w \in a^*\}$. Then $\mathcal{L}$ witnesses Theorem 14. We omit the detailed proof.

Our last result shows that monotonic (even non-automatic) learners cannot learn some automatic classes, even using least counterexamples.

Theorem 15 *There exists an automatic class $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ such that $\mathcal{L} \notin$* **MonLNCEx**.

Proof. Let $\Sigma = \{0\}$. Let $L_0 = 0^*$, $L_{1^j} = \{0^i : i \leq j\}$, $L_{\mathbf{C}onv(1^j,2^k)} = \{0^i : i \leq j\} \cup \{0^k\}$. Let $I = \{0, 1^j, conv(1^j, 2^k) : j, k \in N\}$, and $\mathcal{L} = \{L_\alpha : \alpha \in I\}$.

Clearly, $\mathcal{L}$ is an automatic family. Suppose, by way of contradiction, that $\mathbf{M}$ is a monotonic learner which **LNCEx**-learns $\mathcal{L}$. Consider the shortest σ such that $\mathbf{M}(\sigma, \#^{|\sigma|})$ is for an infinite language. Note that there exists such a σ as $\mathbf{M}$ learns L_0. Now consider a text T extending σ for L_{1^j}, where $j = \max(\text{content}(\sigma) \cup \bigcup_{s<|\sigma|} L_{\mathbf{M}(\sigma[s],\#^s)})$. Let T' be the least-counterexample text for $\mathbf{M}$ on the text T. Then $\mathbf{M}(T, T')$ must converge to a grammar g for L_{1^j}. Thus, content(T') is finite. Let $x \in L_{\mathbf{M}(\sigma,\#^{|\sigma|})} - (\text{content}(T) \cup \text{content}(T'))$. Let m be such that content$(T) =$ content$(T[m])$, content$(T') =$ content$(T'[m])$, $\sigma \subseteq T[m]$, and $\mathbf{M}(T[m], T'[m]) = g$. Then $\mathbf{M}$ is not monotonic on $T[m] \diamond x$, where counterexamples provided are least counterexamples. ∎

4 Conclusions

In this paper we considered learning automatic families by automatic learners which receive negative counterexamples. Various versions of memory restriction and counterexamples were considered. Table 1 gives a summary of results regarding learning all classes of a particular type for various criteria.

Learning Criterion	Aut. Classes of Infinite Languages	All Automatic Classes	Consistent Learning for All Aut. Classes
Auto(Word, Hyp)NCEx **AutoNCIt**	yes	yes	open
Auto(Word)LNCEx	yes	yes	yes
Auto(Word)BNCEx	yes	no	no
AutoHypBNCEx **AutoBNCIt**	open	no	no

Table 1. Summary of results on when all classes of particular type are learnable.

We showed that there is an automatic class which is in **AutoWordBNCEx**$-$**AutoHypBNCEx**, though at this point we do not know if **AutoWordBNCEx** properly contains **AutoHypBNCEx**. It is also open whether **AutoBNCEx** $\subseteq$ **AutoWordBNCEx**. Note that the corresponding problems in **AutoTxtEx** learning (without using negative counterexample) are also open [JLS10]. Regarding monotonic learning, we showed that there are automatic families which cannot be **LNCEx**-learnt by any monotonic (even non-automatic) learners.

Acknowledgements. We thank the anonymous referees for several helpful comments and pointing out a bug in the earlier version of the paper and an improvement of Theorem 8. We thank Frank Stephan for discussion as well as pointing out Theorem 9, which strengthened an earlier version of this theorem.

References

[Ang80] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[Bār74] J. Bārzdiņš. Inductive inference of automata, functions and programs. In *Proceedings of the 20th International Congress of Mathematicians, Vancouver*, pages 455–460, 1974. In Russian. English translation in American Mathematical Society Translations: Series 2, 109:107-112, 1977.

[BG00] A. Blumensath and E. Grädel. Automatic structures. In *15th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 51–62. IEEE Computer Society, 2000.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[HPTS84] K. Hirsh-Pasek, R. Treiman, and M. Schneiderman. Brown and Hanlon revisited: Mothers' sensitivity to ungrammatical forms. *Journal of Child Language*, 11:81–88, 1984.

[JK08] S. Jain and E. Kinber. Learning languages from positive data and negative counterexamples. *Journal of Computer and System Sciences*, 74(4):431–456, 2008. Special Issue: Carl Smith memorial issue.

[JLS10] S. Jain, Q. Luo, and F. Stephan. Learnability of automatic classes. In *Language and Automata Theory and Applications, 4th International Conference, LATA 2010*, volume 6031 of *LNCS*, pages 321–332. Springer, 2010.

[JOPS11] S. Jain, Y. S. Ong, S. Pu, and F. Stephan. On automatic families. In T. Arai, Q. Feng, B. Kim, G. Wu, and Y. Yang, editors, *Proceedings of the 11th Asian Logic Conference, in Honor of Professor Chong Chitat's 60th birthday, 2009*, pages 94–113. World Scientific, 2011.

[KN95] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *Logical and Computational Complexity, (International Workshop LCC 1994)*, volume 960 of *Lecture Notes in Computer Science*, pages 367–392. Springer, 1995.

[LZ93] S. Lange and T. Zeugmann. Language learning in dependence on the space of hypotheses. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 127–136. ACM Press, 1993.

[Wie76] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Journal of Information Processing and Cybernetics (EIK)*, 12(1–2):93–99, 1976.

[Wie90] R. Wiehagen. A thesis in inductive inference. In J. Dix, K. Jantke, and P. Schmitt, editors, *Nonmonotonic and Inductive Logic, 1st International Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 184–207. Springer-Verlag, 1990.