Automatic Learning from Positive Data and Negative Counterexamples*

Sanjay Jain[1], Efim Kinber[2] and Frank Stephan[3]

[1] School of Computing, National University of Singapore, Singapore 117417. 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

[3] Department of Mathematics and Department of Computer Science, National University of Singapore, Singapore 119076. Email: fstephan@comp.nus.edu.sg

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 the 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 which are bounded by the largest positive datum seen so far) can impact automatic learnability.

Keywords: Inductive Inference; Automatic Learning; Automatic Classes; Negative Counterexamples; Iterative Learning.

1 Introduction

Inductive inference [Ang80a,Gol67] studies learning as a limiting process of step by step improved conjectures depending on more and more incoming data. The field, due to its precise and multiple notions of convergence of hypotheses and data presentations, has applications and connections both inside and outside the field of learning:

– related areas within the field of learning are query learning [Ang88,BFK16] and grammatical inference [Cla10,CY13,Fer02];
– the learning of pattern languages [Ang80b,LN05] and elementary formal systems [Shi94], originally studied in inductive inference, have been applied to other areas like molecular biology [AMSKMS93,AMSSY92,ASMS91,ASY92];
– areas which also apply notions from inductive inference outside the field of learning are the area of Weihrauch degrees [BBP12,BGH15], the study of limiting processes in recursion theory [CM09,HA02,HN00,Moe12] and program synthesis [GJTV11,JS15].

* This research was supported in part by MOE Academic Research Fund Tier 1 through grants R252-000-420-112 and R146-000-181-112 to S. Jain and F. Stephan as well as NUS Provost Chair Grant C252-000-087-001 to S. Jain.

1.1 Automatic Inductive Inference

Many inductive inference criteria consider learnability in the general framework of recursion theory and do not impose resource bounds. However, some works do consider limitations of the update time in response to a newly read datum [Pit89] or the amount of memory permitted to store information about previously read data [FKS95,KS95]. Jain, Luo and Stephan [JLS12] introduced an "automatic" variant of inductive inference where these restrictions are formulated in the language of automata theory: the family of target languages is computable by a finite automaton (automatic family) and a learner, more precisely its update function, is a finite automaton itself (automatic learning). This approach is motivated from the fact that

> automatic functions coincide with those computed by a position-faithful one-tape Turing machine in linear time [CJSS13].

Therefore automatic learners are learners whose update function (of long-term memory and hypothesis) satisfies a very natural and restrictive resource bound. This stands in contrast to results of Case and Kötzing [CK09] and Pitt [Pit89] — they showed that for many natural constraints one can slow down learners to polynomial or even linear time without losing learnability according to the given constraints.

Automatic families are the natural counterpart to automatic learners, as they form a natural type of hypothesis space, which is compatible with the notions of automatic learners. Here an automatic family is a family of target languages, which is defined by a regular index set and where the membership problem in these languages is regular in the sense that one finite automaton recognises 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.

Gulwani, Jha, Tiwari and Venkatesan [GJTV11] and Jha and Seshia [JS15] considered a scenario of automated program synthesis (learning), which, in some cases, can be easily modelled with automatic families: One wants to build an acceptor of a language or a program with a certain input-output behaviour (specified by some first-order formula), where the program can be synthesised from basic constraints using bounded number of operations, with parameters, which can be executed in an arbitrary order, but without loops. Here the number of operations is bounded by some prefixed constant c. In the case that all the base operations are from an automatic structure — for example, integers or dyadic rationals with addition and comparison and multiplication by a fixed list of constants, if-then-else statements and operations to manipulate a constant amount of variables — then the resulting programs or acceptors of languages can all be listed within an automatic family, where the choice of the family depends on the constant c and the permitted operations. This could be handled well from both ends: A teacher might use automatic functions to extract counterexamples or draw examples or check correctness of a proposed solution; a learner can access the automatic family in order to find an index (= solution) satisfying the constraints.

Though Gulwani, Jha, Tiwari and Venkatesan [GJTV11] themselves worked within a bit larger scope than automatic structures, their concepts can be implemented quite efficiently when restricted to such a scope. Note that besides finding the right order of the operations, the learner has also to find the right parameters and this makes the process more involved than just dealing with finitely many case distinctions. Furthermore, the oracles mentioned by Jha and Seshia [JS15] can also be handled efficiently in the case of automatic structures, as the first-order theory used to write down the constraints is decidable.

1.2 Memory Limitations

Jain, Luo and Stephan [JLS12], besides introducing the usage of automatic learners, also considered three different natural types of limits on the size of the (long-term) memory available to an automatic 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 (*iterative learning*, see [Wie76]).

Jain, Luo and Stephan [JLS12] established that automatic learners are much weaker than unrestricted recursive learners — even when learning automatic classes. In particular, the class of co-singletons, $\{\{0,1\}^* - \{\alpha\} : \alpha \in \{0,1\}^*\}$, is not automatically learnable. Moreover, they showed the following modification of Angluin's result from [Ang80a]: An automatic class is learnable by a recursive learner iff it satisfies Angluin's tell-tale condition.

Let $L_\epsilon = \{0^j : j > 0\}$ and $L_\alpha = \{0^j : \alpha(j) = 1\} \cup \{\epsilon\}$, where $\alpha \in \{0,1\}^+$. Jain, Luo and Stephan [JLS12] showed that $\{L_\alpha : \alpha \in \{0,1\}^*\}$, separates automatic learning by learners having memory bounded by longest positive input datum seen so far from automatic learning by learners having memory bounded by the size of the current hypothesis. To see this, on the positive side, a learner can keep track of the words 0^j seen in the input by memorizing a word $\beta \in \{0,1\}^*$ such that $\beta(j) = 1$ iff 0^j has been seen in the input text. This information is sufficient to determine the input language in the limit (as $\beta(0) = 1$ implies that the input language is L_β for the limiting value of β; otherwise the input language is L_ϵ). On the other hand, a learner with memory bounded by hypothesis size, while getting input L_ϵ, would forget some of the input words, and thus would not be able to know the target language when ϵ appears in the input later (see [JLS12] for details).

It is open at present if automatic learning by learners having memory bounded by the size of the current hypothesis is different from automatic learning by learners having only the last hypothesis as its memory. Similarly, it is open at present if bounding the memory by the size of the longest word seen so far is a restriction for automatic learners having no memory restriction (except that implicit due to the learner being automatic). Jain, Luo and Stephan [JLS12] also considered how the requirement of *consistency* — where every conjecture of the learner must be consistent with the input seen so far, impacts learnability by automatic learners.

1.3 Inference with Negative Counterexamples

Since automatic learners are not able to learn many automatic classes from positive data alone, it is natural to ask: under which conditions are *all* automatic classes of languages automatically learnable? Jain and Kinber [JK08] 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 its hypothesis contains data that is not a part of the target language. This approach to learning in the limit is arguably 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 different and popular approaches to learning in the limit — learning languages from positive data [Ang80a,Gol67] and learning concepts from subset queries and counterexamples [Ang88,GPS92,LN05,LZ05] or from selected examples [LNW98] whereas none of these approaches by itself adequately represents the process of language acquisition. This model of learning from negative counterexamples was considered in the context of iterative learning by Jain and Kinber [JK07], where the learner keeps in its memory only its latest conjecture.

Jha and Seshia [JS15] reintroduced the above model, and considered it in the context of automated program synthesis. They provided an important practical motivation for the aforementioned model: in a process of automated program synthesis (or learning) from positive examples, a learner interacts with a "verifier" that takes a candidate program and specification and tries to find a counterexample showing that the candidate program does not satisfy specifications.

Jain and Kinber [JK08], as well as Jha and Seshia [JS15], considered three different types of negative counterexamples provided to the learner:

(a) arbitrary counterexamples,
(b) least counterexamples and
(c) counterexamples 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. Jha and Seshia [JS15] provide the following practical motivation for using least counterexamples: the idea of using minimal counterexamples stems from the cases where some validation engines produce counterexamples which aid debugging by localizing errors. As for motivation for counterexamples in (c), they write that in "industrial practice of validation by simulation, . . . the system is often simulated to a finite length to discover bugs. The length of simulation often depends on the traces which illustrate known positive behaviours". Furthermore, also in other areas of query learning, researchers have considered bounds on the size of counterexamples. For example, Ibarra and Jiang [IJ91] investigated for various classes of languages whether learnability of the class depends on the way a counterexample is chosen; for example, always requiring the shortest counterexample permits to learn the class of pattern languages in polynomial time from equivalence queries; always requiring the length-lexicographically first counterexample permits to learn all regular languages in polynomial time.

For explanatory learning, arbitrary and least counterexamples have the same power, and counterexamples bounded by the size of the largest positive datum have lesser power. The above result was first shown by Jain and Kinber [JK08] and later also independently by Jha and Seshia [JS15]. Besides comparing the above types of counterexamples for explanatory learning,

Types of classes	Types of counterexamples	Best types of learning
All automatic classes	Any counterexample	Automatic learning with word-sized memory and Iterative automatic learning
All automatic classes	Least counterexample	Consistent automatic learning with word-sized memory
Automatic classes of infinite languages	Bounded counterexamples	Automatic learning with word-sized memory and Iterative automatic learning

Table 1. Best learners for certain types of classes and negative counterexamples.

Jain and Kinber [JK08] also explored their impact for various other models of learning, such as behaviourally correct learning, learning with informants and learning with anomalies.

1.4 Contributions of This Paper

As mentioned earlier, automatic learners are not able to learn many automatic classes from positive data alone. Thus, one of the motivations for the present paper is to explore what kind of negative counterexamples allow learning of all automatic families by automatic learners. So the counterexamples are needed for both: Overcoming the intrinsic restrictions of learning from positive data and overcoming restrictions stemming from the low computational complexity of automatic learners. Thus, in this paper, we adapt the notion of learning with negative counterexamples to learning of automatic classes and we also mainly consider automatic learners.

Our major result (Theorem 14) is that such automatic learners, even when required to be *iterative*, can learn *every* automatic class. Here note that iterative learners become often more powerful by using a new hypothesis space, as that permits to consider paddings of hypotheses with some information; however, each update of that information results, formally, in a revision of the hypothesis and can therefore be done only finitely often without destroying learning success. So the main difference between iterative learning (using a different hypothesis space) and learning with a hypothesis-sized bound on the memory is that the latter can revise the memory infinitely often while converging to a valid hypothesis.

Theorem 14(b) shows that automatic learners, with the size of the memory bounded by the size of the longest positive datum seen so far (word-sized memory), can learn all automatic families using negative counterexamples. In particular, it shows that bounding the memory of automatic learners by the length of the longest positive datum seen so far is not a restriction for learning automatic families using negative counterexamples, whereas, as mentioned above, whether such a result holds for learning from just positive data is open.

While, on one hand, iterative learners are powerful, on the other hand, interestingly, we have not been able to make such learners consistent with the data seen so far. Yet, on one hand, Theorem 15 shows that consistency can be achieved if the learners always receive the least negative counterexample, while, on the other hand, it follows from a result by Jain and Kinber

[JK08] that there are automatic classes that cannot be learned even by non-automatic consistent learners if the counterexamples are bounded by the largest positive input datum seen so far.

When automatic learners use counterexamples bounded by the largest positive input datum seen so far, they turn out to be more restrictive than when they use arbitrary or least counterexamples. Theorem 18 gives a characterisation of when automatic classes can be learnt by automatic learners using bounded negative counterexamples. This characterisation is based on whether some language in the class has infinitely many initial segments (called cut-offs) of the language included in the class.

That bounded negative counterexamples are more restrictive for automatic learning than unrestricted counterexamples is due to the fact that the learner gets no negative feedback in the case when the conjecture has some elements outside the language to be learnt, but none of these elements is bounded by the data seen so far. Still, even if the memory of the learner is limited by the size of the largest positive input datum seen so far, automatic learners, with this bound on counterexamples, can learn automatic classes consisting of just infinite languages (Corollary 20). A similar result holds for automatic iterative learners (Theorem 21).

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 or, more restrictively, in the case of iterative learning, but can be learned automatically with memory limited by the size of the longest positive datum seen so far even without negative counterexamples (Theorem 24). This result extends the aforementioned separation of automatic learning by learners with memory bounded by the size of the longest word seen so far from those with memory bounded by the size of the current hypothesis for automatic learners not using counterexamples, obtained in [JLS12].

Theorem 25 shows that for automatic classes where the languages are pairwise incomparable, just negative counterexamples are enough to learn the class automatically and iteratively. Our last result (Theorem 26) 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.

Table 1 summarises the results obtained in the sense that it gives what are the best learners found for the various combinations of classes and types of negative counterexamples. Note that iterative learners are also learners where the memory is bounded by the size of the hypothesis. Some problems are left open. In particular, we do not know whether every automatic family can be consistently learnt using arbitrary negative counterexamples.

2 Preliminaries

The set of natural numbers is denoted by $\mathbb{N}$. 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 $|x|$. For any symbol a, a^n denotes a repeated n times. We say $u <_{lex} v$, if the string u comes before v in the lexicographic (dictionary) order (where we assume some canonical ordering of elements of Σ). We use $u \leq_{lex} v$ to denote that $u <_{lex} v$ or $u = v$. We say that a string w is length-lexicographically smaller than string w' (written $w <_{ll} w'$) iff $|w| < |w'|$ or $|w| = |w'|$ and $w <_{lex} w'$. We let $w \leq_{ll} w'$ denote that either $w = w'$ or $w <_{ll} w'$. For any set of strings S, we let $\text{succ}_S(w)$ denote the length-lexicographically least w' such that $w <_{ll} w'$, and $w' \in S$ (if there is no such string, then we let $\text{succ}_S(w)$ to be undefined).

We let $\emptyset, \subseteq$ and $\subset$ respectively denote empty set, subset and proper subset. For a set $L \subseteq \Sigma^*$ and $x \in \Sigma^*$, we let $L(x)$ denote the membership predicate for L, that is, $L(x) = 1$, if $x \in L$ and $L(x) = 0$, if $x \notin L$. We let $\text{card}(S)$ denote the cardinality of the 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). Thus, we take $\# <_{ll} w$, for all $w \in \Sigma^*$.

The 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 (Blumensath and Grädel [BG00], Khoussainov and Nerode [KN95]). *Any relation that is first-order definable from existing automatic relations is automatic.*

We often implicitly use the above fact in our proofs. Note that many common relations such as subset, equality and finiteness are first order definable. For example, for an indexed family $\{L_\alpha : \alpha \in I\}$ over the alphabet Σ, $L_\alpha \subseteq L_\beta$ can be expressed via a first order formula as follows. Let $U = \{conv(\alpha, x) : \alpha \in I, x \in L_\alpha\}$. Now, $L_\alpha \subseteq L_\beta$ iff $(\forall x \in \Sigma^*)[(\alpha, x) \in U \Rightarrow (\beta, x) \in U]$. Furthermore, $L_\alpha = L_\beta$ iff $L_\alpha \subseteq L_\beta$ and $L_\beta \subseteq L_\alpha$. Similarly, L_α is finite iff $(\exists x)(\forall y > x)[y \notin L_\alpha]$. Thus, by above lemma subset, equality and finiteness are automatic. Often, when it is easy to obtain such first order formulas, we do not explicitly give them and just state that the property as automatic.

We now give some well-known examples of automatic families. For this we use the fact that the length-lexicographic ordering $\leq_{ll}$ and the lexicographic ordering $\leq_{lex}$ as well as the prefix-relation $\preceq$ on strings over an alphabet Σ are automatic.

Example 2. **(a)** *For any fixed k, the class of all subsets of Σ^* having at most k elements is an automatic family.*
To see this, let u be a special symbol outside Σ, and fix the constant k. Let I be the set of all convolutions $\mathrm{conv}(a_1, a_2, \ldots, a_h, u, u, \ldots, u)$ of k strings such that $h \leq k$, $a_1 <_{ll} a_2 <_{ll} \ldots <_{ll} a_h$ and $a_1, a_2, \ldots, a_h \in \Sigma^*$. For $\mathrm{conv}(a_1, a_2, \ldots, a_h, u, u, \ldots, u) \in I$, let $L_{\mathrm{conv}(a_1,a_2,\ldots,a_h,u,u,\ldots,u)} = \{a_1, a_2, \ldots, a_h\}$. Then, $\mathcal{L} = \{L_{\mathrm{conv}(a_1,a_2,\ldots,a_h,u,u,\ldots,u)} : \mathrm{conv}(a_1, a_2, \ldots, a_h, u, u, \ldots, u) \in I\}$, is the required automatic family, where the indexing is 1–1.

(b) *The class of all finite and cofinite subsets of $\{0\}^*$ is an automatic family.*
To see this let $I = \{a_0a_1 \ldots a_n : a_0, a_1, \ldots, a_n \in \{0, 1\} \wedge n \in \mathbb{N} \wedge (n = 0 \vee a_{n-1} \neq a_n)\}$. For $a_0a_1 \ldots a_n \in I$, let $L_{a_0a_1\ldots a_n} = \{0^m : a_{\min(\{m,n\})} = 1\}$. Then, $\mathcal{L} = \{L_{a_0a_1\ldots a_n} : a_0a_1 \ldots a_n \in I\}$ is the required automatic family, where the indexing is 1–1.

(c) *Let $I = \Sigma^*$. For $\alpha \in I$, let $L_\alpha = \alpha \cdot \Sigma^*$. Then, $\mathcal{L} = \{L_\alpha : \alpha \in \Sigma^*\}$ is an automatic family.*

(d) *Let $I = \{\mathrm{conv}(\alpha, \beta) : \alpha, \beta \in \Sigma^*$ and $\alpha \leq_{lex} \beta\}$. For $\alpha, \beta \in \Sigma^*$, $\alpha \leq_{lex} \beta$, let $L_{\mathrm{conv}(\alpha,\beta)} = \{x \in \Sigma^* : \alpha \leq_{lex} x \leq_{lex} \beta\}$. Then, $\mathcal{L} = \{L_{\mathrm{conv}(\alpha,\beta)} : \mathrm{conv}(\alpha, \beta) \in I\}$, the class of all non-empty closed intervals is an automatic family. Similarly, the class of all non-empty open intervals is an automatic family.*

(e) *A more involved automatic family is the class of all proper unions of two length-lexicographic intervals. Let $I = \{\mathrm{conv}(\alpha, \beta, \gamma, \delta) : (\exists z)[\alpha \leq_{ll} \beta <_{ll} z <_{ll} \gamma \leq_{ll} \delta]\}$.*
For $\mathrm{conv}(\alpha, \beta, \gamma, \delta) \in I$, let $L_{\mathrm{conv}(\alpha,\beta,\gamma,\delta)} = \{x : \alpha \leq_{ll} x \leq_{ll} \beta\} \cup \{y : \gamma \leq_{ll} y \leq_{ll} \delta\}$.
Then, $\mathcal{L} = \{L_{\mathrm{conv}(\alpha,\beta,\gamma,\delta)} : \mathrm{conv}(\alpha, \beta, \gamma, \delta) \in I\}$ is an automatic family.

(f) *Consider an automatic ordered group $(A, +, \leq)$ and any finite system of equations or inequalities in this group $Eq(\alpha, \beta, \gamma)$ with, say, three parameters $\alpha, \beta, \gamma \in A$ and one free variable x. Now $\mathcal{L} = \{L_{\mathrm{conv}(\alpha,\beta,\gamma)} : \alpha, \beta, \gamma \in A$ and $L_{\mathrm{conv}(\alpha,\beta,\gamma)} = \{x \in A : x$ satisfies $Eq(\alpha, \beta, \gamma)\}\}$ is an automatic family; for many ordered automatic groups like $(\mathbb{Z}, +, <)$ and $(\mathbb{Z}+\sqrt{2}\cdot\mathbb{Z}, +, <)$ one can also permit the multiplication with fixed constants in the system $Eq(\alpha, \beta, \gamma)$ of equations and inequalities, in order to obtain a bit richer structure. The same can be done with other automatic structures like automatic rings and one might allow quantifications within Eq.*

The present work considers learnability of automatic families in the presence of counterexamples. For this, let us consider the following 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 $\mathbb{N}$ 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 $\mathrm{content}(T)$, is $\{T(i) : i \in \mathbb{N}\} - \{\#\}$. Similarly, $\mathrm{content}(T[n]) = \{T(i) : i < n\} - \{\#\}$. A text T is *for a language L* iff $\mathrm{content}(T) = L$.

We let σ, τ range over initial sequences of texts. Let $|\sigma|$ denote the length of σ. For $n \leq |\sigma|$, let $\sigma[n]$ denote $\sigma(0)\sigma(1)\ldots\sigma(n-1)$. 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. In some cases, we want to consider a text formed using longer and longer initial segments $(\sigma_i)_{i\in\mathbb{N}}$, where $\sigma_i \subseteq \sigma_{i+1}$ and $\lim_{i\to\infty} |\sigma_i| = \infty$. In such cases $\bigcup_{i\in\mathbb{N}} \sigma_i$ denotes the text T such that, for all i, $\sigma_i \subseteq T$.

Definition 3. 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 "no conjecture at this point."

(a) A *learner* is a computable (recursive) mapping from $\Delta^* \times (\Sigma^* \cup \{\#\})$ to $\Delta^* \times (J \cup \{?\})$. A learner has an initial memory $mem_0 \in \Delta^*$, and an initial hypothesis $hyp_0 \in J \cup \{?\}$.

(b) Suppose a learner $\mathbf{M}$ with the initial memory mem_0 and the 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 the following:
$\mathbf{M}(\Lambda) = (mem_0, hyp_0)$;
$\mathbf{M}(\sigma\diamond x) = \mathbf{M}(mem, x)$, where $\mathbf{M}(\sigma) = (mem, hyp)$, for some $hyp \in J \cup \{?\}$.
Additionally, for $|\sigma| \geq 1$, we inductively define that $\mathbf{M}(mem, \sigma\diamond x) = \mathbf{M}(mem', x)$, 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 $\beta \in J$ (written: $\mathbf{M}(T)\downarrow_{hyp} = \beta$) iff there exists a t satisfying the following conditions:
(i) $\mathbf{M}(T[t]) \in \Delta^* \times \{\beta\}$ and
(ii) for all $t' \geq t$, $\mathbf{M}(T[t']) \in \Delta^* \times \{\beta\}$.

We always assume that the learner is recursive. In some cases, we will consider additional constraints on the learner.

Intuitively, $\mathbf{M}(\sigma) = (mem, hyp)$ means that the memory and the 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).

If the sequence of hypotheses as above converges to an index 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 4 (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. Suppose $\mathbf{M}$ is a computable learner.

(a) We say that $\mathbf{M}$ **TxtEx**-learns the language L (using the hypothesis space $\mathcal{H}$) from a text T iff $\mathbf{M}(T)\downarrow_{hyp} = \beta$ such that $H_\beta = L$.

(b) We say that **M** **TxtEx**-learns a language L (using the hypothesis space $\mathcal{H}$) iff **M** **TxtEx**-learns L from all texts for the language L (using the hypothesis space $\mathcal{H}$).

(c) We say that **M** **TxtEx**-learns $\mathcal{L}$ (using the hypothesis space $\mathcal{H}$) iff **M** **TxtEx**-learns all languages in $\mathcal{L}$ (using the hypothesis space $\mathcal{H}$).

(d) **TxtEx** $= \{\mathcal{L} : (\exists$ recursive learner $\mathbf{M})[\mathbf{M}$ **TxtEx**-learns $\mathcal{L}$ using some automatic family $\mathcal{H}$ as the hypothesis space$]\}$.

We drop the reference to "using the 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 $\mathcal{L} \subseteq \mathcal{H}$.

The following proposition, based on a classical result by Angluin for learnability of indexed families of languages, is useful. In the proposition, the conditions are slightly simplified compared to the ones given by Angluin, based on decidability of the first order formulas over automatic families. (We do not use this proposition directly in our paper, but it creates an important counterpoint to our research).

Definition 5 (Angluin [Ang80a]). Suppose a class $\mathcal{L}$ of languages is given. Let $L \in \mathcal{L}$. A finite set S is called a *tell-tale* of L (with respect to $\mathcal{L}$) iff $S \subseteq L$ and for all $L' \in \mathcal{L}$, $S \subseteq L' \subseteq L$ implies $L' = L$.

Angluin [Ang80a] had shown that an indexed family (that is a family for which membership question is uniformly decidable in the index for the languages) is **TxtEx**-learnable iff a tell-tale for every language in the family can be uniformly enumerated in the index for the language. Since, for automatic families, tell-tale can be defined using first order formula, the following proposition follows from Angluin's result.

Proposition 6 (Based on Angluin [Ang80a], see also Jain, Luo and Stephan [JLS12]). *An automatic family $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is learnable by a recursive learner iff, for every $\alpha \in I$, there exists a tell-tale set of L_α (with respect to $\mathcal{L}$).*

Definition 7. Suppose a learner **M** using an automatic family $\mathcal{H} = \{H_\beta : \beta \in J\}$ as the hypothesis space is given.

(a) [JLS12] The learner **M** is called an *automatic learner* iff its graph is automatic, that is when $\{conv(mem, x, mem', hyp') : \mathbf{M}(mem, x) = (mem', hyp')\}$ is regular.

(b) [Wie76] **M** is said to be *iterative* iff, for all finite sequences σ, $\mathbf{M}(\sigma) = (mem, hyp)$ implies $mem = hyp$.
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 $|mem| \leq \max(\{|w| : w \in \text{content}(\sigma)\}) + c$.
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 bounded by hypothesis-size, however 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, (i) if $\mathbf{M}(\sigma) = (mem, hyp)$ with $hyp \neq ?$, then $\text{content}(\sigma) \subseteq H_{hyp}$, and (ii) if $\text{content}(\sigma) \subseteq L$, for some L in the class being learnt by $\mathbf{M}$, then $\mathbf{M}(\sigma) \neq ?$.

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

Note that the above constraints (except for monotonicity) are required even on texts for languages outside the class $\mathcal{L}$. In part (c) above, we have allowed the learner to output ? on inputs which are not representative of any language in the class; this was to handle cases where the learner knows that the input is not from the class, but does not have an appropriate hypothesis which is consistent with the input. Also 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 model, 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 8 (Based on Jain and Kinber [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 counterexamples is a computable (recursive) 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 an initial hypothesis $hyp_0 \in J \cup \{?\}$.

(b) Suppose a learner $\mathbf{M}$ with the initial memory mem_0 and the initial hypothesis hyp_0 is given. We extend the definition of $\mathbf{M}$ to sequences as follows, where σ, τ are sequences over $\Sigma^* \cup \{\#\}$ with $|\sigma| = |\tau|$, and $x, y \in \Sigma^* \cup \{\#\}$:
$\mathbf{M}(\Lambda, \Lambda) = (mem_0, hyp_0)$;
$\mathbf{M}(\sigma \diamond x, \tau \diamond y) = \mathbf{M}(mem, x, y)$, 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 a text T with a negative counterexample text T' to a hypothesis $\beta \in J$ (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 are *mem* and *hyp*, respectively. We use $\mathbf{M}$ for both learners using just texts and learners using negative counterexamples in addition to texts. It will be clear from context which learner is meant. Below, $\mathbf{NC}$ in the criteria names denotes learning from negative counterexamples. $\mathbf{B}$ and $\mathbf{L}$ in $\mathbf{BNC}, \mathbf{LNC}$, denote "bounded" and "least", respectively.

Definition 9 (Based on Jain and Kinber [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 $H_{hyp} \cap \{x \in \Sigma^* : x \leq_{ll} \max(\text{content}(T[n]))\} \subseteq L$, then $T'(n) = \#$, and
if $H_{hyp} \cap \{x \in \Sigma^* : x \leq_{ll} \max(\text{content}(T[n]))\} \not\subseteq L$, then $T'(n) \in H_{hyp} \cap \{x \in \Sigma^* : x \leq_{ll} \max(\text{content}(T[n]))\} - L$.
(That is, the counterexample is bounded by the (length-lexicographically) largest positive datum seen so far; consequently, if 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 the hypothesis space $\mathcal{H}$) iff for all texts T for L, for all counterexample texts T' for $\mathbf{M}$ on the 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 the hypothesis space $\mathcal{H}$) if it **NCEx**-learns all languages in $\mathcal{L}$ (using the hypothesis space $\mathcal{H}$).

(d) $\mathbf{NCEx} = \{\mathcal{L} : (\exists$ recursive learner $\mathbf{M})[\mathbf{M}$ **NCEx**-learns $\mathcal{L}$ using some automatic family $\mathcal{H}$ as the hypothesis space$]\}$.

The learnability criteria **LNCEx** and **BNCEx** for learning from least-counterexample or bounded counterexamples are similarly defined by replacing the counterexample text, in definitions (b)–(d) above, by least counterexample text and bounded counterexample text respectively.

Consistent learning from negative counterexamples can be defined by requiring 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' (with respect to some language L), if $\mathbf{M}(T[t], T'[t]) = (mem, hyp)$ with $hyp \neq ?$, then $\text{content}(T[t]) \subseteq H_{hyp}$ and $\text{content}(T'[t]) \cap H_{hyp} = \emptyset$; furthermore, $\mathbf{M}(T[t], T'[t]) \neq ?$, if the input $T[t]$ and corresponding counterexamples $T'[t]$ are for some language in the class $\mathcal{L}$ being learnt.

Similarly, one can define monotonic and memory bounded learners from negative counterexamples, where for the word-size memory bound constraint, we bound the memory of the learner by the largest word seen in either the text (for positive data) or the counterexample text.

Here a learner using negative counterexamples is called automatic if the corresponding graph of the learner, that is, the mapping $(oldmem, datum, negative\ counterexample) \to (newmem, conjecture)$, is automatic.

We use "**Auto**" in the name of the learning criterion 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 criterion to denote that the learners are consistent and monotonic, respectively. We use **Word** and **Hyp** in the name of the learning criteria to denote that the memory of the learners is appropriately bounded. For iterative learning, as is common in the literature, we replace the term "**Ex**" in the name of the criterion by "**It**". For example, **AutoMonNCIt** denotes **NCEx** learnability by a learner which is automatic, monotonic and iterative. **AutoWordConsNCEx** denotes **NCEx** learnability by a learner which is automatic, word memory size bounded, and consistent.

The following proposition bounds the size of memory and conjectures of automatic learners.

Proposition 10 (Jain, Luo and Stephan [JLS12]). *Suppose* $\mathbf{M}$ *is an automatic learner. Then, there exists a constant* c *such that for all texts* T *and all* $n \in \mathbb{N}$, $|\mathbf{M}(T[n])| \leq c*(n+1)+\max(\{|T(m)| : m < n\})$. *Similarly, for learning from negative counterexamples, for a negative counterexample text* T' *for* $\mathbf{M}$ *on input text* T, $|\mathbf{M}(T[n], T'[n])| \leq c*(n+1)+\max(\{|T(m)| : m < n\} \cup \{|T'(m)| : m < n\})$.

3 Negative and Least Negative Counterexamples

We begin with a useful technical proposition.

Proposition 11. *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* $L_\beta \cap \{x : |x| \leq |u|\} = L_\alpha \cap \{x : |x| \leq |u|\}$.

Proof. Below, we show (a) to (c) for possibly different constants c. Then c, as in the proposition, can be taken to be the maximum of these constants.

(a) This was shown by Jain, Ong, Pu and Stephan [JOPS11].

(b) Note that $ProbExt(\alpha, u)$ is finite due to the indexing of $\mathcal{L}$ being 1–1 and that there can be at most finitely many languages L such that $L_\alpha \subseteq L \subseteq L_\alpha \cup \{x : x \leq_{ll} u\}$. Furthermore, $ProbExt(\alpha, u)$ can be expressed as a first order formula: $\beta \in ProbEx(\alpha, u)$ iff $(\forall x)[[x \in L_\beta \Rightarrow [x \in L_\alpha \vee x \leq_{ll} u]] \wedge [x \in L_\alpha \Rightarrow x \in L_\beta]] \wedge (\exists x)[x \in L_\beta \wedge x \notin L_\alpha]$.

Thus, $m_{\alpha,u}$ = length-lexicographically least β such that $(\forall \beta' \in ProbEx(\alpha, u))[\beta' \leq_{ll} \beta]$ can be expressed as a first order formula. Hence, by Lemma 1, $m_{\alpha,u}$ can be obtained automatically from α and u. Thus, for some constant c, for all α, u, $|m_{\alpha,u}| \leq \max(\{|\alpha|, |u|\}) + c$. Part (b) follows.

(c) For any α and u, let β be length-lexicographically least such that $L_\beta \cap \{x : |x| \leq |u|\} = L_\alpha \cap \{x : |x| \leq |u|\}$.

Let M be a DFA with k states which witnesses that $\mathcal{L}$ is an automatic family. Suppose by way of contradiction that $|\beta| > |u| + c$, where $c = k^k + 1$. Thus, there are at most k possibilities for the state which M might be in after seeing input $conv(\beta[|u|+1], x)$, for any x with $|x| \leq |u|$. Hence, by the pigeonhole principle there exist two distinct n', n'', satisfying $|u| < n' < n'' \leq |\beta|$, such that for all x with $|x| \leq |u|$, the state of M after seeing input $conv(\beta[n'], x)$ is the same as the state of M after seeing input $conv(\beta[n''], x)$. Thus, for $\gamma = \beta(0)\beta(1)\ldots\beta(n'-1)\beta(n'')\beta(n''+1)\ldots\beta(|\beta|-1)$, $L_\gamma \cap \{x : |x| \leq |u|\} = L_\beta \cap \{x : |x| \leq |u|\} = L_\alpha \cap \{x : |x| \leq |u|\}$, a contradiction to the assumption of β being length-lexicographically least index satisfying $L_\beta \cap \{x : |x| \leq |u|\} = L_\alpha \cap \{x : |x| \leq |u|\}$. ∎

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 of length up to $|u|$.

For **TxtEx**-learning, *locking-sequence* for a learner $\mathbf{M}$ on a language L is a finite sequence σ such that $\text{content}(\sigma) \subseteq L$, $\mathbf{M}(\text{content}(\sigma))$ is a hypothesis for L, and for all τ such that $\text{content}(\sigma) \subseteq L$, $\mathbf{M}(\sigma\tau) = \mathbf{M}(\sigma)$. That is, the learner $\mathbf{M}$ does not change its mind beyond σ as long as the information is consistent with the language L. Blum and Blum [BB75] showed that for any learner $\mathbf{M}$ for any language L **TxtEx**-learnt by $\mathbf{M}$, there exists a locking-sequence for $\mathbf{M}$ on L. The following Proposition is a modification of this locking-sequence lemma for **TxtEx**-learning adapted to **NCEx**-learning for hypothesis-size memory bounded learners.

In the Proposition (σ, τ) represents the locking sequence with respect to **NCEx** (**LNCEx**, **BNCEx**)-learning for hypothesis-size memory bounded learner $\mathbf{M}$. Due to memory being bounded as above, the convergence to a conjecture α, forces the memory beyond the locking sequence to be from a finite set.

Proposition 12. *Suppose* $\mathbf{M}$ *is a hypothesis-size memory bounded learner which* **LNCEx** *or* **NCEx** *or* **BNCEx**-*learns L with memory using the alphabet Γ. Then there exists a (σ, τ) such that, $\sigma \in SEQ(L)$, τ gives least (least bounded in case of* **BNCEx**-*learning) counterexamples to* $\mathbf{M}$ *for the input σ and the language L, and for some finite $S \subseteq \Gamma^*$ and index α for L,*

(a) $\mathbf{M}(\sigma, \tau) = (mem, \alpha)$, *for some* $mem \in S$,
(b) $(\forall \sigma' \in SEQ(L))[\mathbf{M}(\sigma \diamond \sigma', \tau \diamond \#^{|\sigma'|}) = (mem, \alpha)$, *for some* $mem \in S]$ *and*
(c) $(\forall mem_1, mem_2 \in S)(\exists \gamma \in SEQ(\text{content}(\sigma)))\ [\mathbf{M}(mem_1, \gamma, \#^{|\gamma|}) = (mem_2, \alpha)]$.

Proof. Suppose $\mathbf{M}$ **LNCEx** or **NCEx** or **BNCEx**-learns L. Suppose we always provide least (least bounded) counterexamples to the learner $\mathbf{M}$. Then, for learning of L by $\mathbf{M}$ (according to the relevant criterion) the counterexample text given to $\mathbf{M}$ is uniquely determined from the text for L.

Claim 13. *There exists a sequence σ' with* $\text{content}(\sigma') \subseteq L$ *such that, for all σ'' extending σ' satisfying* $\text{content}(\sigma'') \subseteq L$, *the hypothesis of* $\mathbf{M}$ *on (σ', τ') is the same as the hypothesis of*

$\mathbf{M}$ *on* (σ'', τ'')*, where* τ' *and* τ'' *are the sequences providing the least (least bounded) negative counterexample to* $\mathbf{M}$ *on* σ' *and* σ'' *respectively for the language* L.

Suppose by way of contradiction that the claim does not hold. Let T be a text for L. Let $\sigma_0 = \tau_0 = \Lambda$. Inductively define σ_{i+1} to be the extension of $\sigma_i \diamond T(i)$ such that the hypothesis of $\mathbf{M}$ on $(\sigma_{i+1}, \tau_{i+1})$ is different from the hypothesis of $\mathbf{M}$ on (σ_i, τ_i), where τ_{i+1} is the sequence providing the least (least bounded) negative counterexample to $\mathbf{M}$ on σ_{i+1} for the language L; if there is no such σ_{i+1}, then $\sigma_i = \sigma'$ satisfies the requirements of the claim. Now $\bigcup_{i\in\mathbb{N}} \sigma_i$ is a text for L, with $\bigcup_{i\in\mathbb{N}} \tau_i$ being the corresponding text for least (least bounded) counterexamples. However, $\mathbf{M}$ does not converge to a hypothesis on $(\bigcup_{i\in\mathbb{N}} \sigma_i, \bigcup_{i\in\mathbb{N}} \tau_i)$. This contradiction proves the claim.

Let σ' be as in the claim, and τ' be the sequence providing the least (least bounded) negative counterexample to $\mathbf{M}$ on σ' for the language L. We take α to be the hypothesis of $\mathbf{M}$ on (σ', τ'). As $\mathbf{M}$ is a hypothesis-size memory bounded learner, for all σ'' such that $\sigma' \subseteq \sigma''$ and $\text{content}(\sigma'') \subseteq L$, memory of $\mathbf{M}$ after seeing input (σ'', τ'') is bounded in length by $|\alpha|$ plus a constant, where τ'' is the sequence providing the least (least bounded) negative counterexample to $\mathbf{M}$ on σ'' for the language L. Let S' be the set of possible memory values of $\mathbf{M}$ after seeing some extension σ'' of σ' (along with the corresponding least (least bounded) negative counterexample sequences for the language L) such that $\text{content}(\sigma'') \subseteq L$. For any σ'' extending σ' such that $\text{content}(\sigma'') \subseteq L$, let $S_{\sigma''} = \{mem : (\exists \sigma''')[\text{content}(\sigma''') \subseteq L \wedge \mathbf{M}(\sigma''', \tau''') = (mem, \alpha)]\}$, where τ''' in the above is the sequence of least (least bounded) negative counterexamples to $\mathbf{M}$ on the input σ''' for the language L. Note that $S_{\sigma''} \subseteq S'$. Let S be the minimal among all $S_{\sigma''}$ such that σ'' extends σ' and $\text{content}(\sigma'') \subseteq L$. Intuitively, this S satisfies the requirement that, after the learner has converged to its hypothesis α, the learner can attain all memory values in S from other memory values in S using suitable sequences from L. For $mem, mem' \in S$, let $\gamma_{mem,mem'}$ be such that $\text{content}(\gamma_{mem,mem'}) \subseteq L$ and $\mathbf{M}(mem, \gamma_{mem,mem'}, \#^{|\gamma_{mem,mem'}|}) = (mem', \alpha)$ (by minimality of S, such $\gamma_{mem,mem'}$ exist).

Let σ be an extension of σ'' such that $\text{content}(\gamma_{mem,mem'}) \subseteq \text{content}(\sigma) \subseteq L$ for all $mem, mem' \in S$. Let τ be the sequence of least (least bounded) negative counterexamples to $\mathbf{M}$ on the input σ for language L. Now it is easy to verify that σ, τ, S satisfy the requirements of the proposition. ∎

Intuitively, the above proposition gives a special kind of locking sequence [BB75], where for hypothesis-size memory bounded learners, the memory is also constrained to be from a finite set S, where, for any mem_1 and mem_2 in S, sequences of elements from the locking sequence can be used to transform the memory from mem_1 to mem_2 (beyond the locking sequence).

Our first major result shows that automatic **NCIt**-learners and **NCEx**-learners, with word-size memory bound, can learn any automatic class. Note that this shows that automatic **NCIt**-learning and **NCEx**-learning are much more powerful than automatic learning from positive data alone, as the latter even fails to learn certain classes of finite sets such as $\{\{0,1\}^{|\alpha|} - \{\alpha\} : \alpha \in \{0,1\}^*\}$ [JLS12].

The learning algorithm in this result, as well as several theorems below, is based on the well-known idea of "identification by enumeration", whereby the learner goes through potentially all conjectures in the index set, checking them one by one and doing also a sort of "side checking".

Theorem 14. *Consider any automatic family $\mathcal{L} = \{L_\alpha : \alpha \in I\}$. Then, the following hold:*

(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. (a) 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 by 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 11 (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, 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 $H_{conv(\alpha,u,\beta,b)} = L_\beta$, else $H_{conv(\alpha,u,\beta,b)} = L_\alpha$ (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 needs to further 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, the 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 11(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 and for the m being the least number such that $\alpha_m = \alpha_n$:

(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 $u_n \leq_{ll} u_{n+1}$.
(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 $u_n = u_m$, else $u_n = \max(\{u_m\} \cup \{T(s) : m \leq s < n\} - L_{\alpha_n})$.
(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 **M** computes the new memory and hypothesis

$$conv(\alpha_{n+1}, u_{n+1}, \beta_{n+1}, b_{n+1}),$$

from its previous memory $conv(\alpha_n, u_n, \beta_n, b_n)$ and current positive datum $T(n)$, and counterexample datum $T'(n)$. This updating would be automatic. Note that the learner does not use n for updating; usage of n in the following was more for ease of proving that the learner succeeds. Similar remark applies for update algorithms in later proofs in this paper.

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^*.

Update Algorithm of the learner M

Input: $conv(\alpha_n, u_n, \beta_n, b_n), T(n), T'(n)$

Output: $conv(\alpha_{n+1}, u_{n+1}, \beta_{n+1}, b_{n+1})$, which is both new memory and new hypothesis.

(* Note that $\alpha_{n+1}, u_{n+1}, \beta_{n+1}, b_{n+1}$ as defined below can be computed automatically from $T(n), T'(n), \alpha_n, u_n, \beta_n, b_n$. *)

1. If $[T(n) = \#$ or $[\alpha_n \neq i_0$ and $T(n) \in L_{\alpha_n}]]$, then $u_{n+1} = u_n$, else $u_{n+1} = \max(\{u_n, T(n)\})$.

2. (* $\alpha_{n+1}, \beta_{n+1}, b_{n+1}$ are defined based on the following cases. Note that these cases are first order definable and thus automatic. *)

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 $i_0 \notin I$). *)

$\alpha_{n+1} = \beta_n$.
$\beta_{n+1} = \beta_n$.
$b_{n+1} = 0$.

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. *)

$\alpha_{n+1} = \alpha_n$.
$\beta_{n+1} = \text{succ}_I(\beta_n)$.

If $b_n = 1$ or $T(n) \notin L_{\alpha_n}$, then $b_{n+1} = 1$, else $b_{n+1} = 0$.

Case 2: Not Case 1.

(* In this case $H_{hyp_n} = L_{\alpha_n}$. *)

$\alpha_{n+1} = \alpha_n$.

$\beta_{n+1} = \beta_n$.

If $b_n = 1$ or $T(n) \notin L_{\alpha_n}$, then $b_{n+1} = 1$, else $b_{n+1} = 0$.

End Update Alogrithm

It is now easy to verify that the learner is automatic. The definition of $\alpha_0, \beta_0, b_0, u_0$ clearly maintains 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 ($\beta_n \leq_{ll} \beta_{n+1}$ 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 $L_\alpha = L$, 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, $H_{conv(\alpha,u,\beta,b)} = L_\alpha$. Furthermore, using the invariants (I3), (I4) and Proposition 11(b), we have that $L_\alpha = L$.

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

(b) This proof is a modification of the proof of part (a) to allow the learner to use $\mathcal{L}$ itself as its hypothesis space. This however requires the learner to store in memory the length-lexicographically largest word seen in the input. Thus, the learner is word-size memory bounded and not iterative.

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 11 (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: $conv(\alpha, w, u, \beta)$, 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 $mem_n = conv(\alpha_n, w_n, u_n, \beta_n)$.

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 $L_{\alpha_n} \not\subseteq L$ and $\mathbf{M}$ conjectures L_{α_n}, then the learner will get a counterexample for it. 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 ones). 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 bounded using Proposition 11(b).

We proceed formally now. The invariants maintained by the learner related to the memory are as follows. For all n and for the least number m such that $\alpha_m = \alpha_n$:

(I1) $w_n = \max(\text{content}(T[n]))$.
(I2) $\alpha_n \leq_{ll} \alpha_{n+1}$.
(I3) For all $\alpha' \in I$ such that $\alpha' <_{ll} \alpha_n$, $L_{\alpha'} \neq L$; furthermore, either $\text{content}(T'[n]) \cap L_{\alpha'} \neq \emptyset$ or there exists an $x \leq_{ll} w_n$, $x \in L - L_{\alpha'}$.
(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 with $\alpha_n <_{ll} \beta <_{ll} \beta_n$, if $L_{\alpha_n} \subset L_\beta \subseteq L_{\alpha_n} \cup \{x : x \leq_{ll} u_n\}$, then $L_\beta \not\subseteq L$,
(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 $hyp_n = \beta_n$, else $hyp_n = \alpha_n$. Thus, it is enough to specify how the learner computes $\alpha_n, w_n, u_n, \beta_n$.

Initially, $\alpha_0 = \beta_0 = \min(I)$, $w_0 = u_0 = \#$. We now describe how memory of the learner is updated after receiving input $T(n), T'(n)$ (where the previous memory is $conv(\alpha_n, w_n, u_n, \beta_n)$).

Update Algorithm of the learner M

Input: $conv(\alpha_n, w_n, u_n, \beta_n)$, $T(n)$, $T'(n)$.
Output: $conv(\alpha_{n+1}, w_{n+1}, u_{n+1}, \beta_{n+1})$.

(* $\alpha_{n+1}, w_{n+1}, u_{n+1}, \beta_{n+1}$, are defined based on the following cases. Note that these cases are first order definable and thus automatic. *)

Case 1: $T(n) \notin L_{\alpha_n} \cup \{\#\}$ 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, $[L_{\beta_n} \cap \{x : x \leq_{ll} u_n\} \subseteq L$ and $L_{\beta_n} \cap \{x : x \leq_{ll} u_n\} \not\subseteq L_{\alpha_n}]$. Furthermore, using (I1) and (I4), as $u_n \leq w_n$, note that either $L_{\alpha_n} \cap \text{content}(T'[n+1]) \neq \emptyset$, or there exists a $x \leq_{ll} \max(\{w_n, T(n)\})$ such that $x \in L - L_{\alpha_n}$. *)

$\alpha_{n+1} = \beta_{n+1} = \text{succ}_I(\alpha_n)$.

$w_{n+1} = u_{n+1} = \max(\{w_n, T(n)\})$.

(* Note that $\max(\text{content}(T[n+1])) = \max(\{w_n, T(n)\})$. *)

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). *)

$\alpha_{n+1} = \alpha_n$.

$\beta_{n+1} = \text{succ}_I(\beta_n)$.

$u_{n+1} = u_n$.

$w_{n+1} = \max(\{w_n, T(n)\})$.

(* Note that $\max(\text{content}(T[n+1])) = \max(\{w_n, T(n)\})$. *)

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 $T'(n) = \#$. *)

$\alpha_{n+1} = \alpha_n$,

$\beta_{n+1} = \beta_n$,

$u_{n+1} = u_n$,

$w_{n+1} = \max(\{w_n, T(n)\})$.

(* Note that $\max(\text{content}(T[n+1])) = \max(\{w_n, T(n)\})$. *)

End Update Algorithm

Clearly, the learner **M** is automatic.

The invariants (I1), (I2), (I4), (I5), (I6)(iii), (iv) are clearly maintained by the construction. We now consider (I3) and (I6)(i) and (ii). For (I3) note that Case 1 is the only case where $\alpha_n \neq \alpha_{n+1} = \text{succ}_I(\alpha_n)$, and in this case $L_{\alpha_n} \neq L$, and either $\text{content}(T'[n+1]) \cap L_{\alpha_n} \neq \emptyset$ or there exists an $x \leq_{ll} w_{n+1}$, $x \in L - L_{\alpha_n}$. For (I6)(i), note that if $T(n)$ is not in $L_{\alpha_n} \cup \{\#\}$, 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.

Note that due to the index set I being regular, if I is infinite, then for all $w \in I$, $|\text{succ}_I(w)| \leq |w|+$ a constant. Thus, by (I5), (I6)(iii), (iv), we have that the length of β is at most a constant more than $\max(\{|\alpha_n|, |u_n|\})$, which by (I1) and (I4) is at most $\max(\{|\alpha_n|, |w_n|\})$. Furthermore, by (I1), (I3) and Proposition 11(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, **M** is word-size memory bounded for input texts for languages in the class $\mathcal{L}$.

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. Now, by (I6)(i), $L \subseteq L_\alpha \cup \{x : x \leq u\}$. Furthermore, using the invariants (I5), (I6)(iii) and (I6)(iv), we have that, for all but finitely many n, $|\beta_n| > \max(\{|\alpha|, |u|\}) + c$. Thus, by (I3) and (I6)(ii), for all but finitely many n, for all $\beta <_{ll} \beta_n$ with $\beta \neq \alpha$, $L_\beta \neq L$. Thus, using Proposition 11(b), we have that either, $L_\alpha \not\subseteq L$ or $L_\alpha = L$. Furthermore, by the definition of hyp_n, for all but finitely many n, $hyp_n = \alpha$. Thus, if $L_\alpha \not\subseteq L$, then we must have $T'(n) \in L_\alpha$, and thus Case 1 holding and forcing an update of α, for large enough n, a contradiction.

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

Hypotheses of the learner in the above theorem are not consistent with the data seen so far. As the next theorem shows, for **LNCEx**-learning, we can make the learner consistent, when the learner is allowed a class comprising hypothesis space.

Theorem 15. *Let $\mathcal{L}$ be an automatic class. Then, $\mathcal{L} \in$ **AutoWordConsLNCEx** as witnessed by a learner using a class comprising hypothesis space.*

Proof. The memory used by the learner is of the form $conv(\alpha, \beta, x, y, z, z', b)$, where $\alpha, \beta \in I$, $x, y, z, z' \in \Sigma^* \cup \{\#\}$, $b \in \{0, 1\}$, where $\#$ is a special symbol such that $\# <_{ll} u$, for all $u \in \Sigma^*$.

The hypothesis of the learner is directly linked to its memory. If the memory is $conv(\alpha, \beta, x, y, z, z', b)$, then the hypothesis is $conv(\alpha, \beta, x, y, z', b)$. Note that the hypothesis of the learner does not depend on z. Let

$$H_{conv(\alpha,\beta,x,y,z',0)} = (L_\alpha \cap \{u : u \leq_{ll} x\}) \cup \{u : x <_{ll} u \leq_{ll} y\} \cup (\{u : u >_{ll} y\} \cap L_\beta) \text{ and}$$
$$H_{conv(\alpha,\beta,x,y,z',1)} = ((L_\alpha \cap \{u : u \leq_{ll} x\}) \cup \{u : u >_{ll} x\}) - \{z'\}.$$

Suppose L is the input language. Intuitively, the learner has determined the membership question for all $u \leq_{ll} x$ in the input language (as given by α). The parameter z denotes the largest example/counterexample that the learner has received.

When $b = 0$, the learner has not received any negative counterexamples beyond x. The learner is currently testing if L_β seems like an index for L (at least for data beyond y). Furthermore, y bounds any missing positive data from L_β (that is all $u \geq_{ll} y$ in the positive data seen so far are known to belong to L_β). In case the learner gets an example/counterexample inconsistent with L_β, it will update β to be the next length-lexicographically least element in I (and update y appropriately to be z). In case the learner gets a (least) counterexample z' (where $z' \leq_{ll} y$) to its conjecture $H_{conv(\alpha,\beta,x,y,z',0)}$, it will update x to be the counterexample, and adjust α appropriately (note that, due to the counterexample being least we will have that $\{u : x <_{ll} u <_{ll} z'\} \subseteq L$, and thus an appropriate update of α can be found (assuming $L \in \mathcal{L}$)). In case the counterexample $z' >_{ll} y$, then the learner will set $b = 1$, and through a series of steps (see Case 2 below) it will determine all the elements of the language below z' and update the parameters appropriately; after this, the learner will set $b = 0$ and go back to searching for appropriate β as above.

We now proceed formally.

Suppose L is the input language and T is a text for L and T' is the sequence of counterexamples. Suppose memory of the learner after seeing $T[n], T'[n]$ is $conv(\alpha_n, \beta_n, x_n, y_n, z_n, z'_n, b_n)$.

For $L \in \mathcal{L}$, the invariants maintained by the algorithm are as follows. For all n:

(I1) $\beta_n \leq_{ll} \beta_{n+1}$, $\alpha_n \leq_{ll} \alpha_{n+1}$, $x_n \leq_{ll} x_{n+1}$, $y_n \leq_{ll} y_{n+1}$ and $z_n \leq_{ll} z_{n+1}$.
(I2) $x_n \leq_{ll} y_n \leq_{ll} z_n$.
(I3) $z_n = \max(\text{content}(T[n]) \cup \text{content}(T'[n]))$.
(I4) $\max(\text{content}(T[n]) - L_{\beta_n}) \leq_{ll} y_n$.
(I5) For all $\beta \in I$ such that $\beta <_{ll} \beta_n$, $L_\beta \neq L$.
(I6) $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} = L \cap \{u : u \leq_{ll} x_n\}$; Furthermore, α_n is the least such index in I.
(I7) $\text{content}(T[n]) \cap \{u : y_n <_{ll} u \leq_{ll} z_n\} \subseteq L_{\beta_n} \cap \{u : y_n <_{ll} u \leq_{ll} z_n\}$ and $\text{content}(T'[n]) \cap \{u : y_n <_{ll} u \leq_{ll} z_n\} \cap L_{\beta_n} = \emptyset$.
(I8) If $b_n = 0$, then $x_n \geq_{ll} \max(\text{content}(T'[n]))$.
If $b_n = 1$, then $x_n \geq_{ll} \max(\text{content}(T'[n]) - \{z'_n\})$; furthermore $x_n <_{ll} z'_n$ and $z'_n \notin L$.
(I9) If $b_n = b_{n+1} = 0$, $\beta_n = \beta_{n+1}$, then $y_{n+1} = y_n$.
(I10) If $b_n = 0$ and $b_{n+1} = 1$, then $\beta_{n+1} \neq \beta_n$ and $y_{n+1} = z_{n+1}$.

Initially, $x_0 = y_0 = z_0 = \#$, $b_0 = 0$, $\alpha_0 = \beta_0 = \min(I)$. The value of z'_0 is irrelevant (and we can take it to be $\#$). Note that z'_n is relevant only when $b_n = 1$.

Update of the parameters is done as follows. If $L \notin \mathcal{L}$, then in some cases below, the index set may not contain appropriate α_{n+1} or β_{n+1} as needed; in that case the learner can stop the learning process by just outputting a special symbol to denote that the input language is outside the class. As long as appropriate $\alpha_{n+1}, \beta_{n+1}$ exist, the invariants above will be maintained.

Update Algorithm of the learner M
Input: $T(n), T'(n), conv(\alpha_n, \beta_n, x_n, y_n, z_n, z'_n, b_n)$.
Output: $conv(\alpha_{n+1}, \beta_{n+1}, x_{n+1}, y_{n+1}, z_{n+1}, z'_{n+1}, b_{n+1})$.

$z_{n+1} = \max(\{z_n, T(n), T'(n)\})$.
(* Remaining values are defined based on the following cases. Note that these cases are first order definable and thus automatic. *)

Case 1: $b_n = 0$.
If ($T(n) = \#$ or $T(n) \in L_{\beta_n}$) and $T'(n) \notin L_{\beta_n}$,
Then $\beta_{n+1} = \beta_n$ and $y_{n+1} = y_n$,
Else (* that is, if ($T(n) \neq \#$ and $T(n) \notin L_{\beta_n}$) or $T'(n) \in L_{\beta_n}$; note that in this case $L_{\beta_n} \neq L$ *)
$\beta_{n+1} = \text{succ}_I(\beta_n)$ and $y_{n+1} = z_{n+1}$.

Case 1a: $T'(n) = \#$.
$x_{n+1} = x_n, b_{n+1} = b_n, \alpha_{n+1} = \alpha_n$, $z'_{n+1} = z'_n$.

Case 1b: $T'(n) \neq \#$.
(* Note that here $T'(n) >_{ll} x_n$. *)

Case 1b.1: $T'(n) \leq_{ll} y_n$.
(* Note that $\{u : x_n <_{ll} u <_{ll} T'(n)\} \subseteq L$. *)
$x_{n+1} = T'(n)$.
$b_{n+1} = 0$.

$z'_{n+1} = \#$. (* Value of z'_{n+1} is irrelevant in this case and we can take it to be $\#$. *)

α_{n+1} is the least index such that $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : x_n <_{ll} u <_{ll} x_{n+1}\} = L_{\alpha_{n+1}} \cap \{u : u \leq_{ll} x_{n+1}\}$.

Case 1b.2: $T'(n) >_{ll} y_n$.

(* Note that $\{u : x_n <_{ll} u \leq_{ll} y_n\} \subseteq L$. In this case, the learner needs to figure out membership question for u such that $y_n <_{ll} u < T'(n)$. It will use Case 2 (by setting $b_{n+1} = 1$) to do this. *)

$x_{n+1} = y_n$, $b_{n+1} = 1$ and $z'_{n+1} = T'(n)$.

α_{n+1} is the least index such that $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : x_n <_{ll} u \leq_{ll} x_{n+1}\} = L_{\alpha_{n+1}} \cap \{u : u \leq_{ll} x_{n+1}\}$.

Case 2: $b_n = 1$.

Case 2a: $T'(n) = \#$.

(* In this case the input language is $H_{conv(\alpha_n, \beta_n, x_n, y_n, z'_n, 1)} = L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : u >_{ll} x_n\} - \{z'_n\}$, and we can complete the learning process. *)

$x_{n+1} = z'_n$,

α_{n+1} is the least index such that $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : x_n <_{ll} u <_{ll} x_{n+1}\} = L_{\alpha_{n+1}} \cap \{u : u \leq_{ll} x_{n+1}\}$.

$b_{n+1} = 0$, $y_{n+1} = z_{n+1}$.

β_{n+1} is the least index $\geq_{ll} \beta_n$ such that $\{u : u >_{ll} z_{n+1}\} \subseteq L_{\beta_{n+1}}$.

$z'_{n+1} = \#$. (* Value of z'_{n+1} is irrelevant in this case and we can take it to be $\#$. *)

(* Now, future input will always lead to Case 1a, and thus will update only z_{n+1}. *)

Case 2b: $T'(n) \neq \#$ and $T'(n) <_{ll} z'_n$.

(* Note that $\{u : x_n <_{ll} u <_{ll} T'(n)\} \subseteq L$. *)

$x_{n+1} = T'(n)$, $y_{n+1} = z_{n+1}$, $\beta_{n+1} = \beta_n$, $b_{n+1} = 1$, $z'_{n+1} = z'_n$.

α_{n+1} is the least index such that $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : x_n <_{ll} u <_{ll} x_{n+1}\} = L_{\alpha_{n+1}} \cap \{u : u \leq_{ll} x_{n+1}\}$.

Case 2c: $T'(n) \neq \#$ and $T'(n) \geq_{ll} z'_n$.

(* Note that $(\{u : x_n <_{ll} u <_{ll} T'(n)\} - \{z'_n\}) \subseteq L$. *)

$x_{n+1} = T'(n)$, $y_{n+1} = z_{n+1}$, $\beta_{n+1} = \beta_n$, $b_{n+1} = 0$.

$z'_{n+1} = \#$. (* Value of z'_{n+1} is irrelevant in this case and we can take it to be $\#$. *)

α_{n+1} is the least index such that $L_{\alpha_n} \cap \{u : u \leq_{ll} x_n\} \cup \{u : x_n <_{ll} u <_{ll} x_{n+1}\} - \{z'_n\} = L_{\alpha_{n+1}} \cap \{u : u \leq_{ll} x_{n+1}\}$.

End Update Algorithm

Now, it is easy to verify that the construction maintains invariants (I1), (I2), (I3), (I6), (I10). Note that the requirement for Case 1b.2 implies that $T'(n) \in L_{\beta_n}$. Furthermore, note that $\alpha_n \leq_{ll} \alpha_{n+1}$ in (I1) follows using (I6) and $x_n \leq x_{n+1}$.

For (I5) note that the learner only makes $\beta_{n+1} \neq \beta_n$ in Case 1 or Case 2a: In Case 1, this is done only when $T(n), T'(n)$ witness that $L_{\beta_n} \neq L$; In Case 2a clearly (I5) is maintained as β_{n+1} is chosen appropriately to be the least one such that $L_{\beta_{n+1}}$ contains $\{u : u >_{ll} z_{n+1}\}$ (which is contained in L). For (I4) and (I9), note that Case 2 always keeps $y_{n+1} = z_{n+1}$; furthermore, in Case 1, either, $\beta_{n+1} = \mathrm{succ}_I(\beta_n)$ and $y_{n+1} = z_{n+1}$, or [[$T(n) = \#$ or $T(n) \in L_{\beta_n}$] and $T'(n) \notin L_{\beta_n}$ and $y_{n+1} = y_n$ and $b_{n+1} = 0$]; thus (I4) and (I9) also are maintained (here note (I10)). For (I7), Case 2 trivially maintains the property, and in Case 1, if $T(n) \neq \#$ and $T(n) \notin L_{\beta_n}$ or $T'(n) \in L_{\beta_n}$, then we have $\beta_{n+1} \neq \beta_n$ and $y_{n+1} = z_{n+1}$; thus (I7) is maintained. For (I8) note that, the definition of x_{n+1} in each of the cases ensures that z'_{n+1} (in case $b_{n+1} = 1$) is the only negative datum which may be above x_{n+1}. Thus, all the invariants are satisfied.

Now, suppose $L \in \mathcal{L}$. Thus, by (I1) and (I6) we have that $\lim_{n\to\infty} \alpha_n$ converges, to, say, α and by (I1) and (I5), $\lim_{n\to\infty} \beta_n$ converges, to say β.

Note that if, for some n, Case 2a holds, then the learner converges to a correct hypothesis. Furthermore, we claim that Case 1 holds for infinitely many n. To see this, note that if, for all $n \geq n'$, Case 2 holds, then it must be the case that for all $n \geq n'$ Case 2b holds (as both Case 2a and 2c make $b_{n+1} = 0$). But then we will have that $x_{n'}, x_{n'+1}, x_{n'+2}, \ldots$ form a monotonically increasing sequence below $z'_{n'}$, a contradiction. Thus, Case 1 holds for infinitely many n.

Let n_0 be large enough so that $\alpha_{n_0} = \alpha$ and $\beta_{n_0} = \beta$ and $b_{n_0} = 0$. Thus, for all $n \geq n_0$, we have $b_{n+1} = 0$, $T'(n) \notin L_{\beta_n}$ and [$T(n) = \#$ or $T(n) \in L_{\beta_n}$] (otherwise, $\beta_{n+1} \neq \beta_n$; note that the requirement for Case 1b.2 implies that $T'(n) \in L_{\beta_n}$). Thus, by (I9), $\lim_{n\to\infty} y_n$ converges, to, say, y. It thus follows from (I1) and (I2), that $\lim_{n\to\infty} x_n$ converges, to, say, x. Thus, for all but finitely many n, Case 1a holds, and $\lim_{n\to\infty} z'_n$ also converges, to say z'. Thus, it follows by (I4), (I6), definition of $H_{conv(\alpha_{n_0},\beta_{n_0},x,y,z',0)}$ and the learner not receiving any further counterexamples that the learner converges to a correct hypothesis for L.

Furthermore, by (I2), (I4), (I6), (I7) and (I8), the learner is consistent. The only place the learner above can become inconsistent (rather undefined) is if in some step (i) it does not find an α_{n+1} as needed in Cases 1b, 2a, 2b and 2c, or (ii) it does not find a β as in Case 2a; in such cases the input language is not from the class being learnt. Since these cases can be detected automatically, the learner can output a special symbol to denote the situation (thus, the learner is always consistent when it outputs a non-special symbol as its hypothesis). ∎

Theorem 16. *There exists an automatic class $\mathcal{L}$ which cannot be* **AutoConsLNCEx***-learnt by any automatic learner using a class preserving hypothesis space.*

Proof. For $x \in \{0,1\}^*, y \in \{3,4\}^*$, let $L_{conv(x,y)} = \{z \in \{2,3,4\}^* : |z| \neq |y| \text{ or } z = y\} \cup \{z : z \in \{0,1\}^*, \neg[z \preceq x]\}$.

Let $\mathcal{L} = \{L_{conv(x,y)} : x \in \{0,1\}^*, y \in \{3,4\}^*\}$.

Clearly, $\mathcal{L}$ is an automatic class. For all $n > 0$, note that every $L \in \mathcal{L}$ misses out at most one string in $\{0,1\}^n$.

Suppose by way of contradiction that $\mathbf{M}$ is an automatic learner which **AutoConsLNCEx**-learns $\mathcal{L}$ using a class preserving hypothesis space. As $\mathbf{M}$ is an automatic learner, by Proposition 10, after $\mathbf{M}$ has seen k data items and potentially k counterexamples, each of length at most r, the length of $\mathbf{M}$'s memory and hypothesis are bounded by $c * (k+1) + r$, for some constant c. Let $n > 1$ be large enough so that 2^{n^2} is greater than the number of possibilities of $(memory, hypothesis)$ of length $c * (n+1) + 2n$.

For every set S containing exactly one string from each of $\{3,4\}^m$, for $m = n+1, n+2, \ldots, 2n$, let σ_S denote the sequence of length n containing elements of S in length increasing order.

The negative counterexample sequence τ_S is constructed as follows. For $m < n$, let $\tau_S(m)$ be the length-lexicographically least element extending $\tau_S(m-1)$ (where, $\tau_S(-1)$ is taken to be ϵ) in $L_{\mathbf{M}(\sigma_S[m],\tau_S[m])}$. Note that each $\tau_S(m) \in \{0,1\}^*$. Note that the memory of the learner after (σ_S, τ_S) is bounded in length by $c*(n+1)+2n$. Thus, there exist two different S, S', such that the memory and hypothesis of the learner after seeing (σ_S, τ_S), and $(\sigma_{S'}, \tau_{S'})$ are the same, say these memory and hypothesis are respectively *mem* and *hyp*. Suppose, y, y' are two different strings of the same length such that $y \in S$ and $y' \in S'$. If L_{hyp} does not contain either y or y', then the learner is clearly not consistent. If L_{hyp} contains both y and y', then L_{hyp} also contains $2^{|y|}$. Now, consider $\mathbf{M}(\sigma_S \diamond 2, \tau_S \diamond 2^{|y|})$ and $\mathbf{M}(\sigma_{S'} \diamond 2, \tau_{S'} \diamond 2^{|y|})$. Both hypotheses must be the same, not contain $2^{|y|}$, but contain y and y' for consistency. However, there is no such language in the class. ∎

4 Bounded Negative Counterexamples

Bounded negative counterexamples are not as powerful as negative counterexamples in general; indeed, Jain and Kinber [JK08] provided a family which is not learnable from bounded negative counterexamples, even if one would use recursive in place of automatic learners. This family can be written as $\{L : \forall v, w : v \leq_{ll} w \wedge w \in L \Rightarrow v \in L\}$ and it consists of Σ^* and all length-lexicographic intervals $L_w = \{v : v \leq_{ll} w\}$ and $\emptyset$. This construction can be generalised to any infinite language L rather than just Σ^*, by considering suitable initial parts of L.

The above mechanism can be formalised using cut-offs which are defined as follows. We say that w is a *cut-off* for a language L_α in $\mathcal{L}$ iff $w \in L_\alpha$ and $\{v : v \leq_{ll} w \text{ and } v \in L_\alpha\}$ is also in $\mathcal{L}$. Then a class is not **BNCEx**-learnable if some infinite language in it has infinitely many cut-offs in $\mathcal{L}$. Theorem 18 shows that this criterion is indeed a characterisation.

Definition 17. A class $\mathcal{L}$ is *cut-off sparse* iff each language $L \in \mathcal{L}$ has only finitely many cut-offs in $\mathcal{L}$.

Theorem 18. *An automatic family $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is* **AutoWordBNCEx**-*learnable iff $\mathcal{L}$ is cut-off sparse. In the case that $\mathcal{L}$ is* **AutoWordBNCEx**-*learnable, the learner can use the hypothesis space $(H_\alpha)_{\alpha \in I}$, where $H_\alpha = L_\alpha$.*

Proof. Without loss of generality assume that the indexing I is 1–1 and $\mathcal{L}$ is infinite. Note that only infinite languages can have infinitely many cut-offs in $\mathcal{L}$. If $\mathcal{L}$ contains an infinite language with infinitely many cut-offs in $\mathcal{L}$, then $\mathcal{L}$ is not **BNCEx**-learnable [JK08]. So assume that every infinite L_α has only finitely many cut-offs in $\mathcal{L}$. The task is now to show that this class is **AutoWordBNCEx**-learnable. Without loss of generality, assume that all L_α are non-empty, as a learner can easily be adjusted when the empty language is added to the class.

Define the following function f from I to Σ^*: $f(\alpha) = \min(\{w \in L_\alpha : \text{either } w = \max(L_\alpha) \text{ or } [L_\alpha \text{ is infinite}, |w| > |\alpha| \text{ and } w \text{ properly bounds all cut-offs of } L_\alpha \text{ in } \mathcal{L}]\})$. Clearly, $f(\alpha) \neq \#$, for all $\alpha \in I$.

Note that the "either-case" applies if L_α is finite and that the "or-case" applies if L_α is infinite. It never happens that both cases or none can apply, hence f is total. It should also be noted that only finitely many α are mapped by f to the same value w, as there are only finitely many finite sets with the maximum w in the "either-case" and only finitely many indices shorter than w in the "or-case". Furthermore, the predicate $\{v \in L_\alpha : v \leq_{ll} w\} \notin \mathcal{L}$ can be rewritten as $(\forall \beta \in I)(\exists u \in \Sigma^*)[[(u >_{ll} w \Rightarrow u \in L_\beta)] \wedge [u \leq_{ll} w \Rightarrow [u \in (L_\beta - L_\alpha) \cup (L_\alpha - L_\beta)]]]$. Hence f is first-order definable from automatic parameters and thus automatic. Define the ordering $\sqsubset$ as: $\alpha \sqsubset \beta$ iff $[f(\alpha) <_{ll} f(\beta)$ or $f(\alpha) = f(\beta) \wedge L_\alpha >_{lex} L_\beta]$, where $L_\alpha >_{lex} L_\beta$ means that the length-lexicographically least element u of the symmetric difference of L_α and L_β is in L_α. It is easy to see that $\sqsubset$ has the same order type as the set of natural numbers.

Claim 19. *If $\alpha \sqsubset \gamma$ and $L_\alpha \not\subseteq L_\gamma$, then hypothesis α would eventually get a negative counterexample on any text for L_γ.*

The claim holds as if $\max(L_\gamma) \geq_{ll} \min(L_\alpha - L_\gamma)$, then clearly, there will be a negative counterexample when the input contains an element at least as large as $\min(L_\alpha - L_\gamma)$. On the other hand $\max(L_\gamma) <_{ll} \min(L_\alpha - L_\gamma)$ is not possible, as in this case $\max(L_\gamma)$ is a cut-off for L_α, and thus $f(\gamma) <_{ll} f(\alpha)$ (if L_α is infinite, then this clearly holds; otherwise, both L_γ and L_α are finite and thus $f(\gamma) \geq_{ll} f(\alpha)$ implies $\max(L_\gamma) \geq_{ll} \max(L_\alpha)$ and thus $\max(L_\gamma) \geq_{ll} \min(L_\alpha - L_\gamma)$). The claim follows.

The above claim motivates the below learning procedure to test indices based on $\sqsubset$ ordering. We need two further definitions. For three indices $\alpha, \gamma, \delta \in I$ and a word $v \in \Sigma^*$, let:

$$\gamma \equiv_\alpha \delta \text{ iff } L_\gamma - L_\alpha = L_\delta - L_\alpha;$$
$$I_{\alpha,v} = \{\gamma \in I : \emptyset \subset L_\gamma - L_\alpha \subseteq \{u : u \leq_{ll} v\} \wedge \forall \delta <_{ll} \gamma\, [\gamma \not\equiv_\alpha \delta]\}.$$

Note that $\equiv_\alpha$ is a first-order definable and hence automatic relation. Also, each set $I_{\alpha,v}$ is finite, as there are only finitely many different subsets of $\{u : u \leq_{ll} v\}$, and for each such subset E, only the least index γ, if any, with $L_\gamma - L_\alpha = E$ is in $I_{\alpha,v}$. Hence the following functions are automatic: $\alpha, v \mapsto \min(I_{\alpha,v})$, $\alpha, v \mapsto \max(I_{\alpha,v})$ and $\alpha, v, \gamma \mapsto \mathrm{succ}_{I_{\alpha,v}}(\gamma)$, where each of these functions returns $\#$ in the case that the corresponding index does not exist.

The memory of the learner is of the form $conv(\alpha, \beta, v, w)$, where α, β range over $I \cup \{\#\}$ and v, w range over $\Sigma^* \cup \{\#\}$. Furthermore, the hypothesis of the learner is directly linked to its memory: if $\beta = \#$, then the hypothesis is α else the hypothesis is β (where, if $\alpha = \beta = \#$, then we take the conjecture of the learner as ?). Thus, we only describe how the learner updates its memory.

Intuitively, the learner goes through the possible α in the order given by the ordering $\sqsubset$ and checks whether L_α is the input language to be learnt. The α would be updated only when it has been verified by the learner that the previous α was not an index for the target language. The parameter w denotes the length-lexicographically largest element seen as positive datum so far. The parameter v is used to keep track of the largest positive datum seen by the time the learner starts checking for L_α, as well as to make sure that, before formally starting to check whether L_α is the target language, the learner has seen a positive datum length-lexicographically at least as large as $f(\alpha)$ (see Case 1 below). The value $v = \#$ will indicate that the current α is not an index for the target language and the learner is just waiting to make sure that w has become length-lexicographically large enough before moving to the next α.

In order to address the fact that the learner may have forgotten some past data, the learner needs to check whether some element $x \leq_{ll} v$ satisfying $x \notin L_\alpha$ has earlier appeared in the input text. For this, the learner checks that L_β, $\beta \in I_{\alpha,v}$, are not contained in the target language. This check is needed only when L_α is infinite due to the special ordering, $\sqsubset$, used for going through the various candidates α. The value of $\beta = \#$ will denote that the learner has checked all the relevant β or that no checks were needed. The learner then settles on the hypothesis L_α and checks if there is a negative counterexample to L_α or a future positive example not in L_α.

Suppose T is the input text for target language $L \in \mathcal{L}$, and T' is the corresponding bounded counterexample text. We assume without loss of generality that $T(0) \neq \#$. This is for ease of writing the proof and is without loss of generality as we can easily delay the updating process until an element different from $\#$ is seen in the text (note that by our assumption $\emptyset \notin \mathcal{L}$). Let the memory of the learner after having seen the input $T[n], T'[n]$ be $conv(\alpha_n, \beta_n, v_n, w_n)$. Initially, $\alpha_0 = \beta_0 = \#$ and $v_0 = w_0 = \#$. Note that it is assumed that $\# <_{ll} z$ for all $z \in \Sigma^*$. Similarly, $\# \sqsubset \alpha$ for all $\alpha \in I$.

The following invariants will be satisfied by the learner for all n. Suppose $m \leq n$ is least such that $\alpha_m = \alpha_n$.

- (I1) $w_n = \max(\text{content}(T[n]))$.
- (I2) $\alpha_n \sqsubseteq \alpha_{n+1}$.
- (I3) For all $\alpha \in I$ such that $\alpha \sqsubset \alpha_n$, $L_\alpha \neq L$.
- (I4) If $v_n = \#$, then $L_{\alpha_n} \neq L$.
- (I5) If $v_n \neq \#$, then $v_n \geq f(\alpha_n)$ and $\text{content}(T[n]) - \{x : x \leq v_n\} \subseteq L_{\alpha_n}$. Furthermore, $v_n = w_m$, that is v_n denotes the length-lexicographically largest element seen when the learner started testing whether $L_{\alpha_n} = L$.
- (I6) If $v_n \neq \#$ and $\beta_n = \#$ and $m < n$, then either $\text{content}(T[n]) \subseteq L_{\alpha_n}$ or there exists a $w >_{ll} v_n$ such that $w \in L - L_{\alpha_n}$.
- (I7) If $v_n \neq \#$ and $\beta_n \neq \#$, then L_{α_n} is infinite, $\beta_n \in I_{\alpha_n, v_n}$ and for $\beta <_{ll} \beta_n$ such that $\beta \in I_{\alpha_n, v_n}$, $L_\beta \not\subseteq L$.
- (I8) If $v_n \neq \#$, $\beta_n = \#$ and L_{α_n} is infinite, then for all $\beta \in I_{\alpha_n, v_n}$, $L_\beta \not\subseteq L$. Furthermore, if $v_n \neq \#$, $\alpha_n = \alpha_{n+1}$, then $[[\beta_n = \#$ implies $\beta_{n+1} = \#]$ and $[\beta_{n+1} = \#$ or $\beta_n <_{ll} \beta_{n+1}]]$.
- (I9) There exists a $w \in L$ such that $w \geq f(\alpha_n)$.

The following algorithm gives how the learner updates its memory to $conv(\alpha_{n+1}, \beta_{n+1}, v_{n+1}, w_{n+1})$ from the input positive datum $T(n)$ and negative counterexample $T'(n)$ along with previous memory $conv(\alpha_n, \beta_n, v_n, w_n)$.

Update Algorithm for the learner
Input: positive datum $T(n)$, counterexample $T'(n)$ and previous memory $conv(\alpha_n, \beta_n, v_n, w_n)$
Output: New memory $conv(\alpha_{n+1}, \beta_{n+1}, v_{n+1}, w_{n+1})$
The algorithm takes the first of the following three cases which applies.

Case 1: $\alpha_n = \#$ or $T(n) \in \Sigma^* - L_{\alpha_n}$ or $w_n \in \Sigma^* - L_{\alpha_n}$ or $T'(n) \in L_{\alpha_n}$ or $v_n = \#$ or $\beta_n \neq \# \wedge T'(n) = \#$.

(* Note that in this case L_{α_n} is not the target language L. *)

Let γ be the next index after α_n in I with respect to the ordering $\sqsubset$.
Let $w_{n+1} = \max(\{w_n, T(n)\})$.
If $f(\gamma) \leq_{ll} w_{n+1}$ and L_γ is infinite, then $\alpha_{n+1} = \gamma$, $v_{n+1} = w_{n+1}$ and $\beta_{n+1} = \min(I_{\alpha_{n+1}, v_{n+1}})$.
If $f(\gamma) \leq_{ll} w_{n+1}$ and L_γ is finite, then $\alpha_{n+1} = \gamma$, $v_{n+1} = w_{n+1}$ and $\beta_{n+1} = \#$.
If $f(\alpha_{n+1}) >_{ll} w_{n+1}$, then $\alpha_{n+1} = \alpha_n$, $v_{n+1} = \#$ and $\beta_{n+1} = \#$.

Case 2: Not Case 1 and L_{α_n} is finite.
$\alpha_{n+1} = \alpha_n$, $\beta_{n+1} = \beta_n$, $v_{n+1} = v_n$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 3: Not Case 1 and L_{α_n} is infinite.
$\alpha_{n+1} = \alpha_n$, $v_{n+1} = v_n$ and $w_{n+1} = \max(\{w_n, T(n)\})$.
If $\beta_n \neq \#$ and $\beta_n <_{ll} \max(I_{\alpha_n, v_n})$,
Then $\beta_{n+1} = \mathrm{succ}_{I_{\alpha_n, v_n}}(\beta_n)$,
Else $\beta_{n+1} = \#$.

End Update Algorithm

We now argue that all the invariants are maintained by the above updates assuming that they were true before the update. Note that the invariants are satisfied in the beginning (when $n = 0$). Also, it is easy to verify that the update algorithm maintains the invariants (I1), (I2), (I5), (I9). We now argue that all the cases above maintain the remaining invariants.

First we consider Case 1. Note that only Case 1 makes $\alpha_{n+1} \neq \alpha_n$ and/or sets $v_{n+1} = \#$. Furthermore, Case 1 implies that α_n has been verified to be incorrect index for L. To see this note that, (i) $\alpha_n = \#$ means that α is an invalid initial index, (ii) $T(n) \in \Sigma^* - L_{\alpha_n}$ or $w_n \in \Sigma^* - L_{\alpha_n}$ means that an example outside L_{α_n} had been observed, (iii) $T'(n) \in L_{\alpha_n}$ means that a negative counterexample in L_{α_n} has been observed, (iv) $v_n = \#$ means that α_n has been already found to be incorrect (by invariant (I4) holding before the update), and (v) $\beta_n \neq \# \wedge T'(n) = \#$ means that L_{β_n} has been conjectured and no counterexample was received although all elements in $L_{\beta_n} - L_{\alpha_n}$ are bounded by v_n and thus by w_n; this implies that α_n is incorrect as $L_{\beta_n} - L_{\alpha_n} \neq \emptyset$. The choice of $v_{n+1} = \#$ in the third if-statement is to signal that α_n is incorrect and one has to wait for a larger datum to appear in the input to do the change from α_n to γ. Thus, (I3) is maintained. For (I4), note that $w_{n+1} \neq \#$ (as $T(0) \neq \#$ and thus $\max(\mathrm{content}(T[n+1])) \neq \#$) and thus (I4) is maintained by the update in Case 1. (I7) and (I8) are easily seen to be maintained by Case 1, and (I6) is trivially maintained.

For Case 2, note that in this case $T(n) \in L_{\alpha_n} \cup \{\#\}$ and $T'(n) = \#$ and $w_n = v_n = f(\alpha_n) = \max(L_{\alpha_n})$, as otherwise Case 1 would apply and Case 2 would not have been reached (here note that $w_n \geq v_n \geq f(\alpha_n) = \max(L_{\alpha_n})$ by (I1) and (I5), and $w_n \leq \max(L_{\alpha_n})$ by Case 1 not holding). Thus, (I3), (I4), (I7) and (I8) are easily seen to be maintained by the update in Case 2. For (I6) note that if the target language $L = L_\gamma$ with $\alpha_n \sqsubset \gamma$, then either $f(\gamma) > w_n = f(\alpha_n)$ and $f(\gamma) \in L - L_{\alpha_n}$ or $f(\gamma) = f(\alpha_n) = \max(L_{\alpha_n})$ and $L_\gamma <_{lex} L_{\alpha_n}$. In the later case, $L_{\alpha_n} \not\subseteq L_\gamma$ and $\max(L_{\alpha_n}) \leq_{ll} w_n$, and thus the learner would have received a negative counterexample and Case 1 would have applied. Thus (I6) is maintained by the update in Case 2.

For Case 3, the intuition behind this update rule is that in this case, the learner lets parameter β cycle through the hypotheses in I_{α_n,v_n} in order to verify that each language $L_\gamma \not\subseteq L_{\alpha_n}$ satisfying $\max(L_\gamma - L_{\alpha_n}) \leq_{ll} v_n$ contains some element outside the target language. The invariants (I3), (I4) and (I8) are easily seen to be maintained by the update in Case 3. For (I7) note that Case 1 not holding implies that $\beta_n = \#$ or $L_{\beta_n} \not\subseteq L$. Thus, (I7) is maintained by the update in Case 3. For invariant (I6) note that if $\beta_{n+1} = \#$, then by (I8) holding before the update or by (I7) holding before the update, $\beta_n = \max(I_{\alpha_n,v_n})$, and L_{β_n} receiving negative counterexample due to Case 1 not holding, that for all $\beta \in I_{\alpha_n,v_n}$, L_β is not contained in L. Thus by definition of I_{α_n,v_n} and $\equiv_{\alpha_n}$, $L - L_{\alpha_n}$ does not contain any element $\leq_{ll} v_n$ or it contains some element $>_{ll} v_n$. It thus follows using (I5) that (I6) is maintained by the update in Case 3.

Thus, all the invariants are maintained by the algorithm.

Now we show that the learner **AutoWordBNCEx**-identifies L. Suppose δ is the least index for L with respect to the ordering $\sqsubset$. By (I2) and (I3), $\lim_{n\to\infty} \alpha_n$ converges to say $\alpha \sqsubseteq \delta$. Thus, $\lim_{n\to\infty} v_n$ also converges to some v, as it converges to either $\#$ or w_m, where m is least such that $\alpha_m = \alpha$. If v converges to $\#$, then for γ as in Case 1, eventually the input text would contain an element $\geq f(\gamma)$, (as $f(\gamma) \leq f(\delta)$) and thus α would be updated to γ. So suppose v converges to w_m. If L_α is infinite, then by (I7) and (I8) and finiteness of $I_{\alpha,v}$, we have that $\lim_{n\to\infty} \beta_n$ converges to $\#$. For L_α being finite, trivially, $\lim_{n\to\infty} \beta_n$ converges to $\#$ by (I8) and the second If statement of Case 1 when $\alpha_m = \alpha$ was first made. Thus, by (I5) and (I6) we have that $L \subseteq L_\alpha$. $L_\alpha \subseteq L_\delta = L$ follows from $\alpha \sqsubseteq \delta$ and Claim 19, and the learner not receiving a negative counterexample on hypothesis α (otherwise, Case 1 in the update procedure would have eventually updated α). It follows that the learner **AutoWordBNCEx**-identifies L. ∎

As a corollary to Theorem 18 we have the following result.

Corollary 20. *If $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic class consisting only of infinite languages, then the class $\mathcal{L}$ is* **AutoWordBNCEx***-learnable. The learner uses the hypothesis space $(H_\alpha)_{\alpha\in I}$, where $H_\alpha = L_\alpha$.*

The next result shows that automatic classes which consist only of infinite languages can be **AutoBNCIt**-learnt where the learner uses a class-comprising hypothesis space.

Theorem 21. *If $\mathcal{L}$ is an automatic class consisting only of infinite languages, then the class $\mathcal{L}$ is* **AutoBNCIt***-learnable using a class comprising hypothesis space.*

Proof. Let $\mathcal{L}$ be given by the indexing $\{L_\alpha : \alpha \in I\}$. Now one considers a larger hypothesis space with indices $J = \{conv(\alpha, u, v, w) : \alpha \in I, u, v, w \in \Sigma^* \cup \{\#\}\}$. The hypothesis space is not necessarily 1–1.

- If $u \notin L_\alpha \cup \{\#\}$, then $H_{conv(\alpha,u,v,w)} = L_\alpha \cup \{u\}$;
- If $u = \#$ and $v = \#$, then $H_{conv(\alpha,u,v,w)} = L_\alpha \cup \{z \in \Sigma^* : |z| > |w|\}$;
- If none of the above cases applies, then $H_{conv(\alpha,u,v,w)} = L_\alpha$.

The memory of the learner is always of the form $conv(\alpha, u, v, w)$ (correspondingly, the hypothesis of the learner is $H_{conv(\alpha,u,v,w)}$ as expected by an iterative learner).

The basic idea is the following. The learner goes through all α in length-lexicographic order, discarding them only when they are found to be incorrect. It first conjectures some variants of L_α in order to establish that α seems like a valid hypothesis not contradicting the observed data in an easy way. Once this is done, the learner will then stabilise on the hypothesis α until a contradiction arises by either getting a negative counterexample to L_α or observing a positive datum outside L_α. During the whole "α-cycle", whenever the learner observes a positive datum outside L_α or a counterexample inside L_α, it directly updates to the next index in I and resets the corresponding parameters u, v, w.

Now we describe how the learner, at the beginning of an α-cycle, checks whether the hypothesis α is consistent with the past data. Note that if L_α is not contained in the target language, then eventually the learner will get a counterexample to L_α. So the main problem is to check if the past data is contained in L_α. When starting an α-cycle, $u = \#$, $v = \#$ and w may be any string from $\Sigma^* \cup \{\#\}$. Ideally, the role of w is to be the largest positive example seen so far. However, this is not always possible as the learner is iterative and thus is not able to store the largest datum seen so far. The variable v is used to obtain a suitable w which serves the purpose. Intuitively, v will be set so that, in case L_α is contained in the target language but the positive data seen so far contains an element outside L_α, v bounds at least one such element. The role of w is then to make sure that some positive data seen so far bounds v. Here, it will be the case that w is at most the length-lexicographically largest datum seen so far (where by convention, we take that $\# <_{ll} z$ for all $z \in \Sigma^*$). Afterwards, u will subcycle through all elements of $\{z \in \Sigma^* : z \leq_{ll} v \wedge z \notin L_\alpha\}$ and it will be checked if any of these u is in the target language to be learnt. Once it is verified that none of these are in the target language to be learnt, the hypothesis will then be L_α and remains at it without updating any parameters until either an element outside L_α is observed or a negative counterexample inside L_α is observed.

The above is the general outline of the algorithm. Now it is shown how the memory of the learner is updated. Suppose T is the input text for a target language L, and T' is the corresponding bounded counterexample text. Suppose the memory of the learner after having seen the input $T[n], T'[n]$ is $mem_n = conv(\alpha_n, u_n, v_n, w_n)$. Note that the hypothesis of the learner after seeing the input $T[n], T'[n]$, is $hyp_n = mem_n$ as the learner is iterative. Now, we describe how the learner updates its memory to $mem_{n+1} = conv(\alpha_{n+1}, u_{n+1}, v_{n+1}, w_{n+1})$, after receiving datum $T(n)$ and counterexample $T'(n)$ (along with previous memory $mem_n = conv(\alpha_n, u_n, v_n, w_n)$). The analysis below assumes that the input text/counterexample text is for a language in the class $\mathcal{L}$.

Initially, the memory of the learner has α_0 as the $<_{ll}$-least element from I and $u_0 = v_0 = w_0 = \#$.

Intuitively, in Case 1 below, input does not give any useful information. In Case 2 below, α_n is not a correct index for the target language. In Cases 3 and 4 we consider the α-cycle for L_{α_n} being cofinite or coinfinite respectively.

Update Algorithm of the learner
Input: positive datum $T(n)$, counterexample $T'(n)$, and previous memory $conv(\alpha_n, u_n, v_n, w_n)$.
Output: new memory $conv(\alpha_{n+1}, u_{n+1}, v_{n+1}, w_{n+1})$.

It is always assumed that the learner takes the first of the cases below which applies.

Case 1: $T(n) = \#$ and $T'(n) = \#$.
$(\alpha_{n+1}, u_{n+1}, v_{n+1}, w_{n+1}) = (\alpha_n, u_n, v_n, w_n)$.

Case 2: $T(n) \in \Sigma^* - L_{\alpha_n}$ or $T'(n) \in L_{\alpha_n}$.
(* In this case L_{α_n} is not the target language. *)
$\alpha_{n+1} = \mathrm{succ}_I(\alpha_n)$, $u_{n+1} = \#$, $v_{n+1} = \#$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 3: Not (Case 1 or Case 2) and L_{α_n} is cofinite.

Case 3.1: $v_n = \#$.
(* In this case we choose an appropriate value of v for the next iteration. *)
$v_{n+1} = \min(\{z \in \Sigma^* : \forall z' \geq_{ll} z\, [z' \in L_{\alpha_n}]\})$.
$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \#$ and $w_{n+1} = \max(\{w_n, T(n)\})$.
(* Note that v_{n+1} above strictly bounds any datum not in L_{α_n} which may have been observed earlier by the learner. *)

Case 3.2: $v_n \neq \#$ and $|w_n| \leq |v_n|$.
(* In this case, the learner waits until w_n is longer than v_n (this eventually happens for any target language from the class $\mathcal{L}$). *)
$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \#$, $v_{n+1} = v_n$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 3.3: $v_n \neq \#$ and $|v_n| < |w_n|$ and $[u_n = \#$ or $(u_n \in \Sigma^* - L_{\alpha_n}$ and $T'(n) = u_n)]$.
(* In this case, the learner has verified that u_n is not in the target language and not in L_{α_n}. The learner will now consider the next candidate u_{n+1}. *)
$\alpha_{n+1} = \alpha_n$,
$u_{n+1} = \min(\{v_n\} \cup \{z \in \Sigma^* - L_{\alpha_n} : z >_{ll} u\})$,
$v_{n+1} = v_n$, and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 3.4: $v_n \neq \#$ and $|v_n| < |w_n|$ and $u_n \in \Sigma^* - L_{\alpha_n}$ and $T'(n) = \#$.
(* In this case, the learner has verified that u_n is in the target language although not in L_{α_n}, and hence α_n is abandoned and next index is tried. *)
$\alpha_{n+1} = \mathrm{succ}_I(\alpha_n)$, $u_{n+1} = \#$, $v_{n+1} = \#$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 3.5: $v_n \neq \#$ and $|v_n| < |w_n|$ and $u_n = v_n$.
$(\alpha_{n+1}, u_{n+1}, v_{n+1}, w_{n+1}) = (\alpha_n, u_n, v_n, w_n)$.
(* Note that $H_{conv(\alpha_n,u,v,w)} = L_{\alpha_n}$ and this hypothesis is kept unchanged until it is verified to be wrong. *)

Case 4: Not (Case 1 or Case 2) and L_{α_n} is coinfinite.

Case 4.1: $v_n = \#$ and $T'(n) \in H_{conv(\alpha_n,u_n,v_n,w_n)} - L_{\alpha_n}$.
(* Note that in this case, $u_n = v_n = \#$ and $H_{conv(\alpha_n,u_n,v_n,w_n)} = L_{\alpha_n} \cup \{z \in \Sigma^* : |z| > |w_n|\}$. Furthermore, in this case w_n is too small and thus the learner keeps updating it until no negative counterexample to $H_{conv(\alpha_n,u_n,v_n,w_n)} - L_{\alpha_n}$ is permitted by the largest datum seen so far; this eventually happens, as it happens when the corresponding input element $T(\cdot)$ is length-lexicographically larger than all the earlier elements in the text. *)
$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \#$, $v_{n+1} = \#$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 4.2: $v_n = \#$ and $T'(n) = \#$.

(* That no counterexample has been observed means that either v_{n+1} computed below (which is the least element, greater in length than w_n, in $\Sigma^* - L_{\alpha_n}$) is a strict upper bound on the largest positive example seen so far, or v_{n+1} is in the target language; in the second case it means that the target language is not a subset of L_{α_n}. The learner will find out later which case holds. *)

$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \#$,
$v_{n+1} = \min(\{z \in \Sigma^* - L_{\alpha_n} : |w_n| < |z|\})$ and
$w_{n+1} = \max(\{w_n, T(n)\})$.

Case 4.3: $v_n \neq \#$ and $|w_n| \leq |v_n|$.

(* In this case, the learner waits until w_n is longer than v_n; this eventually happens for any target language from the class $\mathcal{L}$. *)

$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \#$, $v_{n+1} = v_n$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 4.4: $v_n \neq \#$ and $|w_n| > |v_n|$ and $u_n <_{ll} v_n$ and $[u_n = \#$ or $(u_n \in \Sigma^* - L_{\alpha_n}$ and $T'(n) = u_n)]$.

(* The learner has verified that u_n is not in the target language and now looks at the next candidate u_{n+1}. *)

$\alpha_{n+1} = \alpha_n$,
$u_{n+1} = \min(\{z \in \Sigma^* - L_{\alpha_n} : z >_{ll} u_n\})$,
$v_{n+1} = v_n$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 4.5: $v_n \neq \#$ and $|w_n| > |v_n|$ and $u_n \leq_{ll} v_n$ and $u_n \in \Sigma^* - L_{\alpha_n}$ and $T'(n) = \#$.

(* In this case, the learner has verified that u_n is in the target language although not in L_{α_n} and hence α_n is abandoned as an index. *)

$\alpha_{n+1} = \mathrm{succ}_I(\alpha_n)$, $u_{n+1} = \#$, $v_{n+1} = \#$ and $w_{n+1} = \max(\{w_n, T(n)\})$.

Case 4.6: $v_n \neq \#$ and $|w_n| > |v_n|$ and $u_n = v_n$ and $T'(n) = u_n$.

(* The learner has verified that below v_n, all non-elements of L_{α_n} are not in the target language. Furthermore, as v_n is not in the target language, the statement that $T'(n) = \#$ when v_n was defined was due to the fact that at that time the bound of the observed data was below v_n. Hence the learner is now ready to stabilise on the conjecture L_{α_n} until a non-member of L_{α_n} occurs as a positive datum or a member of L_{α_n} occurs as a negative counterexample. *)

$\alpha_{n+1} = \alpha_n$, $u_{n+1} = \min(L_{\alpha_n})$, $v_{n+1} = v_n$ and $w_{n+1} = \max(\{v_n, w_n\})$.

Case 4.7: $u_n \in L_{\alpha_n}$.

$(\alpha_{n+1}, u_{n+1}, v_{n+1}, w_{n+1}) = (\alpha_n, u_n, v_n, w_n)$.

(* Now, $H_{conv(\alpha_n,u_n,v_n,w_n)} = L_{\alpha_n}$ and this hypothesis is kept unchanged until it is verified to be wrong. *)

End Update Algorithm

It is easy to see that the above learner is automatic. Case 3 (when L_{α_n} is cofinite) and Case 4 (when L_{α_n} is coinfinite) go through the α-cycle to check whether L_{α_n} is the target language.

In Case 3.1 (when L_{α_n} is cofinite) and Cases 4.1–4.2 (when L_{α_n} is coinfinite), the learner selects an appropriate v_n for the next iteration such that, if there are any positive data seen so far in $\Sigma^* - L_{\alpha_n}$, then v_n bounds at least one element in $L - L_{\alpha_n}$. In Cases 3.2 and 4.3, the learner waits for w_n to be larger than v_n. In Cases 3.3–3.4, and Cases 4.4–4.6, the learner subcycles through different values of u_n to check whether any non-element of L_{α_n} below v_n has appeared in the positive data. In cases 3.5 and 4.7, the learner has established that past data is consistent with L_{α_n} and stabilises on L_{α_n} until it receives a positive example or a negative counterexample witnessing that the target language is not L_{α_n}.

Thus $\lim_{n\to\infty} \alpha_n$ converges to the least index α for L (that is, $L_\alpha = L$). Furthermore, $\lim_{n\to\infty} v_n$ also converges to v different from $\#$ (as it gets defined via Case 3.1 or Case 4.2 based on L_α being cofinite or coinfinite). Also, $\lim_{n\to\infty} w_n$ also converges to w as $T(n) \geq_{ll} v_n$ for infinitely many n. Thus, via Cases 3.3–3.4 or Cases 4.4–4.6, eventually the learner will verify that all $u \in (\Sigma^* - L_\alpha) \cap \{x : x \leq_{ll} v\}$ do not belong to L, and thus $\lim_{n\to\infty} u_n$ also converges to some $u \in L_\alpha$ as defined by Case 3.3 (when it sets $u_{n+1} = v_n \in L_{\alpha_n}$) or Case 4.6 (when it sets $u_{n+1} = \min(L_{\alpha_n})$). Thus, $H_{conv(\alpha,u,v,w)} = L_\alpha = L$, and the learner converges to an index for L. ∎

Proposition 22. *There exists an automatic class $\mathcal{L}$ consisting only of infinite languages, such that $\mathcal{L}$ is not* **AutoBNCIt***-learnable using any class preserving hypothesis space.*

Proof. Consider the automatic class $\mathcal{L}$ defined as follows. Let $I = \{\epsilon\} \cup \{1^i : i > 0\} \cup \{1^i0^j : i, j > 0\}$. Let $\Sigma = \{0,1\}$. Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where

$$\begin{aligned} L_\epsilon &= \{0,1\}^*; \\ L_{1^i} &= \{x \in \{0,1\}^* : x \leq_{lex} 1^i\}; \\ L_{1^i0^j} &= \{x \in \{0,1\}^* : x \leq_{lex} 1^i\} \cup \{1^i0^j\}. \end{aligned}$$

Now suppose by way of contradiction that an automatic learner $\mathbf{M}$ **AutoBNCIt**-learns $\mathcal{L}$ using a class preserving hypothesis space $(H_\beta)_{\beta\in J}$. Then, there exists a locking sequence $(\sigma, \#^{|\sigma|})$ for $\mathbf{M}$ on L_ϵ, that is $\mathbf{M}(\tau, \#^{|\tau|}) = (mem, \beta)$ for some memory mem, with $H_\beta = L_\epsilon$, for all τ extending σ (note that there exists such a locking-sequence, see Proposition 12). Let m be the length of the longest element in content(σ). Then, let $(\sigma\diamond(0^{m+3})\diamond\#\diamond\sigma', \#^{|\sigma|+1}\diamond(1^{m+1}0)\diamond\tau')$ be a locking sequence for $\mathbf{M}$ for learning $L_{1^{m+1}}$, where the counterexamples given are the least bounded negative counterexamples, if any (note that there exists such a locking-sequence; this can be proven in a way similar to the proof of Proposition 12).

Let $j > 1$ be such that $H_{\mathbf{M}(\sigma'',\tau'')} \neq L_{1^{m+1}0^j}$, for any initial segment σ'' of $\sigma\diamond(0^{m+3})\diamond\#\diamond\sigma'$ and corresponding initial segment τ'' of $\#^{|\sigma|+1}\diamond(1^{m+1}0)\diamond\tau'$, with $|\tau''| = |\sigma''|$.

Let T be a text for $L_{1^{m+1}}$. Now, the behaviour of $\mathbf{M}$ on input $(\sigma\diamond(0^{m+3})\diamond\#\diamond\sigma'\diamond T, \#^{|\sigma|+1}\diamond(1^{m+1}0)\diamond\tau'\diamond\#^\infty)$ is the same as its behaviour on $(\sigma\diamond(1^{m+1}0^j)\diamond\#\diamond\sigma'\diamond T, \#^{|\sigma|+1}\diamond(1^{m+1}0)\diamond\tau'\diamond\#^\infty)$, as $\mathbf{M}$ is iterative. However, these two inputs correspond to the languages $L_{1^{m+1}}$ and $L_{1^{m+1}0^j}$, respectively, where the counterexamples given are the least bounded negative counterexamples. Thus, $\mathbf{M}$ fails to **BNCIt**-learn $\mathcal{L}$ using a class preserving hypothesis space. ∎

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

Proposition 23. **(a)** **AutoNCIt** $\not\subseteq$ **AutoBNCIt**.
(b) **AutoHypNCEx** $\not\subseteq$ **AutoHypBNCEx**.
(c) **AutoWordNCEx** $\not\subseteq$ **AutoWordBNCEx**.
(d) **AutoNCEx** $\not\subseteq$ **AutoBNCEx**.

Proof. Let $L_\epsilon = \{0\}^*$ and, for $i > 0$, let $L_{0^i} = \{0^j : j \leq i\}$. Then, $\mathcal{L} = \{L_\alpha : \alpha \in \{0\}^*\}$ is an automatic family which is **AutoNCIt**-learnable by Theorem 14. For learning $\mathcal{L}$, bounded negative counterexamples are always #. Thus, $\mathcal{L}$ is **BNCEx**-learnable iff $\mathcal{L}$ is **TxtEx**-learnable. As $\mathcal{L}$ is not **TxtEx**-learnable [Gol67], it follows that $\mathcal{L}$ is not **BNCEx**-learnable. The proposition follows. ∎

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

Theorem 24. *There exists an automatic class* $\mathcal{L} = \{L_\alpha : \alpha \in I\}$*, such that*

(a) $\mathcal{L} \notin$ **HypBNCEx**;
(b) $\mathcal{L} \in$ **AutoWordTxtEx**.

Proof. Let $\Sigma = \{0\}$. Let $L_0 = \{0^{2n} : n \geq 0\}$. Let $L_{1^{2i}} = \{0^{2n} : n \leq i\} \cup \{0^{2i+1}\}$. Let $L_{conv(2^{2i},3^{2j})} = L_{1^{2i}} \cup \{0^{2j}\}$. Let $\mathcal{L} = \{L_\alpha : \alpha \in I\}$, where $I = \{0, 1^{2i}, conv(2^{2i}, 3^{2j}) : i, j \in \mathbb{N}\}$.

(a) Suppose by way of contradiction that **M** is a hypothesis-size memory bounded learner which **BNCEx**-identifies $\mathcal{L}$. Without loss of generality assume that **M** never outputs ?. Suppose the alphabet used for memory is Γ.

Let (σ, τ) be as given by Proposition 12 for **M** **BNCEx**-learning L_0.

Suppose i is such that, content$(\sigma) \subseteq \{0^{2j} : j \leq i\}$ and $i > |\max(\text{content}(\tau))|$. Without loss of generality assume that content$(\sigma) = \{0^{2j} : j \leq i\}$.

Let γ be such that $\mathbf{M}(\sigma\diamond 0^{2i+2}\diamond\gamma, \tau\diamond\#\diamond\#^{|\gamma|}) = \mathbf{M}(\sigma, \tau)$ (note that there exists such a γ by Proposition 12).

Note that (i) $\mathbf{M}(\sigma\diamond 0^{2i+2}\diamond\gamma, \tau\diamond\#\diamond\#^{|\gamma|}) = \mathbf{M}(\sigma, \tau)$, where the corresponding hypothesis is an index for L_0, and (ii) the counterexamples (if any) provided by τ on the input σ, for the language $L_{1^{2i}}$, and by $\tau\diamond\#\diamond\#^{|\gamma|}$ on the input $\sigma\diamond 0^{2i+2}\diamond\gamma$, for the language $L_{conv(2^{2i},3^{2i+2})}$, are correct as per the definition of **BNCEx**-learning (note that $L_0 \cap \{x : x \leq_{ll} \max(\text{content}(\sigma))\} = L_{1^{2i}} \cap \{x : x \leq_{ll} \max(\text{content}(\sigma))\} = L_{conv(2^{2i},3^{2i+2})} \cap \{x : x \leq_{ll} \max(\text{content}(\sigma))\} = \text{content}(\sigma)$) and (iii) $0^{2i+2} >_{ll} 0^{2i+1} >_{ll} \max(\text{content}(\sigma))$, and thus $0^{2i+1}, 0^{2i+2}$ are not counterexamples on either of the texts $T_1 = \sigma\diamond 0^{2i+1}\diamond\#^\infty$ and $T_2 = \sigma\diamond 0^{2i+2}\diamond\gamma\diamond 0^{2i+1}\diamond\#^\infty$. Therefore, the behaviour of **M** (hypotheses conjectured and counterexamples received) on the text $\sigma\diamond 0^{2i+1}\diamond\#^\infty$ beyond σ (where counterexamples provided are the least bounded counterexample, if any for the language $L_{1^{2i}}$), is the same as its behaviour (hypotheses conjectured and counterexamples received) on the text $\sigma\diamond 0^{2i+2}\diamond\gamma\diamond 0^{2i+1}\diamond\#^\infty$ beyond $\sigma\diamond 0^{2i+2}\diamond\gamma$, (where counterexamples provided are the least bounded counterexample, if any for the language $L_{conv(2^{2i},3^{2i+2})}$). Thus, $\mathbf{M}$ fails to learn at least one of the sets $L_{1^{2i}}$ and $L_{conv(2^{2i},3^{2i+2})}$.

(b) This can be easily done by memorizing the $\leq_{ll}$-largest word of L_0 seen in the input. Initially, the conjecture of the learner is L_0. If and when it receives 0^{2i+1} in the input, for some i, it continues as follows: at any future point, it conjectures $L_{conv(2^{2i},3^{2k})}$, where k is largest such that 0^{2k} is seen in the input upto that point. ∎

The next theorem presents some conditions under which iterative automatic learners can learn from negative counterexamples alone.

Theorem 25. *Suppose $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ is an automatic family, where the languages in $\mathcal{L}$ are pairwise incomparable with respect to $\subset$. Then $\mathcal{L}$ is* **AutoNCIt**-*learnable and, in the case that all $L \in \mathcal{L}$ are infinite, also* **AutoBNCIt**-*learnable. For the learning algorithm, the learner does not need the positive examples, except that they are used implicitly to permit the counterexamples in the case of* **AutoBNCIt**-*learning.*

Proof. The learner starts conjecturing $\min(I)$ and whenever the learner receives a negative counterexample for the current conjecture α, the learner updates its conjecture from α to $\mathrm{succ}_I(\alpha)$. In the case of **AutoNCIt**-learning, the counterexamples come immediately, in the case of **AutoBNCIt**-learning, we note that the languages are infinite and the negative counterexamples, if any, will be presented once a larger positive datum had been observed. ∎

Strong-monotonic learners [Jan91] only conjecture subsets of the target language to be learnt; hence for strong-monotonic learning, negative counterexamples are irrelevant. Our last result shows, furthermore, that monotonic (even non-automatic) learners are not able to learn all automatic classes, even if they use least counterexamples.

Theorem 26. *There exists an automatic class $\mathcal{L} = \{L_\alpha : \alpha \in I\}$ such that $\mathcal{L} \in$* **AutoTxtIt** *and $\mathcal{L} \notin$* **MonLNCEx**.

Proof. Let $\Sigma = \{0,1\}$. Let $L_0 = \{0\}^*$, $L_{1^j} = \{0^i : i \leq j\} \cup \{1^j\}$ and $L_{1^j2^{k-j}} = \{0^i : i \leq j \vee i = k\} \cup \{1^j, 1^k\}$. Let $I = \{0, 1^j, 1^j2^{k-j} : j,k \in \mathbb{N}, j < k\}$ and $\mathcal{L} = \{L_\alpha : \alpha \in I\}$.

Clearly, $\mathcal{L}$ is an automatic family. $\mathcal{L}$ is also **AutoTxtIt**-learnable by a learner with initial conjecture L_0. Once it sees the first string of the form 1^j then the learner makes a mind change to L_{1^j}. When it sees the second string 1^k with $k \neq j$ then it makes a second mind change either to $L_{1^j2^{k-j}}$ or $L_{1^k2^{j-k}}$, depending on whether j or k is smaller. It is easy to see that this learner **AutoTxtIt**-learns $\mathcal{L}$.

Suppose, by way of contradiction, that $\mathbf{M}$ is a monotonic learner which **LNCEx**-learns $\mathcal{L}$ using some hypothesis space $(H_\alpha)_{\alpha \in J}$. Consider the shortest σ such that $\mathrm{content}(\sigma) \subseteq L_0$ and $H_{\mathbf{M}(\sigma,\tau)} \cap L_0$ is infinite, where τ provides least counterexamples with respect to learning L_0. 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 $j = 1 + \max(|x| : x \in \mathrm{content}(\sigma) \cup \mathrm{content}(\tau) \cup \bigcup_{s<|\sigma|}(H_{\mathbf{M}(\sigma[s],\tau[s])} \cap L_0))$. Note that j is finite as $H_{\mathbf{M}(\sigma[s],\tau[s])} \cap L_0$ is finite for $s < |\sigma|$. Let T' be the least-counterexample text for $\mathbf{M}$ on the text T for the language L_{1^j}. Then $\mathbf{M}(T,T')$ must converge to an index g for L_{1^j}. Thus,

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

Table 2. Summary of results on when all classes of particular type are learnable.

content(T') is finite. Choose k so large that $0^k \in H_{\mathbf{M}(\sigma,\tau)}$ and $0^k, 1^k \notin (\text{content}(T) \cup \text{content}(T'))$. Let m be such that content(T) = content($T[m]$), content(T') = content($T'[m]$), $\sigma \subseteq T[m]$, and $\mathbf{M}(T[m], T'[m]) = g$.

Now $\mathbf{M}$ is not a monotonic learner for $L_{1^j 2^{k-j}}$ as $0^k \in H_{\mathbf{M}(\sigma,\tau)} \cap L_{1^j 2^{k-j}} - H_{\mathbf{M}(T[m],T'[m])}$, even though $\sigma \subseteq T[m]$ and content($T[m]$) $\subseteq L_{1^j 2^{k-j}}$. ∎

5 Conclusions

In this paper we considered learning automatic families by automatic learners which receive negative counterexamples. For these learners, we studied various versions of memory restriction and conditions on the choice of the counterexamples, in particular with respect to the question whether these restrictions hinder learnability in some cases and how severe this hindrance is.

We showed that *all* automatic families can be automatically **NCIt** and **NCEx**-learnt. Furthermore if the automatic class consists *only of infinite* languages, then it can be automatically **BNCIt** and **BNCEx**-learnt. Additionally, *every* automatic family can be automatically and *consistently* **LNCEx**-learnt, though we do not yet know whether every automatic family can be automatically and consistently **NCEx**-learnt. Table 2 gives a summary of results regarding learning all classes of a particular type for various criteria of automatic learning with negative counterexamples.

We showed that there is an automatic class which is **AutoWordBNCEx**-learnable but not **AutoHypBNCEx**-learnable. Also, we showed that **AutoWordBNCEx** contains all automatic families except those where at least one language in the class has infinitely many cut-offs in the class. It is clear that such a class is also not **AutoHypBNCEx**-learnable, hence **AutoHypBNCEx** $\subset$ **AutoWordBNCEx**. Furthermore, we have shown that automatic learning from bounded negative counterexamples is compatible with wordsize memory: **AutoBNCEx** = **AutoWordBNCEx**. Note that the corresponding statements for **AutoTxtEx** learning (without using negative counterexamples) are still open [JLS12]. Negative counterexamples give **AutoWordBNCEx**-learners the ability to overcome the learner's inability to recover the lost memory. This is an important difference between **AutoWordBNCEx** and **AutoWordTxtEx**-

learning. Regarding monotonic learning, we showed that there are automatic families which cannot be **LNCEx**-learnt by any monotonic (even non-automatic) learners.

Acknowledgements. A preliminary version of this paper appeared in ALT 2012 [JK12]. We thank the anonymous referees of both the conference and the journal versions of this paper for several helpful comments.

References

[Ang80a] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[Ang80b] Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

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

[AMSKMS93] Setsuo Arikawa, Satoru Miyano, Ayumi Shinohara, Satoru Kuhara, Yasuhito Mukouchi, Takeshi Shinohara. A machine discovery from amino acid sequences by decision trees over regular patterns. *New Generation Computing* 11(3):361-375, 1993.

[AMSSY92] Setsuo Arikawa, Satoru Miyano, Ayumi Shinohara, Takeshi Shinohara and Akihiro Yamamoto. Algorithmic learning theory with elementary formal systems. *IEICE Transactions on Information and Systems*, E75-D:405-414, 1992.

[ASMS91] Setsuo Arikawa, Takeshi Shinohara, Satoru Miyano and Ayumi Shinohara. More about learning elementary formal systems. *Nonmonotonic and Inductive Logic*, second international workshop, volume 659 of *Lecture Notes in Artificial Intelligence*, pages 107–117. Springer Verlag, 1991.

[ASY92] Setsuo Arikawa, Takeshi Shinohara and Akihiro Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.

[Bār74] Janis 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.

[BFK16] Johanna Björklund, Henning Fernau and Anna Kasprzik. Polynomial inference of universal automata from membership and equivalence queries. *Information and Computation*, 246:3–19, 2016.

[BB75] Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28(2):125–155, 1975.

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

[BBP12] Vasco Brattka, Matthew de Brecht and Arno Pauly. Closed choice and a Uniform Low Basis Theorem. *Annals of Pure and Applied Logic*, 163(8):986–1008, 2012.

[BGH15] Vasco Brattka, Guido Gherardi and Rupert Hölzl. Probabilistic Computability and Choice. *Information and Computation*, 242:249–286, 2015.

[CJSS13] John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.

[CK09] John Case and Timo Kötzing. Difficulties in forcing fairness of polynomial time inductive inference. *Algorithmic Learning Theory*, Twentieth International Conference, ALT 2009, volume 5809 of *Lecture Notes in Computer Science*, pages 263–277. Springer Verlag, 2009.

[CL82] John Case and Christopher Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 107–115. Springer Verlag, 1982.

[CM09] John Case and Samuel E. Moelius. Independence results for n-ary recursion theorems. *Fundamentals of Computation Theory*, Seventeenth International Symposium, FCT 2009. Proceedings, volume 5699 of *Lecture Notes in Computer Science*, pages 38–49. Springer Verlag, 2009.

[Cla10] Alexander Clark. Towards General Algorithms for Grammatical Inference. *Algorithmic Learning Theory*, Twentyfirst International Conference, ALT 2010, volume 6331 of *Lecture Notes in Computer Science*, pages 11–30. Springer Verlag, 2010.

[CY13] Alexander Clark and Ryo Yohshinaka. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96(1):5–31, 2013.

[Fer02] Henning Fernau. Even linear simple matrix languages: formal language properties and grammatical inference. *Theoretical Computer Science*, 289(1):425–456, 2002.

[FKS95] Rusins Freivalds, Efim Kinber and Carl H. Smith. On the Impact of Forgetting on Learning Machines. *Journal of the ACM*, 42(6):1146–1168, 1995.

[GPS92] William I. Gasarch, Mark G. Pleszkoch and Robert Solovay. Learning via queries in $[+, <]$. *The Journal of Symbolic Logic*, 57(1):53–81, 1992.

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

[GJTV11] Sumit Gulwani, Susmit Jha, Ashish Tiwari and Ramarathnam Venkatesan. Synthesis of loop-free programs. *Proceedings of the Thirtysecond ACM SIGPLAN conference on Programming language design and implementation*, PLDP 2011, pages 62-73, 2011.

[HA02] Susumu Hayashi and Yohji Akama. Limit-computable mathematics and its applications. Computer Science Logic, Sixteenth International Workshop, CSL 2002, Eleventh Annual Conference of the EACSL, 2002, volume 2471 of *Lecture Notes in Computer Science*, page 1, Springer Verlag, 2002.

[HN00] Susumu Hayashi and Masahiro Nakata. Towards limit computable mathematics. *Types for Proofs and Programs*, International Workshop, TYPES 2000, Selected Papers, volume 2277 of *Lecture Notes in Computer Science*, pages 125–144. Springer Verlag, 2000.

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

[IJ91] Oscar H. Ibarra and Tao Jiang. Learning regular languages from counterexamples. *Journal of Computer and System Sciences*, 43(2):299–316, 1991.

[JK07] Sanjay Jain and Efim Kinber. Iterative Learning from Positive Data and Negative Counterexamples. *Information and Computation*, 205(12):1777–1805, 2007.

[JK08] Sanjay Jain and Efim 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.

[JK12] Sanjay Jain and Efim Kinber. Automatic learning from positive data and negative counterexamples. In Nader Bshouty, Gilles Stoltz, Nicolas Vayatis and Thomas Zeugmann, editors, *Algorithmic Learning Theory, 23nd International Conference, ALT 2012*, pages 66–80. Lecture Notes in Artificial Intelligence 7568. Springer Verlag, 2012.

[JLS12] Sanjay Jain, Qinglong Luo and Frank Stephan. Learnability of automatic classes. *Journal of Computer and System Sciences*, 78:1910–1927, 2012.

[JOPS11] Sanjay Jain, Yuh Shin Ong, Shi Pu and Frank 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.

[Jan91] Klaus P. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.

[JS15] Susmit Jha and Sanjit A. Seshia. *A Theory of Formal Synthesis via Inductive Learning.* Technical report on http://arxiv.org/abs/1505.03953.

[KS95] Efim Kinber and Frank Stephan. Language learning from texts: mind changes, limited memory and monotonicity. *Information and Computation*, 123:224–241, 1995.

[KN95] Bakhadyr Khoussainov and Anil 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 Verlag, 1995.

[LN05] Steffen Lange and Jochen Nessel. Learning erasing pattern languages with queries. Theoretical Computer Science 348(1):41–57, 2005.

[LNW98] Steffen Lange, Jochen Nessel and Rolf Wiehagen. Learning recursive languages from good examples. *Annals of Mathematics and Artificial Intelligence*, 23:27–52, 1998.

[LZ93] Steffen Lange and Thomas 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.

[LZ05] Steffen Lange and Sandra Zilles. Relations between Gold-style learning and query learning. *Information and Computation*, 203(2):211–237, 2005.

[Moe12] Samuel E. Moelius. Characteristics of minimal effective programming systems. How the World Computes - Turing Centenary Conference and Eighth Conference on Computability in Europe, CiE 2012, volume 7318 of *Lecture Notes in Computer Science*, pages 507–512. Springer Verlag, 2012.

[Pit89] Lenny Pitt. Inductive inference, DFAs, and computational complexity. *Analogical and Inductive Inference, Proceedings of the Second International Workshop*, AII 1989, volume 397 of *Lecture Notes in Artificial Intelligence*, pages 18–44. Springer Verlag, 1989.

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

[Shi94] Takeshi Shinohara. Rich classes inferable from positive data: Length-Bounded Elementary Formal Systems. *Information and Computation*, 108:175–186, 1994.

[Wie90] Rolf 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.