THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Lower Kent Ridge Road, Singapore 119260

TRA3/06

Iterative Learning from Positive Data and Negative Counterexamples

Sanjay JAIN and Efim KIMBER

March 2006

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Iterative Learning from Positive Data and Negative Counterexamples

Sanjay Jain [a,1] and Efim Kinber [b]

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

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

Abstract

A model for learning in the limit is defined where a (so-called *iterative*) learner gets all positive examples from the target language, tests every new conjecture with a teacher (oracle) if it is a subset of the target language (and if it is not, then it receives a negative counterexample), and uses only limited long-term memory (incorporated in conjectures). Three variants of this model are compared: when a learner receives least negative counterexamples, the ones whose size is bounded by the maximum size of input seen so far, and arbitrary ones. A surprising result is that sometimes *absence* of short counterexamples can help an iterative learner in a situation, where arbitrary counterexamples are useless. We also compare our learnability model with other relevant models of learnability in the limit, study how our model works for indexed classes of recursive languages, and show that learners in our model can work in *non-U-shaped* way — never abandoning the first right conjecture.

1 Introduction

In 1967 E. M. Gold [Gol67] suggested an algorithmic model for learning languages and other possibly infinite concepts. This model, **TxtEx**, where a learner gets all *positive* examples and stabilizes on the right description (a grammar) for the target concept, was adopted by computer and cognitive scientists (see, for example, [Pin79]) as a basis for discussion on algorithmic modeling of certain cognitive processes. Since then different other formal models of algorithmic learning in the limit have been defined and discussed in the literature. One of the major questions stimulating this discussion is what type of

[1] Supported in part by NUS grant number R252-000-127-112.

input information can be considered reasonable in various potentially infinite learning processes. Another important question is what amount of inputted data a learner can store in its (long-term) memory. Yet another issue is the way how input data is communicated to the learner. In the original Gold's model the learner is able to store potentially *all* inputted (positive) examples in its long-term memory; still, the latter assumption may be unrealistic for certain learning processes. Gold also considered a variant of his model where the learner receives all positive and *all negative* examples. However, while it is natural to assume that *some* negative data may be available to the learner, this variant, **InfEx**, though interesting from theoretical standpoint (for example, it can be used as a formal model for learning classes of functions, see [JORS99]), can hardly be regarded as adequate for most of the learning processes in question. R. Wiehagen in [Wie76] (see also [LZ96]) suggested a variant of the Gold's original model, so-called *iterative* learners, whose long-term memory cannot grow indefinitely (in fact, it is incorporated into the learner's conjectures). This model has been considered for learnability from all positive examples (denoted as **TxtIt**) and from all positive and all negative examples (**InfIt**). In her paper [Ang88], D. Angluin suggested a model of learnability, where data about the target concept is communicated to a learner in a way different from the Gold's model - it is supplied to the learner by a *minimally adequate teacher* (oracle) in response to *queries* from a learner. Angluin considered different type of queries, in particular, *membership* queries, where the learner asks if a particular word is in the target concept, and *subset* queries, where the learner tests if the current conjecture is a subset of the target language — if not, then the learner may get a *negative counterexample* from a teacher (subset queries and corresponding counterexamples help a learner to refute *overgeneralizing* wrong conjectures; K. Popper [Pop68] regarded refutation of overgeneralizing conjectures as a vital part of learning and discovery processes).

In [JK04], the authors introduced the model (**NCEx**) combining the Gold's model, **TxtEx**, and the Angluin's model: a **NCEx**-learner receives all positive examples of the target concept and makes subset query about each conjecture — receiving a negative counterexample if the answer is negative. This model is along the line of research related to the Gold's model for learnability from positive data in presence of *some* negative data (see also [Mot91,BCJ95]). Three variants of negative examples supplied by the teacher were considered: negative counterexamples of arbitrary size, if any (the main model **NCEx**), least counterexamples (**LNCEx**), and counterexamples whose size would be bounded by the maximum size of positive input data seen so far (**BNCEx**) — thus, reflecting complexity issues that the teacher might have. In this paper, we incorporate the limitation on the long-term memory reflected in the **It**-approach into all three above variants of learning from positive data and negative counterexamples: in our new model, **NCIt** (and its variations), the learner gets full positive data and asks a subset query about every conjecture,

however, the long-term memory is a part of a conjecture, and, thus, cannot store indefinitely growing amount of input data (since, otherwise, the learner cannot stabilize to a single right conjecture). Thus, the learners in our model, while still getting full positive data, get just as many negative examples as necessary (a finite number, if the learner succeeds) and can use only finite amount of long-term memory. We explore different aspects of our model. In particular, we compare all three variants between themselves and with other relevant models of algorithmic learning in the limit discussed above. We also study how our model works in the context of learning *indexed* (that is, effectively enumerable) classes of recursive languages (such popular classes as *pattern* languages (see [Ang80]) and regular languages are among them). In the end, we show that learners in our model can work in *non-U-shaped* way — not ever abandoning a right conjecture.

The paper is structured as follows. In Sections 2 and 3 we introduce necessary notation and formally introduce our and other relevant learnability models and establish trivial relationships between them. Section 4 is devoted to relationships between three abovementioned variants of **NCIt**. First, we show that least examples do not have advantage over arbitrary ones — this result is similar to the corresponding result for **NCEx** obtained in [JK04], however, the proof is more complex. Then we show that capabilities of iterative learners getting counterexamples of arbitrary size and those getting short counterexamples are incomparable. The fact that short counterexamples can sometimes help more than arbitrary ones is quite surprising: if a short counterexample is available, then an arbitrary one is trivially available, but not vice versa — this circumstance can be easily used by **NCEx**-learners to simulate **BNCEx**-learners, but not vice versa, as shown in [JK04]. However, it turns out that iterative learners can sometimes use the fact that a short counterexample *is not* available to learn concepts, for which arbitrary counterexamples are of no help at all!

Section 5 compares our models with other popular models of learnability in the limit. First, we show that **TxtEx**-learners, capable of storing potentially all positive input data, can learn sometimes more than **NCIt**-learners, even if the latter ones are allowed to make a finite number of errors in the final conjecture. On the other hand, **NCIt**-learners can sometimes do more than the **TxtEx**-learners (being able to store all positive data). In addition to exhibiting a class of languages witnessing the latter difference, we establish this difference on yet another level: it turns out that adding an arbitrary recursive language to a **NCIt**-learnable class preserves its **NCIt**-learnability, while it is trivially not true for **TxtEx**-learners. An interesting — and quite unexpected — result is that **NCIt**-learners can simulate any **InfIt**-learner. Note that **InfIt** gets access to *full* negative data, whereas an **NCIt**-learner gets only finite number of negative counterexamples (although both of them are not capable of storing all input data)! Moreover, **NCIt**-learners can sometimes learn more than any

InfIt-learner. The fact that **NCIt**-learners receive negative counterexamples to wrong "overinclusive" conjectures (that is conjectures which include elments outside the language) is exploited in the relevant proof. Here note that for **NCEx** and **InfEx**-learning where all data can be remembered, **NCEx** $\subset$ **InfEx**. So the relationship between negative counterexamples and complete negative data differs quite a bit from the noniterative case.

In Section 6, we show that if a teacher delays answers to an **NCIt**-learner, this can diminish its learning capability. This result contrasts a result from [JK04] showing that delays do not affect **NCEx**-learners — thus, delaying answers from a teacher can be detrimental to learners being not able to store full input data in the long-term memory.

In Section 7, we consider **NCIt**-learnability of indexed classes of recursive languages. Our main result here is that all such classes are **NCIt**-learnable. Note that it is typically not the case when just positive data is available — even with unbounded long-term memory. On the other hand, interestingly, there are indexed classes that are not learnable if a learner uses the set of programs computing just the languages from the given class as its hypotheses space (so-called *class-preserving* type of learning, see [ZL95]). The latter result is interesting, as usually if an indexed class is learnable within a formal learnability model, it is learnable class-preservingly (for example, it is trivially the case for **InfEx**-learnability, where the learner uses a natural *identification by enumeration* strategy, simply trying one target concept after another — as long as it does not contradict input data).

In Section 8, we prove that **NCIt**-learning can be done so that a learner never abandons a right conjecture (so-called *non-U-shaped* learning, see [BCM$^+$05], became recently a popular subject in developmental psychology, see [Bow82]). Note that the same problem for **TxtIt**-learners (not using negative counterexamples) remains open ([CCJS05]).

2 Notation and Preliminaries

Any unexplained recursion theoretic notation is from [Rog67]. The symbol N denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$. Symbols $\emptyset$, $\subseteq$, $\subsetneq$, $\supseteq$, and $\supsetneq$ denote empty set, subset, proper subset, superset, and proper superset, respectively. Cardinality of a set S is denoted by $\mathrm{card}(S)$. The maximum and minimum of a set are denoted by $\max(\cdot), \min(\cdot)$, respectively, where $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. $L_1 \mathbf{\Delta} L_2$ denotes the symmetric difference of L_1 and L_2, that is $L_1 \mathbf{\Delta} L_2 = (L_1 - L_2) \cup (L_2 - L_1)$. For a natural number a, we say that $L_1 =^a L_2$, iff $\mathrm{card}(L_1 \mathbf{\Delta} L_2) \leq a$. We say that $L_1 =^* L_2$, iff $\mathrm{card}(L_1 \mathbf{\Delta} L_2) < \infty$. Thus, we take $n < * < \infty$, for all $n \in N$. If $L_1 =^a L_2$, then we say that L_1 is

an a-variant of L_2.

We let $\langle \cdot, \cdot \rangle$ stand for an arbitrary, computable, bijective mapping from $N \times N$ onto N [Rog67]. We assume without loss of generality that $\langle \cdot, \cdot \rangle$ is monotonically increasing in both of its arguments. We define $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$. Pairing function can be extended to n-tuples in a natural way. Let $\text{cyl}_i = \{\langle i, x \rangle \mid x \in N\}$.

By φ we denote a fixed *acceptable* programming system for the partial computable functions mapping N to N [Rog67,MY78]. By φ_i we denote the partial computable function computed by the program with number i in the φ-system. Symbol $\mathcal{R}$ denotes the set of all recursive functions, that is total computable functions. Symbol $\mathcal{R}^n_{0,1}$ denotes the set of all recursive functions, with n input parameters and range being $\{0, 1\}$. By Φ we denote an arbitrary fixed Blum complexity measure [Blu67,HU79] for the φ-system. A partial recursive function $\Phi(\cdot, \cdot)$ is said to be a Blum complexity measure for φ, iff the following two conditions are satisfied:

(a) for all i and x, $\Phi(i, x)\downarrow$ iff $\varphi_i(x)\downarrow$.

(b) the predicate: $P(i, x, t) \equiv \Phi(i, x) \leq t$ is decidable.

By convention we use Φ_i to denote the partial recursive function $\lambda x.\Phi(i, x)$. Intuitively, $\Phi_i(x)$ may be thought as the number of steps it takes to compute $\varphi_i(x)$.

By W_i we denote domain(φ_i). W_i is, then, the recursively enumerable (r.e.) set/language ($\subseteq N$) accepted (or equivalently, generated) by the φ-program i. We also say that i is a grammar for W_i. Symbol $\mathcal{E}$ will denote the set of all r.e. languages. Symbol L, with or without decorations, ranges over $\mathcal{E}$. By χ_L we denote the characteristic function of L. By $\overline{L}$, we denote the complement of L, that is $N - L$. Symbol $\mathcal{L}$, with or without decorations, ranges over subsets of $\mathcal{E}$. By $W_{i,s}$ we denote the set $\{x < s \mid \Phi_i(x) < s\}$.

We often need to use padding to be able to attach some relevant information to a grammar. $pad(j, \cdot, \cdot, \ldots)$ denotes a 1–1 recursive function (of appropriate number of arguments) such that $W_{pad(j,\cdot,\cdot,\ldots)} = W_j$. Such recursive functions can easily be shown to exist [Rog67].

We now present concepts from language learning theory. The next definition introduces the concept of a *sequence* of data.

Definition 1 (a) A *sequence* σ is a mapping from an initial segment of N into $(N \cup \{\#\})$. The empty sequence is denoted by Λ.

(b) The *content* of a sequence σ, denoted content(σ), is the set of natural

numbers in the range of σ.

(c) The *length* of σ, denoted by $|\sigma|$, is the number of elements in σ. So, $|\Lambda| = 0$.

(d) For $n \leq |\sigma|$, the initial sequence of σ of length n is denoted by $\sigma[n]$. So, $\sigma[0]$ is Λ.

Intuitively, #'s represent pauses in the presentation of data. We let σ, τ, and γ, with or without decorations, range over finite sequences. We denote the sequence formed by the concatenation of τ at the end of σ by $\sigma\diamond\tau$. For simplicity of notation, sometimes we omit $\diamond$, when it is clear that concatenation is meant. SEQ denotes the set of all finite sequences.

Definition 2 [Gol67] (a) A *text* T for a language L is a mapping from N into $(N \cup \{\#\})$ such that L is the set of natural numbers in the range of T. $T(i)$ represents the $(i+1)$-th element in the text.

(b) The *content* of a text T, denoted by content(T), is the set of natural numbers in the range of T; that is, the language which T is a text for.

(c) $T[n]$ denotes the finite initial sequence of T with length n.

Definition 3 [Gol67] A *language learning machine from texts* is an algorithmic device which computes a (possibly partial) mapping from SEQ into N.

Definition 4 [Gol67] (a) An *informant* I is a mapping from N to $(N \times \{0,1\}) \cup \#$ such that for no $x \in N$, both $(x,0)$ and $(x,1)$ are in the range of I.

(b) content(I) = set of pairs in the range of I (that is range$(I) - \{\#\}$).

(c) We say that a I is *an informant* for L iff content$(I) = \{(x, \chi_L(x)) \mid x \in N\}$.

(d) A *canonical informant* for L is the informant $(0, \chi_L(0))(1, \chi_L(1))\ldots$.

Intuitively, informants give both all positive and all negative data for the language being learned. $I[n]$ is the first n elements of the informant I. One can similarly define language learning machines from informants.

We let $\mathbf{M}$, with or without decorations, range over learning machines. $\mathbf{M}(T[n])$ (or $\mathbf{M}(I[n])$) is interpreted as the grammar (index for an accepting program) conjectured by the learning machine $\mathbf{M}$ on the initial sequence $T[n]$ (or $I[n]$). We say that $\mathbf{M}$ converges on T to i, (written: $\mathbf{M}(T)\downarrow = i$) iff $(\forall^\infty n)[\mathbf{M}(T[n]) = i]$. Convergence on informants is similarly defined.

We let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denote a recursive enumeration of all (iterative) learning machines from texts/informants or negative counterexamples, based on con-

text.

There are several criteria for a learning machine to be successful on a language. Below we define some of them. All of the criteria defined below are variants of the **Ex**-style learning described in Introduction and its extension, *behaviourally correct*, or **Bc**-style learning (where a learner produces conjectures, almost all of which are correct, but not necessarily the same, see [JORS99] for formal definition); in addition, they allow a finite number of errors in almost all conjectures (uniformly bounded number, or arbitrary).

Definition 5 [Gol67,CL82] Suppose $a \in N \cup \{*\}$.

(a) **M** $\mathbf{TxtEx}^a$*-identifies a text* T just in case $(\exists i \mid W_i =^a \text{content}(T))$ $(\forall^\infty n)[\mathbf{M}(T[n]) = i]$.

(b) **M** $\mathbf{TxtEx}^a$*-identifies an r.e. language* L (written: $L \in \mathbf{TxtEx}^a(\mathbf{M})$) just in case **M** $\mathbf{TxtEx}^a$-identifies each text for L.

(c) **M** $\mathbf{TxtEx}^a$*-identifies a class* $\mathcal{L}$ *of r.e. languages* (written: $\mathcal{L} \subseteq \mathbf{TxtEx}^a(\mathbf{M})$) just in case **M** $\mathbf{TxtEx}^a$-identifies each language from $\mathcal{L}$.

(d) $\mathbf{TxtEx}^a = \{\mathcal{L} \subseteq \mathcal{E} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}^a(\mathbf{M})]\}$.

If instead of convergence to a grammar on text T, we just require that all but finitely many grammars output by **M** on T are for an a-variant of content(T), (that is, $(\forall^\infty n)[W_{\mathbf{M}(T[n])} =^a L]$), then we get $\mathbf{TxtBc}^a$-identification. We refer the reader to [CL82] or [JORS99] for details.

Definition 6 [Gol67,CL82] Suppose $a \in N \cup \{*\}$.

(a) **M** $\mathbf{InfEx}^a$*-identifies* L (written: $L \in \mathbf{InfEx}^a(L)$), just in case for all informants I for L, $(\exists i \mid W_i =^a L)$ $(\forall^\infty n)[\mathbf{M}(I[n]) = i]$.

(b) **M** $\mathbf{InfEx}^a$*-identifies a class* $\mathcal{L}$ *of r.e. languages* (written: $\mathcal{L} \subseteq \mathbf{InfEx}^a(\mathbf{M})$) just in case **M** $\mathbf{InfEx}^a$-identifies each language from $\mathcal{L}$.

(c) $\mathbf{InfEx}^a = \{\mathcal{L} \subseteq \mathcal{E} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{InfEx}^a(\mathbf{M})]\}$.

One can similarly define $\mathbf{InfBc}^a$-identification [CL82].

Intuitively, an iterative learner [Wie76,LZ96] is a learner whose hypothesis depends only on its last conjecture and current input. That is, for $n \geq 0$, $\mathbf{M}(T[n+1])$ can be computed algorithmically from $\mathbf{M}(T[n])$ and $T(n)$. Thus, one can describe the behaviour of **M** via a partial recursive function $F(p, x)$, where $\mathbf{M}(T[n+1]) = F(\mathbf{M}(T[n]), T(n))$. Here, note that $\mathbf{M}(T[0])$ is predefined to be some constant value. We will often identify F above with **M** (that is use $\mathbf{M}(p, x)$ to describe $\mathbf{M}(T[n+1])$, where $p = \mathbf{M}(T[n])$ and $x = T(n)$). This is

for ease of notation.

Below we formally define $\mathbf{TxtIt}^a$. $\mathbf{InfIt}^a$ can be defined similarly.

Definition 7 [Wie76,LZ96]

(a) $\mathbf{M}$ $\mathbf{TxtIt}^a$-*identifies* $\mathcal{L}$, iff $\mathbf{M}$ $\mathbf{TxtEx}^a$-identifies $\mathcal{L}$, and for all σ, τ and x, if $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$, then $\mathbf{M}(\sigma x) = \mathbf{M}(\tau x)$. We further assume that $\mathbf{M}(\sigma)$ is defined for all σ such that for some $L \in \mathcal{L}$, content$(\sigma) \subseteq L$.

(b) $\mathbf{TxtIt}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ $\mathbf{TxtIt}^a$-identifies $\mathcal{L}]\}$.

For $\mathbf{Ex}^a$ and $\mathbf{Bc}^a$ models of learning (for learning from texts or informants or their variants when learning from negative examples, as defined below), one may assume without loss of generality that the learners are total. However for iterative learning one cannot assume so. Thus, we explicitly require in the definition that iterative learners are defined on all inputs which are initial segments of texts (informants) for a language in the class.

Note that, although it is not stated explicitly, an **It**-type learner might store some input data in its conjecture (thus serving as a limited long-term memory). However, the amount of stored data cannot grow indefinitely, as the learner must stabilize to one (right) conjecture.

For $a = 0$, we often write $\mathbf{TxtEx}, \mathbf{TxtBc}, \mathbf{TxtIt}, \mathbf{InfEx}, \mathbf{InfBc}, \mathbf{InfIt}$ instead of $\mathbf{TxtEx}^0, \mathbf{TxtBc}^0, \mathbf{TxtIt}^0, \mathbf{InfEx}^0, \mathbf{InfBc}^0, \mathbf{InfIt}^0$, respectively.

Definition 8 [Ful90] σ is said to be a $\mathbf{TxtEx}$-*stabilizing sequence* for $\mathbf{M}$ on L, iff (a) content$(\sigma) \subseteq L$, and (b) for all τ such that content$(\tau) \subseteq L$, $\mathbf{M}(\sigma\tau) = \mathbf{M}(\sigma)$.

Definition 9 [BB75,Ful90] σ is said to be a $\mathbf{TxtEx}$-*locking sequence* for $\mathbf{M}$ on L, iff (a) σ is a $\mathbf{TxtEx}$-stabilizing sequence for $\mathbf{M}$ on L and (b) $W_{\mathbf{M}(\sigma)} = L$.

If $\mathbf{M}$ $\mathbf{TxtEx}$-identifies L, then every $\mathbf{TxtEx}$-stabilizing sequence for $\mathbf{M}$ on L is a $\mathbf{TxtEx}$-locking sequence for $\mathbf{M}$ on L. Furthermore, one can show that if $\mathbf{M}$ $\mathbf{TxtEx}$-identifies L, then for every σ such that content$(\sigma) \subseteq L$, there exists a $\mathbf{TxtEx}$-locking sequence for $\mathbf{M}$ on L which extends σ (see [BB75,Ful90]).

Similar result can be shown for $\mathbf{InfEx}$, $\mathbf{TxtBc}$, $\mathbf{InfBc}$ and other criteria of learning discussed in this paper. We often will drop $\mathbf{TxtEx}$ (and other criteria notation) from $\mathbf{TxtEx}$-stabilizing sequence and $\mathbf{TxtEx}$-locking sequence, when the criterion is clear from context.

$\mathcal{L}$ is said to be an *indexed family* of languages iff there exists an indexing $L_0, L_1, \ldots$ of languages in $\mathcal{L}$ such that the question $x \in L_i$ is uniformly decidable (i.e., there exists a recursive function f such that $f(i, x) = \chi_{L_i}(x)$).

We let INIT $= \{L \mid (\exists i)[L = \{x \mid x \leq i\}]\}$.

3 Learning with Negative Counterexamples

In this section we formally define our models of learning from full positive data and negative counterexamples as given by [JK04]. Intuitively, for learning with negative counterexamples, we may consider the learner being provided a text, one element at a time, along with a negative counterexample to the latest conjecture, if any. (One may view this negative counterexample as a response of the teacher to the *subset query* when it is tested if the language generated by the conjecture is a subset of the target language). One may model the list of negative counterexamples as a second text for negative counterexamples being provided to the learner. Thus the learning machines get as input two texts, one for positive data, and other for negative counterexamples.

We say that $\mathbf{M}(T, T')$ converges to a grammar i, iff for all but finitely many n, $\mathbf{M}(T[n], T'[n]) = i$.

First, we define the basic model of learning from positive data and negative counterexamples. In this model, if a conjecture contains elements not in the target language, then a negative counterexample is provided to the learner. **NC** in the definition below stands for *negative counterexample*.

Definition 10 [JK04] Suppose $a \in N \cup \{*\}$.

(a) **M NCEx**^{a}*-identifies a language* L (written: $L \in \mathbf{NCEx}^a(\mathbf{M})$) iff for all texts T for L, and for all T' satisfying the condition:

$T'(n) \in S_n$, if $S_n \neq \emptyset$ and $T'(n) = \#$, if $S_n = \emptyset$,
where $S_n = \overline{L} \cap W_{\mathbf{M}(T[n],T'[n])}$

$\mathbf{M}(T, T')$ converges to a grammar i such that $W_i =^a L$.

(b) **M NCEx**^{a}*-identifies a class* $\mathcal{L}$ *of languages* (written: $\mathcal{L} \subseteq \mathbf{NCEx}^a(\mathbf{M})$), iff **M NCEx**^{a}-identifies each language in the class.

(c) $\mathbf{NCEx}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{NCEx}^a(\mathbf{M})]\}$.

For ease of notation, we sometimes define $\mathbf{M}(T[n], T'[n])$ also as $\mathbf{M}(T[n])$, where we separately describe how the counterexamples $T'(n)$ are presented to the conjecture of **M** on input $T[n]$.

One can similarly define **NCIt**^{a}-learning, where the learner's output depends only on the previous conjecture and the latest positive data and counterex-

ample provided. In these cases, we sometimes denote the output $\mathbf{M}(T[n+1], T'[n+1])$, with $\mathbf{M}(p, T(n), T'(n))$, where $p = \mathbf{M}(T[n], T'[n])$ (here note that $\mathbf{M}(T[0], T'[0])$ is some predefined constant p_0).

Jain and Kinber [JK04] also considered the cases where

(i) negative counterexamples provided are the least ones (that is, in Definition 10(a), one uses $T'(n) = \min(S_n)$, instead of $T'(n) \in S_n$); The corresponding learning criteria is referred to as $\mathbf{LNCEx}^a$, and

(ii) negative counterexamples are provided iff they are bounded by the largest element seen in $T[n]$ (that is, in Definition 10(a), one uses $S_n = \overline{L} \cap W_{\mathbf{M}(T[n],T'[n])} \cap \{x \mid x \leq \max(\text{content}(T[n]))\}$); The corresponding learning criteria is referred to as $\mathbf{BNCEx}^a$.

We refer the reader to [JK04] for details. One can similarly define $\mathbf{LNCIt}^a$, $\mathbf{BNCIt}^a$, and $\mathbf{BNCBc}^a$, $\mathbf{LNCBc}^a$, $\mathbf{BNCBc}^a$ criteria of learning.

Below, we state a number of relationships between the formal models of learning discussed above which trivially follow from their definitions.

Proposition 11 *[JK04] Suppose $a \in N \cup \{*\}$.*

(a) $\mathbf{TxtEx}^a \subseteq \mathbf{BNCEx}^a \subseteq \mathbf{NCEx}^a \subseteq \mathbf{LNCEx}^a$.

(b) $\mathbf{TxtBc}^a \subseteq \mathbf{BNCBc}^a \subseteq \mathbf{NCBc}^a \subseteq \mathbf{LNCBc}^a$.

(c) $\mathbf{TxtIt}^a \subseteq \mathbf{BNCIt}^a$*;*

(d) $\mathbf{TxtIt}^a \subseteq \mathbf{NCIt}^a \subseteq \mathbf{LNCIt}^a$.

Note that $\mathbf{BNCIt}^a \not\subseteq \mathbf{NCIt}^a$, as we will show below.

Proposition 12 *Suppose $a \in N \cup \{*\}$.*

(a) $\mathbf{BNCIt}^a \subseteq \mathbf{BNCEx}^a \subseteq \mathbf{BNCBc}^a$.

(b) $\mathbf{NCIt}^a \subseteq \mathbf{NCEx}^a \subseteq \mathbf{NCBc}^a$.

(c) $\mathbf{LNCIt}^a \subseteq \mathbf{LNCEx}^a \subseteq \mathbf{LNCBc}^a$.

4 Relationship Among Different Variations of NCIt-Criteria

In this section we compare all three variants of iterative learners using negative counterexamples. Our first result shows that least counterexamples do not give

advantage to learners in our model. This result is similar to the corresponding result for **NCEx**-learners ([JK04]), however, the proof is more complex.

Following notation is used in Theorem 13 and Remark 30.

Notation: $\mathbf{M}^{\text{ncex}}(p, a_0 a_1 \ldots a_n)$ denotes the following computation. Define $q_0 = p$. Let $q_{i+1} = \mathbf{M}(q_i, a_i, \text{ncex}(q_i))$ (that is output of $\mathbf{M}$ when previous conjecture was q_i, a_i is the positive data and $\text{ncex}(q_i)$ is the negative counterexample (or #) provided). Then, the grammar output by $\mathbf{M}^{ncex}(p, a_0 a_1 \ldots, a_n) = q_{n+1}$, if all $\text{ncex}(q_i)$, $i \leq n$, in the above computation are defined (as well as $\mathbf{M}$ converges in all of above computations — this later part of $\mathbf{M}$ converging in all computations would always hold, when inputs are defined, as the input data will be from the class, as long as counterexamples provided by ncex are correct). Otherwise, $\mathbf{M}(p, a_0 a_1 \ldots a_n)$ is not defined.

Theorem 13 *For all $a \in N \cup \{*\}$,* $\mathbf{LNCIt}^a = \mathbf{NCIt}^a$

Proof. The proof idea is similar to that of showing $\mathbf{LNCEx} \subseteq \mathbf{NCEx}$, in [JK04] except that we need to keep track of the backlog in the simulation when we are trying to find the least counterexample.

We state the proof only for $a = 0$. The proof can easily be seen to work for arbitrary a. Suppose $\mathbf{M}$ **LNCIt**-identifies $\mathcal{L}$. We define $\mathbf{M}'$ which **NCIt**-identifies $\mathcal{L}$. Suppose T is a text for $L \in \mathcal{L}$. The conjectures of $\mathbf{M}'$ on input $T[m]$ would be of form $pad(p_m, r_m, \text{ncex}_m, \tau_m)$.

Invariants (A) to (D) would hold.

(A) τ_m is the backlog in simulation. That is, $T[m] = \tau'_m \tau_m$, where $\mathbf{M}'$ has, as yet, only simulated $\mathbf{M}$ on τ'_m, and simulation on τ_m is yet to be done. It will be the case that $\tau'_m \subseteq \tau'_{m+1}$. Thus, τ_{m+1} is a suffix of $\tau_m T(m+1)$. Furthermore, if r_{m+1} is 0, then $\tau'_m \subset \tau'_{m+1}$, and if r_{m+1} is not 0, then $\tau'_m = \tau'_{m+1}$.

(B) ncex_m is a partial mapping from conjectures to least counterexamples (or #) as known to $\mathbf{M}'$. Domain of ncex_m is all conjectures made by $\mathbf{M}$ on proper prefixes of τ'_m (here τ'_m is as in (a) above, and counterexamples provided to $\mathbf{M}$ are the least counterexamples).

(C) r_m is used to check the mode in which $\mathbf{M}'$ is currently in. If $r_m = 0$, it means $\mathbf{M}'$ is currently simulating $\mathbf{M}$, and the last conjecture p_m is the conjecture of $\mathbf{M}$ on τ'_m (when the counterexamples in the simulation are given using ncex_m).

If $r_m = 1 + \langle p, s \rangle$, it means that $\mathbf{M}'$ is currently trying to find out the least counterexample for the grammar p, which was output by $\mathbf{M}$ after seeing τ'_m (where the counterexamples were given by ncex_m). Here s denotes that until

now, we have verified that $W_p \cap \{x < s\} \subseteq L$, and are currently testing whether $W_p \cap \{s\} \subseteq L$ (in this case p_m is the grammar for $W_p \cap \{s\}$). Also, it is known that $W_p \not\subseteq L$ (as $\mathbf{M}'$ would have earlier received a negative counterexample to its conjecture of form $pad(p, \cdot, \cdot, \cdot)$).

(D) If $r_m = 1 + \langle p, w\rangle$ and $r_{m+1} \neq 0$, then $r_{m+1} = 1 + \langle p, w+1\rangle$.

Initially, $\mathbf{M}'$ on empty input is $pad(p_0, 0, \emptyset, \Lambda)$, where p_0 is the output of $\mathbf{M}$ on empty input. Note that the invariants hold.

We now describe how to determine parameters in $pad(p_{m+1}, r_{m+1}, \text{ncex}_{m+1}, \tau_{m+1})$, the output of $\mathbf{M}'$ on input $T(m)$, with counterexample y to previous conjecture $pad(p_m, r_m, \text{ncex}_m, \tau_m)$.

Note that in Case 1 below, the input positive data in the simulation of $\mathbf{M}$ is from the text T and ncex_{m+1}, where defined, is correct; thus one can determine whether $\mathbf{M}^{\text{ncex}_{m+1}}(p_m, \gamma)$, is defined or not under the assumption that input text is for a language from the class $\mathcal{L}$ (since in this case only reason for $\mathbf{M}^{\text{ncex}_{m+1}}(p_m, \gamma)$ to be undefined would be that ncex_{m+1} is undefined for one of the intermediate conjectures). Similar reasoning applies to Case 4.

Case 1: $r_m = 0$ and ($y = \#$ or $\text{ncex}_m(p_m)$ is defined).

If $\text{ncex}_m(p_m)$ is not defined, then extend ncex_m to ncex_{m+1} by additionally defining $\text{ncex}_{m+1}(p_m) = \#$. Otherwise let $\text{ncex}_{m+1} = \text{ncex}_m$.

Let γ be the largest prefix of $\tau_m T(m)$ such that $\mathbf{M}^{\text{ncex}_{m+1}}(p_m, \gamma)$ is defined. Note that γ above is not empty. Let $p_{m+1} = \mathbf{M}^{\text{ncex}_{m+1}}(p_m, \gamma)$. Let τ_{m+1} be such that $\tau_m T(m) = \gamma\tau_{m+1}$. Let $r_{m+1} = 0$.

Case 2: $r_m = 0$ and ($y \neq \#$ and $\text{ncex}_m(p_m)$ is not defined).

Let $\text{ncex}_{m+1} = \text{ncex}_m$. Let $r_{m+1} = 1 + \langle p_m, 0\rangle$. Let $\tau_{m+1} = \tau_m T(m)$. Let p_{m+1} be such that $W_{p_{m+1}} = \{0\} \cap W_{p_m}$.

Case 3: $r_m = 1 + \langle p, w\rangle$ and $y = \#$.

Let $\text{ncex}_{m+1} = \text{ncex}_m$. Let $r_{m+1} = 1 + \langle p, w+1\rangle$. Let $\tau_{m+1} = \tau_m T(m)$. Let p_{m+1} be such that $W_{p_{m+1}} = \{w+1\} \cap W_p$.

Case 4: $r_m = 1 + \langle p, w\rangle$ and $y \neq \#$.

Let ncex_{m+1} be extension of ncex_m by defining $\text{ncex}_{m+1}(p) = w$.

Let γ be largest prefix of $\tau_m T(m)$ such that $\mathbf{M}^{\text{ncex}_{m+1}}(p, \gamma)$ is defined. Let $p_{m+1} = \mathbf{M}^{\text{ncex}_{m+1}}(p, \gamma)$. Let τ_{m+1} be such that $\tau_m T(m) = \gamma\tau_{m+1}$. Let $r_{m+1} = 0$.

It is easy to verify that the invariants are maintained. Also, using invariants

(C), (D), it follows that if $r_m \neq 0$, then there exists a $m' > m$, such that $r_{m'} = 0$, and thus $\tau'_m \subset \tau'_{m'}$.

Now suppose n is such that $\mathbf{M}$ (when receiving least counterexamples) converges on T at n (that is, for all $n' \geq n$, $\mathbf{M}(T[n']) = \mathbf{M}(T[n])$. Then, once $\tau'_m \supseteq T[n+1]$, we have, using invariant (B), that ncex_m is defined on all conjectures of $\mathbf{M}$ on T. It follows that for all $m' > m$, on input $T(m')$ only Case 1 will apply, and value of $\tau_{m'}$ would be Λ, and $p_{m'}$ would be $\mathbf{M}(T)$ and $r_{m'}$ would be 0. Thus, $\mathbf{M}'$ on T converges to a grammar for L. ∎

One of the variants of teacher's answers to subset queries in [Ang88] was just "yes" or "no". That is, the teacher just tells the learner that a counterexample exists, but does not provide it. Note that the above proof works also under these conditions, as it did not use the exact value of numerical counterexample, but just the fact that it existed.

Now we will compare **NCIt**-learning with its variant where size of counterexamples is limited by the maximum size of the input seen so far.

First we show that, surprisingly, short counterexamples can sometimes help to iteratively learn classes of languages not learnable by any **NCIt**-learner. The proof exploits the fact that sometimes actually *absence* of short counterexamples can help in a situation when arbitrary counterexamples are useless! Note that a short counterexample is available to a learner, then an arbitrary counterexample is trivially available, and **NCEx**-learners easily utilize this circumstance to simulate any **BNCEx**-learner, as shown in [JK04].

Theorem 14 $\mathbf{BNCIt} - \mathbf{NCIt}^* \neq \emptyset$.

The following lemma gives the diagonalizing class $\mathcal{L}$. For ease of presentation, the diagonalization proof is split into two parts.

Lemma 15 *Let* $\mathcal{L}_1 = \{\{\langle 0, x\rangle \mid x \in W_e\} \mid e = \min(W_e), e \in N\}$.

$\mathcal{L}_2 = \{L \mid (\exists i, j \in N)[$
$\text{card}(L \cap \text{cyl}_0) < \infty$ *and* $L \cap \text{cyl}_1 = \{\langle 1, \langle i, j\rangle\rangle\}$ *and*
$(\forall w)[\langle 0, \langle i, w\rangle\rangle \notin L]$ *and* $(L - (\text{cyl}_0 \cup \text{cyl}_1)) = \{\langle 2, \langle x, k\rangle\rangle \mid x \in W_j, k \in N\}]\}$.

$\mathcal{L}_3 = \{L \mid (\exists i, j \in N,$ *finite set* $D)[$
$\text{card}(L \cap \text{cyl}_0) < \infty$ *and* $L \cap \text{cyl}_1 = \{\langle 1, \langle i, j\rangle\rangle\}$ *and*
$(\exists w)[\langle 0, \langle i, w\rangle\rangle \in L)]$ *and* $(L - (\text{cyl}_0 \cup \text{cyl}_1)) = \{\langle 2, \langle x, k\rangle\rangle \mid x \in D, k \in N\}]\}$.

Let $\mathcal{L}_4 = \{L \mid \text{card}(L) < \infty,\ L \subseteq \text{cyl}_0 \cup \text{cyl}_1$ *and* $L \not\subseteq \text{cyl}_0\}$.

Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$.

(a) $\mathcal{L}_1 \cup \mathcal{L}_4 \notin \mathbf{NCIt}$.

(b) $\mathcal{L} \notin \mathbf{NCIt}^*$.

PROOF. (a) Suppose by way of contradiction that $\mathcal{L}_1 \cup \mathcal{L}_4 \in \mathbf{NCIt}$ as witnessed by machine $\mathbf{M}$. Then, by implicit use of Kleene Recursion Theorem [Rog67], there exists an e such that W_e may be defined as follows. Initially enumerate e in W_e, and let σ_0 be such that content$(\sigma_0) = \{\langle 0, e\rangle\}$. Intuitively Cntrexmpls denotes the set of elements frozen to be outside the diagonalizing language being constructed. Initially, Cntrexmpls $= \{\langle i, x\rangle \mid i \geq 2$ and $i, x \in N\} \cup \{\langle 0, x\rangle \mid x < e\}$. Intuitively, NegSet is the set of conjectured grammars for which we have found a negative counterexample (in Cntrexmpls). Initially let NegSet $= \emptyset$. $ncex(j)$ is a function which gives, for $j \in$ NegSet, a negative counterexample from Cntrexmpls. For the following, let γ_τ be a sequence of length $|\tau|$ defined as follows. For $i < |\tau|$,

$$\gamma_\tau(i) = \begin{cases} ncex(\mathbf{M}(\tau[i], \gamma_\tau[i])), & \text{if } \mathbf{M}(\tau[i], \gamma_\tau[i]) \in \text{NegSet;} \\ \#, & \text{otherwise.} \end{cases}$$

(where the value of NegSet is as at the time of above usage). Here note that we will be using the above definition only for cases when $\mathbf{M}(\tau[i], \gamma_{\gamma_\tau}[i])$ is defined for all $i < |\tau|$.

Note that at the beginning of any stage s, NegSet will be finite, and Cntrexmpls will be a finite set plus $\{\langle i, x\rangle \mid i \geq 2$ and $i, x \in N\} \cup \{\langle 0, x\rangle \mid x < e\}$. Also we will have the invariant that at start of stage s, content$(\sigma_s) = \{\langle 0, x\rangle \mid x \in W_e$ enumerated before stage $s\}$.

Go to stage 0.

Stage s

1. Dovetail steps 2 and 3 until step 2 or 3 succeed. If step 2 succeeds before step 3, if ever, then go to step 4. If step 3 succeeds before step 2, if ever, then go to step 5.
   Here we assume that if step 3 can succeed by simulating $\mathbf{M}(\tau, \gamma_\tau)$ for s steps, then step 3 succeeded first (and for the shortest such τ), otherwise whichever of these steps succeeds first is taken. (So some priority is given to step 3 in the dovetailing).
2. Search for a $\tau \supseteq \sigma_s$ such that
   content$(\tau) \subseteq \{\langle 0, x\rangle \mid x \geq e\}$ − Cntrexmpls,
   $\mathbf{M}(\tau[i], \gamma_\tau[i])$, is defined for all $i \leq |\tau|$, and
   $\mathbf{M}(\tau, \gamma_\tau) \neq \mathbf{M}(\sigma_s, \gamma_{\sigma_s})$.
3. Search for a $\tau \subseteq \sigma_s$ such that $\mathbf{M}(\tau, \gamma_\tau) \notin$ NegSet and $W_{\mathbf{M}(\tau, \gamma_\tau)}$ enumerates an element not in content(σ_s).

4. Let τ be as found in step 2. Let $\sigma_{s+1} = \tau$, and enumerate $\{x \mid \langle 0, x\rangle \in \text{content}(\tau)\}$ into W_e.
   Go to stage $s+1$.
5. Let τ be as found in step 3, and $j = \mathbf{M}(\tau, \gamma_\tau)$, and z be the element found to be enumerated by W_j which is not in content(σ_s).
   Let NegSet = NegSet $\cup \{j\}$.
   Let Cntrexmpls = Cntrexmpls $\cup \{z\}$.
   Let $ncex(j) = z$.
   Let $\sigma_{s+1} = \sigma_s$.
   Go to stage $s+1$.

End stage s

We now consider the following cases:

Case 1: Stage s starts but does not finish.

Note that $\mathbf{M}$ needs to converge on all inputs, where positive data is contained in cyl_0 and negative counterexamples are consistent (due to it being defined on all inputs from $\mathcal{L}_4$). Thus, for any sequence σ extending σ_s, such that content$(\sigma) \subseteq \{\langle 0, x\rangle \mid x \in N\} -$ Cntrexmpls, we have that $\mathbf{M}(\sigma, \gamma_\sigma)\downarrow = \mathbf{M}(\sigma_s, \gamma_{\sigma_s})$, and the counterexamples given according to γ_{σ_s} is correct for any language which is not a subset of content(σ_s) (since otherwise step 3 would have succeeded). In other words, all conjectures by $\mathbf{M}$ on prefixes of $(\sigma_s, \gamma_{\sigma_s})$ are either contained in content(σ_s) or contain an element from Cntrexmpls (as given by ncex$(\cdot)$.

Now fix x such that $\langle 1, x\rangle \notin$ Cntrexmpls. Consider the behaviour of $\mathbf{M}$ on $T = \sigma_s\#(\langle 1, x\rangle)^\infty$, where the counterexamples beyond $\sigma_s\#$ are given based on the input language being $L = \text{content}(\sigma_s) \cup \{\langle 1, x\rangle\}$ and choosing the least counterexample (counterexamples on initial segments of $\sigma_s\#$ are provided based on $\gamma_{\sigma_s\#}$). If $\mathbf{M}$ makes infinitely many mind changes, then we have that $\mathbf{M}$ does not **NCIt**-identify content$(\sigma_s) \cup \{\langle 1, x\rangle\}$. On the other hand, if $\mathbf{M}$ makes only finitely many mind changes on T, then let S be the set of counterexamples provided on the input text T. Let $\langle 0, w\rangle$ be such that $\langle 0, w\rangle$ is not in S. Then, the behaviour of $\mathbf{M}$ on $T' = \sigma_s\diamond(\langle 0, w\rangle)\diamond(\langle 1, x\rangle)^\infty$, is same as that on T (here the counterexamples on input $\sigma_s\diamond(\langle 0, w\rangle)$ are based on $\gamma_{\sigma_s\diamond(\langle 0,w\rangle)}$; counterexample provided is the least counterexample once $\langle 1, x\rangle$ appears in the input). Thus, $\mathbf{M}$ does not **NCIt**-identify at least one of content$(\sigma_s) \cup \{\langle 0, w\rangle, \langle 1, x\rangle\}$ and content$(\sigma_s) \cup \{\langle 1, x\rangle\}$, both of which belong to $\mathcal{L}_4$.

Case 2: All stages finish.

Let $L = \{\langle 0, x\rangle \mid x \in W_e\}$. Let $T = \bigcup_{s \in N} \sigma_s$. Note that T is a text for L. Let Cntrexmpls (NegSet) denote the set of all elements which are ever placed in

Cntrexmpls (NegSet) by the above construction. Let γ_T be defined as follows.

$$\gamma_T(i) = \begin{cases} ncex(\mathbf{M}(T[i], \gamma_T[i])), & \text{if } \mathbf{M}(T[i], \gamma_T[i]) \in \text{NegSet}; \\ \#, & \text{otherwise.} \end{cases}$$

For $\tau \subseteq T$, let $\gamma_\tau = \gamma_T[|\tau|]$. Note that eventually, any conjecture j by $\mathbf{M}$ on (T, γ_T) which enumerates an element not in L, belongs to NegSet, with a negative counterexample for it belonging to Cntrexmpls (given by $ncex(j)$). This is due to eventual success of step 3, for all $\tau \subseteq T$, for which $\mathbf{M}(\tau, \gamma_\tau) \not\subseteq L$ (due to priority assigned to step 3).

If $\mathbf{M}(T, \gamma_T)$ makes infinitely many mind changes, then, clearly, $\mathbf{M}$ does not **NCIt**-identify L. On the other hand, if $\mathbf{M}$ makes only finitely many mind changes on (T, γ_T), then eventually, any conjecture j by $\mathbf{M}$ on (T, γ_T) which enumerates an element not in L, belongs to NegSet, with a negative counterexample for it belonging to Cntrexmpls (given by ncex(j)). Thus, beyond some large enough stage s, step 3 would never succeed, and, for any stage $s' \geq s$, simulation of $\mathbf{M}(\tau, \gamma_\tau)$, as at stage s' of step 2, is correct (i.e., negative counterexamples are given, whenever the conjectured language is not a subset of L), and step 2 succeeds in all but finitely many stages — a contradiction to $\mathbf{M}$ making only finitely many mind changes.

From above cases it follows that $\mathbf{M}$ does not **NCIt**-identify L. Part (a) follows.

(b) Proof of part (b) is an extension of the proof of part (a).

Intuitively, $\mathcal{L}_2 \cup \mathcal{L}_3$ cannot be learned from informants, if the learner does not get the information about the elements $\{\langle 0, \langle i, w\rangle\rangle \mid w \in N\}$. This is what is exploited in the following.

In the class used in the proof of part (a), we exploit the fact that in Case 1, the iterative learner is not able to remember all the elements of form $\langle 0, x\rangle$ that it has seen (and is also not able to find these elements by using its future conjectures/counterexamples). In the definition of $\mathcal{L}$, such crucial information (whether the input language contains an element of form $\langle 0, \langle i, w\rangle\rangle$ for some w) has been used to hide information whether the input language is coming from $\mathcal{L}_2$ or $\mathcal{L}_3$. An **NCIt***-learner is not able to detect this information, and thus not able to learn $\mathcal{L}_2 \cup \mathcal{L}_3$. We now informally present the details.

Suppose by way of contradiction that $\mathbf{M}$ **NCIt***-identifies $\mathcal{L}$. One then proceeds with the staging construction for defining W_e as in the proof of part (a). If the construction has infinitely many stages, then as in Case 2 of part (a), one can argue that $\mathbf{M}$ does not **NCIt***-identify $\{\langle 0, x\rangle \mid x \in W_e\}$ which is in $\mathcal{L}_1$.

On the other hand, if there are only finitely many stages (i.e. Case 1), then fix

σ_s as in Case 1. Then, by implicit use of Kleene Recursion Theorem [Rog67]), one can choose a large enough i and j such that $\langle 0, \langle i, w\rangle\rangle$ does not appear in σ_s, for any w, and W_j can be described as follows. W_j essentially repeats the diagonalization against $\mathbf{M}$ using the construction similar to that of W_e in proof of part (a). The only difference is that it uses cyl_2 instead of cyl_0, and instead of just using $\langle 0, x\rangle$, it uses $\langle 2, \langle x, k\rangle\rangle$, for all k, (that is, for each x either $\langle 2, \langle x, k\rangle\rangle$ would be placed in the diagonalizing language for all k, or none of $\langle 2, \langle x, k\rangle\rangle$ would be placed in the diagonalizing language). The elements in σ_s along with $\langle 1, \langle i, j\rangle\rangle$ are used as predefined elements committed to be in the diagonalizing language.

Again, if infinitely many stages are there, then we can argue as in Case 2 of part (a) that content$(\sigma_s) \cup \{\langle 1, \langle i, j\rangle\rangle\} \cup \{\langle 2, \langle x, k\rangle\rangle \mid x \in W_j, k \in N\}$ (which is in $\mathcal{L}_2$) is not $\mathbf{NCIt}^*$-identified by $\mathbf{M}$. Otherwise, the learner converges at some σ_t. Then, one can find an element of form $\langle 0, \langle i, w\rangle\rangle$ which has not appeared in Cntrexmpls along with an appropriate set D such that the last conjecture of $\mathbf{M}$ on σ_t is infinitely different from $L = \text{content}(\sigma_s) \cup \{\langle 1, \langle i, j\rangle\rangle, \langle 0, \langle i, w\rangle\rangle\} \cup \{\langle 2, \langle x, k\rangle\rangle \mid x \in D, k \in N\}$, which is a member of $\mathcal{L}_3$. (Here D would contain (i) all x such that $\langle 2, \langle x, k\rangle\rangle$ in content(σ_t), for some k, and (ii) one special x which ensures that the last conjecture of $\mathbf{M}$ is infinitely different from L). We omit the details. ∎

We are now ready to give the proof of Theorem 14.

Proof. (of Theorem 14) Without loss of generality assume that the pairing function, $\langle \cdot, \cdot \rangle$, is monotonically increasing in both the arguments.

It suffices to show that $\mathcal{L}$ as defined in Lemma 15(b) belongs to $\mathbf{BNCIt}$. To see this consider the following strategy for learner $\mathbf{M}$.

Phase 1: Initially, $\mathbf{M}$ keeps outputting a grammar for $\{\langle 0, x\rangle \mid x \in W_e\}$, where x is the minimal value seen so far such that $\langle 0, x\rangle$ belongs to the input language. This process continues, until $\mathbf{M}$ gets an input an element of form $\langle 1, \langle i, j\rangle\rangle$. If and when $\mathbf{M}$ receives $\langle 1, \langle i, j\rangle\rangle$ in its input, it remembers it, and proceeds to Phase 2.

Phase 2: In this phase $\mathbf{M}$ tries to determine an upper bound on the elements of $L \cap \text{cyl}_0$ (at least the ones which have already been received by $\mathbf{M}$). To do this, on input positive data x, $\mathbf{M}$ outputs a grammar for $\{\langle x + 3, 0\rangle\}$, until it does not receive a negative counterexample. Note that this will eventually happen as the first time $\mathbf{M}$ receives a positive data which is larger than all the elements seen in Phase 1, grammar for $\{\langle x+3, 0\rangle\}$ would not receive a negative counterexample, as $\langle x + 3, 0\rangle$ would then be larger than all the inputs seen so far (here note that all languages in $\mathcal{L}$, which contain $\langle 1, \langle i, j\rangle\rangle$ are infinite). Let $m = \langle x + 3, 0\rangle$ and go to Phase 3.

Phase 3: In this phase $\mathbf{M}$ tries to determine all the elements of form $\langle 0, x\rangle$, such that $\langle 0, x\rangle \leq m$ and $\langle 0, x\rangle \in L$. To do this, $\mathbf{M}$ first waits for an input element which is at least as large as m (note that this will eventually happen, if $L \in \mathcal{L} - \mathcal{L}_1$). Then, $\mathbf{M}$ can determine whether $\langle 0, x\rangle \in L$, for $\langle 0, x\rangle \leq m$, by outputting a grammar for $\{\langle 0, x\rangle\}$. $\langle 0, x\rangle \in L$, iff the above conjecture does not receive a negative counterexample. Let S_1 denote $L \cap \{\langle 0, x\rangle \mid \langle 0, x\rangle \leq m\}$.

Additionally $\mathbf{M}$ will also remember elements of form $\langle 0, x\rangle$, which are $\geq m$, and are received in Phase 3 (or later). Let the set of such elements be S_2.

Once all the elements of form $\langle 0, x\rangle \leq m$, which belong to L are determined: If $S_1 \cup S_2$ contains an element of form $\langle 0, \langle i, w\rangle\rangle$, then go to Phase 5. Otherwise go to Phase 4.

Phase 4: In Phase 4, $\mathbf{M}$ will keep updating S_2, in case it receives an element of form $\langle 0, x\rangle \geq m$. Additionally, it will output a grammar for $S_1 \cup S_2 \cup \{\langle 1, \langle i, j\rangle\rangle\} \cup \{\langle 2, \langle x, k\rangle\rangle \mid x \in W_j, k \in N\}$,

If in Phase 4, $\mathbf{M}$ ever receives an element of form $\langle 0, \langle i, w\rangle\rangle$ then it will go to Phase 5.

Phase 5: In Phase 5, $\mathbf{M}$ will keep updating S_2, in case it receives an element of form $\langle 0, x\rangle \geq m$. Additionally, it will keep track of set D consisting of elements x such that $\langle 2, \langle x, k\rangle\rangle$ is seen in input in Phase 5.

It will then output a grammar for $S \cup \{\langle 1, \langle i, j\rangle\rangle\} \cup \{\langle 2, \langle x, k\rangle\rangle \mid x \in D, k \in N\}$.

This completes the description of $\mathbf{M}$. Now suppose the input language $L \in \mathcal{L}_1$. Then clearly Phase 1 will never end and $\mathbf{M}$ will **BNCIt**-identify L.

If $L \in \mathcal{L}_1$, then $\mathbf{M}$ will never leave Phase 1, and $\mathbf{M}$ will eventually determine the least e such that $\langle 0, e\rangle$ belongs to L, and thus correctly output a grammar for L in the limit.

If $L \in \mathcal{L}_2 \cup \mathcal{L}_3$, then eventually $\mathbf{M}$ will receive an element of form $\langle 1, \langle i, j\rangle\rangle$, and then $\mathbf{M}$ will proceed to Phase 2. In Phase 2, it will eventually get an input x such that its conjecture of $\{\langle x + 3, 0\rangle\}$, does not get a counterexample, and thus it will proceed to Phase 3. In Phase 3, it will successfully determine all elements $\langle 0, x\rangle \in L$, which are $\leq m$.

If $L \in \mathcal{L}_2$, then $\mathbf{M}$ will proceed to Phase 4 (as $S_1 \cup S_2$ will not contain any element of form $\langle 0, \langle i, w\rangle\rangle$) and then eventually output only correct (canonical) grammar for L, as it will eventually have $S_1 \cup S_2 = L \cap \mathrm{cyl}_0$.

If $L \in \mathcal{L}_3$, then either at end of Phase 3, or in Phase 4, it will eventually find an element in the input set which is of form $\langle 0, \langle i, w\rangle\rangle$, and thus proceed to

Phase 5. In Phase 5, it will eventually have $S_1 \cup S_2 = L \cap \mathrm{cyl}_0$, and $D = \{x \mid (\exists k)[\langle 2, \langle x, k\rangle]\rangle \in L\}$, as L either contains $\langle 2, \langle x, k\rangle]\rangle$, for all k or no such k. Thus, $\mathbf{M}$ will then output a correct (canonical) grammar for L and not change its mind thereafter. ∎

The next theorem shows that **NCIt**-learners can sometimes do more than any **BNCBc**-learner, even if the latter one is allowed to make finite number of errors in almost all conjectures.

Theorem 16 $(\mathbf{NCIt} \cap \mathbf{InfIt}) - \mathbf{BNCBc}^* \neq \emptyset$.

Proof. $\mathrm{INIT} \cup \{N\}$ can be easily seen to be in $\mathbf{NCIt} \cap \mathbf{InfIt}$. $\mathrm{INIT} \cup \{N\} \notin \mathbf{BNCBc}^*$ was shown in [JK05]. ∎

Now we will compare **NCIt** and **BNCIt** for classes consisting of only infinite languages. Note that **NCEx** and its bounded variant have same power for such classes, as established in [JK05]. As our next theorem shows, for iterative learners, arbitrary counterexamples give a learner an advantage.

Theorem 17 *There exists a class of infinite languages* $\mathcal{L} \in \mathbf{NCIt} - \mathbf{BNCIt}$.

Proof. Let Let $\mathcal{L} = \{\mathrm{cyl}_0 \cup \mathrm{cyl}_1\} \cup \{\mathrm{cyl}_0 \cup D \mid \mathrm{card}(D) < \infty\}$.

Clearly, $\mathcal{L}$ consists only of infinite languages, as every language in $\mathcal{L}$ contains cyl_0.

It is easy to verify that $\mathcal{L} \in \mathbf{NCIt}$ (learner first conjectures $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$. If there is a counterexample, then the learner just switches to learning $\{\mathrm{cyl}_0 \cup D \mid D$ is finite$\}$.

Now suppose by way of contradiction, that $\mathbf{M}$ **BNCIt**-identifies $\mathcal{L}$. Then let (σ, σ') be a **BNCIt**-locking sequence for $\mathbf{M}$ on $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$, where the counterexamples provided in σ' are the least ones, if valid. (that is, for any $m < |\sigma|$, $\sigma'(m) = \#$, if $W_{\mathbf{M}(\sigma[m], \sigma'[m])} \cap \{x \mid x \leq \max(\mathrm{content}(\sigma[m]))\} \subseteq \mathrm{cyl}_0 \cup \mathrm{cyl}_1$; otherwise $\sigma'(m) = \min(W_{\mathbf{M}(\sigma[m], \sigma'[m])} - (\mathrm{cyl}_0 \cup \mathrm{cyl}_1))$). Thus, for all $z \in \mathrm{cyl}_0 \cup \mathrm{cyl}_1 \cup \{\#\}$, $\mathbf{M}(\sigma z, \sigma'\#) = \mathbf{M}(\sigma, \sigma')$, which is a grammar for $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$. Without loss of generality assume that $\mathrm{content}(\sigma) = \{\langle 1, x\rangle \mid \langle 1, x\rangle \leq \max(\mathrm{content}(\sigma))\} \cup \{\langle 0, x\rangle \mid \langle 0, x\rangle \leq \max(\mathrm{content}(\sigma))\}$. Let w be the least number such that $\langle 1, w\rangle \notin \mathrm{content}(\sigma)$. Let w' be such that $\langle 0, w'\rangle > \langle 1, w\rangle$. Let T_{cyl_0} be a text for cyl_0. Now consider the behaviour of $\mathbf{M}$ on text $T = \sigma\diamond\langle 0, w'\rangle\diamond T_{cyl_0}$ for $L = \mathrm{content}(\sigma) \cup \mathrm{cyl}_0$, where counterexamples provided are the least valid ones. Thus, the sequence of counterexamples, T', starts with $\sigma'\diamond\#\diamond\langle 1, w\rangle)$. (Note that $\mathbf{M}(\sigma\diamond\langle 0, w'\rangle, \sigma'\diamond\#) = \mathbf{M}(\sigma, \sigma')$, which is a grammar for $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$). If $\mathbf{M}(T, T')$ does not converge to a grammar for L, then we have that $\mathbf{M}$ does not **BNCIt**-identify $L \in \mathcal{L}$. On the other hand, if $\mathbf{M}(T, T')$

converges to a grammar for L, then let S be the set of counterexamples provided. Let $\langle 1, z\rangle$ be such that $\langle 1, z\rangle > \max((L \cup S \cup \{\langle 1, w\rangle\}) \cap \mathrm{cyl}_1)$. Then, as (σ, σ') was a **BNCIt**-locking sequence for $\mathbf{M}$ on $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$, the behaviour of $\mathbf{M}$ on (T'', T') is same as that on (T, T'), where $T'' = \sigma \diamond \langle 1, z\rangle \diamond T_{cyl_0}$. (Note that the counterexamples in T' are valid for $\mathrm{content}(T'') = L \cup \{\langle 1, z\rangle\}$). It follows that $\mathbf{M}$ fails to **BNCIt**-identify $L \cup \{\langle 1, z\rangle\}$, which is a member of $\mathcal{L}$. ∎

Note that by Theorem 36, the class used in the above separation does not belong to $\mathbf{InfIt}^n$, for all n.

Remark 18 The result above is quite sensitive to the exact definition of **BNC**. To illustrate this, note the following property.

(Property 1) If a **BNC**-learner gets no counterexample for some conjecture for a superset of W_p on input $T[s]$, but gets a counterexample for conjecture p on input $T[s']$, then $\max(T(s), T(s+1), \ldots, T(s'-1)) = \max(\mathrm{content}(T[s']))$. In particular, if $s' = s + 1$ in the previous statement, then $T(s) = \max(\mathrm{content}(T[s']))$.

Consequently, if in the definition of **BNC**, we had chosen $S_n = \overline{L} \cap W_{\mathbf{M}(T[n], T'[n])} \cap \{x \mid x \leq \max(\mathrm{content}(T[n+1]))\}$, then the above result changes to its opposite as shown by following proof idea.

We want to build a **BNCIt**-learner $\mathbf{M}'$ that simulates a **NCIt**-learner $\mathbf{M}$. In the original definition, by the time $\mathbf{M}'$ knows that conjecture is not a subset of the input language, it is too late (we just got the large data $T(n-1)$, and lost it since $\mathbf{M}'$ did not change mind — and then got to know that conjecture was not a subset). On the other hand, if the non-subset answer is based on $T[n+1]$, then when $\mathbf{M}'$ knows that the conjecture is not a subset, $\mathbf{M}'$ also knows the large data causing it — $T(n)$ — it is the crucial difference. Once $\mathbf{M}'$ knows that it has to remember the maximum element of (this section of) the input, then it can use standard methods of buffering (used in other simulations in our paper), and getting back lost data. A bit more specifically, $\mathbf{M}'$ has two different modes of operation and keeps track of all past conjectures, known counterexamples corresponding to them. In normal mode of operation, new conjectures are unions of past conjectures which have not yet received counterexamples. When $\mathbf{M}'$ gets a counterexample, we find out the offending earliest past conjecture (via non-normal mode where we buffer input data). If a conjecture made prior to the conjecture made at $T[n]$ was offending then we can recover all past data using the fact that $T(n)$ (the maximum of the input $T[n]$) is available (by testing whether $\{x\} \subseteq L$, for each $x \leq T(n)$, again in non-normal mode, where we buffer input data). Then one can consider an appropriate modification of the input text (by replacing past data with an initial segment of elements $\leq T(n)$), and restart the simulation on the modified

text. One needs to be slightly careful in doing simulation on buffered data. The full proof is technically complex, and needs some further adjustments to the above basic idea (as the actual testing of prior conjecture may happen at input beyond $T[n+1]$, however this is not crucial), and we omit the details.

Note that showing the difference between **BNCIt** and **NCIt** in the general case does not depend on particular definition of **BNCIt**: Theorem 16 that states this difference, holds no matter whether we use $T[m+1]$ (or any other $T[m+c]$) or $T[m]$ in the definition of $\mathbf{BNCBc}^*$, which, in turn, contains **BNCIt** no matter how the B-bound is interpreted in both $\mathbf{BNCBc}^*$ and **BNCIt**.

5 Comparison With Other Criteria of Learning

In this section we compare our model with other close relevant models of learnability in the limit.

Our first aim is to establish a hierarchy of learnability in our model based on the number of errors in the final correct conjecture. First, we need an auxiliary similar result for regular iterative learners.

Theorem 19 *(Based on [CS83])*

(a) $\mathbf{TxtIt}^{n+1} - \mathbf{InfEx}^n \neq \emptyset$.

(b) $\mathbf{TxtIt}^* - \bigcup_{n\in N} \mathbf{InfEx}^n \neq \emptyset$.

Proof. Let $L_f = \{\langle x, f(x)\rangle \mid x \in N\}$.

For $a \in N \cup \{*\}$, let $\mathcal{C}_a = \{f \mid \varphi_{f(0)} =^a f\}$.

Let $\mathcal{L}_a = \{L_f \mid f \in \mathcal{C}_a\}$.

It is easy to verify that $\mathcal{L}_a \in \mathbf{TxtIt}^a$. [CS83] showed that $\mathcal{L}_{n+1} \notin \mathbf{InfEx}^n$ and $\mathcal{L}_* \notin \bigcup_{n\in N} \mathbf{InfEx}^n$. Theorem follows. ∎

As $\mathbf{LNCIt}^a \subseteq \mathbf{LNCEx}^a \subseteq \mathbf{InfEx}^a$ (first inequality follows by definition; see [JK04] for the second inequality), we have:

Corollary 20 *(a)* $\mathbf{TxtIt}^{n+1} - (\mathbf{LNCIt}^n \cup \mathbf{BNCIt}^n) \neq \emptyset$.

(b) $\mathbf{TxtIt}^* - \bigcup_{n\in N}(\mathbf{LNCIt}^n \cup \mathbf{BNCIt}^n) \neq \emptyset$.

Our next two results show that learners that can store in their long-term

memory potentially all positive data can sometimes learn more than any **BNCIt**/**NCIt**-learner.

Theorem 21 $\mathbf{TxtEx} - \mathbf{BNCIt}^* \neq \emptyset$.

Proof. Let $L_{e,e'} = \{\langle e, e', x\rangle \mid x \in W_e\}$. $X^y_{e,e'} = L_{e,e'} \cup \{\langle e, e', x\rangle \mid x \geq y, x \in W_{e'}\}$.

Let $\mathcal{L}_1 = \{L_{e,e'}, X^y_{e,e'} \mid W_e \cap W_{e'} = \emptyset\}$.

Let $\mathcal{L}_2 = \{L \mid \text{card}(L) < \infty \text{ and } (\forall e, e')[L \not\subseteq \{\langle e, e', x\rangle \mid x \in N\}]\}$.

Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$.

It is easy to verify that $\mathcal{L} \in \mathbf{TxtEx}$. If the input language is not a subset of $\{\langle e, e', x\rangle \mid x \in N\}$, for any e, e', then one can use the strategy for learning finite sets. Otherwise, the learner outputs a grammar for $L_{e,e'}$ until an element of form $\langle e, e', x\rangle$ appears in the input, for some $x \in W_{e'}$. In this case learner outputs a grammar for $X^y_{e,e'}$, for y being minimum such x.

We now show that $\mathcal{L} \notin \mathbf{BNCIt}^*$. Suppose by way of contradiction that $\mathcal{L} \in \mathbf{BNCIt}$ as witnessed by machine $\mathbf{M}$. Then, by implicit use of Double Recursion Theorem [Rog67], there exist e and e' such that $W_e, W_{e'}$ may be defined as follows. We will have that each x belongs to at most one of $W_e, W_{e'}$. In stage s we wish to place s in one of W_e or $W_{e'}$ (if stage s completes, then s will be placed in one of W_e or $W_{e'}$). We will also construct (an initial segment of) a text T. We will have the invariant that $T(s) = \langle e, e', s\rangle$, iff $s \in W_e$. $T(s) = \#$ or undefined otherwise (where $T(s)$ is defined if stage s completes). It thus follows that T is a text for $L_{e,e'}$ (assuming $T(s)$ is defined for all s).

For the following, let cseq^t_τ be a sequence of length $|\tau|$ defined as follows. For $i < |\tau|$,

$$\text{cseq}^t_\tau(i) = \begin{cases} \min(W_{\mathbf{M}(\tau[i], \text{cseq}^t_\tau[i]), t} - \text{content}(\tau)), & \text{if } W_{\mathbf{M}(\tau[i], \text{cseq}^t_\tau[i]), t} \cap \\ & \{x \mid x \leq \max(\text{content}(\tau[i]))\} \\ & \not\subseteq \text{content}(\tau); \\ \#, & \text{otherwise.} \end{cases}$$

Intuitively, cseq gives the sequence of counterexamples using t enumeration steps for the conjectures.

Go to stage 0.

Stage s

For $t = s$ to ∞ do

1. If $\mathbf{M}(T[r], \text{cseq}^t_{T[r]})$, for $r \leq s$, or $\mathbf{M}(T[s]\diamond\langle e, e', s\rangle, \text{cseq}^t_{T[s]\diamond\langle e,e',s\rangle})$ or $\mathbf{M}(T[s]\#, \text{cseq}^t_{T[s]\#})$, do not converge within t steps, then go to next iteration of for loop. Otherwise, proceed to step 2.
2. If $\mathbf{M}(T[s], \text{cseq}^t_{T[s]})\downarrow \neq \mathbf{M}(T[s]\diamond\langle e, e', s\rangle, \text{cseq}^t_{T[s]\diamond\langle e,e',s\rangle})\downarrow$, then enumerate s in W_e, let $T(s) = \langle e, e', s\rangle$, and go to stage $s + 1$.
   Else, enumerate s in $W_{e'}$, let $T(s) = \#$, and go to stage $s + 1$.

EndFor

End stage s

If some stage s starts but does not finish, then $\mathbf{M}$ is not defined on some valid positive data/counterexample sequence. To see this let σ be $T[r]$, $r \leq s$, or $T[s]\#$, or $T[s]\diamond\langle e, e', s\rangle$, such that $\mathbf{M}(\sigma, \text{cseq}^t_\sigma)$ does not converge, and x be large enough such that $\langle e + 1, e', x\rangle > \max(\text{content}(\sigma))$. Now we have that $\mathbf{M}$ is not defined on some prefix of $\sigma\langle e + 1, e', x\rangle\#^\infty$, where counterexamples provided are the least (bounded) ones, if any. However, as $\text{content}(\sigma) \cup \{\langle e + 1, e', x\rangle\} \in \mathcal{L}_2$, we have that $\mathbf{M}$ does not $\mathbf{BNCIt}^*$-identify $\mathcal{L}$. So assume that all stages finish. Note that T is a text for $L_{e,e'}$, and the non-# elements of T form an increasing sequence of elements of $L_{e,e'}$. Let cseq_T be defined as follows:

$$\text{cseq}_T(i) = \begin{cases} \min(W_{\mathbf{M}(T[i],\text{cseq}_T[i])} - \text{content}(T)), & \text{if } W_{\mathbf{M}(T[i],\text{cseq}_T[i])} \cap \\ & \{x \mid x \leq \max(\text{content}(T[i]))\} \\ & \not\subseteq \text{content}(T); \\ \#, & \text{otherwise.} \end{cases}$$

If $\mathbf{M}(T, \text{cseq}_T)$ makes infinitely many mind changes, then clearly $\mathbf{M}$ does not $\mathbf{BNCIt}^*$-identifies $L_{e,e'} \in \mathcal{L}$. So assume that n is such that for all $m \geq n$, $\mathbf{M}(T[m], \text{cseq}_T[m])\downarrow = \mathbf{M}(T[n], \text{cseq}_T[n])\downarrow$, and $\text{cseq}_T(m) = \text{cseq}_T(n)$. Also note that if $W_{\mathbf{M}(T,\text{cseq}_T)} \not\subseteq \text{content}(T)$, then $\text{cseq}_T(n) \neq \#$. Let t be large enough such that for all $m \leq n$, $\mathbf{M}(T[m], \text{cseq}_T[m])\downarrow$ in t time steps, and if $\text{cseq}_T(m) \neq \#$, then $\text{cseq}_T(m) \in W_{\mathbf{M}(T[m],\text{cseq}_T[m]),t}$. Furthermore, assume that all members of $\text{content}(T)$ which are $\leq \max(\text{content}(\text{cseq}_T))$ belong to $\text{content}(T[n])$. It follows that

(A) for all $t' > t$, $s > t$, $\text{cseq}^{t'}_{T[s]} \subseteq \text{cseq}_T$, and thus $\mathbf{M}(T[s], \text{cseq}^{t'}_{T[s]}) = \mathbf{M}(T[n], \text{cseq}_T[n]) = \mathbf{M}(T, \text{cseq}_T)$.

It follows that in stages $s > t$, If statement in step 1 does not hold (since otherwise, we will have $\mathbf{M}(T[s], \text{cseq}^{t'}_{T[s]}) \neq \mathbf{M}(T[s + 1], \text{cseq}^{t'}_{T[s+1]})$, for some $t' \geq s$, in contradiction to (A)). Thus, $T(s) = \#$, for all $s > t$. Thus, (using (A), and If statement of step 2, Stage s) we have for all $s > t$, $\mathbf{M}(T[s]\diamond\langle e, e', s\rangle, \text{cseq}^{t'}_{T[s]\diamond\langle e,e',s\rangle}) = \mathbf{M}(T[s], \text{cseq}^{t'}_{T[s]}) = \mathbf{M}(T, \text{cseq}_T)$, for some

appropriate $t' \geq s$. It follows that for large enough s, $\mathbf{M}(T[s]\diamond\langle e,e',s\rangle\langle e,e',s+1\rangle\langle e,e',s+2\rangle\ldots,\mathrm{cseq}_{T[s]\diamond\langle e,e',s\rangle\langle e,e',s+1\rangle\langle e,e',s+2\rangle\ldots}) = \mathbf{M}(T,\mathrm{cseq}_T)$. (Note that for large enough s, if $W_{\mathbf{M}(T[n],\mathrm{cseq}_{T[n]})}$ contains an element not content(T), then it contains such an element $\leq \langle e,e',s\rangle$ which belongs to $W_{\mathbf{M}(T[n],\mathrm{cseq}_{T[n]}),s})$. Thus, $\mathbf{M}$ fails to $\mathbf{BNCIt}^*$-identify at least one of $L_{e,e'}$ and $X^s_{e,e'}$. ∎

Theorem 22 $\mathbf{TxtEx} - \mathbf{NCIt}^* \neq \emptyset$.

Proof. Consider $\mathcal{L}$ as defined in Lemma 15(b). $\mathcal{L} \in \mathbf{TxtEx}$ can be shown as follows. The learner first checks if the input contains an element of form $\langle 1,\langle i,j\rangle\rangle$. If not, then the language must be from $\mathcal{L}_1$, which can be easily **TxtEx**-identified. On the other hand, suppose input contains an element $\langle 1,\langle i,j\rangle\rangle$. Then, by checking whether the input contains an element of form $\langle 0,\langle i,w\rangle\rangle$ or not, one can use the **TxtEx**-learning algorithm for $\mathcal{L}_3$ or $\mathcal{L}_2$ respectively, which are individually easily seen to be in **TxtEx**. ∎

Now we show that **NCIt**-learners (even **BNCIt**-learners) can sometimes be more powerful than any $\mathbf{TxtBc}^*$-learner.

Theorem 23 $\mathbf{BNCIt} \cap \mathbf{NCIt} - \mathbf{TxtBc}^* \neq \emptyset$.

Proof. $\{L \mid (\exists D \mid \mathrm{card}(D) < \infty)[L = \{\langle i,x\rangle \mid x \notin D, i, x \in N\}]\}$. It is easy to verify that $\mathcal{L} \in \mathbf{BNCIt} \cap \mathbf{NCIt}$. Gold [Gol67] showed that $\{L \mid \mathrm{card}(N - L) < \infty\} \notin \mathbf{TxtBc}$ (even by non-effective learners), which implies that $\mathcal{L} \notin \mathbf{TxtBc}^*$. ∎

Now we will compare our model with iterative learners from informants. First, we show that there are **BNCIt**-learnable classes that cannot be learned from informants by any iterative learner. Thus, even just finite number of short negative data (received when necessary) can help iterative learners sometimes more than full negative data (most of it being forgotten by the learner).

Theorem 24 $\mathbf{BNCIt} \cap \mathbf{NCIt} - \mathbf{InfIt} \neq \emptyset$.

Proof. Let $\mathcal{L} = \{\{\langle 0,x\rangle \mid x \in W_e\} \mid e = \min(W_e), e \in N\} \cup \{L \mid (\exists x)[\langle 1,x\rangle \in L \text{ and } L - \{\langle 1,x\rangle\} \subseteq \{\langle 0,y\rangle \mid \langle 0,y\rangle \leq \langle 1,x\rangle\}]\}$.

It is easy to verify that the above class is in $\mathbf{BNCIt} \cap \mathbf{NCIt}$. (Initially just output minimal e such that $\langle 0,e\rangle$ is in the input, until it is found that the input language contains $\langle 1,x\rangle$ for some x. Then using the conjectures for $\{\langle 0,y\rangle\}$, for $\langle 0,y\rangle \leq \langle 1,x\rangle$, one can determine the elements of L.)

$\mathcal{L} \notin \mathbf{InfIt}$ can be shown as follows. Suppose by way of contradiction $\mathbf{M}$ **InfIt**-identifies $\mathcal{L}$. Note that $\mathbf{M}$ must be defined on all information segments

σ such that $\{x \mid (x,1) \in \text{content}(\sigma)\} \subseteq \{\langle 0,y\rangle \mid y \in N\}$, as $\mathbf{M}$ is defined on information segments for languages in $\mathcal{L}$. Now, by implicit use of Kleene Recursion Theorem [Rog67], there exists an e such that W_e may be described as follows. Initially, $e \in W_e$. Let σ_0 be information segment such that $\text{content}(\sigma_0) = \{(\langle 0,x\rangle,0) \mid x < 0\} \cup \{(\langle 0,e\rangle,1)\}$. Let $z_0, z_1, \ldots$ be an enumeration of elements of $N - \{\langle 0,x\rangle \mid x \in N\}$. Suppose σ_s has been defined. Define σ_{s+1} as follows. If one of $\mathbf{M}(\sigma_s \diamond (z_s,0) \diamond (\langle 0, e+s+1\rangle, 0))$ and $\mathbf{M}(\sigma_s \diamond (z_s,0) \diamond (\langle 0, e+s+1\rangle, 1))$ is different from $\mathbf{M}(\sigma_s)$, then (i) let σ_{s+1} be $\sigma_s \diamond (z_s,0) \diamond (\langle 0, e+s+1\rangle, w)$, where $w \in \{0,1\}$ and $\mathbf{M}(\sigma_s) \neq \mathbf{M}(\sigma_{s+1})$ and (ii) enumerate $e+s+1$ in W_e iff w chosen above is 1.

Now if σ_s is defined, for all s, then $\mathbf{M}$ diverges on $\bigcup_{s \in N} \sigma_s$, an informant for $\{\langle 0,x\rangle \mid x \in W_e\}$. On the other hand, if σ_{s+1} does not get defined (but σ_s does get defined), then fix k such that $\langle 1,k\rangle > \max(\{\langle 0,x\rangle \mid x \leq e+s+1\} \cup \{z_r \mid r \leq s\})$, and let I be such that $\text{content}(I) = \{(\langle 0,x\rangle,0) \mid x > e+s+1\} \cup \{\langle z_r, 0\rangle \mid r \in N, z_r \neq \langle 1,k\rangle\}$. Let $I_w = \sigma_s \diamond (z_s,0) \diamond (\langle 0, e+s+1\rangle, w)(\langle 1,k\rangle, 1)I$. Note that I_1 is an informant for $L_1 = \{\langle 0,x\rangle \mid x \in W_e\} \cup \{\langle 1,k\rangle\} \cup \{\langle 0, e+s+1\rangle\}$ and I_0 is an informant for $L_0 = \{\langle 0,x\rangle \mid x \in W_e\} \cup \{\langle 1,k\rangle\}$.

It is easy to verify that $\mathbf{M}$ behaves in the same way on both above informants, and thus fails to **InfIt**-identify at least one of L_0 and L_1, both of which are in $\mathcal{L}$. ∎

Similar generalization idea as in Lemma 15(b) can be used to show that

Theorem 25 $\mathbf{BNCIt} \cap \mathbf{NCIt} - \mathbf{InfIt}^* \neq \emptyset$.

Our next result, together with the above two theorems, shows that **NCIt**-learners are stronger than **InfIt**-learners. Thus, just finite number of negative counterexamples received when the learner attempts to be "overinclusive" can do more than all negative counterexamples! Note that this is not true for **BNCIt**-learners, as follows from Theorem 16. First, we prove a useful technical lemma.

Definition 26 An *initial information segment for L* is an initial information segment of canonical informant for L.

Lemma 27 *Suppose* $\mathbf{M}$ **InfIt**-*identifies L. Then for any initial information segment σ for L, if the following properties (a) to (c) are satisfied, then* $W_{\mathbf{M}(\sigma)} = L$.

(a) For all $x \in L$ such that $(x,1) \notin \text{content}(\sigma)$, for some $\tau \subseteq \sigma$, $\mathbf{M}(\tau \diamond (x,1)) = \mathbf{M}(\tau)$.

(b) For all but finitely many $x \in L$, $\mathbf{M}(\sigma \diamond (x,1)) = \mathbf{M}(\sigma)$,

(c) $\{x \mid (x,0) \notin \text{content}(\sigma)$ *and* $\mathbf{M}(\sigma\diamond(x,0))\downarrow \neq \mathbf{M}(\sigma)\} \subseteq L$.

PROOF. Let $S = \{x \in L \mid \mathbf{M}(\sigma\diamond(x,1)) = \mathbf{M}(\sigma)\}$. Now $L-S$ is finite. Let τ be a sequence formed by inserting elements of $L-S$ such that $(x,1) \notin \text{content}(\sigma)$, in σ at places so that it does not cause a mind change (i.e., $x \in L-S$ such that $(x,1) \notin \text{content}(\sigma)$ is inserted after $\sigma' \subseteq \sigma$, such that $\mathbf{M}(\sigma'\diamond(x,1)) = \mathbf{M}(\sigma')$). Note that for all $x \in L-S$ such that $(x,1) \notin \text{content}(\sigma)$, there exists such a σ' by clause (a). Now consider information sequence $I = \tau I'$, where $\text{content}(I') = \{(x,1) \mid x \in S\} \cup \{x,0) \mid (x,0) \notin \text{content}(\sigma)$ and $x \notin L\}$. It is now easy to verify that $\mathbf{M}(I) = \mathbf{M}(\sigma)$, and I is an information sequence for L. Lemma follows. ∎

Now we show that any **InfIt**-learner can be simulated by a **NCIt**-learner.

Theorem 28 $\mathbf{InfIt} \subseteq \mathbf{NCIt}$.

PROOF. Suppose $\mathbf{M}$ **InfIt**-identifies $\mathcal{L}$. We construct $\mathbf{M}'$ which **NCIt**-identifies $\mathcal{L}$. Given a text T for $L \in \mathcal{L}$, the aim is to construct σ satisfying (a) to (c) of Lemma 27.

Output of $\mathbf{M}'$ on $T[m]$ will be of form $pad(p_m, q_m, R_m, \sigma_m)$. The following invariants will be satisfied for all m.

(A) σ_m is an initial information segment for L. Moreover, $\sigma_m \subseteq \sigma_{m+1}$.

(B) $R_m \subseteq \text{content}(T[m])$, and for all $x \in \text{content}(T[m]) - R_m$, either $(x,1) \in \text{content}(\sigma_m)$ or for some $\tau \subseteq \sigma_m$, $\mathbf{M}(\tau\diamond(x,1)) = \mathbf{M}(\tau)$.

(C) If $q_m = 0$, then p_m is a grammar for the set $\{|\sigma_m|\}$. Note that $|\sigma_m|$ is the least element x such that neither $(x,0)$ nor $(x,1)$ belongs to $\text{content}(\sigma_m)$.

(D) If $q_m = 1$, then p_m is a grammar for $\{x \mid (x,0) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m\diamond(x,0))\downarrow \neq \mathbf{M}(\sigma_m)\}$. In this case, we will have additionally that $R_m = \emptyset$.

(E) If $q_m = 2$, then we have already tested that $\{x \mid (x,0) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m\diamond(x,0))\downarrow \neq \mathbf{M}(\sigma_m)\} \subseteq L$. Additionally, $R_m = \emptyset$. Also in this case, $p_m = \mathbf{M}(\sigma_m)$.

Intuitively, we eventually want to search for σ_m which satisfies Lemma 27. We want to make sure that elements of L satisfy clause (a) in Lemma 27. R_m intuitively denotes the set of elements which may not (and thus need to be taken care of by extending σ_m). Note that we need to remember this set, as iterative learner could lose data. q_m intuitively keeps track of whether we are building up larger and larger σ_m or whether we are checking clause (c) in Lemma 27, or if this checking has already been done.

Initially on input Λ, $\mathbf{M}'$ outputs $(p, 1, \emptyset, \Lambda)$, where p is a grammar for $\{x \mid \mathbf{M}((x,0)) \neq \mathbf{M}(\Lambda)\}$. Clearly, invariants (A) to (E) are satisfied.

Now $\mathbf{M}'$ on input $x = T(m)$, counterexample y (on conjecture of $\mathbf{M}'$ on $T[m]$) with previous conjecture being $pad(p_m, q_m, R_m, \sigma_m)$, outputs $pad(p_{m+1}, q_{m+1}, R_{m+1}, \sigma_{m+1})$ where the parameters $p_{m+1}, q_{m+1}, R_{m+1}, \sigma_{m+1}$ are defined as follows.

Below note that, for $L \in \mathcal{L}$, and T being a text for L, $\mathbf{M}$ would always be defined on input being $\sigma_m \diamond (x, 1)$, where σ_m is an initial information segment for L, $x \in \text{content}(T)$. Thus, we will not explicitly check for convergence of $\mathbf{M}$ on such inputs, but assume that it converges.

Case 1: $q_m = 0$.

Let $\sigma_{m+1} = \sigma_m \diamond (|\sigma_m|, w)$, where w is 1 or 0 based on whether counterexample is # or a numerical value. Note that p_m was a grammar for $\{|\sigma_m|\}$.

Let $R_{m+1} = (R_m \cup \{x\}) - (\{\#\} \cup \{x' \mid (x', 1) \in \text{content}(\sigma_{m+1})\} \cup \{x' \mid \mathbf{M}(\sigma_{m+1} \diamond (x', 1)) = \mathbf{M}(\sigma_{m+1})\})$.

If R_{m+1} is $\emptyset$, then let $q_{m+1} = 1$ and p_{m+1} be a grammar for $\{x' \mid (x', 0) \notin \text{content}(\sigma_{m+1})$ and $\mathbf{M}(\sigma_{m+1} \diamond (x', 0))\downarrow \neq \mathbf{M}(\sigma_{m+1})\}$. Else, let $q_{m+1} = 0$ and p_{m+1} be a grammar for $\{|\sigma_{m+1}|\}$.

Invariants (A), (C), (D) and (E) are easily seen to be satisfied. To see that invariant (B) is satisfied, note that by induction all $z \in \text{content}(T[m]) - R_m$, satisfied $[(z, 1) \in \text{content}(\sigma_m)$ or for some $\tau \subseteq \sigma_m$, $\mathbf{M}(\tau \diamond (z, 1)) = \mathbf{M}(\tau)]$. On the other hand if $z = T(m+1) = x, z \neq \#$ or if $z \in R_m$, then z is missing from R_{m+1} iff $(z, 1) \in \text{content}(\sigma_m)$ or $\mathbf{M}(\sigma_{m+1} \diamond (z, 1)) = \mathbf{M}(\sigma_{m+1})$. Thus, (B) is satisfied.

Case 2: $q_m = 1$.

Let $\sigma_{m+1} = \sigma_m$.

If there was a counterexample (i.e., $y \neq \#$), or $[x \neq \#$ and $(x, 1) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m \diamond (x, 1)) \neq \mathbf{M}(\sigma_m)]$, then let $R_{m+1} = \{x\} - \{\#\}$, $q_{m+1} = 0$, and p_{m+1} be a grammar for $\{|\sigma_{m+1}|\}$.

Else (i.e, $y = \#$, and $[x = \#$ or $(x, 1) \in \text{content}(\sigma_m)$ or $\mathbf{M}(\sigma_m \diamond (x, 1)) = \mathbf{M}(\sigma_m)]$), then let $R_{m+1} = \emptyset$, $q_{m+1} = 2$, and $p_{m+1} = \mathbf{M}(\sigma_m)$.

Invariants (A), (C), (D) and (E) are easily seen to be satisfied. To see that invariant (B) is satisfied, note that by induction all $z \in \text{content}(T[m])$, satisfied $(z, 1) \in \text{content}(\sigma_m)$ or for some $\tau \subseteq \sigma_m$, $\mathbf{M}(\tau \diamond (z, 1)) = \mathbf{M}(\tau)$. Also, $T(m+1) = x$ is placed in R_{m+1} if $(x \neq \#$ and $(x, 1) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m \diamond (x, 1)) \neq \mathbf{M}(\sigma_m))$. Thus invariant (B) is also satisfied.

Case 3: $q_m = 2$.

Let $\sigma_{m+1} = \sigma_m$.

If $x \neq \#$ and $(x,1) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m \diamond (x,1)) \neq \mathbf{M}(\sigma_m)$, then let $R_{m+1} = \{x\}$, $q_{m+1} = 0$ and p_{m+1} be a grammar for $\{|\sigma_{m+1}|\}$.

Else, let $R_{m+1} = \emptyset$, $q_{m+1} = 2$, and $p_{m+1} = \mathbf{M}(\sigma_m)$.

Invariants (A), (C), (D) are easily seen to be satisfied. If $q_{m+1} = 2$, then (E) also remains satisfied since q_m was also 2. To see that invariant (B) is satisfied, note that by induction all $z \in \text{content}(T[m])$ satisfied $(z,1) \in \text{content}(\sigma_m)$ or for some $\tau \subseteq \sigma_m$, $\mathbf{M}(\tau \diamond (z,1)) = \mathbf{M}(\tau)$. Also, $T(m+1) = x$ is placed in R_{m+1} if ($x \neq \#$ and $(x,1) \notin \text{content}(\sigma_m)$ and $\mathbf{M}(\sigma_m \diamond (x,1)) \neq \mathbf{M}(\sigma_m)$). Thus invariant (B) is also satisfied.

Thus, invariants are satisfied in all cases. Moreover, $\lim_{m\to\infty} \sigma_m$ converges, as for large enough initial information segment σ_m for L, $\mathbf{M}(\sigma_m \diamond (x, \chi_L(x)) = \mathbf{M}(\sigma_m)$, for $(x, \chi_L(x)) \notin \text{content}(\sigma_m)$.

Also, it is easy to verify that if $q_m = 0$, then $\sigma_m \subset \sigma_{m+1}$. Thus, for all but finitely many m, $q_m \neq 0$. Also, if $q_m = 1$ or 2, then either $q_{m+1} = 2$ or $q_{m+1} = 0$. It follows that $\lim_{m\to\infty} q_m = 2$. Thus, by property (E), $\lim_{m\to\infty} R_m = \emptyset$. Hence, $\mathbf{M}'$ stabilizes to a conjecture of form $(p, 2, \emptyset, \sigma)$, for some initial information segment σ for L — this σ satisfies (a) — (c) in Lemma 27, as otherwise Case 3 (along with properties (B) and (E)) would eventually ensure change of q_m, and the conjecture. Thus, $\mathbf{M}'$ identifies L, as it converges to padded version of grammar $\mathbf{M}(\sigma)$. ∎

Proof of the above theorem also shows,

Theorem 29 *For all $a \in N \cup \{*\}$, $\mathbf{InfIt}^a \subseteq \mathbf{NCIt}^a$.*

We already established that learners from full positive data with indefinitely growing long-term memory (**TxtEx**) can sometimes learn more than any **NCIt**-learner (Theorem 22). Now we will show this difference on yet another level. It can be easily demonstrated that adding a recursive language to a **TxtEx**-learnable class does not preserve its **TxtEx**-learnability (see, for example, [Gol67]). Our next result shows that adding one recursive language to a class in **NCIt**, still leaves it in **NCIt**. (Note that the same result was obtained in [JK04] for **NCEx**-learners, however, the algorithm witnessing the simulation there was nearly trivial — unlike our simulation in the proof below).

We begin with a useful remark describing a way how an **NCIt** learning machine, being fed a text T, can simulate another **NCIt** machine working on an effectively modified input text T'. This simulation will be utilized in the proof of our next theorem.

Remark 30 In the following Theorem 31, when we say that some machine $\mathbf{M}'$ simulates $\mathbf{M}$ on text T' — where T' is formed from the remain-

ing input text T'' (to be received by $\mathbf{M}'$) in a nice way (that is $T' = \beta\alpha(T''(0))\alpha(T''(1))\alpha(T''(2))\ldots$ for some effective mapping α) — the simulation is done as follows.

Intuitively in the simulation, before seeing the input $T''(s)$, the last conjecture of $\mathbf{M}'$ would have been $pad(p_s, \text{ncex}_s, \tau_s)$, where

(a) $\beta\alpha(T(0))\alpha(T(1))\ldots\alpha(T(s-1)) = \tau'_s\tau_s$, for some τ'_s. Intuitively, we have already simulated $\mathbf{M}$ on τ'_s, and we have a backlog of τ_s.

(b) ncex_s denotes a function mapping some conjectures to counterexamples for them (here counterexamples are with respect to the input language L). $\mathbf{M}'$ would have obtained these counterexamples by conjecturing some padded version of these conjectures. It will be the case that all conjectures of $\mathbf{M}$ on proper prefixes of τ'_s (as defined in (a) above) are in the domain of ncex_s.

(c) p_s is the last conjecture of $\mathbf{M}$ on input τ'_s (as defined in (a) above), when the counterexamples provided in the simulation are via the function ncex_s.

(d) Furthermore, we will have the property that $\tau'_s \subset \tau'_{s+1}$ (and thus, τ_{s+1} is a proper suffix of $\tau_s\alpha(T''(s))$).

Intuitively, conjecture $pad(p_s, \text{ncex}_s, \tau_s)$ denoted that τ_s is the backlog in the simulation, and ncex_s is the mapping of conjectures to counterexamples (as known to $\mathbf{M}'$).

Initially, these conditions can be satisfied by having $\text{ncex}_0 = \emptyset$, p_0 being conjecture of $\mathbf{M}$ on empty sequence, and $\tau_0 = \beta$.

Suppose $\mathbf{M}'$ had output $pad(p_s, \text{ncex}_s, \tau_s)$ after having read $T''[s]$. We now describe how $\mathbf{M}'$ outputs after reading $T''(s)$ and counterexample y (to its conjecture $pad(p_s, \text{ncex}_s, \tau_s)$). $\mathbf{M}'$ will output $pad(p_{s+1}, \text{ncex}_{s+1}, \tau_{s+1})$, where $p_{s+1}, \text{ncex}_{s+1}$ and τ_{s+1} are described as follows.

If ncex_s is already defined on p_s, then $\text{ncex}_{s+1} = \text{ncex}_s$; otherwise ncex_{s+1} extends ncex_s by defining $\text{ncex}_{s+1}(p_s) = y$.

Let γ be the largest initial segment of $\tau_s\alpha(T''(s))$ such that $\mathbf{M}^{\text{ncex}_{s+1}}(p_s, \gamma)$ is defined. Here note that, since $\mathbf{M}$ is defined on all inputs consistent with some language in the class, as long as the input language is from the class, the only reason for $\mathbf{M}^{\text{ncex}_{s+1}}(p_s, \gamma')$ to be undefined is that for some intermediate conjecture p, $\text{ncex}_{s+1}(p)$ is undefined. Let $p_{s+1} = \mathbf{M}(p_s, \gamma)$. Let τ_{s+1} be such that $\tau_s\alpha(T''(s)) = \gamma\tau_{s+1}$. (Note that γ is non-empty, thus $\tau'_{s+1} = \tau'_s\gamma$ is a proper extension of τ'_s). Intuitively, γ above represents the largest initial segment of $\tau_s\alpha(T''(s))$ such that $\mathbf{M}'$ can simulate $\mathbf{M}$ on γ, as it knows the value of counterexamples for each of the intermediate conjectures, if any. γ thus rep-

resents the whole of $\tau_s\alpha(T''(s))$, if $\mathbf{M}'$ knows (via ncex) the counterexamples (if any) for all the intermediate conjectures (in this case τ_{s+1} becomes Λ). On the other hand if $\mathbf{M}'$ does not know the counterexample for some intermediate conjecture, then γ represents this initial part, where we need to output the conjecture of $\mathbf{M}$ to learn the counterexample value.

This completes the description of simulation.

We now argue that if $\mathbf{M}$ **NCIt**-identifies L, and T' is a text for L, and $\mathbf{M}'$ gets counterexamples based on input language being L, then $\mathbf{M}'$ in the above simulation will converge to a grammar of form $pad(p, \cdot, \cdot)$, where p is a grammar for L.

To see this, suppose the input language to $\mathbf{M}'$ is L, T' is a text for L, and $\mathbf{M}$ **NCIt**-identifies L. Let ncex$'$ denote the function such that ncex$'(p)$ is the counterexample (or #) which $\mathbf{M}'$ receives in the above simulation when it first outputs (if ever) a conjecture of the form $pad(p, \cdot, \cdot)$. Then, it is easy to verify that the simulation of $\mathbf{M}$ in the above construction uses the counterexamples based on ncex$'$. Furthermore, the output of $\mathbf{M}'$ after seeing $T''[s]$ is of form $pad(\mathbf{M}(\tau'_s), \text{ncex}_s, \tau_s)$, where ncex$_s$ is restriction of ncex$'$ to the domain being set of conjectures of $\mathbf{M}$ on proper prefixes of τ'_s. Suppose $\mathbf{M}$ on T' converges to a grammar p (when counterexamples are provided according to ncex$'$). Suppose n is large enough such that, for all $n' \geq n$, $\mathbf{M}$ on $T'[n']$ outputs p (when counterexamples are provided according to ncex$'$). Thus, once τ'_s extends $T'[n+1]$, we will have that $\tau_s = \Lambda$, (as ncex$_{s+1}$ will then contain all the counterexamples needed for the simulation of $\mathbf{M}$). This will happen at or before the time when $s = n$, since τ'_s form monotonically increasing sequence of initial segments (see property (d) above). It follows that the sequence of conjectures of $\mathbf{M}'$ on T'' converges to $pad(p, \text{ncex}, \emptyset)$, where ncex is restriction of ncex$'$ to the domain being the set of conjectures made by $\mathbf{M}$ on T' (when counterexamples are provided according to ncex$'$).

Theorem 31 *If $\mathcal{L} \in$* **NCIt** *and X is recursive, then $\mathcal{L} \cup \{X\} \in$* **NCIt**.

Proof. Suppose $\mathbf{M}$ **NCIt**-learns $\mathcal{L}$. Then $\mathbf{M}'$ first checks (on input Λ) if X is a subset of the input — if not then $\mathbf{M}'$ simulates $\mathbf{M}$ on $T' = aT''$, where a is the first element of the input text T, and T'' is the remaining text to be seen. If X is a subset of the input language, then the behaviour of $\mathbf{M}'$ depends on the following cases.

Case 1: X is finite.

$\mathbf{M}'$ conjectures a (standard) grammar for X until a non-element a of X is seen. If such a non-element is never seen, then $\mathbf{M}'$ keeps on outputting the (standard) grammar for X. Otherwise $\mathbf{M}'$ simulates $\mathbf{M}$ on $T' = a_0a_1 \ldots a_kaT''$, where $a_0, a_1, \ldots a_k$ are elements of X, a is the first non-element of X observed

in the input data, and T'' is the remaining text beyond a. As T' is a text for input language, we get **NCIt**-identification of the input language by $\mathbf{M}'$ based on Remark 30.

Case 2: X is infinite.

As in Case 1, except that in the case of seeing a non-element a of X, $\mathbf{M}'$ simulates $\mathbf{M}$ on the text $T' = a\alpha(T''(0))\alpha(T''(1))\ldots$ (here $T''(x)$ are elements of the remaining input text, as in Remark 30), where $\alpha(w)$ is an element w followed by elements of X which are $\leq w$ (note that, since X is infinite, this way the learner M' receives growing segments of the language X after getting every new element $T''(x)$ of the input — thus being compensated for possible loss of positive data from intersection of input language with X on earlier stages of learning).

As T' is a text for input language, we get **NCIt**-identification of the input language by $\mathbf{M}'$ based on Remark 30. ∎

Note that the above result cannot be extended to r.e. X. For all r.e., but non-recursive sets A, $\{A \cup \{x\} \mid x \notin A\}$ can be shown to be in **NCIt**. However [JK04] showed that, for r.e. but non-recursive A, $\{A\} \cup \{A \cup \{x\} \mid x \notin A\}$ is not in **LNCEx**.

6 Delays

In this section we consider a situation when the teacher provides a counterexample (if any) to a **NCIt**-learner with possible delay. There are few different ways how a possibility of delays can be formalized within the framework of our model (see relevant discussion for **NCEx**-learners in [JK05]). The "worst" case of delay can happen when a learner converges to a hypothesis that enumerates a non-subset of the target language and eventually receives a negative counterexample (in particular, some previous conjecture thus might never receive a counterexample). Another possible interpretation: every non-subset conjecture eventually gets a counterexample — we will adhere to this interpretation below. In order to avoid unnecessary formal details, we present the problem of delays for **NCIt**-learners as an informal remark.

Remark 32 Delays do matter for **NCIt**. To see this, consider $\mathcal{L} = \{N\} \cup \{D \mid D \text{ is finite }\}$. Then it is easy to see that $\mathcal{L} \in$ **NCIt**. However, $\mathcal{L}$ is not in **NCIt** with delays. To see this, suppose $\mathbf{M}$ does learning with delays. Then consider a **NCIt**-locking sequence σ when the input is N and no counterexamples are provided. Now consider the text $T = \sigma\#\#^\infty$, for content(σ), where the counterexamples are all delayed beyond the $\sigma\#$ part. Let the set of counterex-

amples provided be S. Since $\mathbf{M}$ needs to converge on T, S is finite. Note that for all conjectures p made by $\mathbf{M}$ on T, either $W_p \subseteq \text{content}(\sigma)$, or $W_p \cap S \neq \emptyset$. Now consider some element x which is not in $S \cup \text{content}(\sigma)$. Then, behaviour of M is same on $T = \sigma\#\#^\infty$ and $T' = \sigma x\#^\infty$, if on T' one provides the same sequence of counterexamples as done for T. Note that the sequence of counterexamples on T' would be valid, as for all conjectures p made by $\mathbf{M}$ on T (T'), either $W_p \subseteq \text{content}(\sigma)$, or $W_p \cap S \neq \emptyset$. Thus, $\mathbf{M}$ fails to **NCIt**-identify with delays at least one of the languages $\text{content}(\sigma)$ and $\text{content}(\sigma) \cup \{x\}$, both of which are in $\mathcal{L}$.

7 Results Related to Indexed Families

In this section we consider **NCIt**-learning for indexed classes of recursive languages — one of the popular learning tasks (as it was mentioned in the Introduction, such popular subjects of learning as *patterns* and *regular languages* are examples of indexed classes). Note that these classes are often not learnable if only (full) positive and no negative data is available even if a learner can potentially hold all input in the long-term memory, as was established yet by Gold ([Gol67]). Note also that there exist indexed families which are not in $\mathbf{BNCBc}^*$ (see [JK04]). Our main result in this section is that all such classes are **NCIt**-learnable.

Theorem 33 *Every indexed family $\mathcal{L}$ is in* **NCIt**.

Proof. Suppose $L_0, L_1, \ldots$ is recursive indexing of $\mathcal{L}$ such that one can effectively decide from x and i whether $x \in L_i$. Let P be a recursive function such that $W_{P(j,S,X)} = L_j \cup X \cup \bigcup_{i \in S} L_i$. We define a learner $\mathbf{M}$ which **NCIt**-identifies $\mathcal{L}$. Conjectures of $\mathbf{M}$ will be of form $P(j, S, X)$, for some finite sets S, X. On input (T, T'), where $L = \text{content}(T)$, $\mathbf{M}$ is defined as follows. Suppose $\mathbf{M}(T[n], T'[n]) = P(j_n, S_n, X_n)$, then by induction we will have the following properties: (i) $j_n \leq j_{n+1}$, $S_n \subseteq S_{n+1}$, and $X_n \subseteq X_{n+1}$, (ii) $S_n \subseteq \{i \mid i \leq j_n\}$, (iii) $X_{n+1} \neq X_n$ iff $j_{n+1} \neq j_n$, (iv) for all $i < j_n$, $L_i \neq L$, (v) $X_n \cup \bigcup_{i \in S_n} L_i \subseteq L$, (vi) $\text{content}(T[n]) \subseteq X_n \cup \bigcup_{i \in S_n} L_i$, (vii) if $j_n \notin S_n$, then ($n = 0$ or $j_{n-1} = j_n - 1$).

Initially let $\mathbf{M}(\Lambda, \Lambda) = P(0, \emptyset, \emptyset)$. $\mathbf{M}(T[n+1], T'[n+1]) = P(j_{n+1}, S_{n+1}, X_{n+1})$, is defined as follows.

If $T'(n) \neq \#$, then let $j_{n+1} = j_n + 1$, $S_{n+1} = S_n$, $X_{n+1} = X_n \cup \{T(n)\}$. ElseIf $T'(n) = \#$ and $T(n) \notin L_{j_n}$, then let $j_{n+1} = j_n + 1$, $S_{n+1} = S_n \cup \{j_n\}$, $X_{n+1} = X_n \cup \{T(n)\}$. Else (i.e., $T'(n) = \#$ and $T(n) \in L_{j_n}$), then let $j_{n+1} = j_n$, $S_{n+1} = S_n \cup \{j_n\}$, $X_{n+1} = X_n$.

It is easy to verify that induction hypotheses are satisfied. Now suppose $L \in \mathcal{L}$. By (iv) we have that $\lim_{n\to\infty} j_n$ converges, and thus by (i), (ii) and (iii), we have that $\lim_{n\to\infty} S_n$ and $\lim_{n\to\infty} X_n$ also converge. Let $(j, S, X) = \lim_{n\to\infty}(j_n, S_n, X_n)$. By (vii), we have that $j \in S$. By (v) and (vi) we have that $P(j, S, X)$ is a grammar for L. ∎

Here note that we needed to carry along the set S_n in the above construction, as one may otherwise lose data which are included in L_{j_n}, when $T(n) \in L_{j_n} \subseteq L$ (carrying all inputs $T(n)$ instead would require indefinitely growing long-term memory).

Complexity of the algorithm witnessing the Theorem above is underscored by the following result, showing that **NCIt**-learning of some indexed classes becomes impossible if a learner wants to use grammars describing just the languages from the target class as its hypotheses space (so-called *class-preserving* learnability, see [ZL95]). For class-preserving learnability, instead of using $W_0, W_1, \ldots$ as hypothesis space for interpreting the conjectures of the learner (for example, see Definition 5), one uses a hypothesis space $H_0, H_1, \ldots$ such that (a) $\{H_i \mid i \in N\} = \mathcal{L}$, and (b) for all i, x, one can effectively decide in i and x whether $x \in H_i$. Thus, one could consider $H_0, H_1, \ldots$ to be represented by a numbering $\varphi_i(\cdot,\cdot)$, such that $\varphi_i(j, x) = 1$ iff $x \in H_j$.

Let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ denote a recursive enumeration of all iterative learners.

Theorem 34 *There exists an indexed family $\mathcal{L}$ such that for all e, i,*

(a) $\varphi_i \notin \mathcal{R}^2_{0,1}$ or

(b) $\{\{x \mid \varphi_i(j, x) = 1\} \mid j \in N\} \neq \mathcal{L}$ or

(c) $\mathbf{M}_e$ does not **NCIt***-identify $\mathcal{L}$ using hypothesis space $(\lambda x.\varphi_i(j, x))_{j\in N}$.*

Proof. By s-m-n theorem [Rog67], there exists a recursive p such that $\varphi_{p(e,i,s)}$ may be defined as follows. We will have the property that if $\varphi_{p(e,i,s)}$ is non-empty, then it will be total. We will take $\mathcal{L}$ to be $\{\varphi^{-1}_{p(e,i,s)}(1) \mid \varphi_{p(e,i,s)} \neq \emptyset\}$. It follows that $\mathcal{L}$ is an indexed family of recursive languages. The aim of functions $\varphi_{p(e,i,\cdot)}$ is to diagonalize against $\mathbf{M}_e$ being **NCIt** learner for $\mathcal{L}$ using hypothesis space $(\lambda x.\varphi_i(j, x))_{j\in N}$.

Below is the description of $\varphi_{p(e,i,\cdot)}$. Initially, let $\varphi_{p(e,i,0)}(x) = 0$, for $x \notin \{\langle e, i, y\rangle \mid y \in N\}$, Let $\sigma_0, \sigma'_0 = \Lambda$, and $x_s = 0$.

The following invariants will be satisfied, for all s, at the beginning of stage s.

(A) $\varphi^{-1}_{p(e,i,j)}(1) \subseteq \{\langle e, i, x\rangle \mid x \in N\}$.

(B) $\varphi_{p(e,i,0)}(x)$ has been defined (at the start of stage s) for $x \in \{\langle e', i', y\rangle \mid (e', i') \neq (e, i)\} \cup \{\langle e, i, y\rangle \mid y < x_s\}$.

(C) $\varphi_{p(e,i,j)}$, $0 < j \leq s$, have been defined on all inputs, and for each of these j, for some x, $\varphi_{p(e,i,j)}(x) = 1$, $\varphi_{p(e,i,0)} = 0$.

(D) $\varphi_{p(e,i,j)}$, has not been defined on any input for $j > s$.

(E) content$(\sigma_s) = \{\langle e, i, x\rangle \mid \varphi_{p(e,i,0)}(\langle e, i, x\rangle) = 1, x < x_s\}$.

(F) For $r < |\sigma_s|$, if $\sigma'_s(r) \neq \#$, then there exists an x such that $\varphi_i(\mathbf{M}_e(\sigma_s[r], \sigma'_s[r]), x)\downarrow = 1$, and $\varphi_{p(e,i,0)}(x) = 0$.

(G) For $r < |\sigma_s|$, if $\sigma'_s(r) = \#$, then (i) for e', i', j' such that $(e', i') \neq (e, i)$, $\varphi_i(\mathbf{M}_e(\sigma_s[r], \sigma'_s[r]), \langle e, i, 0\rangle)\downarrow \neq \varphi_{p(e',i',j')}(\langle e, i, 0\rangle)$, and (ii) for all j, $0 < j \leq s$, there exists an $x \leq \langle e, i, x_s - 1\rangle$ such that $\varphi_i(\mathbf{M}_e(\sigma_s[r], \sigma'_s[r]), x)\downarrow \neq \varphi_{p(e,i,j)}(x)$ (Thus, the only possible language in $\mathcal{L}$, as of now, that could be consistent with $\varphi_i(\mathbf{M}_e(\sigma_s[r], \sigma'_s[r]), \cdot)$, is $\varphi^{-1}_{p(e,i,0)}(1)$).

Intuition behind the stage s is as follows: Initially we place an element a in a basic set A ($\varphi^{-1}_{p(e,i,0)}(1)$ in out context) and b outside A, and search for a segment σ containing (past data plus) a such that $\mathbf{M}_e$ on σ and $\sigma\#$ outputs the same conjecture, which is consistent with A, (in particular it has a but not b) (see steps 1–2). Furthermore, all conjectures output on prefixes of σ, which contain a either do not contain b or contain another element $< b$, which does not belong to A. At that point we add a set B ($\varphi^{-1}_{p(e,i,s+1)}(1)$ in our context) to $\mathcal{L}$, where B contains both a and b (along with past data), and search for an extension τ of σ containing both a and b such that $\mathbf{M}_e$ on τ and $\tau\#$ outputs the same conjecture containing both a and b (see steps 3–4). Once τ is found, we place an element c in A, and check if on σc, $\mathbf{M}_e$ makes a mind change (see step 5). If so, then we can continue to next stage (as we have been able to force one more mind change on basic set A). If no mind change was found, then, we check if any conjecture of $\mathbf{M}_e$ on τ contains both b and c (see step 6), if so, then $\mathbf{M}_e$ has produced a conjecture not in class. Otherwise, we add a set C containing a, b, c (along with past data), to the class $\mathcal{L}$: now C is not identified by $\mathbf{M}_e$ due to it being iterative learner (as we could take a text for C which is modification of τ with c inserted right after σ). We now proceed with the details.

Go to stage 0.

Stage s

1. Let $\varphi_{p(e,i,0)}(\langle e, i, x_s\rangle) = 1$.
2. For $z = x_s + 1$ to ∞ do
   2.1. Let $\varphi_{p(e,i,0)}(\langle e, i, z\rangle) = 0$.

(* Below we try to search for a "seeming" stabilizing sequence (σ, σ'), on $\varphi^{-1}_{p(e,i,0)}(1)$ which has some additional properties (see (d) and (g) below). *)

2.2. For each y_s satisfying (a), search for z steps for a σ extending σ_s and a σ' extending σ'_s such that (b) — (g) are satisfied.

(a) $x_s < y_s \leq z$,

(b) $|\sigma| = |\sigma'|$,

(c) $\text{content}(\sigma) = \text{content}(\sigma_s) \cup \{\langle e, i, x_s\rangle\}$,

(d) $\mathbf{M}_e(\sigma[r], \sigma'[r])\downarrow$ and $\varphi_i(\mathbf{M}_e(\sigma[r], \sigma'[r]), x)\downarrow$, for all $x \leq \langle e, i, y_s\rangle$, for all $r \leq |\sigma|$,

(e) $\mathbf{M}_e(\sigma\#, \sigma'\#)\downarrow = \mathbf{M}_e(\sigma, \sigma')$,

(f) for all $x \leq \langle e, i, y_s\rangle$, $\varphi_i(\mathbf{M}_e(\sigma, \sigma'), x)\downarrow = \varphi_{p(e,i,0)}(x)$,

(g) for all $r < |\sigma|$, either:

(i) $\sigma'(r) = \#$ and for all $x \leq \langle e, i, y_s\rangle$, $\varphi_i(\mathbf{M}_e(\sigma[r], \sigma'[r]), x)\downarrow = \varphi_{p((e,i,0)}(x)$ or

(ii) $\# \neq \sigma'(r) \leq \langle e, i, y_s - 1\rangle$, and $\varphi_i(\mathbf{M}_e(\sigma[r], \sigma'[r]), \sigma'(r)) = 1$ and $\varphi_{p(e,i,0)}(\sigma'(r)) = 0$.

2.3. If such a σ, σ' (with corresponding y_s) are found, go to step 3.

EndFor

3. Let $\varphi_{p(e,i,s+1)}(\langle e, i, x\rangle) = \varphi_{p(e,i,0)}(\langle e, i, x\rangle)$, for $x \leq z$, $x \neq y_s$.
Let $\varphi_{p(e,i,s+1)}(x) = 0$, for $x \notin \{\langle e, i, y\rangle \mid y \in N\}$.
Let $\varphi_{p(e,i,s+1)}(\langle e, i, y_s\rangle) = 1$.

4. For $w = z + 1$ to ∞ do

4.1. Let $\varphi_{p(e,i,0)}(\langle e, i, w\rangle) = \varphi_{p(e,i,s+1)}(\langle e, i, w\rangle) = 0$.
(* Below we try to search for a "seeming" stabilizing sequence (σ, σ'), on $\varphi^{-1}_{p(e,i,s+1)}(1)$ which has some additional properties (see (d) and (g) below). *)

4.2. For each t_s satisfying (a), search for w steps for a τ extending σ and τ' extending σ' such that (b) — (g) are satisfied.

(a) $z < t_s \leq w$,

(b) $|\tau| = |\tau'|$,

(c) $\text{content}(\tau) = \text{content}(\sigma) \cup \{\langle e, i, y_s\rangle\}$.

(d) $\mathbf{M}_e(\tau[r], \tau'[r])\downarrow$ and $\varphi_i(\mathbf{M}_e(\tau[r], \tau'[r]), x)\downarrow$, for all $x \leq \langle e, i, t_s\rangle$, for all $r \leq |\tau|$,

(e) $\mathbf{M}_e(\tau\#, \tau'\#)\downarrow = \mathbf{M}_e(\tau, \tau')$,

(f) for all $x \leq \langle e, i, t_s\rangle$, $\varphi_i(\mathbf{M}_e(\tau, \tau'), x)\downarrow = \varphi_{p(e,i,s+1)}(x)$,

(g) for all $r < |\tau|$, either:

(i) $\tau'(r) = \#$ and for all $x \leq \langle e, i, t_s\rangle$ such that $x \neq \langle e, i, y_s\rangle$, $\varphi_i(\mathbf{M}_e(\sigma[r], \sigma'[r]), x) = \varphi_{p(e,i,0)}(x)$ or

(ii) $\# \neq \tau'(r) \leq \langle e, i, t_s\rangle$, $\varphi_i(\mathbf{M}_e(\tau[r], \tau'[r]), \tau'(r)) = 1$ and $\varphi_{p(e,i,s+1)}(\tau'(r)) = 0$.

4.3. If such a τ, τ' (with corresponding t_s) are found, go to step 5.

EndFor

5. Define $\varphi_{p(e,i,0)}(\langle e, i, w + 1\rangle) = 1$.

Define $\varphi_{p(e,i,s+1)}(\langle e,i,x\rangle) = 0$, for all $x > w$.
For $w' = w+2$ to ∞ do
Let $\varphi_{p(e,i,0)}(\langle e,i,w'\rangle) = 0$.
If $\mathbf{M}_e(\sigma\diamond\langle e,i,w+1\rangle, \sigma'\#)$ does not converges within w' steps, then go to next iteration of For loop.
Else if $\mathbf{M}_e(\sigma\diamond\langle e,i,w+1\rangle, \sigma'\#)\downarrow \neq \mathbf{M}_e(\sigma,\sigma')$, then go to stage $s+1$, with $x_{s+1} = w'+1$, $\sigma_{s+1} = \sigma\diamond\langle e,i,w+1\rangle$, $\sigma'_{s+1} = \sigma'\#$.
(* We have found a mind change. *)
Else go to step 6.
EndFor
6.
Define $\varphi_{p(e,i,0)}(\langle e,i,x\rangle) = 0$, for $x > w'+1$.
If, for all $r \in \{|\tau|\} \cup \{r' \mid \tau'(r') = \#\}$, $\varphi_i(\mathbf{M}(\tau[r],\tau'[r]),\langle e,i,w+1\rangle) = 1$ iff $\varphi_i(\mathbf{M}(\tau[r],\tau'[r]),\langle e,i,y_s\rangle) = 0$,
Then let $\varphi_{p(e,i,s+2)}(y_s) = 1$, and $\varphi_{p(e,i,s+2)}(x) = \varphi_{p(e,i,0)}(x)$, for $x \neq y_s$.
Quit (we have done the diagonalization, and no need to go to stage $s+1$).
End stage s.

It is easy to verify that that the invariants (A), (B), (C), (D), (E) are satisfied. Invariant (F) is satisfied as σ'_{s+1} is then defined via step 5, and in step 2.2, this property is explicitly ensured when defining σ'. Invariant (G) holds, using the fact that whenever $\sigma'(r) = \#$ or $r = |\sigma|$, in step 2.2, we have verified that $\varphi_i(\mathbf{M}_e(\sigma[r],\sigma'[r]),x) = \varphi_{p(e,i,0)}(x)$, for $x \leq \langle e,i,y_s\rangle$, and $\varphi_{p(e,i,j)}$, $0 < j \leq s$, are inconsistent with $\varphi_{p(e,i,0)}[\langle e,i,y_s\rangle + 1]$ by induction (see invariant (C)), and $\varphi_{p(e,i,s+1)}(\langle e,i,y_s\rangle) \neq \varphi_{p(e,i,0)}(\langle e,i,y_s\rangle)$, by definition in step 3.

Thus, all invariants hold. Also, it is easy to verify that all $\varphi_{p(e,i,j)}$ are either total, or empty. ($\varphi_{p(e,i,0)}$ is made total by having infinitely many stages, or at step 2, 4, 5 or 6, if stage s does not end or finishes with a Quit. $\varphi_{p(e,i,s+1)}$ is made total in stage s itself (if stage s is executed, either at step 4 or at step 5). The only remaining case is, $\varphi_{p(e,i,s+2)}$ being started in step 6 of stage s, but then it is clearly made total in step 6.

Suppose $\varphi_i \in \mathcal{R}_{0,1}$, and $\{\{x \mid \varphi_i(j,x) = 1\} \mid j \in N\} = \mathcal{L}$. We now consider the following cases.

Case 1: There are infinitely many stages.

Consider $T = \bigcup_{s\in N} \sigma_s$ and $T' = \bigcup_{s\in N} \sigma'_s$. Thus, T is a text for $\varphi^{-1}_{p(e,i,0)}(1)$, and T' is valid sequence of counterexamples for $\mathbf{M}_e$ when provided with T as input text (by invariant (F) and (G), since $\varphi^{-1}_{p(e,i,0)}(1)$ is the only language in $\mathcal{L}$ which could be consistent with the hypothesis $\mathbf{M}(T[r],T'[r])$ (in numbering $(\varphi_i(j,\cdot))_{j\in N}$), whenever $T'(r) = \#$). However, $\mathbf{M}_e(T,T')$ makes infinitely many mind changes (see step 5, which is the only step which changes stage).

Case 2: Stage s starts but does not end.

In case step 2 does not succeed, consider $L = \varphi^{-1}_{p(e,i,0)}(1)$. But then, we have that there is no **NCIt**-locking sequence (extending (σ_s, σ'_s)) for $\mathbf{M}_e$ on L, (here note that σ'_s is valid sequence of counterexamples using invariants (F) and (G), and the fact that no $\varphi_{p(e,i,j)}$, with $j > s$, is defined on any input.) Thus, $\mathbf{M}_e$ does not **NCIt**-identify $\mathcal{L}$.

Similarly, in case step 4 is started but does not finish then $\mathbf{M}_e$ does not have a **NCIt**-locking sequence for $L = \varphi^{-1}_{p(e,i,s+1)}(1)$.

If step 5 does not finish, then $\mathbf{M}_e$ is not defined on $(\sigma\diamond\langle e, i, w+1\rangle, \sigma'\#)$, a valid input for $\varphi^{-1}_{p(e,i,0)}(1)$ (as there are infinitely many iterations of for loop at step 5, and answers given by σ' are correct due to check at 2.2 (f), 2.2 (g) — due to which only $\varphi^{-1}_{p(e,i,0)}(1)$ in $\mathcal{L}$ is consistent with $\varphi_i(\mathbf{M}_e(\sigma[r], \sigma'[r]), \cdot)$, for $r \in \{|\sigma|\} \cup \{r' \mid \sigma'(r') = \#\}$).

Now suppose the construction reaches step 6. Let γ, γ' be such that $\sigma\gamma = \tau$ and $\sigma'\gamma' = \tau'$. Now, we have that $\mathbf{M}_e(\sigma\diamond\langle e, i, w+1\rangle, \sigma'\diamond\#) = \mathbf{M}_e(\sigma, \sigma')$, and $\mathbf{M}_e(\tau\#, \tau'\#) = \mathbf{M}_e(\tau, \tau')$. Thus, since $\mathbf{M}_e$ is iterative learner, we have $\mathbf{M}_e(\sigma\diamond\langle e, i, w+1\rangle\diamond\gamma, \sigma'\diamond\#\diamond\gamma') = \mathbf{M}_e(\tau, \tau')$. If the If statement in step 6 does not hold, then for the offending r, we have that $\{x \mid \varphi_i(\mathbf{M}(\tau[r], \tau[r]), x) = 1\}$ is not in $\mathcal{L}$, since no language in $\mathcal{L}$ contains both $\langle e, i, y_s\rangle$ and $\langle e, i, w+1\rangle$ (note that in this case $\varphi_{p(e,i,s+2)}$ does not get defined). On the other hand, if the If statement holds, then consider $L = \varphi^{-1}_{p(e,i,s+2)}(1)$. Let $T = \sigma\diamond\langle e, i, w+1\rangle\diamond\gamma\diamond\#^\infty$ and $T' = \sigma'\#\gamma'\#^\infty$. As shown above, $\mathbf{M}_e(T, T') = \mathbf{M}_e(\tau, \tau')$, and the answers as given by T' are correct, as whenever $T'(r) \neq \#$, by step 4.2 (g) (ii) we have verified that there is indeed a counterexample. When, $T'(r) = \#$, by step 4 g (i), only hypotheses in $\mathcal{L}$ which could be consistent with $\mathbf{M}(T[r], T'[r])$, are $\varphi^{-1}_{p(e,i,j)}(x)$, $j \in \{0, s+1, s+2\}$, all of which are subsets of L. However, since If statement holds, we have that $\mathbf{M}_e(T, T')$ converges to a conjecture which is not for L (as L contains both $\langle e, i, y_s\rangle$ and $\langle e, i, w+1\rangle$).

From above cases we have that (a) or (b) or (c) holds for (e, i). Theorem follows. ∎

Note that if we only consider indexed families consisting of infinite languages, then class preserving learning can be done. This can be shown along the lines of Theorem 33, where instead of outputting conjectures $P(j, S, X)$ we output conjectures of form $pad(j, S, X)$ (with initial conjecture being $(pad(0, \emptyset, \emptyset))$. Update of conjectures on input $T(n), T'(n)$ is done as follows: If $T'(n) \neq \#$, then $j_{n+1} = j_n + 1$, $S_{n+1} = S_n$, $X_{n+1} = X_n \cup \{T(n)\}$. ElseIf $T'(n) = \#$ and ($T(n) \notin L_{j_n}$ or $X_n \not\subseteq L_{j_n}$ or $L_i \cap \{w \mid w \leq T(n)\} \not\subseteq L_j$, for some $i \in S_n$), then let $j_{n+1} = j_n + 1$, $S_{n+1} = S_n \cup \{j_n\}$, $X_{n+1} = X_n \cup \{T(n)\}$. Else (i.e., $T'(n) = \#$ and $T(n) \in L_{j_n}$ and $X_n \subseteq L_{j_n}$ and $L_i \cap \{w \mid w \leq T(n)\} \subseteq L_j$, for all $i \in S_n$), then let $j_{n+1} = j_n$, $S_{n+1} = S_n \cup \{j_n\}$, $X_{n+1} = X_n$. One can then verify that j_n

would converge to the minimal $\mathcal{L}$-grammar for L. We omit the details.

If we drop requirement of **NCIt**-learner being algorithmic, then the whole class of recursively enumerable languages can be learned — since then this class can be viewed as an indexed set.

Theorem 35 *There exists a non-recursive learner which* **NCIt***-identifies* $\mathcal{E}$.

Proof. For non-recursive learners, $\mathcal{E}$ can be considered as indexed family. Thus, using the analogue of Theorem 33 for non-recursive learners, we have the theorem. ∎

As it was shown in Theorem 25, in the general case, **NCIt**-learners can sometimes do more than **InfIt***-learners. However, as the next theorem shows, their capabilities on indexed classes are the same. Still, **InfIt**^{n}-learners cannot learn some indexed classes.

Theorem 36 *(a) If* $\mathcal{L}$ *is an indexed family, then* $\mathcal{L} \in$ **InfIt***.

(b) For all $n \in N$, $\mathcal{L} = \{\mathrm{cyl}_0 \cup \mathrm{cyl}_1\} \cup \{\mathrm{cyl}_0 \cup D \mid D \subseteq \mathrm{cyl}_1, \mathrm{card}(D) < \infty\} \notin$ **InfIt**^{n}.

Proof. (a) Suppose $\mathcal{L} = \{L_0, L_1, \ldots\}$, where one can effectively decide, given x, i, whether $x \in L_i$. Then one can show that $\mathcal{L} \in$ **InfIt*** (using hypothesis space $L_0, L_1, \ldots$ itself), via learner $\mathbf{M}$ defined as follows. $\mathbf{M}(\Lambda) = 0$. If $\mathbf{M}(I[n]) = j$ and $I(n) = (x, w)$, then $\mathbf{M}(I[n+1]) = j$, if $L_j(x) = w$; $\mathbf{M}(I[n+1]) = j+1$, otherwise. It is easy to verify that $\mathbf{M}$ is iterative. Clearly, $\mathbf{M}$ converges on any informant I for $L \in \mathcal{L}$ (as it would converge once it outputs the least index in $\mathcal{L}$ for L, if not earlier). If $\mathbf{M}(I) = j$, then for all but finitely many n, $I[n:\infty]$ is consistent with L_j. It follows that, L_j is a finite variant of L.

(b) Suppose by way of contradiction that $\mathbf{M}$ **InfIt**^{n}-identifies $\mathcal{L}$. Let σ be **InfIt**^{n}-locking information segment for $\mathbf{M}$ on $\mathrm{cyl}_0 \cup \mathrm{cyl}_1$. Let $D = \{x \mid x \in \mathrm{cyl}_1, (x,1) \in \mathrm{content}(\sigma)\}$. Let τ be such that $\sigma\tau$ is a **InfIt**-locking information segment for $\mathbf{M}$ on $\mathrm{cyl}_0 \cup D$. Let S be a subset of cyl_1 such that $\mathrm{card}(S) = 2n+1$ and, for all $x \in S$, $(x,1)$ and $(x,0)$ do not belong to $\mathrm{content}(\sigma\tau)$. Let I be such that $\mathrm{content}(I) = \{(x,1) \mid x \in \mathrm{cyl}_0\} \cup \{(x,0) \mid x \notin \mathrm{cyl}_0 \cup D \cup S\}$. Let γ_1, γ_2 be information segment such that $\mathrm{content}(\gamma_1) = \{(x,1) \mid x \in S\}$, and $\mathrm{content}(\gamma_2) = \{(x,0) \mid x \in S\}$. Now, $\mathbf{M}$ converges to the same grammar on $I_1 = \sigma\gamma_1\tau I$ and $I_2 = \sigma\tau\gamma_2 I$ (since $\mathbf{M}$ is iterative learner and $\mathbf{M}(\sigma\gamma_1) = \mathbf{M}(\sigma)$ and $\mathbf{M}(\sigma\tau) = \mathbf{M}(\sigma\tau\gamma_2)$, by **InfIt**-locking information segment property for σ and $\sigma\tau$). Now, I_1 and I_2 are information sequences for $\mathrm{cyl}_0 \cup D \cup S$ and $\mathrm{cyl}_0 \cup D$ respectively, which differ on $2n+1$ elements. Thus, $\mathbf{M}$ fails to **InfIt**^{n}-identify at least one of them. ∎

8 Non-Ushaped Learning

A learner is said to be *non-U-shaped* if it does not abandon a correct hypothesis ([BCM$^+$05]). That is, its sequence of conjectures does not show a pattern of ..., correct conjecture, wrong conjecture, ..., correct conjecture.

In this section, we will show that requirement of being non-U-shaped does not hurt **NCIt**-learning. Note that the problem of whether it is possible to convert every **TxtIt**-learner into a non-U-shaped one remains open (cf. [CCJS05]).

The main idea of the proof is that one searches for a type of locking sequence (called pseudo-locking sequence) which is a prefix of a some canonical texts for languages in the class. The canonical texts should satisfy the properties that (i) **NCIt**-learner should be able to obtain initial segments of canonical text using arbitrary text for a language L, (ii) **NCIt**-learner should be able to check for these initial segments, whether they are pseudo-locking sequences. (ii) above is not easy to do, however, one can do this for certain "good" pseudo-locking sequences, which suffices for our purposes. The canonical texts can be obtained by considering the elements of L being provided in increasing order. To handle checking pseudo-locking property for finite sets, we need to insert $\#$ at infinitely many positions in the canonical text (irrespective of whether L is finite or not, as the checking mechanism developed below does not know in advance whether L is finite or not). This leads to the following definition of canonical texts (can-text for short).

We say that T is *can-text* for L iff for all $x \in N$, $T(x) = x/2$, if x is even and $x/2 \in L$, and $T(x) = \#$ otherwise.

Let Canseq $= \{\sigma \mid |\sigma|$ is even and $(\forall x < |\sigma|)[\sigma(x) \neq \# \Rightarrow x$ is even and $\sigma(x) = x/2]\}$. Intuitively, Canseq denotes even length initial sequences of can-texts.

We now define pseudo-locking sequence. (This definition is specific to **NCIt** (**LNCIt**)-identification).

Definition 37 A sequence (σ, σ'), where $|\sigma| = |\sigma'|$ is said to be *pseudo-locking sequence for* $\mathbf{M}$ *on* L iff the following four properties are satisfied:

(i) content$(\sigma) \subseteq L$;

(ii) for $w < |\sigma|$, $\sigma'(w) = \#$, if $W_{\mathbf{M}(\sigma[w],\sigma'[w])} \subseteq L$; $\sigma'(w) = \min(W_{\mathbf{M}(\sigma[w],\sigma'[w])} - L)$, otherwise;

(iii) for all $w \in (L \cup \{\#\}) -$ content(σ), $\mathbf{M}(\sigma w, \sigma'\#) = \mathbf{M}(\sigma, \sigma')$;

(iv) $W_{\mathbf{M}(\sigma,\sigma')} = L$.

Below, we fix a **NCIt**-learner $\mathbf{M}$, so as to avoid giving the parameter $\mathbf{M}$ in various functions such as cseq etc. Assume $\mathbf{M}(\Lambda, \Lambda)$ can be computed in 0 time steps (this is just for ease of notation).

For a can-text T such that $\mathbf{M}$ **NCIt**-identifies content(T), let

$$\mathrm{cseq}_T(r) = \begin{cases} \#, & \text{if } W_{\mathbf{M}(T[r],\mathrm{cseq}_T[r])} \subseteq \\ & \mathrm{content}(T); \\ \min(W_{\mathbf{M}(T[r],\mathrm{cseq}_T[r])} - \mathrm{content}(T)), & \text{otherwise.} \end{cases}$$

For $\sigma \in$ Canseq, denote by cseq_σ the following sequence of counterexamples: For $r < |\sigma|$, let

$$\mathrm{cseq}_\sigma(r) = \begin{cases} \uparrow, & \text{if } \mathrm{cseq}_\sigma(r') \text{ is not defined for} \\ & \text{some } r' < r, \text{ or } \mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r]) \\ & \text{is not defined within } |\sigma| \text{ steps;} \\ \#, & \text{if } \mathrm{cseq}_\sigma(r') \text{ is defined for all} \\ & r' < r \text{ and } \mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r]) \text{ is} \\ & \text{defined within } |\sigma| \text{ steps, and} \\ & (W_{\mathbf{M}(\sigma[r],\mathrm{cseq}_\sigma[r]),|\sigma|} \cap \\ & \{x \mid x < |\sigma|/2\}) \subseteq \mathrm{content}(\sigma); \\ \min((W_{\mathbf{M}(\sigma[r],\mathrm{cseq}_\sigma[r]),|\sigma|} \cap & \\ \{x \mid x < |\sigma|/2\}) - \mathrm{content}(\sigma)), & \text{otherwise.} \end{cases}$$

Intuitively, cseq_σ gives the sequence of negative counterexamples, for learner $\mathbf{M}$ when receiving positive examples from σ, if there exist such counterexamples $< |\sigma|/2$ that can be witnessed using $|\sigma|$ simulation steps.

Note that cseq_T is just the extension of above definition for infinite sequence T such that $\mathbf{M}$ is defined on $(T[n], \mathrm{cseq}_T[n])$, for all n (which is the case if $\mathbf{M}$ **NCIt**-identifies content(T)).

Also note that, for can-text T for L which is **NCIt**-identified by $\mathbf{M}$, $\mathrm{cseq}_{T[n]}$ can be considered as approximation to cseq_T (it approximates it from above, where we do lexicographic comparison and # is considered as being bigger than any natural number).

Let m_σ be the largest value of r such that $\mathrm{cseq}_\sigma(r)$ is defined (and thus $\mathbf{M}(\sigma[r'], \mathrm{cseq}_\sigma[r'])$ is known to be defined for $r' \leq m_\sigma$).

Let prog be a recursive function such that $W_{\mathrm{prog}(\mathbf{M},\sigma)}$, for $\sigma \in$ Canseq

is defined as follows. Intuitively, $\mathrm{prog}(\mathbf{M}, \sigma)$ checks certain properties of $(\sigma[m_\sigma], \mathrm{cseq}_\sigma[m_\sigma])$ being a pseudo-locking sequence for $\mathbf{M}$ on some language X. If so, then it enumerates X; otherwise it enumerates N.

Let $X = \bigcup_{r \leq m_\sigma, \mathrm{cseq}_\sigma(r)=\#} W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r])}$.

$W_{prog(\mathbf{M},\sigma)} = X$, if the following conditions (A) to (C) are satisfied; Otherwise, $W_{prog(\mathbf{M},\sigma)} = N$.

(A) for all $r \leq m_\sigma$, $\min((W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r]), |\sigma|} \cap \{x \mid x < |\sigma|/2\}) - \mathrm{content}(\sigma)) = \min((W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r])} \cap \{x \mid x < |\sigma|/2\}) - \mathrm{content}(\sigma))$

(that is, the least counterexamples provided by cseq_σ seem to be correct, unless the minimal counterexamples are $\geq |\sigma|/2$);

(B) $\mathrm{cseq}_\sigma(m_\sigma) = \#$

(that is, $W_{\mathbf{M}(\sigma, \mathrm{cseq}_\sigma)}$ seems to be a subset of the input language (potentially correct));

(C) for all $w \in (X \cup \{\mathrm{content}(\sigma)\} \cup \{\#\}) - \mathrm{content}(\sigma[m_\sigma])$, $\mathbf{M}(\sigma[m_\sigma]w, \mathrm{cseq}_\sigma[m_\sigma]\#) = \mathbf{M}(\sigma[m_\sigma], \mathrm{cseq}_\sigma[m_\sigma])$ or $\mathbf{M}(\sigma[m_\sigma]w, \mathrm{cseq}_\sigma[m_\sigma]\#)\uparrow$.

(that is, $(\sigma[m_\sigma], \mathrm{cseq}_\sigma[m_\sigma])$ seems like a pseudo-stabilizing sequence for $\mathbf{M}$ on $X \cup \{\mathrm{content}(\sigma)\}$.)

Proposition 38 *Suppose σ is an initial segment of even length of the can-text for L, and $\mathbf{M}$ **LNCIt**-identifies L, and $L \neq N$. If $W_{\mathrm{prog}(\mathbf{M},\sigma)} \subseteq L$, then (A), (B) and (C) as well as (D)–(F) hold.*

(D) $cseq_\sigma \subseteq cseq_T$, where T is can-text for L.

(E) for $m_\sigma \leq r \leq |\sigma|$, $\mathbf{M}(\sigma[r], cseq_\sigma[m_\sigma]\#^{r-m_\sigma})\downarrow = \mathbf{M}(\sigma[m_\sigma], cseq_\sigma[m_\sigma])$,

(F) if $W_{prog(\mathbf{M},\sigma)} = L$, then $(\sigma[m_\sigma], cseq_\sigma[m_\sigma])$ is a pseudo-locking sequence for $\mathbf{M}$ on L. Thus $(\sigma, cseq_\sigma[m_\sigma]\#^{|\sigma|-m_\sigma})$ is also a pseudo-locking sequence for $\mathbf{M}$ on L.

Proof. If $\mathrm{cseq}_\sigma(r) = \#$, then by definition of prog and hypothesis of the proposition, we have $W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r])} \subseteq W_{\mathrm{prog}(\mathbf{M},\sigma)} \subseteq L$. If $\mathrm{cseq}_\sigma(r) \neq \#$, then by definition of cseq_σ and (A), we have that $\mathrm{cseq}_\sigma(r) = \min((W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r]), |\sigma|} \cap \{x \mid x < |\sigma|/2\}) - \mathrm{content}(\sigma)) = \min((W_{\mathbf{M}(\sigma[r], \mathrm{cseq}_\sigma[r])} \cap \{x \mid x < |\sigma|/2\}) - \mathrm{content}(\sigma))$. Now as $L \cap \{x \mid x < |\sigma|/2\} = \mathrm{content}(\sigma)$, (by definition of can-text), (D) follows.

(E) follows using (C), as $W_{\mathbf{M}(\sigma[m_\sigma], \mathrm{cseq}_\sigma[m_\sigma])} \subseteq L$ (see (B), (D)).

(F) follows using (C), and the fact that **M NCIt**-identifies L. ∎

Proposition 39 *Let T be can-text for L such that* **M LNCIt***-identifies L. Then, for all but finitely many s,* $\text{prog}(\mathbf{M}, T[2s])$ *is a grammar for L, and $cseq_{T[2s]}$ is an initial sequence of $cseq_T$.*

Proof. Let n be large enough such that

(i) $(\forall n' \geq n)[\mathbf{M}(T[n'], \text{cseq}_T[n']) = \mathbf{M}(T[n], \text{cseq}_T[n])]$,

(ii) $(\forall n' \leq n)$, if $\text{cseq}_T(n') \neq \#$, then $\text{cseq}_T(n') \in W_{\mathbf{M}(T[n'],\text{cseq}_T[n']),n}$.

Let $s > n$ be large enough such that for all $n' \leq n$, $\mathbf{M}(T[n'], \text{cseq}_T[n'])$ can be computed within s steps.

Then, it is easy to verify that for all even $s' > s$, $\text{cseq}_{T[s']}$, is an extension of $\text{cseq}_T[n+1]$, and $\text{cseq}_{T[s']} \subseteq \text{cseq}_T$, and thus, $L = W_{\mathbf{M}(T[n],\text{cseq}_T[n])} \subseteq W_{\text{prog}(\mathbf{M},T[s'])} \subseteq L$ (last inequality holds as, for all r such that $\text{cseq}_T(r) = \#$, we have $W_{\mathbf{M}(T[r],\text{cseq}_T[r])} \subseteq L$). Proposition follows. ∎

In the following theorem our aim is to find a prefix $T''[n]$ of can-text T'' for L such that (i) $\text{prog}(\mathbf{M}, T''[n])$ does not produce a counterexample, and (ii) for all $x \in L$, for some $n' \leq n$, $\mathbf{M}(T''[n']x, \text{cseq}_{T''}[n'+1]) = \mathbf{M}(T''[n'], \text{cseq}_{T''}[n'])$, and (iii) for all but finitely many $x \in L$, $\mathbf{M}(T[n]x, \text{cseq}_{T''}[n+1]) = \mathbf{M}(T''[n], \text{cseq}_{T''}[n])$. The conditions (ii) and (iii) along with **NCIt**-identification of L by $\mathbf{M}$ would ensure that $\mathbf{M}(T''[n'], \text{cseq}_{T''}[n'])$ is a grammar for L. Thus, using (i) and definition of $\text{prog}(\mathbf{M}, T''[n])$ will ensure that $\text{prog}(\mathbf{M}, T''[n])$ is a grammar for L.

Theorem 40 LNCIt $\subseteq$ NUNCIt.

Proof. Suppose **M LNCIt**-identifies $\mathcal{L}$. Without loss of generality assume $\mathcal{L}$ does not contain N and every language in $\mathcal{L}$ consists of at least one element (otherwise, we could just modify the following learner $\mathbf{M}'$ to first output a grammar for N, and in case of a counterexample, output the input set until at least two elements are discovered in the input, in which case the following technique can be applied to learn the input).

The output of $\mathbf{M}'$ will be of the form:

$pad(p, S, \sigma, mode)$

Where, S denotes the backlog, σ denotes an even length initial segment of can-text for the input language, and mode denotes the mode of output/operation.

mode=0, means that we are testing whether a particular element $|\sigma|/2$ belongs

to the input language or not. Thus, $W_p = \{|\sigma|/2\}$.

mode=1 means regular output, which would mean that $p = prog(\mathbf{M}, \sigma)$.

Suppose T is a text for $L \in \mathcal{L}$. Suppose $\mathbf{M}'(T[n])$ is $pad(p_n, S_n, \sigma_n, mode_n)$. We will have the invariant that

(G) σ_n is even length initial segment of can-text of L. Furthermore, $\sigma_n \subseteq \sigma_{n+1}$, and if $mode_n = 0$, then $\sigma_n \subset \sigma_{n+1}$.

(H) For all $x \in \text{content}(T[n]) - S_n$, either $x \in \text{content}(\sigma_n)$, or for some $r \leq |\sigma_n|$, $\mathbf{M}(\sigma_n[r]x, \text{cseq}_{T''}[r+1]) = \mathbf{M}(\sigma_n[r], \text{cseq}_{T''}[r])$, where T'' is can-text for L.

Let $\mathbf{M}'(T[0]) = pad(p_0, \emptyset, \Lambda, 0)$, where $W_{p_0} = \{0\}$.

We show how to compute $\mathbf{M}'(T[n+1])$, based on $p_n, S_n, \sigma_n, mode_n$, the input element $T(n)$, and the counterexample c_n.

Case 1: $mode_n = 0$

If $c_n = \#$, then let $\sigma_{n+1} = \sigma_n \diamond (|\sigma_n|/2)\#$. Else, let $\sigma_{n+1} = \sigma_n \#\#$.
Let $S_{n+1} = S_n \cup \{T(n)\} - \{\#\}$
Let $mode_{n+1} = 1$.
Let $p_{n+1} = \text{prog}(\mathbf{M}, \sigma_{n+1})$.

Case 2: $mode_n = 1$, $c_n \neq \#$.

Let $\sigma_{n+1} = \sigma_n$.
Let $S_{n+1} = S_n \cup \{T(n)\} - \{\#\}$
Let $mode_{n+1} = 0$.
Let p_{n+1} be such that $W_{p_{n+1}} = \{|\sigma_{n+1}|/2\}$.

Case 3: $mode_n = 1$, $c_n = \#$.

Note that by Proposition 38 (D), in this case we have that $\text{cseq}_{\sigma_n} \subseteq \text{cseq}_{T''}$, where T'' is can-text for L.
Let $\sigma_{n+1} = \sigma_n$.
Let $S_{n+1} = (S_n \cup \{T(n)\}) - (\text{content}(\sigma_n) \cup \{\#\} \cup \{w \mid \mathbf{M}(\sigma_n[m_{\sigma_n}]w, \text{cseq}_{\sigma_n}[m_{\sigma_n}]\#) = \mathbf{M}(\sigma_n[m_{\sigma_n}], \text{cseq}_{\sigma_n}[m_{\sigma_n}])\})$.
If $S_{n+1} \neq \emptyset$, then let $mode_{n+1} = 0$, and p_{n+1} be such that $W_{p_{n+1}} = \{|\sigma_{n+1}|/2\}$.
Else (i.e., if $S_{n+1} = \emptyset$), let $mode_{m+1} = 1$, and $p_{m+1} = p_m$ (which is $= \mathbf{M}(\sigma_n[m_{\sigma_n}], \text{cseq}_{\sigma_n}[m_{\sigma_n}]) = \mathbf{M}(\sigma_n, \text{cseq}_{T''}[|\sigma_n|])$ where T'' is the can-text for input language, by properties (D) and (E) in Proposition 38.)

It is easy to verify that invariants are satisfied.

Now, if $mode_n = 0$, for infinitely many n, then for large enough n, using invariant (G), we have that (i) σ_n is a pseudo-locking sequence for $\mathbf{M}$ on L (since $\mathbf{M}$ **NCIt**-identifies L on can-text for L), and (ii) using, Proposition 39, $\text{prog}(\mathbf{M}, \sigma_n)$ is a grammar for L. Fix large enough n, such that above properties hold for all bigger n. Thus, for $n' > n$, we will always be in case 3, with $S_{n'+1}$ as computed there being $\emptyset$, and thus $mode_{n'+1} = 1$. A contradiction.

Thus, for all but finitely many n, $mode_n = 1$. It follows that eventually we are always in case 3, with $S_{n+1} = \emptyset$. Let $\sigma = \lim_{n\to\infty} = \sigma_n$. By Proposition 38 we have that, for T'' being can-text for L,

(i) $\text{cseq}_\sigma \subseteq \text{cseq}_{T''}$

(ii) $(\forall^\infty n)[\mathbf{M}(\sigma T(n), \text{cseq}_{T''}[|\sigma|]\#) = \mathbf{M}(\sigma, \text{cseq}_{T''}[|\sigma|]) = \mathbf{M}(\sigma[m_\sigma], \text{cseq}_\sigma[m_\sigma])]$ (by Case 3, and $mode_{n+1}$ being 1), and

(iii) for all $x \in L$, there exists a $w \leq |\sigma|$ such that $\mathbf{M}(\sigma[w]x, \text{cseq}_{T''}[w+1]) = \mathbf{M}(\sigma[w], \text{cseq}_{T''}[w])$ (by invariant (H)).

Thus, $\mathbf{M}(\sigma, \text{cseq}_{T''}[|\sigma|]) = \mathbf{M}(\sigma[m_\sigma], \text{cseq}_\sigma[m_\sigma])$ is a grammar for L (by **LNCIt**-identification of L by $\mathbf{M}$), and thus, $W_{\text{prog}(\mathbf{M},\sigma)} \supseteq L$, by definition of $W_{\text{prog}(\mathbf{M},\sigma)}$. Thus $\text{prog}(\mathbf{M}, \sigma)$ is a grammar for L as $\mathbf{M}'$ does not receive a counterexample for conjecture $\text{prog}(\mathbf{M}, \sigma)$.

It follows from the above that $\mathbf{M}'$ **NCIt**-identifies L. Now suppose n is the least such that $\mathbf{M}'(T[n]) = (p_n, S_n, \sigma_n, mode_n)$ is a grammar for L. Then, we have that $mode_n = 1$, and $p_n = \text{prog}(\mathbf{M}, \sigma_n)$. Thus, by Proposition 38, $(\sigma_n[m_{\sigma_n}], \text{cseq}_{\sigma_n}[m_{\sigma_n}])$ is a pseudo-locking sequence for $\mathbf{M}$ on L. It follows that for all $n' \geq n$, on input $T[n']$, $\mathbf{M}'$ will be in Case 3, and $S_{n'+1} = \emptyset$. Thus, $\mathbf{M}'$ does not change its conjecture on inputs $T[n']$, $n' \geq n$. Thus, $\mathbf{M}'$ is non-U-shaped on L. ∎

References

[Ang80] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

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

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[BCJ95] G. Baliga, J. Case, and S. Jain. Language learning with some negative information. *Journal of Computer and System Sciences*, 51(5):273–285, 1995.

[BCM+05] G. Baliga, J. Case, W. Merkle, F. Stephan, and R. Wiehagen. When unlearning helps. Manuscript, http://www.cis.udel.edu/~case/papers/decisive.ps, 2005.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[Bow82] M. Bowerman. Starting to talk worse: Clues to language acquisition from children's late speech errors. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, New York, 1982.

[CCJS05] L. Carlucci, J. Case, S. Jain, and F. Stephan. Memory limited U-shaped learning. Technical Report TR51/05, School of Computing, National University of Singapore, Nov 2005.

[CL82] J. Case and C. 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.

[CS83] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[Ful90] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.

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

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[JK04] S. Jain and E. Kinber. Learning languages from positive data and negative counterexamples. In Shai Ben-David, John Case, and Akira Maruoka, editors, *Algorithmic Learning Theory: Fifteenth International Conference (ALT' 2004)*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 54–68. Springer-Verlag, 2004.

[JK05] S. Jain and E. Kinber. Learning languages from positive data and negative counterexamples. *Journal of Computer and System Sciences*, 2005. To appear.

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Mass., second edition, 1999.

[LZ96] S. Lange and T. Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53:88–103, 1996.

[Mot91] T. Motoki. Inductive inference from all positive and some negative data. *Information Processing Letters*, 39(4):177–182, 1991.

[MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms*. North Holland, New York, 1978.

[Pin79] S. Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.

[Pop68] K. Popper. *The Logic of Scientific Discovery*. Harper Torch Books, New York, second edition, 1968.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.

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

[ZL95] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In K. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer-Verlag, 1995.