Theoretical Computer Science 473 (2013) 100–123

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Memory-limited non-U-shaped learning with solved open problems

John Case [a], Timo Kötzing [b,*]

[a] Department of Computer and Information Science, University of Delaware, Newark, DE 19716, USA

[b] Department 1: Algorithms and Complexity, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

ARTICLE INFO

Keywords:
Learning from positive data
U-shaped learning

ABSTRACT

In empirical cognitive science, for human learning, a semantic or behavioral *U-shape* occurs when a learner first learns, then unlearns, and, finally, relearns, some target concept.

Within the formal framework of Inductive Inference, for learning from positive data, previous results have shown, for example, that such U-shapes are unnecessary for explanatory learning, but are necessary for behaviorally correct and non-trivial vacillatory learning.

Herein we also distinguish between semantic and syntactic U-shapes. We answer a number of open questions in the prior literature as well as provide new results regarding syntactic U-shapes. Importantly for cognitive science, we see more of a previously noticed pattern that, for parameterized learning criteria, beyond very few initial parameter values, U-shapes *are necessary* for full learning power.

We analyze the necessity of U-shapes in two memory-limited settings.

The first setting is Bounded Memory State (BMS) learning, where a learner has an explicitly-bounded state memory, and otherwise only knows its current datum. We show that there are classes learnable with three (or more) memory states that are not learnable non-U-shapedly with any finite number of memory states. This result is surprising, since, for learning with one or two memory states, U-shapes are known to be unnecessary. This solves an open question from the literature.

The second setting is that of Memoryless Feedback (MLF) learning, where a learner may ask a bounded number of questions about what data has been seen so far, and otherwise only knows its current datum. We show that there is a class learnable memorylessly with a single feedback query such that this class is not learnable non-U-shapedly memorylessly with any finite number of feedback queries.

We employ self-learning classes together with the Operator Recursion Theorem for many of our results, but we also introduce two new techniques for obtaining results. The first is for transferring inclusion results from one setting to another. The main part of the second is the *Hybrid Operator Recursion Theorem*, which enables us to separate some learning criteria featuring *complexity-bounded* learners, employing *self-learning classes*. Both techniques are not specific to U-shaped learning, but applicable for a wide range of settings.

1. Introduction

In Section 1.1 we explain and motivate *U-shaped learning* and present a conjecture significant for cognitive science and based in part on new results in the present paper.

* Corresponding author.
E-mail addresses: case@cis.udel.edu (J. Case), koetzing@mpi-inf.mpg.de (T. Kötzing).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.10.010

In Section 1.2 we discuss the general techniques of the present paper including two new, interesting techniques.

Section 1.3 summarizes our results regarding the necessity of U-shaped learning in memory limited contexts and indicates which are our *main* results. These main results provide cognitive science insight about U-shaped learning, and they include those answering interesting, previously open questions.

1.1. U-shaped learning

U-shaped learning occurs when a learner first learns a correct *behavior*, then abandons that correct behavior and finally returns to it once again. This pattern of learning has been observed by cognitive and developmental psychologists in a variety of child development phenomena, such as language learning [6,37,44], understanding of temperature [44,45], weight conservation [5,44], object permanence [5,44] and face recognition [7]. The case of language acquisition is paradigmatic. For example, in the case of the past tense of English verbs, it has been observed that children learn correct syntactic forms (call/called, break/broken), then undergo a period of overregularization in which they attach regular verb endings such as 'ed' to the present tense forms even in the case of irregular verbs (break/breaked, speak/speaked) and finally reach a final phase in which they correctly handle both regular and irregular verbs. This example of U-shaped learning behavior has figured prominently in cognitive science [37,40,46].[1]

While the prior cognitive science literature on U-shaped learning was typically concerned with the reality of such learning behavior or with modeling *how* humans achieve this U-shaped behavior, [4,12] are motivated by the question of *why* humans exhibit this seemingly inefficient behavior. Is it a mere evolved but harmless inefficiency or is it *necessary* for full human learning power? A technically answerable version of this question is: are there some formal learning tasks for which U-shaped behavior is logically necessary? We first need to describe some formal criteria of successful learning.

An algorithmic learning function h is, typically, fed an infinite sequence consisting of the elements of a (formal) language L in arbitrary order with possibly some pause symbols # in between elements. During this process h outputs a corresponding sequence $p(0), p(1), \ldots$ of hypotheses (grammars) which may generate the language L to be learned. A fundamental criterion of successful learning of a language is called *explanatory learning* (TxtEx-*learning*) and was introduced by Gold [27]. Explanatory learning requires that the learner's output conjectures stabilize in the limit to a *single* conjecture (grammar/program, description/explanation) that generates the input language. *Behaviorally correct learning (TxtBc-learning)* [19,39] requires, for successful learning, only convergence in the limit to possibly infinitely many syntactically distinct but correct conjectures. Another interesting class of criteria features *vacillatory learning* [10,31]. This paradigm involves learning criteria which allow the learner to vacillate in the limit between *at most* some bounded, finite number of syntactically distinct but correct conjectures. For each criterion that we consider above (and below), a *non-U-shaped learner* is naturally modeled as a learner that never *semantically* returns to a previously abandoned correct conjecture on languages it learns according to that criterion.

[4] showed that every TxtEx-learnable class of languages is TxtEx-learnable by a non-U-shaped learner, that is, for TxtEx-learnability, U-shaped learning is *not* necessary. Furthermore, based on a proof in [23], [4] noted that, by contrast, for behaviorally correct learning [19,39], U-shaped learning *is* necessary for full learning power. In [12] it is shown that, for non-trivial vacillatory learning, U-shaped learning is again *necessary* (for full learning power). In the present paper we will see that, for the learning criteria for which it presents new results, again, beyond a very few initial parameter values, seemingly inefficient U-shaped learning *does* increase learning power. This leads us to a conjecture significant for cognitive science: for the still open problems usefully shown in Table 1 in Section 5 below, beyond initial, small parameter values, we will again see that U-shapes are *necessary* for full learning power.

What turns out to be a variant of non-U-shaped learning is *strongly non-U-shaped* learning essentially defined in [49],[2] where the learner is required never to *syntactically* abandon a correct conjecture on languages it learns according to that criterion. Clearly, *strong* non-U-shaped learnability implies non-U-shaped learnability.[3] In our experience, for theoretical purposes, it is frequently easier to show non-U-shaped learnability by showing *strong* non-U-shaped learnability.

A notion similar to non-U-shapedness is that of *decisiveness* [38], where a learner may not semantically return to *any* semantically abandoned conjecture (not just abandoned *correct* conjectures) on *any* text (not just on texts for learnable languages). It is intuitively clear that this is a very strong requirement for memory-limited learning criteria, (see Theorem 4.9 below, which shows, in certain memory-limited settings anyhow, that only finite classes of languages can be decisively learned).[4] However, the notion herein of *class-decisive*, which requires decisive behavior only on texts for successfully learned languages, might be more interesting.[5] Note that class-decisive learning is closer to non-U-shaped learning than

1 U-shapes are such an important topic in cognitive science that a Special Issue of the *Journal of Cognition and Development* (Volume 5, Issue 1, 2004) was devoted to them.

2 Wiehagen actually used the term *semantically finite* in place of *strongly non-U-shaped*. However, there is a clear connection between this notion and that of *non-U-shapedness*. Our choice of terminology is meant to expose this connection. See also [20].

3 For non-U-shaped learning, the learner (on the way to success) must not *semantically* abandon a correct conjecture. In general, semantic change of conjecture is not algorithmically detectable, but syntactic change is. However, in the cognitive science lab we can many times observe a *behavioral/semantic* change, but it is beyond the current state of the art to see, for example, grammars in people's heads — so we cannot *yet* observe mere syntactic changes in people's heads.

4 In [4] it is shown that decisiveness even limits the power of TxtEx-learning.

5 The term 'class-decisive' is used in [31], but in the context of function learning.

decisive learning.[6] Furthermore, as for *strongly* non-U-shaped learning, we say a learner is *strongly class-decisive* iff, on texts for learned languages, the learner never *semantically* returns to a *syntactically* abandoned conjecture. With regard to this notion, herein we have only Remark 3.13 below.

1.2. Presented techniques

In this paper we employ general techniques to tackle problems regarding U-shaped learning, and two of them are new to the present paper.

In [17] we presented a very general technique which can be used to show separations of learning criteria power, for example, to show the *necessity* of U-shapes. We employ it again herein. This technique employs so-called *self-learning classes of languages* as witnesses for separations of learning criteria power. A self-learning class of languages is such that each element of each language of the class provides instructions for what to compute and output as a new hypothesis. The technique itself is to employ, with self-learning classes, subtle infinitary self-and-other program reference arguments employing (variants of) the Operator Recursion Theorem (ORT) from [8,9] (see also [31]). This technique of self-learning classes together with the ORT can also be employed in the setting of learning computable functions, for which the technique was extensively studied in its own right in [18]. Both papers (in different settings) show self-learning classes to be very strong in the following sense. For learners *not* restricted to be total, for example, to run in usefully bounded time, two associated learning criteria separate if and only if the associated self-learning class witnesses the separation.

In Section 2.2, new to the present paper, we importantly also extend the powerful tool of self-learning classes so as to separate learning criteria restricted to *linear time computable* learners from other criteria. In that section we, then, provide a new and interesting theorem to help carry out the separation proofs employing the associated self-learning classes — for the case of learners restricted to be linear time computable. This result is called the Hybrid Operator Recursion Theorem (HORT) (Theorem 2.4 below). It is a generalization of the ORT mentioned just above. The HORT permits self-and-other program reference between certain *highly restricted* complexity-bounded programming systems and a general purpose programming system.[7]

Just as the original tool of self-learning classes was not restricted to analyzing U-shaped learning, one could apply our HORT-based extension (and variants) to other learning criteria as well.

Introduced in [11] is *Bounded Memory State (BMS) learning* (see Section 1.3 below for a discussion), and, in [11], it is asserted that BMS-learning with an arbitrary (but finite) number of memory states per learning process is equivalent to iterative learning (see also Theorem 3.5 below). But does the result from [21] that iterative learning does not require U-shaped behavior carry over to BMS-learning? Below we present another technique, also new to the present paper, the Transfer Lemma, Lemma 2.7, for formally transferring inclusion results. Often, one can say something like "It is easy to see that the proof we gave for X can be modified to be a proof for Y". The Transfer Lemma is about making these kind of statements more formal, so that one can say "Because of X and the Transfer Lemma, we immediately obtain Y". In particular, we use the Transfer Lemma to obtain that U-shapes are unnecessary for BMS-learning with arbitrary but finitely many memory states (see Corollary 3.6). This second technique is also not specialized only to analyze U-shaped learning, but can be applied to other learning criteria as well. The technique is developed and discussed further in Section 2.3.

1.3. Memory-limited learning results

It is clear that human learning involves memory limitations. In the present paper (as in [11]) we consider the necessity of U-shaped learning in some formally *memory-limited* versions of language learning. In the prior literature many types of memory-limited learning have been studied [36,47,48,38,23,14,11,30]. Herein we study the types from [11] about which that paper has some results, and answer the open questions from that paper about those types.

As noted above, [11] introduces *Bounded Memory State (BMS) learning*. Associated learners do *not* have access to any previously seen data. Instead, after each datum presented, the learner may choose one out of a bounded number of memory states, and, when presented with another datum, will be passed this memory state along with the new datum. Thus, each output of new conjecture and new memory state may depend only on the new datum and the just previous memory state. Intuitively, such a learner can be pictured as a finite state machine, where the transitions depend on each new datum.[8]

---

6 The difference is that non-U-shaped learning requires (class) decisiveness only for the correct conjectures of a language being learned and does not require (class) decisiveness for the correct conjectures of a learnable language when it is not being learned.

Nonetheless, we (mostly) do not explore in the present paper trivial or other comparisons of criteria classes featuring class decisiveness versus those featuring non-U-shapedness.

7 It differs a bit conceptually from the Hybrid Recursion Theorem of [41] in that the latter involves *bounded finitary* self-reference between, for example, a relatively *un*restricted complexity-bounded programming system and a general purpose one.

8 For such a learner with a number of memory states equal $c \geq 1$, intuitively, the learner can store any one out of c different *values* in its long term memory [24,34]. For example, when c is 2^k, the memory is equivalent to the learner having k bits of memory.

For the criteria studied for example in [48,14,29,30], learning functions also have access to their just prior output conjecture (if any), *but*, for the criteria studied herein, learning functions have no such access.

In [11], the authors show that Bounded Memory State learning with up to two memory states does not require U-shapes.

As an open problem (Problem 40) they ask whether or not U-shapes are similarly unnecessary for higher numbers of memory states. Surprisingly, one of our main results herein, Theorem 3.3 says that there is a class learnable with *three* memory states which is not learnable for *any* number of memory states and with*out* U-shapes. Hence, in all but the bottom two cases for number of memory states available, *U-shapes are necessary* for increased learning power.

Furthermore, from another main result herein, Theorem 3.10, we have that there is a class learnable with just two memory states, which is not *strongly* non-U-shapedly learnable *even by an iterative learner*. However, as we show in Theorem 3.11, this latter result does no longer hold if we assume total learners, where every learnable class with a bounded number of memory states can be learned strongly non-U-shapedly with finitely (but unboundedly) many memory states. Corollary 3.6 gives that U-shapes are not necessary for learning with finitely many memory states; our proof employs the Transfer Lemma, and uses the extensional equivalence of iterative learning and learning with finitely many memory states ([11], see also Theorem 3.5).

We conclude the analysis of BMS cases with two remarks on class-decisive learning criteria with one and two memory states (Remarks 3.12 and 3.13).

Also in [11], *Memoryless Feedback (MLF) learning* is introduced. This is similar to BMS-learning in that a learner does not have direct access to any strictly previously seen data. Instead, for a given natural number n, the learner may *query*, in parallel, for up to n different data points, whether those data points have been seen previously. No query may depend on the outcome of another query, and all queries are individually answered truthfully, so that the learner knows, for each queried datum, whether it has been seen before.

In [11], the authors show that, for each choice of parameter $n > 0$, U-shapes are necessary for the full learning power of MLF-learning. As an open problem (Problem 39), they ask, for any given parameter $m > 0$, whether there is a parameter $n > m$ such that MLF-learning with a (possibly high) parameter of n allows for non-U-shaped learning of all classes of languages that are MLF learnable with parameter m. We answer this question negatively, and show a much stronger result, another of our main results herein: Theorem 4.2 below, says that there is a class of languages learnable memorylessly with a *single* feedback query which is not non-U-shapedly MLF learnable with *arbitrarily many sequential recall queries*. For this result, the learner may even continue asking queries, *dependent on the outcome of previous queries*, and not be limited to any finite number.

We complement this latter result by showing that any class of *infinite* languages learnable memorylessly with finitely many feedback queries is so learnable without U-shapes. Even stronger, Theorem 4.3 states that each TxtEx-learnable class of infinite languages is learnable memorylessly with arbitrarily many feedback queries and without U-shapes.

For this latter theorem, it is essential that the number of feedback queries is not bounded: Theorem 4.6 states that there is a class of infinite languages learnable memorylessly with a single feedback query, which is *not* learnable with*out* U-shapes by any *particular bounded* number of feedback queries.

We conclude our analysis of MLF-learning by showing that it is *essential* that a query can be used to find out whether the *current* datum has been seen before (see Theorem 4.8) and that (all and) only finite collections of languages can be MLF-learned decisively.

This paper is an extension of [16].

2. Mathematical preliminaries

Complexity-theoretic notions follow [41]. Unintroduced computability-theoretic notions follow [42].

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$.

The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets. The symbol $\setminus$ denotes set-difference.

The quantifier $\forall^\infty x$ means "for all but finitely many x", the quantifier $\exists^\infty x$ means "for infinitely many x". For any set A, $\mathrm{card}(A)$ denotes its cardinality.

With $\mathfrak{P}$ and $\mathfrak{R}$ we denote, respectively, the set of all partial and of all total functions $\mathbb{N} \to \mathbb{N}$. With dom and rng we denote, respectively, domain and range of a given function.

We sometimes denote a partial function f of $n > 0$ arguments $x_1, \ldots, x_n$ in lambda notation (as in Lisp) as $\lambda x_1, \ldots, x_n . f(x_1, \ldots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x . c$ is the constantly c function of one argument.

For any predicate P, we let $\mu x . P(x)$ denote the least x such that $P(x)$.

We fix any computable 1–1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[9] We let π_1 and π_2 be the decoding functions for the first and second component, respectively. Whenever we consider tuples of natural numbers as input to $f \in \mathfrak{P}$, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to (left-associatively) code the tuples into a single natural number. We similarly fix a coding for finite sets and sequences, so that we can use those as input as well.

If $f \in \mathfrak{P}$ is not defined for some argument x, then we denote this fact by $f(x)\uparrow$, and we say that f on x *diverges*; the opposite is denoted by $f(x)\downarrow$, and we say that f on x *converges*. If f on x converges to p, then we denote this fact by $f(x)\downarrow = p$. If f

[9] For a linear-time example, see [41, Section 2.3].

is computable and we can fix a program for f, then we use, for all x, t, $f(x)\downarrow_t$ to say that the fixed program for f converges within t steps.

The special symbol ? is used as a possible hypothesis (meaning "no change of hypothesis"). We say that $f \in \mathfrak{P}$ *converges to* p iff $\exists x_0 : \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$; we write $f \to p$ to denote this.[10]

A partial function $\psi \in \mathfrak{P}$ is *partial computable* iff there is a deterministic, multi-tape Turing machine which, on input x, returns $\psi(x)$ if $\psi(x)\downarrow$, and loops infinitely if $\psi(x)\uparrow$. $\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all computable functions $\mathbb{N} \to \mathbb{N}$. The functions in $\mathcal{R}$ are called *computable functions*.

We let φ be any fixed acceptable programming system for the partial computable functions $\mathbb{N} \to \mathbb{N}$ with associated complexity measure Φ. Further, we let φ_p denote the partial computable function computed by the φ-program with code number p, and we let Φ_p denote the partial computable *complexity* function of the φ-program with code number p.

In this paper, we consider *linear time* computability with respect to $(\varphi^{\mathrm{TM}}, \Phi^{\mathrm{TM}})$. We call a function f *linlin* iff f is linear time computable and there is a linear time computable function $\hat{f}$ such that $\hat{f} \circ f$ is the identity.

A set $L \subseteq \mathbb{N}$ is *computably enumerable (*ce*)* iff it is the domain of a computable function. Let $\mathcal{E}$ denote the set of all ce sets. We let W be the mapping such that $\forall e : W(e) = \mathrm{dom}(\varphi_e)$. For each e, we write W_e instead of $W(e)$. W is, then, a mapping from $\mathbb{N}$ *onto* $\mathcal{E}$. We say that e is an index, or program, (in W) for W_e.

In this paper, a *computable operator* is a mapping from any one (respectively, two) partial function(s) $\mathbb{N} \to \mathbb{N}$ into one such partial function such that there exists an algorithm which, when fed *any* enumeration(s) of the graph(s) of the input(s), outputs *some* enumeration of the graph of the output. Rogers [42] extensively treats the one-ary case of these operators and calls them *recursive operators*.

An *effective operator* is a mapping $\Theta : \mathcal{P} \to \mathcal{P}$ such that there is an $f \in \mathcal{R}$ with, for all e, $\Theta(\varphi_e) = \varphi_{f(e)}$. Note that every effective operator has a unique extension to a computable operator.

A *finite sequence* is a mapping with a finite initial segment of $\mathbb{N}$ as domain (and range, $\subseteq \mathbb{N}$). $\emptyset$ denotes the empty sequence (and, also, the empty set). The set of all finite sequences is denoted by $\mathbb{S}\mathrm{eq}$. For any given set $A \subseteq \mathbb{N}$, the set of all finite sequences of elements in A is denoted with $\mathbb{S}\mathrm{eq}(A)$. For each finite sequence σ, we will denote the first element, if any, of that sequence by $\sigma(0)$, the second, if any, by $\sigma(1)$ and so on. $\mathrm{len}(\sigma)$ denotes the number of elements in a finite sequence σ, that is, the cardinality of its domain. We use $\diamond$ (with infix notation) to denote concatenation on sequences. With a slight abuse of notation, for a sequence σ and a natural number x, we let $\sigma \diamond x$ denote the sequence that starts with the sequence σ and then ends with x. For any finite sequence σ such that $\mathrm{len}(\sigma) > 0$, we let $\mathrm{last}(\sigma)$ be the last element of σ and σ^- be σ with its last element deleted. By convention, we set $\emptyset^- = \emptyset$.

For a partial function $f \in \mathfrak{P}$ and $i \in \mathbb{N}$, if $\forall j < i : f(j)\downarrow$, then $f[i]$ is defined to be the finite sequence $f(0), \ldots, f(i-1)$.

The symbol # is pronounced *pause* and is used to symbolize "no new input data" in a text. For each (possibly infinite) sequence q, let $\mathrm{content}(q) = (\mathrm{rng}(q) \setminus \{\#\})$.

Later, we will type infinite sequences as being in $\mathfrak{R}$, *but*, technically, *texts* (for languages $\subseteq \mathbb{N}$) *are infinite sequences*, which may contain pauses (#s) which are not natural numbers. We will assume the pauses to be appropriately coded as natural numbers. The special symbol ? mentioned above is treated in a way analogous to that of #.

From now on, by convention, f, g and h with or without decoration range over (partial) functions $\mathbb{N} \to \mathbb{N}$; x, y with or without decorations range over $\mathbb{N}$; σ, τ with or without decorations range over finite sequences of natural numbers; D with or without decorations ranges over finite subsets of $\mathbb{N}$.

We will make use of a padded variant of the s-m-n Theorem [42]. Intuitively, s-m-n permits algorithmic storage of arbitrary data (and, hence, programs) inside any program. The suitable padded variant of s-m-n we use herein states that there is a strictly monotonically increasing total computable function s such that s is linlin and

$$\forall a, b, c : \varphi_{s(a,b)}(c) = \varphi_a(b, c). \tag{1}$$

In (1), φ-program $s(a, b)$ is essentially φ-program a with datum b stored inside. We will also use variants of Case's *Operator Recursion Theorem* (ORT), providing *infinitary* self-and-other program reference [8,9,31]. ORT itself states that, for all computable operators $\Theta : \mathfrak{P} \to \mathfrak{P}$,

$$\exists e \in \mathcal{R}\ \forall a, b : \varphi_{e(a)}(b) = \Theta(\varphi_e)(a, b). \tag{2}$$

We will employ a version where the function e is linlin.

2.1. Computability-theoretic learning

In this section we formally define several criteria for computability-theoretic learning.

The prior literature (as exemplified in [31]) usually defines each learning criterion independently of others, even though many criteria share some concepts. We prefer a unified notation for learning criteria and use an approach introduced in [32]; this approach covers many criteria defined in the prior literature, in particular all criteria discussed in this paper.

[10] $f(x)$ converges should not be confused with f converges *to*.

The benefits from this level of abstraction are manifold:

(i) Achieving better understanding of particular learning criteria and their *relations with one another*.
(ii) Development of *unified problem solving techniques*.
(iii) Discovery of *new but natural learning criteria*.
(iv) Improvements in mathematical clarity, precision, *economy* and rigor.

Item (i) is a general goal also in many other publications; we believe that our unified notation facilitates this goal by making the definitions of different criteria easier to compare for humans, as well as making variants easier to define by exchanging or adding parts of a learning criterion.

Item (ii) is exemplified in Section 2.3, where we give one such technique; others are given in [17], of which we cite one in Lemma 2.2. For new learning criteria mentioned in Item (iii), we refer to [15], where *dynamic modeling* was introduced.

Finally, we want to stress Item (iv) by saying that we discuss many criteria in this paper, all of which are succinctly defined in this section without reference to "analogous" or "similar" definitions. In particular, we give an overview over many criteria and their properties in Table 1 in Section 5.

In the introduction we referred to learning with a bounded number of memory states as BMS-learning, Memoryless Feedback learning was called MLF-learning. Thus, both BMS and MLF refer to how a learner processes information about the target language. On the other hand we saw TxtEx-learning, where "Txt" refers to data presentation from texts, and "Ex" refers to the sequence of conjectures converging to a correct hypothesis. This is the classical way of denoting these criteria, but this naming scheme is inconsistent: names sometimes refer to how data is processed, sometimes to data presentation and restrictions on conjectures. Our unified notation always makes all parts of a learning criterion explicit. For example, BMS-learning will be denoted as TxtBMSEx-learning (to denote learning from text, with data processing with bounded memory states and Ex-style convergence), and TxtEx-learning will be denoted by TxtGEx ("G" denotes the model of data processing where all data presented so far is accessible, including the order of presentation; "G" stands for Gold, who first formalized this method of data processing [27]).

We are now ready to give our unified notation.

A *learner* is a partial computable function.

A *language* is a ce set $L \subseteq \mathbb{N}$. Any total function $T : \mathbb{N} \to \mathbb{N} \cup \{\#\}$ is called a *text*. For any given language L, a *text for* L is a text T such that $\mathrm{content}(T) = L$. With $\mathbf{Txt}(L)$ we denote the set of all texts for L.

For this paper, a *learning criterion* is a triple (α, β, δ), where α, β and δ are of the appropriate type; we proceed by defining these types, before returning to the formal definition of learning criteria.

A *sequence generating operator* is an operator β taking as arguments a function h (the learner) and a text T and that outputs a function p. We call p the *learning sequence* of h given T. Intuitively, β defines how a learner processes a given text to produce a sequence of conjectures.

We define the sequence generating operators **G**, **Psd**, **Sd**, **ItCtr** and **It** as follows. **G**, **Psd**, **Sd**, **ItCtr** and **It**, respectively, stand for Gold [27], partially set-driven [43,25,26,31], set-driven [47,31] iterative with counter [21] and iterative [47,48], respectively. For all h, T, i,

$$\mathbf{G}(h,T)(i) = h(T[i]);$$

$$\mathbf{Psd}(h,T)(i) = h(\mathrm{content}(T[i]), i);$$

$$\mathbf{Sd}(h,T)(i) = h(\mathrm{content}(T[i]));$$

$$\mathbf{ItCtr}(h,T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0;\ ^{11} \\ h(\mathbf{ItCtr}(h,T)(i-1), T(i-1), i-1), & \text{otherwise;} \end{cases}$$

$$\mathbf{It}(h,T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0; \\ h(\mathbf{It}(h,T)(i-1), T(i-1)), & \text{otherwise.} \end{cases}$$

We note in passing another family of sequence generating operator called *bounded example memory*, denoted $(\mathbf{Bem}_n)_{n\in\mathbb{N}}$, due to John Canny as cited in [38] (see also [14]), where a learner may store up to a fixed number n of seen data points for later use.

For Bounded Memory States learning with $n \geq 1$ states, learners are functions of the kind $\langle h, f\rangle$, i.e., learners with two outputs: the first for a new conjecture, the second for a new memory state. Given such a learner $\langle h, f\rangle$ and a text T, we define recursively the $\mathbf{BMS}_n$ learning sequence p and the sequence q of states[12] of $\langle h, f\rangle$ given T thus.

$$p(0) = h(\emptyset); \tag{3}$$

$$q(0) = f(\emptyset); \tag{4}$$

$$\forall i : p(i+1) = h(q(i), T(i)); \tag{5}$$

$$\forall i : q(i+1) = \min(n-1, f(q(i), T(i))). \tag{6}$$

11 h on $\emptyset$ provides an initial conjecture.

12 Without loss of generality, the set of states is $\{0, \ldots, n-1\}$ and 0 is the initial state.

The sequence generating operator $\mathbf{BMS}_n$ is defined accordingly. $\mathbf{BMS}_*$ learning replaces (6) by

$$\forall i : q(i+1) = f(q(i), T(i)), \tag{7}$$

and returns as conjecture sequence $\lambda x.\uparrow$ if $\mathrm{rng}(q)$ is infinite and p otherwise.

Memoryless Feedback learning, as given in [11], is a learning criterion where the learner works in two stages. In the first stage, the learner is presented a datum and uses it to compute a finite *set*. In the second stage, the learner computes a new conjecture, given the same datum and, additionally, for each element x of the finite set computed in the first stage, an indicator of whether x occurred previously in the current text.

Intuitively, each element x in the set resulting from the first stage represents the question "have I seen datum x previously?".

Variants of Memoryless Feedback learning, where the size of each such set is bounded by a fixed parameter $n \in \mathbb{N}$, are also studied in [11]. Herein, we additionally study a variant where a learner is allowed to make queries *sequentially*, which we call memoryless *recall* learning (MLR). Variants of feedback learning where the learner has the additional information of the last conjecture made (as in iterative learning) are called *feedback learning* and the associated family of sequence generating operators is denoted $(\mathbf{Fb}_n)_{n\in\mathbb{N}}$ [48,36,14]. The variant allowing sequential recalls is called *recall learning*, denoted $(\mathbf{Rcl}_n)_{n\in\mathbb{N}}$, and is not studied in the prior literature.

We will not give formal details for modeling the associated sequence generating operators. Instead, we employ an informal query function `rcl` described below. For each $n \in \mathbb{N}\cup\{*\}$, let $\mathbf{MLF}_n$ be the sequence generating operator associated with allowing n *parallel* queries (feedback learning). Further, for all $m \in \mathbb{N}\cup\{*\}$, we let $\mathbf{MLR}_m$ be the sequence generating operator associated with allowing m *sequential* queries (recall learning).

As indicated, for specifying learners with feedback or recall queries, we introduce the use of `rcl` as follows.[13]

For $a, b, c \in \mathcal{P}$ and $d \in \mathbb{N}$ we will frequently make statements such as the following.

$$\forall x : \varphi_d(x) = \begin{cases} a(x), & \text{if } \mathtt{rcl}(c(x)); \\ b(x), & \text{otherwise.} \end{cases} \tag{8}$$

Intuitively, this means that φ_d on input (new datum) x will first recall $c(x)$, and then, if $c(x)$ was seen previously, output $a(x)$, otherwise $b(x)$. Furthermore, for a finite set D, we use $\mathtt{rcl}(D)$ to denote the set $\{x \in D \mid \mathtt{rcl}(x)\}$.

Successful learning might require the learner to observe certain restrictions, for example non-U-shapedness. These restrictions are formalized in our next definition.

A *sequence acceptance criterion* is a predicate δ on a learning sequence and a text. We give the following examples: Explanatory learning [27] (**Ex**), Vacillatory learning [10,31] (**Fex**), Behaviorally correct learning [19,39] (**Bc**), Postdictively complete learning [2,3,1] (**Pcp**), non-U-shaped [4] (**NU**), strongly non-U-shaped [20] (**SNU**), class decisive (**CDec**) and strongly class decisive (**SCDec**).[14] Formally, we let, for all p, T,

$$\mathbf{Ex}(p,T) \Leftrightarrow [\forall^\infty i : W_{p(i)} = \mathrm{content}(T) \wedge p(i) = p(i+1)];$$

$$\forall n : \mathbf{Fex}_n(p,T) \Leftrightarrow \begin{array}{l}[\exists \text{ a set } D \text{ of at most } n \text{ grammars for } \mathrm{content}(T) : \\ \forall^\infty i : p(i) \in D];\end{array}$$

$$\mathbf{Bc}(p,T) \Leftrightarrow [\forall^\infty i : W_{p(i)} = \mathrm{content}(T)];$$

$$\mathbf{Pcp}(p,T) \Leftrightarrow [\forall i : \mathrm{content}(T[i]) \subseteq W_{p(i)}];$$

$$\mathbf{NU}(p,T) \Leftrightarrow [\forall i : W_{p(i)} = \mathrm{content}(T) \Rightarrow W_{p(i+1)} = W_{p(i)}];$$

$$\mathbf{SNU}(p,T) \Leftrightarrow [\forall i : W_{p(i)} = \mathrm{content}(T) \Rightarrow p(i+1) = p(i)];$$

$$\mathbf{CDec}(p,T) \Leftrightarrow [\forall i \le j \le k : W_{p(i)} = W_{p(k)} \Rightarrow W_{p(j)} = W_{p(i)}];$$

$$\mathbf{SCDec}(p,T) \Leftrightarrow [\forall i \le j \le k : W_{p(i)} = W_{p(k)} \Rightarrow p(j) = p(i)].$$

We combine any two sequence acceptance criteria δ and δ' by intersecting them, and we denote this combination by $\delta\delta'$.

A *learning restriction* is a predicate α on a learner and a language, parameterized with a sequence generating operator β. We write the parameter β as a subscript and give the following examples.

- No restriction: The constantly true predicate of the appropriate type **T**.
- Total Learner: $\forall \beta, h, L : \mathcal{R}_\beta(h, L) \Leftrightarrow h \in \mathcal{R}$.

Similarly to the total learner, we denote the restriction to linear time computable learners with $\mathbf{LinF}_\beta$.

We will now give the formal definitions for successful learning.

Definition 2.1. A *learning criterion* is a triple (α, β, δ) such that α is a learning restriction, β a sequence generating operator and δ a sequence acceptance criterion. We also write $\alpha\mathbf{Txt}\beta\delta$ to denote this learning criterion. Furthermore, to denote criteria as in the previous literature, we may list α, β and δ in a different order.[15]

[13] The use of `rcl` is "syntactic sugar".

[14] In [38], the notion of *decisiveness* was introduced. The difference between decisive and *class* decisive learning is that, for the latter, a learner need only be decisive on the class to be learned. This notation was used in [31] in the context of function learning.

[15] In particular, **NU** is usually denoted as a prefix to a learning criterion.

Let a learning criterion $I = (\alpha, \beta, \delta)$ be given. We use α_I, β_I and δ_I to denote α, β and δ, respectively. For δ' a sequence acceptance criterion, we let $\delta' I = I\delta' = (\alpha, \beta, \delta\delta')$. For a learner h we say that h *I-learns* a language L iff the restriction α holds for h and L given β and, for all texts T for L, the learning sequence of h on T (as defined by β) and the text T fulfill restriction δ.

For each learning criterion I, we denote the class of all languages I-learned by h with $I(h)$. Abusing notation, we use $\alpha\mathbf{Txt}\beta\delta$ to denote the set of all classes of languages I-learnable by some learner (as well as the learning criterion).

We omit α if $\alpha = \mathbf{T}$.

From [17] we take the following definition (inspired by the notion of "stabilizer segments" [26]).

Let β be sequence generating operator. For a learner h and a language L, we define a *β-sink of h on L* to be a conjecture e such that

$$\forall T \in \mathbf{Txt}(L)\forall i_0 : [\beta(h, T)(i_0) = e \Rightarrow (\forall i \geq i_0)(\beta(h, T)(i) = e)]. \tag{9}$$

Intuitively, a sink is a conjecture never abandoned on texts for L.

We use this notion of a sink to give another restriction on learning. Intuitively, it requires a learner to not just converge to any correct conjecture, but to a sink. For all β, h, L,

$$\mathbf{Sink}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L)\exists e : (\beta(h, T) \rightarrow e \ \wedge \ e \text{ is a } \beta\text{-sink of } h \text{ on } L)].$$

Sink-stabilizing is of interest as strong non-U-shaped learning can be characterized in terms of sink-stabilizing learning in the theorem just below. This lemma is a special case of a theorem from [17]; we will use it in the proof of Theorem 3.11.

Lemma 2.2 (*Sink Locking Lemma*)**.** *Let $\beta \in \{\mathbf{G}, \mathbf{Psd}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{It}\}$ and $\alpha \in \{\mathbf{T}, \mathcal{R}\}$. Then*

$$\mathbf{Sink}\alpha\mathbf{Txt}\beta\mathbf{Ex} = \mathbf{SNU}\alpha\mathbf{Txt}\beta\mathbf{Ex}.$$

2.2. A programming system with linear time universal function

In this section we will develop a technique to separate learning criteria with *linear time computable* learners from other criteria.

We would like to encode complete sets of instructions for learners to perform in data. Encoding these instructions in the φ-system would, in general, require φ-universality to execute these instructions; this universality is computationally expensive [41, Section 3.1.6].

Frequently separation proofs do not require the full generality of the φ-system to encode the necessary instructions, and thus a learner might not need to have access to a φ-universality. We formalize this with complexity-bounded programming systems ψ which are much more restricted than both φ and standard complexity-bounded systems (the latter as in [33,41]). Our systems ψ crucially have linear time universality! However, they are strong enough to suffice for our associated separation results.

We proceed by giving a family of such systems. Theorem 2.3 will note its qualities, and the ψ/φ-Hybrid ORT given in Theorem 2.4 is extremely useful for several proofs below.

For each finite set of linear time computable functions D and each $n > 1$, $\psi^{D,n}$ will denote a programming system in which all elements of D are computable. Let a finite set $D = \{f_1, \ldots, f_k\}$ of linear time computable functions be given.

For all n, $\psi^{D,n}$-programs are basically trees of computation, where the depth of each tree is bounded by n. Each inner node of the tree has one of six types, defining its semantics. The type 0 nodes correspond to one of the base functions (all of D and one more function f_0, which will compute an s-m-n function in the $\psi^{D,n}$)-system), 1 to a projection, and 2 to a constant function; 3 is reserved for if-then-else, 4 for pairing and 5 for composition.

- Nodes of type 0, 1 or 2 have exactly one child; this child is a leaf and labeled, respectively, with the index of the function to use, the index of the projection or the associated constant.
- Nodes of type 3 have exactly three children, none of them a leaf, denoting the condition and the two branches of the if-then-else.
- Nodes of type 4 have exactly two children, none of them a leaf, denoting the two components of the pair.
- Nodes of type 5 also have two children (none a leaf), denoting the $\psi^{D,n}$-programs to compose.

Furthermore, each tree is labeled by some number; this number has no semantic meaning and is used for technical purposes (to get a linear time invertible s-m-n function).

It is clear that all such labeled trees with type annotations can be coded into the natural numbers in an efficient way (see [41] for an example of efficient coding). The semantics are as suggested by the list above. Note that each function from the $\psi^{D,n}$)-system expects its input as a coded tuple of multiple inputs; it cannot access the *value* of this coded tuple, but has to apply projections on it.

We now define a linlin s-m-n function. Let f_0 be the following (linlin) function. Given a (coded) computation tree t as above and a number x, traverse the tree. For every projection node, if the index of the projection is >1, reduce that index by 1; otherwise, replace that projection node by a node of type 2, with associated constant x. Label the resulting tree with

the triple of the old label, the (code of) t and x. It is clear that this function is linlin (as there are only constantly many inner nodes on a tree of constant depth, and assuming all the codings of the tree to be efficient).

Note that f_0 does not increase the depth of the tree, and thus turns $\psi^{D,n}$-programs into $\psi^{D,n}$-programs.[16] It is easy to see that an application of f_0 on t and x returns a tree computation tree t' which, in each computation, works just like t, but with the first argument constantly x and all other arguments are filled with the arguments to t' (the first argument to t' will be used as the second argument to t and so on). Thus, f_0 is an s-m-n function, and we will refer to it also as s_ψ.

It is now straightforward to see the following theorem.

Theorem 2.3. *Let D be a finite set of linear time computable functions and let $n \in \mathbb{N}$. We have that*

(i) *$\psi^{D,n}$ has a linear time universal function;*
(ii) *all elements of D are $\psi^{D,n}$-computable;*
(iii) *$\psi^{D,n}$ has a $\psi^{D,n}$-computable instance s_ψ of s-m-n.*

Let s_φ be a linear time, linear time left-invertible instance of φ-s-m-n. Let D include unpairing, s_φ, a left-inverse for s_φ, a left-inverse for s_ψ and plus, minus, less than or equal, equality and logical connectives. From now on (including later sections), we fix

$$\psi = \psi^{D,100}. \tag{10}$$

Note that we chose 100 for no particular reason other than that it is easily big enough for our purposes.

For any function g for which there is an n such that g is $\psi^{D,n}$-computable, we let $d(g)$ be the minimum such n (intuitively, g describes the complexity of g in the ψ-system).

For any j and m, we say that a j-ary operator Θ is *m-ψ admissible* iff there are $t \in \mathbb{N}$, $1 \leq i_1, \ldots, i_t \leq j$, $r_0, \ldots, r_t \in \mathcal{R}$ such that

$$\forall g_1, \ldots, g_j : \Theta(g_1, \ldots, g_j) = \lambda x . r_0(g_{i_1}(r_1(x)), \ldots, g_{i_t}(r_t(x))), \tag{11}$$

and $2 + t + \max_{i \leq t} d(r_i) \leq 100 - m$.

Intuitively, an operator is m-ψ admissible if it is given by a ψ-program additionally with placeholders for "calls" to the arguments, where the total depth of the resulting program will still be low enough (below 100) if the arguments have defining programs with a low enough depth (at most m). In particular, we will use operators which make a few calls to the argument functions and otherwise are defined in terms of if-then-else, composition, equality, less than or equal and logical connectives, which will then easily lead to 80-ψ admissible binary operators.

Next we give our ψ/φ-Hybrid ORT.

Theorem 2.4 (ψ/φ-Hybrid ORT)**.** *There are ψ-computable functions $\overline{e}$ and $\overline{f}$ such that, for all effective operators Θ_0, Θ_1 in two arguments such that Θ_0 is 20-ψ admissible, there are strictly monotonically increasing linear time computable functions e, f such that*

$$\forall n, x : \psi_{e(n)}(x) = \Theta_0(e, f)(n, x); \tag{12}$$
$$\forall n, x : \varphi_{f(n)}(x) = \Theta_1(e, f)(n, x); \tag{13}$$

$\overline{e} \circ e$ and $\overline{f} \circ f$ are the identity and $d(\overline{e}), d(\overline{f}) \leq 5$.

Proof. Recall that s_ψ, s_φ are ψ computable s-m-n functions for ψ and φ, respectively. Further, let a_ψ and a_φ be such that, for all b_0, b_1, $a_\psi(b_0, b_1) = \lambda m . s_\psi(b_0, \langle b_0, b_1\rangle, m))$ and $a_\varphi(b_0, b_1) = \lambda m . s_\varphi(b_1, \langle\langle b_0, b_1\rangle, m\rangle)$.

There are a ψ-program d_0 and a φ-program d_1 such that, for all b, b_0, b_1, n, x,[17]

$$\psi_{d_0}(\langle\langle\langle b_0, b_1\rangle, n\rangle, x\rangle) = \Theta_0(a_\psi(b_0, b_1), a_\varphi(b_0, b_1))(\langle n, x\rangle); \tag{14}$$
$$\varphi_{d_1}(\langle\langle\langle b_0, b_1\rangle, n\rangle, x\rangle) = \Theta_1(a_\psi(b_0, b_1), a_\varphi(b_0, b_1))(\langle n, x\rangle). \tag{15}$$

Applying the respective s-m-n functions to the left-hand-sides of (14) and (15), we have, for all b_0, b_1, n, x,

$$\psi_{s_\psi(d_0, \langle\langle b_0, b_1\rangle, n\rangle)}(x) = \psi_{d_0}(\langle\langle\langle b_0, b_1\rangle, n\rangle, x\rangle); \tag{16}$$
$$\varphi_{s_\varphi(d_1, \langle\langle b_0, b_1\rangle, n\rangle)}(x) = \varphi_{d_1}(\langle\langle\langle b_0, b_1\rangle, n\rangle, x\rangle). \tag{17}$$

Hence, by (14) through (17), for all b, b_0, b_1, n, x,

$$\psi_{s_\psi(d_0, \langle\langle b_0, b_1\rangle, n\rangle)}(x) = \Theta_0(a_\psi(b_0, b_1), a_\varphi(b_0, b_1))(\langle n, x\rangle); \tag{18}$$
$$\varphi_{s_\varphi(d_1, \langle\langle b_0, b_1\rangle, n\rangle)}(x) = \Theta_1(a_\psi(b_0, b_1), a_\varphi(b_0, b_1))(\langle n, x\rangle). \tag{19}$$

16 Other approaches at defining an s-m-n function for $\psi^{D,n}$ in a more classical way might increase the depth.

17 For the existence of d_0, we use Θ_0 being 20-ψ admissible.

Then, setting $b_0 = d_0$ and $b_1 = d_1$ in (18) and (19), we have, for all n, x,

$$\psi_{s_\psi(d_0,\langle\langle d_0,d_1\rangle,n\rangle)}(x) = \Theta_0(a_\psi(d_0,d_1), a_\varphi(d_0,d_1))(\langle n,x\rangle); \tag{20}$$

$$\varphi_{s_\varphi(d_1,\langle\langle d_0,d_1\rangle,n\rangle)}(x) = \Theta_1(a_\psi(d_0,d_1), a_\varphi(d_0,d_1))(\langle n,x\rangle). \tag{21}$$

Clearly,

$$e = a_\psi(d_0,d_1); \tag{22}$$

$$f = a_\varphi(d_0,d_1) \tag{23}$$

define 1-1 linear time computable functions e, f as desired.

Let $\overline{s_\psi}$ and $\overline{s_\varphi}$ be ψ-computable left-inverses for s_ψ and s_φ, respectively. Then $\overline{e} = \pi_2 \circ \pi_2 \circ \overline{s_\psi}$ and $\overline{f} = \pi_2 \circ \pi_2 \circ \overline{s_\varphi}$ are as desired. □

We will use ψ for several proofs in this paper. In particular, we will construct classes learnable by a linear time computable learner in one learning criterion, and we use the ψ/φ-Hybrid ORT to diagonalize out of these classes in another learning criterion.

2.3. Transfer of inclusion results

From [11] we know that BMS-learning with an arbitrary (but finite) number of memory states per learning process is equivalent to iterative learning (see Theorem 3.5 below). But does the result that iterative learning does not require U-shaped behavior carry over to BMS-learning? In this section we present a theorem (the Transfer Lemma, Lemma 2.7) for formally transferring inclusion results. Often, one can say something like "It is easy to see that the proof we gave for X can be modified to be a proof for Y". This section is about making these kind of statements more formal, so that one can say "Because of X and the Transfer Lemma, we immediately get Y". In particular, we use the Transfer Lemma to obtain that U-shapes are unnecessary for BMS-learning (see Corollary 3.6).

For this, we introduce *inclusion qualifiers* with which we augment inclusions between learning criteria. For example, we use c as an indicator for a *constructive* inclusion (a learner for one learning criterion can be algorithmically turned into a learner for another).

The following definition formally presents all of our inclusion qualifiers of the present paper.

Definition 2.5. Let I, I' be learning criteria.

- We write $I \subseteq_c I'$ iff there is a computable operator Θ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I'(\Theta(h))$.[18]
- We write $I \subseteq_i I'$ iff there is a computable operator Θ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I'(\Theta(h))$ and, for all texts T, there is a text T' for the same language as T such that $\beta_I(h,T') = \beta_{I'}(\Theta(h),T)$.[19]
- We write $I \subseteq_{si} I'$ iff there is a computable operator Θ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I'(\Theta(h))$ and, for all texts T, there is a text T' for the same language as T such that, for all i, $W_{\beta_I(h,T')(i)} = W_{\beta_{I'}(\Theta(h),T)(i)}$.[20]
- We write $I \subseteq_{mi} I'$ iff there is a computable operator Θ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I'(\Theta(h))$ and, for all texts T, there is a text T' for the same language as T and a strictly monotone function r such that $\beta_I(h,T') \circ r = \beta_{I'}(\Theta(h),T)$.[21]
- We write $I \subseteq_{msi} I'$ iff there is a computable operator Θ such that, for all $h \in \mathcal{P}$, $I(h) \subseteq I'(\Theta(h))$ and, for all texts T, there is a text T' for the same language as T and a strictly monotone function r such that, for all i, $W_{\beta_I(h,T')(r(i))} = W_{\beta_{I'}(\Theta(h),T)(i)}$.[22]

Note that one could define many more such inclusion qualifiers; in particular, it might be interesting to consider cases where we require $T' = T$. However, for the present paper, the above definitions are sufficient.

We use $I =_c I'$ to denote $I \subseteq_c I'$ and $I' \subseteq_c I$, and so on. We denote with $I \underset{i}{\overset{c}{\leftrightarrows}} I'$ that $I \subseteq_c I'$ and $I' \subseteq_i I$, and so on.

We give the following definition regarding properties of sequence acceptance criteria.

Definition 2.6. A sequence acceptance criterion δ is called

- *monotonic* iff, for all strictly monotone functions r and $(p,L) \in \delta$, $(p \circ r, L) \in \delta$[23,24];
- *semantic* iff, for all $(p,L) \in \delta$ and p' such that $\forall i : W_{p(i)} = W_{p'(i)}$, $(p',L) \in \delta$.[25]

---

18 Intuitively, Θ gives us a constructive way to transform an I-learner into an at least as powerful I'-learner. The letter "c" stands for "constructive".

19 Intuitively, $\Theta(h)$ only constructs learning (conjecture) sequences that are constructed by h. The letter "i" stands for "identical".

20 The letters "si" stand for "semantically identical".

21 Intuitively, $\Theta(h)$ produces the same conjecture sequences as h, just maybe skipping conjectures. The letters "mi" stand for "monotonically identical". Note that this has nothing to do with *monotone learning* as discussed in the literature [28,49,35].

22 The letters "msi" stand for "monotonically semantically identical".

23 Intuitively, leaving out some conjectures does not prevent learning.

24 Note that, again, this definition of monotonic has no relation to monotone learning as discussed in the literature (see Footnote 21).

25 Intuitively, all that matters about the conjectures is their semantics.

It is easy to see that all sequence acceptance criteria explicitly given in this paper are monotonic; furthermore, **Pcp**, **NU** and **CDec** are semantic, while **Ex**, **SNU** and **SCDec** are not.

We now present the central lemma of this section, the Transfer Lemma.

Lemma 2.7 (*Transfer Lemma*)**.** *Let I, I' be learning criteria and δ a sequence acceptance criterion.*

$I \subseteq_i I' \Rightarrow I\delta \subseteq_i I'\delta$.
$I \subseteq_{si} I' \wedge \delta$ *semantic restriction* $\Rightarrow I\delta \subseteq_{si} I'\delta$.
$I \subseteq_{mi} I' \wedge \delta$ *monotonic restriction* $\Rightarrow I\delta \subseteq_{mi} I'\delta$.
$I \subseteq_{msi} I' \wedge \delta$ *monotonic, semantic restriction* $\Rightarrow I\delta \subseteq_{msi} I'\delta$.

The proof is straightforward.

3. Bounded memory states learning (BMS)

In this section we discuss and present our results on bounded memory states learning.

The first main theorem of this section is Theorem 3.3, which solves Problem 40 from [11], as mentioned in Section 1. The theorem implies that U-shapes are necessary for full learning power whenever at least three memory states are available.

In Theorem 3.5 we state the equivalence of learning with finitely many memory states and iterative learning [11]. We use this result and the Transfer Lemma (Lemma 2.7) to show that U-shapes are unnecessary for learning with finitely many memory states (Corollary 3.6).

The second main theorem of this section is Theorem 3.10, which implies that there is a class learnable with two memory states which is not *strongly* non-U-shapedly learnable even with an iterative learner.

In contrast to the second main result, we have Theorem 3.11: for *total* learners, all learning with a bounded number of memory states can be done strongly non-U-shapedly with a finite number of memory states (finite on each text).

The remaining results concern BMS-learning with only one or two memory states.

For BMS-learning we do not address the question of learning infinite ce sets only. However, it is easy to see that each separation we give is witnessed by a set of infinite ce sets.

The following definition is useful for the proof of Theorem 3.3. Lemma 3.2 gives some useful observations regarding this definition.

Definition 3.1. Let $f \in \mathcal{P}$. We think of f as the state transition function of a BMS-learner. For this definition, we let $f^* \in \mathcal{P}$ be such that $f^*(\emptyset) = 0$ and $\forall \sigma, x : f^*(\sigma \diamond x) = f(f^*(\sigma), x)$.[26]

For all $g \in \mathcal{R}$, let Y_g be such that

$$Y_g = \{j \mid \ (\forall k \leq j+1 : f^*(g[k])\downarrow) \ \wedge \ (\forall k \leq j) f^*(g[k]) \neq f^*(g[j+1])\}. \tag{24}$$

Intuitively, Y_g is the set of all j such that f, when presented the text g, changes into a previously not visited state after seeing element $g(j)$.

Lemma 3.2. *Let $f \in \mathcal{P}$. Let f^* be as in Definition 3.1 above. For $g \in \mathcal{P}$, we will below refer to the following statement.*

$$\forall k : f^*(g[k])\downarrow. \tag{25}$$

(i) *There is a computable operator $\Theta : \mathcal{P} \to \mathcal{P}$[27] such that, for all $g \in \mathcal{R}$, if (25), then $\Theta(g)$ is total and decides Y_g.*
(ii) *If $\mathrm{rng}(f)$ is finite, then, for all $g \in \mathcal{R}$, Y_g is finite.*
(iii) *For all $g \in \mathcal{R}$, if (25), then*

$$\forall \tau \subset g \ \exists \sigma \in \mathbb{S}\mathrm{eq}(\{g(j) \mid j \in Y_g\}) : f^*(\tau) = f^*(\sigma). \tag{26}$$

Proof. Obviously, Θ as follows satisfies (i).

$$\forall x, j : \Theta(\varphi_x)(j) = \begin{cases} \uparrow, & \text{if } (\exists k \leq j+1 : f^*(\varphi_x[k])\uparrow); \\ 1, & \text{else if } (\forall k \leq j) f^*(\varphi_x[k]) \neq f^*(\varphi_x[j+1]); \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

(ii) is easy to see.

(iii) can be seen by $\subseteq$-induction on τ as follows. Let $\tau \subset g$ be such that, for all $\hat{\tau} \subset \tau$, $\exists \sigma \in \mathbb{S}\mathrm{eq}(\{g(j) \mid j \in Y_g\}) : f^*(\hat{\tau}) = f^*(\sigma)$. Let $\tau_0 \subseteq \tau$ be the $\subseteq$-minimum such that $f^*(\tau_0) = f^*(\tau)$. The conclusion is trivial if $\tau_0 = \emptyset$. Else, $\mathrm{len}(\tau_0) - 1 \in Y_s$. Let $\sigma \in \mathbb{S}\mathrm{eq}(\{s(j) \mid j \in Y_s\})$ such that $f^*(\tau_0^-) = f^*(\sigma)$. Therefore,

$$\begin{aligned} f^*(\tau) \quad &= f^*(\tau_0) = f(\mathrm{last}(\tau_0), f^*(\tau_0^-)) = f(\mathrm{last}(\tau_0), f^*(\sigma)) \\ &= f^*(\sigma \diamond \mathrm{last}(\tau_0)). \end{aligned} \tag{28}$$

Hence, $(\sigma \diamond \mathrm{last}(\tau_0)) \in \mathbb{S}\mathrm{eq}(\{g(j) \mid j \in Y_g\})$ is the desired sequence witnessing (iii) for τ. □

[26] Note that $f^*(\emptyset) = 0$ is the initial state of any given **BMS**-learner.

[27] I.e. there exists a computable $f \in \mathcal{R}$ such that, for all φ-programs q, $\Theta(\varphi_q) = \varphi_{f(q)}$ [42].

Contrasting $\mathbf{TxtBMS}_1\mathbf{Ex} = \mathbf{NUTxtBMS}_1\mathbf{Ex}$ and $\mathbf{TxtBMS}_2\mathbf{Ex} = \mathbf{NUTxtBMS}_2\mathbf{Ex}$ from [11], we have the following theorem, solving an open problem, Problem 40, from [11].

Theorem 3.3. *We have*

$$\mathbf{TxtBMS}_3\mathbf{Ex} \setminus \bigcup_{n>0} \mathbf{NUTxtBMS}_n\mathbf{Ex} \neq \emptyset.$$

Proof. Let $h_0 \in \mathcal{P}$ be such that

$$\forall v, x : h_0(x, v) = \begin{cases} \langle ?, v\rangle, & \text{if } x = \#; \\ \varphi_x(v), & \text{otherwise.} \end{cases} \tag{29}$$

Let $\mathcal{L} = \mathbf{TxtBMS}_3\mathbf{Ex}(h_0)$. Let $n > 0$. Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{NUTxtBMS}_n\mathbf{Ex}$, as witnessed by $\langle h, f\rangle$ (h returns the new conjecture, f the new state). Suppose, without loss of generality, $\mathrm{rng}(f)$ is finite. Let f^* be as in Definition 3.1 above. Let h^* be such that $h^*(\emptyset) = ?$ and $\forall \sigma, x : h^*(\sigma \diamond x) = h(f^*(\sigma), x)$. Let, for all $s \in \mathcal{R}$, Y_s be as in Definition 3.1 above, and let Θ be as shown existent in Lemma 3.2(i).

Intuitively, $h^*(\sigma)$ is the hypothesis of the learner after seeing σ as input.

We define ce sets P, Q in uniform dependence of $r, s \in \mathcal{P}$ (we abbreviate, for all i, $s_i = \lambda j.s(i, j)$) such that, for all a, b, σ, τ,

$$Q(b, \tau) \Leftrightarrow \exists \xi \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(\tau)) : \emptyset \neq W_{h^*(\tau \diamond r(b) \diamond \xi)} \cap (\mathrm{rng}(s_b) \setminus \mathrm{rng}(\tau)); \tag{30}$$

and

$$P(a, b, \sigma, \tau) \Leftrightarrow \begin{cases} a \neq b & \wedge \\ \sigma \in \mathbb{S}\mathrm{eq}(\{s_a(j) \mid \Theta(s_a)(j) = 1\}) & \wedge \\ \tau \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(s_b)) & \wedge \\ f^*(\sigma) = f^*(\tau) & \wedge \\ f^*(\sigma \diamond r(a)) = f^*(\tau \diamond r(b)) & \wedge \\ Q(b, \tau). \end{cases} \tag{31}$$

Fix a ce-index for P. By 1-1 ORT, there are 1-1 $e, r, s, t, y, z \in \mathcal{R}$ with pairwise disjoint ranges and $p \in \mathbb{N}$ such that a number of restrictions are satisfied. The first group of restrictions is given by the following four equations. $\forall x, i$:

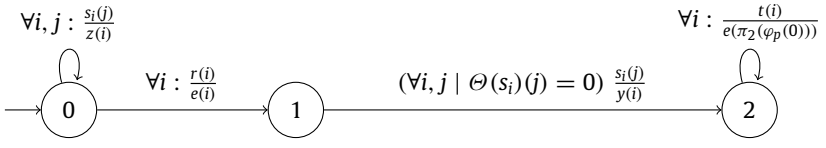
$$\varphi_p(x) = \mu\langle a, b, \sigma, \tau, d\rangle.[P(a, b, \sigma, \tau) \text{ in} \leq d \text{ steps}]; \tag{32}$$

$$W_{y(i)} = \mathrm{rng}(s_i) \cup \{r(i)\}; \tag{33}$$

$$W_{z(i)} = \mathrm{rng}(s_i); \tag{34}$$

$$W_{e(i)} = \{r(i)\} \cup \mathrm{rng}(t) \cup \begin{cases} \emptyset, & \text{if } \varphi_p(0)\uparrow; \\ \mathrm{content}(\pi_3(\varphi_p(0))), & \text{else if } i = \pi_1(\varphi_p(0)); \\ \mathrm{content}(\pi_4(\varphi_p(0))), & \text{otherwise.} \end{cases} \tag{35}$$

The second group of restrictions in our application of ORT is indicated by a labeled graph using vertices $\{0, 1, 2\}$. For all elements $x \in \mathrm{rng}(r) \cup \mathrm{rng}(s) \cup \mathrm{rng}(t)$ and $\ell \in \mathbb{N}$, an edge from vertex v to vertex w labeled $\frac{x}{\ell}$ (we use this kind of label for readability; $\frac{x}{\ell}$ is not to be confused with a fraction) in the graph just below adds the restriction $\varphi_x(v) = \langle \ell, w\rangle$ as part of our application of ORT.



The third and last group of restrictions is as follows.

$$(\forall i, j \mid \Theta(s_i)(j)\uparrow)\varphi_{s_i(j)}(1)\uparrow \tag{36}$$

and, for all $x \in \mathrm{rng}(r) \cup \mathrm{rng}(s) \cup \mathrm{rng}(t)$ and vertices v such that $\varphi_x(v)$ was not previously specified, we have the restriction $\varphi_x(v) = \langle ?, v\rangle$.

It is easy to verify that these three groups are not contradictory and embody a valid application of ORT.

The above graph now allows us to easily determine whether certain interesting subsets of $\mathrm{rng}(r) \cup \mathrm{rng}(s) \cup \mathrm{rng}(t)$ are in $\mathcal{L}$. For example,

$$\forall i \in \mathbb{N} : \mathrm{rng}(s_i) = W_{z(i)} \in \mathcal{L}. \tag{37}$$

Statement (37) can be derived with the help of the graph, as it shows that, for all i, M, on any element from $\mathrm{rng}(s_i)$, stays in state 0 and outputs $z(i)$ as hypothesis.

From (37) we get, for all i and $\sigma \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(s_i))$, $f^*(\sigma)\downarrow$; in particular, for all i, we have (25) with s_i in the place of g. Therefore, for all i, using Lemma 3.2(i), $\Theta(s_i)$ is total and (36) is vacuous. By Lemma 3.2(ii), for each i, Y_{s_i} is finite, and, thus,

the above graph easily shows

$$\forall i \in \mathbb{N} : \mathrm{rng}(s_i) \cup \{r(i)\} = W_{y(i)} \in \mathcal{L}. \tag{38}$$

Hence,

$$\forall i \in \mathbb{N}, \forall \rho \in \mathbb{S}\mathrm{eq}(W_{y(i)}) : \langle h^*, f^* \rangle(\rho)\downarrow. \tag{39}$$

Claim 1. $\forall b \in \mathbb{N}\, \exists \tau \subset s_b : Q(b, \tau)$.

Proof of Claim 1. Let $b \in \mathbb{N}$. By the Pigeonhole Principle, as $\mathrm{rng}(f)$ is finite, there is v such that $\exists^\infty k : f^*(s_b[k]) = v$. By (37), there is j_0 such that

$$W_{h^*(s_b[j_0])} = W_{z(b)}; \text{ and } \forall j \geq j_0 : h^*(s_b[j]) \in \{?, h^*(s_b[j_0])\}. \tag{40}$$

Let $j_1 > j_0$ be such that $f^*(s_b[j_1]) = v$. By (38), there is $k \in \mathbb{N}$ such that

$$W_{h^*(s_b[j_1] \diamond r(b) \diamond s_b[k])} = W_{y(b)}. \tag{41}$$

Let $j_2 > k$ be such that $f^*(s_b[j_2]) = v$. Then

$$W_{h^*(s_b[j_2] \diamond r(b) \diamond s_b[k])} = W_{y(b)}. \tag{42}$$

Hence, $\exists \tau \subset s_b : Q(b, \tau)$ as witnessed by $s_b[j_2]$ for τ and $s_b[k]$ for ξ. □ (for Claim 1)

Claim 2. $\varphi_p(0)\downarrow$.

Proof of Claim 2. For the proof of this claim only, for each $b \in \mathbb{N}$, we fix τ_b as shown existent by Claim 1.

There are only finitely many pairs of states (elements of $\mathrm{rng}(f)$), while there are infinitely many $b \in \mathbb{N}$. Hence, by the Pigeonhole Principle, there are $a, b \in \mathbb{N}$ such that $a \neq b, f^*(\tau_a) = f^*(\tau_b)$ and $f^*(\tau_a \diamond r(a)) = f^*(\tau_b \diamond r(b))$. To show the claim, we use Lemma 3.2(iii) with s_a in place of g to replace τ_a by σ such that $\sigma \in \mathbb{S}\mathrm{eq}(\{s_a(j) \mid j \in Y_{s_a}\})$ and $f^*(\tau_a) = f^*(\sigma)$. Thus,

$$f^*(\sigma \diamond r(a)) = f^*(\tau_a \diamond r(a)) = f^*(\tau_b \diamond r(b)) \tag{43}$$

and

$$f^*(\sigma) = f^*(\tau_a) = f^*(\tau_b). \tag{44}$$

Now we see that $P(a, b, \sigma, \tau_b)$, as

$a \neq b$	by choice of a, b;
$\sigma \in \mathbb{S}\mathrm{eq}(\{s_a(j) \mid \Theta(s_a)(j) = 1\})$	by choice of σ and $\Theta(s_a)$ decides Y_{s_a};
$\tau_b \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(s_b))$	as $\tau_b \subset s_b$;
$f^*(\sigma) = f^*(\tau_b)$	as (44);
$f^*(\sigma \diamond r(a)) = f^*(\tau_b \diamond r(b))$	because of (43);
$Q(b, \tau)$	by choice of τ_b. □ (for Claim 2)

Let

$$\langle a, b, \sigma, \tau, d \rangle = \varphi_p(0), \tag{45}$$

and let ξ be as stated existent by $Q(b, \tau)$. We have

$$W_{e(a)} = \{r(a)\} \cup \mathrm{rng}(t) \cup \mathrm{content}(\sigma), \tag{46}$$

and

$$W_{e(b)} = \{r(b)\} \cup \mathrm{rng}(t) \cup \mathrm{content}(\tau). \tag{47}$$

It is easy to see (from the graph above) that $W_{e(a)}, W_{e(b)} \in \mathcal{L}$.

Let

$$\rho_a = \sigma \diamond r(a), \tag{48}$$

$$\rho_b = \tau \diamond r(b), \tag{49}$$

$$T_a = \rho_a \diamond t, \tag{50}$$

$$T_b = \rho_b \diamond t, \tag{51}$$

and

$$T_b' = \rho_b \diamond \xi \diamond t. \tag{52}$$

Then T_a is a text for $W_{e(a)}$ and T_b, T'_b are texts for $W_{e(b)}$. As $f^*(\rho_a) = f^*(\rho_b)$, and, as h has to identify $W_{e(a)}$ from $\rho_a \diamond t = T_a$ and $W_{e(b)}$ from $\rho_b \diamond t = T_b$, we have, for all $k > 0$, $h^*(\rho_a \diamond t[k]) =?$ and $h^*(\rho_b \diamond t[k]) =?$. Thus, there is ℓ such that $W_{h^*(\rho_b[\ell])} = W_{e(b)}$.
To see that we have a U-shape with text T'_b:

(i) $\exists \ell : W_{h^*(\rho_b[\ell])} = W_{e(b)}$ (as stated just above);
(ii) $W_{h^*(\rho_b \diamond \xi)} \neq W_{e(b)}$ (by ξ witnessing $Q(b, \tau_b)$); and
(iii) $\exists \ell : W_{h^*(\rho_b \diamond \xi \diamond t[\ell])} = W_{e(b)}$ (as T'_b is a text for $W_{e(b)} \in \mathcal{L}$).

This is a contradiction to $\mathcal{L} \in \mathbf{NUBMS}_n(\langle h, f\rangle)$. □

With $\mathbf{BMS}'_*$ we denote the variant of $\mathbf{BMS}_*$ where a learner is not allowed to return to a state that it left earlier. We give the following lemma to help with a later proof. Recall that the decorations on the equality signs just below are inclusion qualifiers, introduced in Definition 2.5.

Lemma 3.4. *We have*

$$\mathbf{TxtBMS}_*\mathbf{Ex} =_{mi} \mathbf{TxtBMS}'_*\mathbf{Ex}. \tag{53}$$

Proof. The inclusion "$\supseteq$" is trivial.
Regarding "$\subseteq$", let $h, f \in \mathcal{P}$. Let $h', f' \in \mathcal{P}$ be such that, for all k, D, x,

$$h'(\emptyset) = h(\varepsilon); \tag{54}$$
$$f'(\emptyset) = \langle f(\emptyset), \{f(\emptyset)\}\rangle; \tag{55}$$
$$h'(\langle k, D\rangle, x) = h(k, x); \tag{56}$$
$$f'(\langle k, D\rangle, x) = \begin{cases} \langle h(k,x), \langle k, D\rangle\rangle, & \text{if } f(k,x) \in D; \\ \langle h(k,x), \langle f(k,x), D \cup \{f(k,x)\}\rangle\rangle, & \text{otherwise.} \end{cases} \tag{57}$$

Note that $\langle h', f'\rangle$ will never visit the same state again after leaving it.
Furthermore, it is straightforward to see inductively the following. Suppose $\langle h', f'\rangle$ on a text for a language L outputs, for some k, D, $\langle k, D\rangle$. Then, for all $k' \in D$, there is a sequence σ of elements from L such that h, starting from state k', will end in state k after processing σ.
Thus, for all languages L and each text T for L, we can construct a text $T' = T(0)\sigma_0 T(1)\sigma_1 T(2)\ldots$ for L such that, for all i, $h'^*(T[i]) = h^*(T(0)\sigma_0 \ldots T(i-1)\sigma_{i-1})$.
Therefore, Lemma 3.4 follows. □

Using Lemma 3.4, we can assume, without loss of generality, that $\mathbf{BMS}_*$-learners never come back to a once abandoned state.
The following theorem was mentioned without proof (and without inclusion qualifiers) in [11]. We will need it (with associated inclusion qualifiers, see Definition 2.5) for later conclusions.

Theorem 3.5 (*[11]*)**.** *We have*

$$\mathbf{TxtBMS}_*\mathbf{Ex} \underset{i}{\overset{msi}{\rightleftarrows}} \mathbf{TxtItEx}. \tag{58}$$

Proof. The inclusion "$\supseteq$" is straightforward by storing the last hypothesis in the state.
Regarding "$\subseteq$", let $\langle h, f\rangle \in \mathcal{P}$. We use a 1-1 computable function pad such that, for all e, x, $W_{\mathrm{pad}(e,x)} = W_e$. Let $h' \in \mathcal{P}$ be such that

$$h'(\emptyset) = \mathrm{pad}(h(\emptyset), f(\emptyset)); \tag{59}$$
$$h'(\mathrm{pad}(e,k), x) = \mathrm{pad}(h(k,x), f(k,x)). \tag{60}$$

A straightforward induction shows, for all texts T and all i, $\mathbf{BMS}(h, T)(i)$ is semantically equivalent to $\mathbf{It}(h', T)(i)$ (the latter is a padded version of the former). Using Theorem 3.4, both claims follow. □

Note that Theorem 3.5 shows the equivalence of *unrestricted* **It**- and $\mathbf{BMS}_*$-learning, while it leaves open whether $\mathbf{BMS}_*$-learning allows for learning more classes of languages than **It**-learning in cases where not just **Ex** is the sequence acceptance criterion, but **Ex** intersected with some non-semantic or non-monotonic sequence acceptance criterion. For example, **SNU** is a non-semantic sequence acceptance criterion; see the paragraph just after Corollary 3.6 below.
Furthermore, note that both Lemma 3.4 and its proof carry over to the version of $\mathbf{BMS}_*$-learning, where a learner has to use only finitely many memory states on *any* text (not just on texts for learned languages); Theorem 3.5 carries over with the modification of *confident* **TxtItEx**, as already stated in [11] (a learner is confident iff it converges on any text, regardless for what language; see [38,31]).
As we gave the equivalence of **It**- and $\mathbf{BMS}_*$-learning with inclusion qualifiers, we can derive the following corollary. Not employing the inclusion qualifiers, we have $\mathbf{TxtBMS}_*\mathbf{Ex} \subseteq \mathbf{TxtItEx} \subseteq \mathbf{NUTxtItEx}$ (by Theorem 3.5 and [21]); furthermore, *using the inclusion qualifier*, we can apply the Transfer Lemma (Lemma 2.7) to get $\mathbf{NUTxtItEx} \subseteq \mathbf{NUTxtBMS}_*\mathbf{Ex}$. Altogether, this gives the following corollary.

Corollary 3.6. *We have*

$$\mathbf{NUTxtBMS}_*\mathbf{Ex} = \mathbf{TxtBMS}_*\mathbf{Ex}.$$

One might expect to be able to use the Transfer Lemma to show $\mathbf{SNUTxtBMS}_*\mathbf{Ex} \subset \mathbf{TxtBMS}_*\mathbf{Ex}$ using $\mathbf{SNUTxtItEx} \subset \mathbf{TxtItEx}$ – analogously to the justification of Corollary 3.6. However, note the "s" for *semantic* in the inclusion qualifier in one direction of Theorem 3.5, which does not allow for transferring results regarding non-semantic sequence acceptance criteria (such as **SNU**).

However, we can give a much stronger result, which also entails that even for *two* memory states, we cannot avoid U-shapes in a strong way, see Theorem 3.10.[28]

But before we get to Theorem 3.10, we give Definition 3.7 and Lemma 3.8.

Definition 3.7. For all $h^* \in \mathcal{P}$, h^* is called *canny* iff, for all σ, (i) through (ii) below.

(i) $h^*(\sigma)\downarrow \Rightarrow h^*(\sigma) \in \mathbb{N}$, i.e., h^* never outputs ?.
(ii) $h^*(\sigma \diamond \#) = h^*(\sigma)$.
(iii) For all $x \in \mathbb{N}$, if $h^*(\sigma \diamond x) \neq h^*(\sigma)$, then, for all $\tau \supseteq \sigma \diamond x$, $h^*(\tau \diamond x) = h^*(\tau)$.

With any learner h we can associate a function h^* that maps a sequence σ of data input to the current conjecture of learner h after seeing σ; with **Can** we denote the learner restriction where only learners are allowed which, when starred, are canny.

Lemma 3.8 (*Theorem 1 in [21]*)**.** *We have*

$$\mathbf{CanTxtItEx} = \mathbf{TxtItEx}.$$

The proof of [21, Theorem 1] gives a construction to turn an iterative learner into a canny one learning at least as much. It is easy to see that this construction preserves strong non-U-shapedness. Note that the construction does not, in general, preserve (not necessarily strong) non-U-shapedness.

Using a slight modification of the proof of [21, Theorem 1], we get the following remark.

Remark 3.9. We have

$$\mathbf{CanSNUTxtItEx} = \mathbf{SNUTxtItEx};$$
$$\mathbf{CanTxtBMS}_*\mathbf{Ex} = \mathbf{TxtBMS}_*\mathbf{Ex};$$
$$\mathbf{CanSNUTxtBMS}_*\mathbf{Ex} = \mathbf{SNUTxtBMS}_*\mathbf{Ex};$$
$$\mathbf{CanTxtBMS}'_*\mathbf{Ex} = \mathbf{TxtBMS}'_*\mathbf{Ex};$$
$$\mathbf{CanSNUTxtBMS}'_*\mathbf{Ex} = \mathbf{SNUTxtBMS}'_*\mathbf{Ex}.$$

Theorem 3.10. *We have*

$$\mathbf{TxtBMS}_2\mathbf{Ex} \setminus \mathbf{SNUTxtBMS}_*\mathbf{Ex} \neq \emptyset.$$

Proof. Let h_0 as in the proof of Theorem 3.3. Let $\mathcal{L} = \mathbf{TxtBMS}_2\mathbf{Ex}(h_0)$. Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{SNUTxtBMS}_*\mathbf{Ex}$, as witnessed by $\langle h, f\rangle$. Without loss of generality, by Lemma 3.4 and the Transfer Lemma (Lemma 2.7), we assume $\langle h, f\rangle$ never to go back to a previous conjecture. By Remark 3.9, any strongly non-U-shaped $\mathbf{TxtBMS}'_*\mathbf{Ex}$-learner can be assumed to be canny (without loss of generality), so suppose h is canny.

Let h^* be such that, for all sequences σ, $h^*(\sigma)$ is conjecture of the learner h after seeing σ as input (in particular, h^* is undefined on all extensions of sequences on which it is undefined); similarly, $h^*(z, \sigma)$ is just the conjecture of the learner after seeing σ as input, starting from the state z. We use $f^*(\sigma)$ to denote the state of $\langle h, f\rangle$ after seeing σ and.

We give an ORT-argument similar to that in the proof of Theorem 3.3, using the same graphical notation.

By 1-1 ORT, there are a 1-1 function $a \in \mathcal{R}$ and $e_0, e_1, e_2, b \in \mathbb{N}$ with $b \notin \mathrm{rng}(a)$ such that a number of restrictions are satisfied. We abbreviate, for all i, t,

$$P(i) \Leftrightarrow \langle h^*, f^*\rangle(a[i+1])\downarrow = \langle h^*, f^*\rangle(a[i])\downarrow, \tag{61}$$
$$\overline{P}(i) \Leftrightarrow \langle h^*, f^*\rangle(a[i+1])\downarrow \neq \langle h^*, f^*\rangle(a[i])\downarrow, \tag{62}$$
$$Q(i) \Leftrightarrow \exists \sigma \leq i : \sigma \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(a[i])) \wedge h^*(f^*(a[i]), b \diamond \sigma)\downarrow_i \neq h^*(f^*(a[i]), b)\downarrow_i, \tag{63}$$
$$R(i) \Leftrightarrow \exists i' \leq i : Q(i') \wedge P(i'),\ ^{29} \tag{64}$$
$$A = \{a(i) \mid \overline{P}(i)\}. \tag{65}$$

[28] As we shall see, this contrasts with the case of just one memory state, where every learner is strongly class-decisive (Remark 3.13).

[29] Note that the existential quantifier is used only to make R monotone in the case of (67) below, for convenience.

The first restriction is given by the following equation.

$$W_{e_0} = \mathrm{rng}(a). \tag{66}$$

Eqs. (68)–(71) have to be satisfied *if*

$$\forall i : \langle h^*, f^*\rangle(a[i])\downarrow, \tag{67}$$

and we leave e_1 and e_2 unrestricted otherwise. Note that, given (67), P, Q and R are decidable.

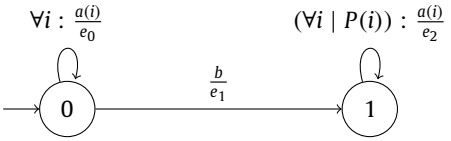
$$\begin{aligned} W_{e_1} &= \{b\} \cup \{a(i) \in A \mid \neg R(i)\} \\ &\quad \cup \{a(i) \mid \neg R(i) \wedge \exists i' : R(i')\}; \end{aligned} \tag{68}$$

$$= \{b\} \cup \begin{cases} \mathrm{rng}(a[i]), & \text{if there is a least } i \text{ with } R(i); \\ A, & \text{otherwise;} \end{cases} \tag{69}$$

$$W_{e_2} = \{b\} \cup \{a(i) \mid \neg R(i)\}; \tag{70}$$

$$= \{b\} \cup \begin{cases} \mathrm{rng}(a[i]), & \text{if there is a least } i \text{ with } R(i); \\ \mathrm{rng}(a), & \text{otherwise.} \end{cases} \tag{71}$$

The next group of restrictions is given by the following diagram.



The last group of restrictions is as follows. For all i with $\langle h^*, f^*\rangle(a[i+1])\uparrow$ or $\langle h^*, f^*\rangle(a[i])\uparrow$, $\varphi_{a(i)}(1)\uparrow$. For each $x \in \mathrm{rng}(a) \cup \{b\}$ and vertices v such that $\varphi_x(v)$ was not previously specified, we have the restriction $\varphi_x(v) = \langle ?, v\rangle$.

It is straightforward to see $W_{e_0} \in \mathcal{L}$. Thus, as a is a text for W_{e_0}, $\forall i : h^*(a[i])\downarrow$ (i.e., (67) holds) and we have $\forall^\infty i : P(i)$. We distinguish two cases as in (69) and (71).

Case 1: There is i with $R(i)$.

Let i be the least such example. Then $Q(i)$. Let σ be as given by $Q(i)$. We have $W_{e_1} = W_{e_2} = \mathrm{rng}(a[i]) \cup \{b\}$. Clearly, this language is in $\mathcal{L}$. As h is canny (see Definition 3.7), we have that $h^*(a[i] \diamond b)$ is an index for W_{e_1} ($a[i] \diamond b \diamond \#^\infty$ is a text for W_{e_1}). Furthermore, since $Q(i)$ as witnessed by σ, $h^*(f^*(a[i]), b) \neq h^*(f^*(a[i]), b \diamond \sigma)$, a contradiction to h being strongly non-U-shaped.

Case 2: Otherwise.

We have $W_{e_1} = A \cup \{b\}$. Clearly, this language is in $\mathcal{L}$. Furthermore, $W_{e_2} = \mathrm{rng}(a) \cup \{b\}$, which is clearly in $\mathcal{L}$ as well. From $W_{e_0} \in \mathcal{L}$, there is i_0 such that, for all $i \geq i_0$, $h^*(a[i]) = h^*(a[i_0])$. Consider the text T for W_{e_1} which starts with the subsequence of $a[i_0]$ which lists all and only the elements of A, then b, and then infinitely many #. T is a text for W_{e_1}; thus, as h is canny, after processing the element b in T, h outputs an index for W_{e_1}. Consider changing T by replacing the initial subsequence of $a[i_0]$ with all of $a[i_0]$. With this modified text, we only feed additional elements to the iteratively learning h, which do not change h's conjectures as compared with those on T. Hence, $h^*(f^*(a[i_0]), b)$ is an index for W_{e_1}.

Consider the text $T = a[i_0] \diamond b \diamond \Diamond_{i \geq 0}\, a(i_0 + i)$ for W_{e_2}. We have $\forall i \geq i_0 : P(i) \wedge \neg(Q(i) \wedge P(i))$. Thus, $\forall i \geq i_0 : \neg Q(i)$.

On T, h will have to change its conjecture after $a[i_0] \diamond b$, to identify W_{e_2}. Let $i_1 > i_0$ be such that $h^*(a[i_0] \diamond b \diamond \Diamond_{i_0 \leq i \leq i_1} a(i)) \neq h^*(a[i_0] \diamond b)$. Let t be large enough such that $h^*(f^*(a[i_0]), b \diamond \Diamond_{i_0 \leq i \leq i_1} a(i))\downarrow_t$, $h^*(f^*(a[i_0]), b)\downarrow_t$ and $\Diamond_{i_0 \leq i \leq i_1} a(i) \leq t$. We now have $Q(t)$, a contradiction. □

Note that Theorem 3.10 entails that iterative learning cannot be done strongly non-U-shapedly, using Theorem 3.5 and the Transfer Lemma (Lemma 2.7). Next we show that any class of languages learnable with a *bounded* number of memory states by a *total* learner is also learnable by a strongly non-U-shaped learner, if we allow an *arbitrary* (but finite) number of memory states; see the following theorem.

Theorem 3.11. *We have*

$$\bigcup_{n \in \mathbb{N}} \mathcal{R}\mathbf{TxtBMS}_n\mathbf{Ex} \subseteq_i \mathbf{SNU}\mathcal{R}\mathbf{TxtBMS}_*\mathbf{Ex}.$$

Furthermore, each learner witnessing inclusion in the right-hand-side learning criterion uses only finitely many memory states on all *texts.*

Proof. Let $n \in \mathbb{N}$, $\mathcal{L} \in \mathcal{R}\mathbf{TxtBMS}_n\mathbf{Ex}$ as witnessed by $h \in \mathcal{R}$. Using Lemma 3.4, we assume, without loss of generality, h makes no cycles. Similarly, we assume all of h's conjectures in different states to be syntactically different. We will now build a $\mathbf{BMS}_*$-learner h'. For this we need a data structure which is a labeled graph on n vertices. Initially, there are only the n vertices $\{0, \ldots, n-1\}$ and no edges. Further, the data structure contains a pointer to one of the vertices, labeling it as the "current" one (initially 0).

Whenever h' sees a new datum x, the data structure is updated as follows. For each possible state $k \in \{0, \ldots, n-1\}$ that h could be in, we run h on x from state k; suppose the result is $\langle e, m\rangle$ (recall that h is total). If in the graph there is an edge from k to m labeled ? or there is no edge at all, replace the edge (or insert new edge) from k to m labeled e. It is clear that the graph is only updated finitely many times on any text. Further, the pointer to "current" traverses the outgoing edge to the numerically least successor state (choosing the least is an arbitrary decision; any definite choice would do).

We let the new state of h' after seeing a new datum x be given by the updated data structure (and initially by the initial data structure), paired with the label of the edge most recently traversed by the pointer to "current," which had then a non-? label (and ? if there is no such edge). The new hypothesis is this label, padded with the current state (i.e., the current graph), ? if there is no such label. Thus, h' changes its state (and its conjecture) only finitely often on any text.

It is straightforward to see that h' $\mathcal{R}$**TxtBMS**$_*$**Ex**-learns $\mathcal{L}$. Further, it is clear that h' is sink-locking, as each conjecture is padded with the current state. We now get the result by the Sink-Locking Lemma (Theorem 2.2). □

We conclude this section by giving two results on BMS-learning for both the two and one memory states cases.

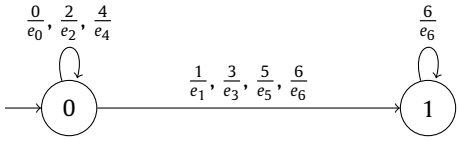
Remark 3.12. We have

$$\textbf{LinFTxtBMS}_2\textbf{Ex} \setminus \textbf{CDecTxtBMS}_2\textbf{Ex} \neq \emptyset;$$

in particular,

$$\textbf{CDecTxtBMS}_2\textbf{Ex} \subset \textbf{TxtBMS}_2\textbf{Ex}.$$

Proof. Consider the languages {0}, {0, 1}, {2}, {2, 3}, {4}, {4, 5} and {0, 1, 2, 3, 4, 5, 6}. Fix respective indices $e_0, \ldots, e_6$ for these languages. The diagram depicts a linear time computable learner h which learns all these languages using two states (an edge labeled $\frac{x}{e}$ from state u to v denotes $h(u, x) = \langle e, v\rangle$; for all u, x such that there is no such outgoing edge, then $h(u, x) = \langle ?, u\rangle$).



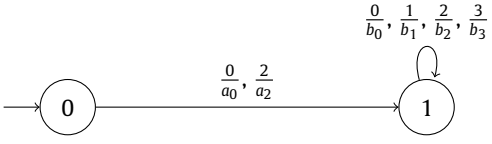
Suppose, by way of contradiction, some **BMS**$_2$-learner h learns these seven languages class-decisively. Without loss of generality, h never leaves State 1 (for all state transitions from State 1 to State 0, we change this into a transition from State 1 to State 1; see the proof of Theorem 3.4 for an argument for the correctness).

Case 1: h stays in State 0 for at least two of {0}, {2} and {4}.

Without loss of generality, h stays in State 0 for {0} and {2}. Thus, h outputs indices for {0} and {2} on 0 and 2, respectively. Therefore, h is not decisive on the text $0, 2, 0, 1, 3, 4, 5, 6^\infty$ for {0, 1, 2, 3, 4, 5, 6}.

Case 2: Otherwise.

Without loss of generality, h changes states on 0 and 2. Thus, for some conjectures $a_0, a_2, b_0, \ldots, b_3 \in \mathbb{N} \cup \{?\}$, h has a transition diagram as follows (some edges are omitted, but, for the rest of the proof, we do not need to know about them).



As h identifies {0}, we have that b_0 is an index for {0} or ?. As h identifies {0, 1}, we have that $\{b_0, b_1\}$ does *not* contain an index for {0}. Thus, $b_0 =?$. Therefore, as h identifies {0}, we have that a_0 is an index for {0}. However, $\{a_0, b_1\}$ *does* contain an index for {0, 1} (because no other conjectures are made on the text 01^∞ for {0, 1}). Hence, b_1 is an index for {0, 1}.

Analogously, b_3 is an index for {2, 3}. Therefore, h is not decisive on the text $0, 1, 3, 1, 2, 4, 5, 6^\infty$ for {0, 1, 2, 3, 4, 5, 6}. □

Any **TxtBMS**$_1$**Ex**-learner on # or any element x from a learned language L outputs an index for L or ?; furthermore, all non-? outputs must be the same. Thus, we have the following remark.

Remark 3.13. For all $\mathcal{C} \subseteq \mathcal{P}$, we have

$$\textbf{SCDec}\mathcal{C}\textbf{TxtBMS}_1\textbf{Ex} =_i \textbf{Txt}\mathcal{C}\textbf{BMS}_1\textbf{Ex}.$$

4. Memoryless Feedback learning (MLF)

The main theorem in this section is Theorem 4.2 just below. This theorem answers the open question mentioned in Section 1 above regarding memory-less feedback learning. Theorem 4.3 gives that memoryless learning with arbitrarily many feedback queries is equivalent to **TxtGEx**-learning *on classes of infinite languages*.

Memoryless Feedback learning, as defined above, trivially allows a learner to query for whether the *current input element* has been seen *strictly previously*. In Definition 4.7 below we give a variant of Memoryless Feedback learning, called MLF′-learning, where all queries are answered based on *all* data seen so far, *including* the current datum. For MLF′-learning, it is no longer possible to query to see if the current datum has been seen previously. From Theorem 4.8 we have that the learning power of MLF′-learning is strictly lower.

We conclude this section with a theorem regarding decisive memoryless learning.

Some results in this section make use of the following definition of a self-learning class of languages.

Definition 4.1. We use ψ as defined in Section 2.2. We assume that, in this system, we can appropriately use `rcl` to recall previous data as introduced in Section 2.1 for the φ-system; recall that `rcl` is syntactic sugar. Let $h_0 \in \mathscr{P}$ be such that

$$\forall i, x : h_0(x) = \begin{cases} ?, & \text{if } x = \#; \\ \psi_x(0), & \text{otherwise.} \end{cases} \tag{72}$$

Let $\mathscr{L}_0 = \mathbf{TxtMLF}_1\mathbf{Ex}(h_0)$.

Theorem 4.2.

$$\mathbf{LinFTxtMLF}_1\mathbf{Ex} \setminus \mathbf{NUTxtMLR}_*\mathbf{Ex} \neq \emptyset.$$

Proof. We consider $\mathscr{L}_0$ from Definition 4.1, which is clearly learnable in linear time. Suppose $h_1 \in \mathscr{P}$ **TxtMLR**$_*$**Ex**-learns $\mathscr{L}_0$. We show that h_1 is *not* non-U-shaped. For any sequence of data σ, we let $h_1^*(\sigma)$ denote the current conjecture of h_1 after having seen the sequence σ. It is easy to see that we can assume, without loss of generality,

$$\forall \tau, x : h_1^*(\tau \diamond x \diamond x) \in \mathbb{N} \Rightarrow h_1^*(\tau \diamond x) = h_1^*(\tau \diamond x \diamond x). \tag{73}$$

Let $f \in \mathscr{P}$ be such that, on input x, f first computes $h_1(x)$ where all queries are answered with "false". If $h_1(x)\downarrow$, f outputs the maximum recalled element plus 1 (or 0, if no queries were asked). Then we have

$$\forall x : h_1^*(x)\downarrow \Rightarrow f(x)\downarrow \text{ and } h_1^* \text{ on } x \text{ does not recall any } y \geq f(x). \tag{74}$$

By ψ/φ-Hybrid ORT, there are $e_0, e_1, a \in \mathbb{N}$ and strictly monotonic increasing functions $\hat{b}, \hat{c} \in \mathscr{R}$ such that $\hat{b}$ and $\hat{c}$ have disjoint ranges, neither containing a, and, abbreviating

$$b = \hat{b}(f(a));$$
$$c = \lambda i.\hat{c}(i, f(a));$$
$$E = \{c(i) \mid \exists j \geq i : h_1^*(a \diamond c[j])\downarrow \neq h_1^*(a \diamond c[j+1])\downarrow\}$$

we have $\forall i, t, x$:[30]

$$W_{e_0} = \{a\} \cup \operatorname{rng}(c); \tag{75}$$

$$W_{e_1} = \begin{cases} \{a\}, & \text{if } h_1^*(a)\uparrow \text{ or } h_1^*(a) = ?; \\ \{a, b\} \cup E, & \text{otherwise;} \end{cases} \tag{76}$$

$$\psi_a(x) = \begin{cases} ?, & \text{if } \mathtt{rcl}(a); \\ e_1, & \text{otherwise;} \end{cases} \tag{77}$$

$$\psi_{\hat{b}(i)}(x) = \begin{cases} ?, & \text{if } \mathtt{rcl}(\hat{b}(i)); \\ e_1, & \text{otherwise;} \end{cases} \tag{78}$$

$$\psi_{\hat{c}(i,t)}(x) = \begin{cases} e_1, & \text{if } \mathtt{rcl}(\hat{b}(t)); \\ e_0, & \text{otherwise.} \end{cases} \tag{79}$$

Claim 1. $h_1^*(a) \in \mathbb{N}$.

Proof of Claim 1. Suppose, by way of contradiction, otherwise. We have $\{a\} = W_{e_1} \in \mathscr{L}_0$. If $h_1^*(a)\uparrow$, then h_1 would not learn $W_{e_1} \in \mathscr{L}_0$, a contradiction. Hence, $h_1^*(a) = ?$. Using (73), we get that h_1, on the text $\lambda i.a$, does not learn $\{a\} = W_{e_1} \in \mathscr{L}_0$, a contradiction. □ (FOR CLAIM 1)

[30] Note that we use the shorthand `rcl` as introduced in Section 2.1 also for the ψ-system.

Using (74) and Claim 1, we have $f(a)\downarrow$. Hence, b is defined and c is total. It is now easy to see that

$$W_{e_0} \in \mathcal{L}_0. \tag{80}$$

Therefore, h_1 **TxtMLR**$_*$**Ex**-learns W_{e_0}. Let k be minimal such that

$$\forall i \geq k : h_1^*(a \diamond c[i]) \in \{?, h_1^*(a \diamond c[k])\}. \tag{81}$$

With reasoning similar to that in Claim 1, we get $k > 0$ and $c(0) \in E$. Furthermore,

$$W_{h_1^*(a \diamond c[k])} = W_{e_0}. \tag{82}$$

Hence, W_{e_1} is the finite set $\{a, b, c(0)\} \cup \text{content}(c[k])$. It is easily verified that $W_{e_1} \in \mathcal{L}_0$.

Note that, as $\hat{b}$ and $\hat{c}$ are strictly monotonically increasing,

$$\forall x \in \{b\} \cup \text{rng}(c) : x \geq f(a). \tag{83}$$

Hence, by choice of f (see (74)), h_1 on a cannot recall any $x \in \{b\} \cup \text{rng}(c)$. Therefore,

$$h_1^*(a) = h_1^*(c[k] \diamond b \diamond a). \tag{84}$$

Claim 2. $W_{h_1^*(a)} = W_{e_1}$.

Proof of Claim 2. From Claim 1 we have that $h_1^*(a) \in \mathbb{N}$. As h_1 **TxtMLR**$_*$**Ex**-identifies W_{e_1} from the text $c[k] \diamond b \diamond \lambda i.a$ and using (73),

$$W_{h_1^*(c[k] \diamond b \diamond a)} = W_{e_1}. \tag{85}$$

Using (84), we get the claim. □ (for Claim 2)

We consider the text $T = a \diamond c[k] \diamond b \diamond \lambda x.\#$ for W_{e_1}. The following shows that h_1 has a U-shape on T.

(i) $W_{h_1^*(a)} = W_{e_1}$ by Claim 2;
(ii) $W_{h_1^*(a \diamond c[k])} = W_{e_0}$ by (82);
(iii) $\exists j \geq k+1 : W_{h_1^*(T[j])} = W_{e_1}$ as h_1 **TxtMLR**$_*$**Ex**-identifies W_{e_1}.

Therefore, h_1 is not non-U-shaped on $\mathcal{L}_0$. □

With $\text{Pow}(\mathcal{E}_\infty)$ we denote the powerset of all infinite ce sets.

Theorem 4.3.

$$\text{Pow}(\mathcal{E}_\infty) \cap \textbf{NUTxtMLF}_*\textbf{Ex} =_c \text{Pow}(\mathcal{E}_\infty) \cap \textbf{TxtGEx}.$$

Proof. "$\subseteq$" is trivial.
"$\supseteq$": Let $h_0 \in \mathcal{P}$, let $\mathcal{L} = \text{Pow}(\mathcal{E}_\infty) \cap \textbf{TxtSdEx}(h_0)$. Without loss of generality, we can assume $h_0 \in \mathcal{R}$.[31] We use a function pad as in the proof of Theorem 3.5. Let $f, m, h_1 \in \mathcal{P}$ be such that, for all x,

$$f(x) = \texttt{rcl}(\{y \leq x\});$$
$$m(x) = \mu D.(\forall D' \mid D \subseteq D' \subseteq f(x)) h_0(D) = h_0(D');\ ^{32}$$
$$h_1(x) = \begin{cases} ?, & \text{if } x = \# \text{ or } x \in f(x); \\ \text{pad}(h_0(m(x)), m(x)), & \text{otherwise.} \end{cases}$$

Let $L \in \mathcal{L}$ and let T be a text for L. Let D be the $\leq$-minimum locking set for h_0 on L.

Claim 1. $\forall^\infty i : h_1^*(T[i]) \in \{?, \text{pad}(h_0(D), D)\}$ and $\exists^\infty i : h_1^*(T[i]) = \text{pad}(h_0(D), D)$.

Proof of Claim 1. Let c_0 be such that $D \subseteq \text{content}(T[c_0])$. Let $c_1 \geq c_0$ be such that $\forall c \geq c_1$:

$$T(c) \notin \text{content}(T[c_1]) \Rightarrow T(c) > \max(D).^{33} \tag{86}$$

Intuitively, after $T[c_1]$, the only new elements presented to the learner are strictly bigger than $\max(D)$. This implies that, after $T[c_1]$,

$$\forall c \geq c_1 : D \subseteq f(T(c)). \tag{87}$$

31 This can be seen using standard delaying tricks.
32 Note that, for purposes of minimization, D is treated as a natural number and minimized with respect to $\leq$.
33 We use the convention $\max(\emptyset) = -1$.

From D being the $\leq$-minimum locking set we get $\forall D' < D, \forall^\infty i$:

$$\neg(\forall D'' \mid D' \subseteq D'' \subseteq f(T(i)))h_0(f(T(i))) = h_0(D'').$$

Putting (87) and (4) together with D is a locking set, we get the claim. □ (for Claim 1)

As D is a locking set, an application of the Sink Locking Lemma (Lemma 2.2) gives the desired result. □

The proof of Theorem 4.6 makes use of some graph theory, so we give the just below definitions and a lemma (Lemma 4.5) to support the proof of Theorem 4.6.

Definition 4.4. A *directed graph* is a pair (V, E), where $V \subseteq \mathbb{N}$ is a set of *vertices* and $E \subseteq V \times V$ is a set of *edges*; for each vertex $v \in V$, we call $\mathrm{card}(\{y \in V \mid (v, y) \in E\})$ the *out-degree of* v; the supremum over all vertices' out-degrees is the *out-degree of* (V, E). A graph is called *infinite* iff its set of vertices is infinite.

For a graph $G = (V, E)$, a set $D \subseteq V$ is called an *independent set of* G iff, for all $x, y \in D$, $(x, y) \notin E$.

Lemma 4.5. *Let G be an infinite directed graph of finite out-degree. Then, for each $k > 0$ and each $j \in \mathbb{N}$, there is a collection of size j of mutually disjoint sets of vertices D with $\mathrm{card}(D) = k$ such that D is an independent set of G.*

Proof. [34] Let n be the out-degree of G. By induction on k, clear for $k = 1$. Suppose the claim is true for some $k \in \mathbb{N}$. Let $j \in \mathbb{N}$. We show that there are j independent sets of G of cardinality $k + 1$.

Let $D_0, \ldots, D_{j+n-1}$ be $j + n$ mutually disjoint independent sets of G of cardinality k. Let M be the (infinite) set of vertices of G that is not in $\bigcup_{i<j+n} D_i$ or reachable in G *from* $\bigcup_{i<j+n} D_i$ in one step. Then, for each $x \in M$, as x has out-degree at most n, there are at least j different $i < j + n$ such that $D_i \cup \{x\}$ is an independent set of G of cardinality $k + 1$. As M is infinite, the induction step follows. □

Theorem 4.6.

$$(\mathbf{LinFTxtMLF}_1\mathbf{Ex} \cap \mathrm{Pow}(\mathcal{E}_\infty)) \setminus \bigcup_{n\in\mathbb{N}} \mathbf{NUTxtMLF}_n\mathbf{Ex} \neq \emptyset.$$

Proof. We consider h_0 and $\mathcal{L}_0$ from Definition 4.1. Let $\mathcal{L} = \mathcal{L}_0 \cap \mathrm{Pow}(\mathcal{E}_\infty)$. Suppose, by way of contradiction, $\mathcal{L} \in \bigcup_{n\in\mathbb{N}} \mathbf{NUTxtMLF}_n\mathbf{Ex}$, as witnessed by n and h.

For $L \subseteq \mathbb{N}$ and $x \in \mathbb{N}$, we write $h_0^L(x)$ and $h^L(x)$, respectively, to denote the output of h_0 and h, respectively, when presented with x and all recalls to elements of L are answered with "yes", all others with "no".

Below, we introduce an infinite ce set of elements on which h is forced to output ?. Then we will use some or all of these elements to make certain sets infinite. By ψ/φ-Hybrid ORT, there are $a, w \in \mathbb{N}$ and a 1-1 $z \in \mathcal{R}$ such that

$$W_a = \{w\} \cup \mathrm{rng}(z); \tag{88}$$

$$\forall x : \psi_w(x) = a; \tag{89}$$

$$\forall j, x : \psi_{z(j)}(x) = ?. \tag{90}$$

Obviously, $W_a \in \mathcal{L}$; hence, for all $\rho \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(z))$, $h^*(\rho)\downarrow$.

Claim 1. For all $\rho \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(z))$ with $\rho \neq \emptyset$ we have $h^*(\rho) = ?$.

Proof of Claim 1. Let $\rho \in \mathbb{S}\mathrm{eq}(\mathrm{rng}(z)) \setminus \{\emptyset\}$. By ψ/φ-Hybrid ORT, there are 1-1 $b, u \in \mathcal{R}$ such that, for all j, x,

$$W_{b(j)} = \{u(j)\} \cup \mathrm{content}(\rho); \tag{91}$$

$$\psi_{u(j)}(x) = b(j). \tag{92}$$

For each j, let $L_j = W_{b(j)}$. For all j, and for all texts T for L_j, h_0 on T outputs $b(j)$ at least once and otherwise only ?. Thus, $L_j \in \mathcal{L}_0$.

Let $x = \mathrm{last}(\rho)$ and let j_0 be large enough such that, for all $j \geq j_0$, h on x does not recall $u(j)$. For all j, let e_j be the output of h on x when all queries for elements from L_j are answered positively, all others negatively. Note that, for all $j \neq j_0$, we have $(L_{j_0} \cup L_j) \setminus (L_{j_0} \cap L_j) = \{u(j_0), u(j)\}$. Therefore, for all $j \geq j_0$, $e_j = e_{j_0}$. Thus, for all $j \geq j_0$ and all texts T for $W_{b(j)}$ such that T contains infinitely many occurrences of x, we have that h on T infinitely often outputs e_{j_0}. In particular, there are two different languages in $\mathcal{L}$ and texts for these languages, such that h on both of these texts infinitely often outputs e_{j_0}. Thus, $e_{j_0} = ?$. Obviously, by choice of L_{j_0}, $e_{j_0} = h^*(\rho)$. □ (for Claim 1)

Now we can use $\mathrm{rng}(z)$ as the desired infinite ce set of elements on which h is forced to output ?.

Let, for each $v \in \mathbb{N}$,

$$E_v = \{z(j) \mid \begin{array}{l} h \text{ on } z(j) \text{ does not recall } v \text{ and} \\ h \text{ on } v \text{ converges and does not recall } z(j) \end{array}\}. \tag{93}$$

[34] This result can be derived as a consequence from Turán's Theorem [22, Theorem 7.1.1]. However, the according derivation is roughly as complex as the below proof of this lemma. Hence, also in the interest of self-containment, we prove the lemma without reference to Turán's Theorem.

By ψ/φ-Hybrid ORT, there are 1-1 $p, r, s \in \mathcal{R}$ with pairwise disjoint ranges (and, without loss of generality, ranges disjoint of $\mathrm{rng}(z)$) such that, for all x, y,

$$W_{p(x)} = \{r(x)\} \cup E_{r(x)}; \tag{94}$$

$$\varphi_{r(x)}(y) = \begin{cases} p(x), & \text{if not } \mathtt{rcl}(s(x)); \\ ?, & \text{otherwise;} \end{cases} \tag{95}$$

$$\psi_{s(x)}(y) = ?. \tag{96}$$

Claim 2. Let $x \in \mathbb{N}$. Suppose $E_{r(x)}$ is infinite. Then h **NUTxtMLF**$_n$**Ex**-identifies $W_{p(x)} = \{r(x)\} \cup E_{r(x)}$.

Proof of Claim 2.

Let T be a text for $W_{p(x)}$. For i such that $T(i) = r(x)$ we have $W_{h_0^*(T[i+1])} = W_{p(x)}$. Further, for all $j > i$, $h_0^*(T[j+1]) \in \{?, p(x)\}$. Therefore, $W_{p(x)} \in \mathcal{L}$. □ (FOR CLAIM 2)

Claim 3. Let $x \in \mathbb{N}$ be such that $E_{r(x)}$ is infinite. Then $h(r(x))\downarrow$ and $W_{h(r(x))} = W_{p(x)} = \{r(x)\} \cup E_{r(x)}$.

Proof of Claim 3. From Claim 2, we get that h **NUTxtMLF**$_n$**Ex**-identifies $W_{p(x)}$. From Claim 1 and the definition of $E_{r(x)}$, we get that, for all $\rho \in \mathbb{S}\mathrm{eq}(W_{p(x)})$ with $\mathrm{last}(\rho) \in E_{r(x)}$, $h^*(\rho) = ?$. Intuitively, h cannot give any conjecture other than ? when presented with an element from $E_{r(x)}$, as it does not recall $r(x)$. Thus, when h is presented $r(x)$ for the first time, it has to give a correct conjecture, i.e., $W_{h(r(x))} = \{r(x)\} \cup E_{r(x)}$ (the text might contain only one occurrence of $r(x)$). □ (FOR CLAIM 3)

Claim 4. There is a finite set D of cardinality $n+2$, such that

(i) for all $x \in D$, h on $r(x)$ does not recall any element from $\{r(y), s(y) \mid y \in D \setminus \{x\}\}$; and
(ii) for all $x \in D$, $E_{r(x)}$ is infinite.

Proof of Claim 4. Let G be the graph with vertex set $\mathbb{N}$ such that, for all $x, y \in \mathbb{N}$, there is an edge from x to y in G iff: $x \neq y$ and h on $r(x)$ recalls $r(y)$ or $s(y)$. Clearly, the out-degree of G is bounded by n, as h recalls at most n elements on any input, and r, s are 1-1 and have disjoint ranges. We apply Lemma 4.5 with $n+2$ in the place of k and $n+1$ in the place of j to obtain $n+1$ mutually disjoint sets $D_0, \ldots, D_n$, each fulfilling (i) of the claim. We proceed by showing that there are at most n different $x \in \mathbb{N}$ such that $E_{r(x)}$ is finite in order to show that at least one of these sets fulfills (ii).

As, for all x, $h(r(x))\downarrow$, we have

$$E_{r(x)} \text{ finite } \Rightarrow \forall^\infty j : h \text{ on } z(j) \text{ recalls } r(x). \tag{97}$$

Suppose, by way of contradiction, there are at least $n+1$ different $x \in \mathbb{N}$ such that $E_{r(x)}$ is finite. Thus, for all but finitely many j, h on $z(j)$ recalls at least $n+1$ elements from $\mathrm{rng}(r)$, a contradiction to h only making n recalls. □ (FOR CLAIM 4)

Fix a D as shown existent in Claim 4.

By 1-1 ORT there are 1-1 $t \in \mathcal{R}$, $e \in \mathbb{N}$ and $\mathtt{ce}$ sets F, L such that

$$F = \{i \mid \forall x \in D : h \text{ on } r(x) \text{ does not recall } t(i)\}; \tag{98}$$

$$L = W_e = \{r(x), s(x) \mid x \in D\} \cup \{t(i) \mid i \in F\}; \tag{99}$$

$$\forall i, y : \varphi_{t(i)}(y) = e. \tag{100}$$

Note that, for all $x \in D$, h on $r(x)$ does not recall any element from $(L \setminus \{r(x), s(x)\})$. Furthermore, F is infinite.

Claim 5. h **NUTxtMLF**$_n$**Ex**-identifies L.

Proof of Claim 5. Let T be a text for L, let i be such that, for all $x \in D$, $s(x) \in \mathrm{content}(T[i])$. As F is infinite, there is $j > i$ such that $T(j) \in \{t(z) \mid z \in F\}$. Thus, $h_0^*(T[j+1]) = e$ and, for all $j' > j$, we have

$$h_0^*(T[j'+1]) = \begin{cases} e, & \text{if } T(j') \in \{t(z) \mid z \in F\}; \\ ?, & \text{if } T(j') \in \{r(x) \mid x \in D\}; \\ ?, & \text{if } T(j') \in \{\#\} \cup \{s(x) \mid x \in D\}. \end{cases}$$

To see the second case, note that $s(x) \in \mathrm{content}(T[j'])$. This shows that h_0 on T converges to e. □ (FOR CLAIM 5)

We consider the text $T = \Diamond_{x \in D}(r(x) \diamond s(x)) \diamond \Diamond_{i \in F} t(i)$ for L. Let i_0 be such that $W_{h^*(T[i_0])} = L$. We have $i_0 \neq 0$, as otherwise there would be a U-shape from Claim 3.

Let $x \in D$ be such that h on $T(i_0 - 1)$ does not recall $r(x)$ or $s(x)$ and $T(i_0 - 1) \notin \{r(x), s(x)\}$ (possible as $\mathrm{card}(D) = n+2$). Let τ contain all and only the elements from $\mathrm{rng}(T[i_0 - 1])$ recalled by h on $T(i_0 - 1)$.

Now we consider the text $T' = \tau \diamond T(i_0 - 1) \diamond r(x) \diamond T$ for L. We have that

(i) $W_{h^*(\tau \diamond T(i_0-1))} = L$ (by choice of τ);
(ii) $W_{h^*(\tau \diamond T(i_0-1) \diamond r(x))} = W_{p(x)} \neq L$ (by Claim 3) and
(iii) $\exists j > \mathrm{len}(\tau) + 2 : W_{h^*(T[j])} = L$ (as h^* on T' has to converge to a correct ce-index for L).

Thus, h has a U-shape on T', a contradiction. □

Note that **MLF** allows for a learner to determine (at the cost of a query) whether the current datum has been seen previously (i.e., is a repetition), and the previous proofs in this section sometimes made use of this ability. The next theorem states that this ability is important for learning power.

Definition 4.7. Call **MLF**′ the sequence generating functional that one gets by modifying **MLF** to answer *true* to recalls for the current datum.

The sequence generating functional **MLF**′ destroys a learners ability to determine whether the current datum has been seen previously (i.e., is a repetition).

Theorem 4.8.

$$\mathbf{TxtMLF}_1\mathbf{Ex} \setminus \mathbf{TxtMLF}'_*\mathbf{Ex} \neq \emptyset.$$

Proof. We consider $\mathcal{L}_0$ from Definition 4.1.

By padded ORT there are 1-1 $r, e \in \mathcal{R}$ such that $\forall n, k, x$:

$$\varphi_{r(0)}(x) = \begin{cases} ?, & \text{if } \mathtt{rcl}(r(0)); \\ e(0), & \text{otherwise;} \end{cases} \tag{101}$$

$$\varphi_{r(\langle n,k\rangle+1)}(x) = e(n+1); \tag{102}$$

$$W_{e(0)} = \{r(0)\}; \tag{103}$$

$$W_{e(n+1)} = \{r(0)\} \cup \{r(\langle n, x\rangle + 1) \mid x \in \mathbb{N}\}. \tag{104}$$

It is easy to see that $\{W_{e(n)} \mid n \in \mathbb{N}\} \subseteq \mathcal{L}_0$. Suppose, by way of contradiction, $\mathcal{L}_0 \in \mathbf{MLF}'_*\mathbf{Ex}$ as witnessed by h. We have that $r(0)^\infty$, the infinite sequence of repetitions of $r(0)$, is a text for $W_{e(0)}$. Therefore, h on this text converges to a number p such that $W_p = W_{e(0)}$. Furthermore, for all $k > 0$, h^* on $r(0)^k$ will get "yes" answers to a recall for $r(0)$, and "no" answers to all other recalls. Hence, $\forall k : h^*(r(0)^k) = h^*(r(0)) = p$.

As h on $r(0)$ recalls only finitely many elements, there is n large enough such that h on $r(0)$ does not recall any element in $W_{e(n+1)}$ different from $r(0)$. Let T be a text for $W_{e(n+1)}$ containing $r(0)$ infinitely often. We have, for all i with $T(i) = r(0)$, h on $T(i)$ will get "yes" answers to a recall for $r(0)$, and "no" answers to all other recalls; thus,

$$h^*(T[i+1]) = h^*(r(0)) = p. \tag{105}$$

Therefore, h on T outputs p infinitely often, which is a contradiction as $W_p = W_{e(0)} \neq W_{e(n+1)}$. □

We conclude this section with the following theorem regarding decisive memoryless learning.

Theorem 4.9.

$$\bigcup_{n\in\mathbb{N}} \mathbf{DecTxtMLF}_n\mathbf{Ex} = \bigcup_{n\in\mathbb{N}} \mathbf{DecTxtMLR}_n\mathbf{Ex} = \{\mathcal{L} \mid \mathcal{L} \text{ finite}\}.$$

Proof. We show the inclusions $(1) \subseteq (2) \subseteq (3) \subseteq (1)$, where $(1), \ldots$ denote the three terms in the claim of the theorem.

The first inclusion is clear. Regarding the third inclusion: Let $\mathcal{L}$ be a finite collection of languages. Let D be a finite set of data such that

$$\forall L, L' \in \mathcal{L} : L \neq L' \Rightarrow \exists x \in D : x \in (L \setminus L') \cup (L' \setminus L). \tag{106}$$

Clearly, any learner which recalls all of D and outputs a conjecture from $\mathcal{L}$ consistent with the answers (including consistent with the negative answers) whenever such a conjecture exists (and otherwise repeating the previous conjecture) will identify all of $\mathcal{L}$ decisively.

Regarding the second inclusion: Let $n \in \mathbb{N}$ and let $\mathcal{L} \in \mathbf{DecTxtMLF}_n\mathbf{Ex}$ as witnessed by $h \in \mathcal{P}$. Suppose, by way of contradiction, $\mathcal{L}$ is infinite. Then h outputs an infinite number of different conjectures for the members of elements of $\mathcal{L}$. Thus, there is an infinite set M and, for each $x \in M$, a set D_x, such that h, after having seen D_x and no other data, will output a conjecture (for a different language for each $x \in M$) on x. Using Lemma 4.5, there are x and y such that h on x, if all of D_x and D_y were presented previously, does not recall any of $D_y \cup \{y\}$ and vice versa. Thus h, after being presented all of $D_x \cup D_y$, will repeat a conjecture after abandoning it on the input sequence x, y, x. □

Table 1
Known results and open problems.

	Criterion	NU		SNU	
Full	**TxtGEx**	=	[4]	=	[20]
	$n \geq 2$: **TxtGFex**$_n$	⊂	[12]	⊂	[12]
	TxtGBc	⊂	[23]	⊂	[23]
	TxtGPcpEx	=	[13]	=	(unpublished)[36]
	TxtPsdEx	=	[17]	=	[17]
	TxtSdEx	=	[17]	=	[17]
Iter+	**TxtItEx**	=	[21]	⊂	[17]
	TxtItCtrEx				
	TxtBem$_n$**Ex**				
	TxtFb$_n$**Ex**				
	TxtRcl$_n$**Ex**				
None	$n \geq 1$: **TxtMLF**$_n$**Ex**	⊂	[11]	⊂	[11]
	TxtMLF$_*$**Ex**	⊂	Theorem 4.2	⊂	Theorem 4.2
	TxtBMS$_1$**Ex**	=	[11]	=	Remark 3.13
	TxtBMS$_2$**Ex**	=	[11]	⊂	Theorem 3.10
	$n \geq 3$: **TxtBMS**$_n$**Ex**	⊂	Theorem 3.3	⊂	Theorem 3.3
	TxtBMS$_*$**Ex**	=	Corollary 3.6	⊂	Theorem 3.10

5. Some known results and open problems

We conclude the present paper with a table (Table 1) on *a core of* known results and open problems regarding necessity of U-shapes.[35]

This table groups various learning criteria according to how they can access strictly previous data and any prior conjecture. The group 'None' denotes no direct access to prior conjectures (and limited access to prior data or states), the group 'Iter+' denotes access to any immediately prior conjecture (with no or limited access to strictly previous data), and 'Full' denotes direct access to all previous data points.

An entry "=" denotes that, without loss of generality, a learner can have the property of the header of that column, while "⊂" denotes the opposite. An empty cell indicates an *open problem*. We would like to see these problems solved, also because the associated proofs will likely give new insights into their respective learning criteria. The definition of the different learning criteria can be found in Section 2.1.

Acknowledgments

We would like to thank the anonymous referees of the conference publication that this paper is based on, as well as the referees of this journal version, for their comments and suggestions; they have greatly enhanced this paper. Furthermore, we want to thank Samuel E. Moelius III and Thomas Zeugmann for discussions on various theorems and their proofs.

Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. NE 1182/5-1.

References

[1] D. Angluin, Inductive inference of formal languages from positive data, Information and Control 45 (1980) 117–135.
[2] J. Bārzdiņš, Inductive inference of automata, functions and programs, in: Proc. of the International Congress of Mathematicians, 1974, pp. 455–560 (English translation in, American Mathematical Society Translations: Series 2 109 (1977), pp. 107–112).
[3] L. Blum, M. Blum, Toward a mathematical theory of inductive inference, Information and Control 28 (1975) 125–155.
[4] G. Baliga, J. Case, W. Merkle, F. Stephan, W. Wiehagen, When unlearning helps, Information and Computation 206 (2008) 694–709.
[5] T. Bower, Concepts of development, in: Proceedings of the 21st International Congress of Psychology, Presses Universitaires de France, Paris, 1978.
[6] M. Bower, Starting to talk worse: Clues to language development from children's late speech errors, in: S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, in: Developmental Psychology Series, Academic Press, NY, 1982.
[7] S. Carey, Face perception: anomalies of development, in: S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, in: Developmental Psychology Series, Academic Press, NY, 1982.
[8] J. Case, Periodicity in generations of automata, Mathematical Systems Theory 8 (1974) 15–32.
[9] J. Case, Infinitary self-reference in learning theory, Journal of Experimental and Theoretical Artificial Intelligence 6 (1994) 3–16.
[10] J. Case, The power of vacillation in language learning, SIAM Journal on Computing 28 (6) (1999) 1941–1969.
[11] L. Carlucci, J Case, S. Jain, F. Stephan, Results on memory-limited U-shaped learning, Information and Computation 205 (2007) 1551–1573.
[12] L. Carlucci, J. Case, S. Jain, F. Stephan, Non-U-shaped vacillatory and team learning, Journal of Computer and System Sciences 74 (2008) 409–430.
[13] L. Carlucci, S. Jain, E. Kinber, F. Stephan, Variations on U-shaped learning, Information and Computation 204 (8) (2006) 1264–1294.
[14] J. Case, S. Jain, S. Lange, T. Zeugmann, Incremental concept learning for bounded data mining, Information and Computation 152 (1999) 74–110.

35 The table omits mention of the various criteria featuring linear time learners or some form of decisiveness. It also does not present results such as Theorem 4.2, pertaining to the impossibility of gaining non-U-shaped learnability by providing more resources to the learner.

36 This can be shown using techniques from [17], suitably modified.

[15] J. Case, T. Kötzing, Dynamic modeling in inductive inference, in: Proc. of ALT (Algorithmic Learning Theory), 2008, pp. 404–418.
[16] J. Case, T. Kötzing, Solutions to open questions for non-U-shaped learning with memory limitations, in: Proc. of ALT (Algorithmic Learning Theory), 2010.
[17] J. Case, T. Kötzing, Strongly non-U-shaped learning results by general techniques, in: Proc. of COLT (Conference on Learning Theory), 2010, pp. 181–193.
[18] J. Case, T. Kötzing, Measuring learning complexity with criteria epitomizers, in: Proc. of STACS (Symposium on Theoretical Aspects of Computer Science), 2011, pp. 320–331.
[19] J. Case, C. Lynes, Machine inductive inference and language identification, in: M. Nielsen, E. Schmidt (Eds.), Proceedings of the 9th International Colloquium on Automata, Languages and Programming, in: Lecture Notes in Computer Science, vol. 140, Springer-Verlag, Berlin, 1982, pp. 107–115.
[20] J. Case, S. Moelius, Optimal language learning, in: Proc. of ALT (Algorithmic Learning Theory), 2008, pp. 419–433.
[21] J. Case, S. Moelius, U-shaped, iterative, and iterative-with-counter learning, Machine Learning 72 (2008) 63–88.
[22] R. Diestel, Graph Theory, Springer, 2000.
[23] M. Fulk, S. Jain, D. Osherson, Open problems in systems that learn, Journal of Computer and System Sciences 49 (3) (1994) 589–604.
[24] R. Freivalds, E. Kinber, C. Smith, On the impact of forgetting on learning machines, Journal of the ACM 42 (1995) 1146–1168.
[25] M. Fulk, A Study of Inductive Inference Machines, Ph.D. Thesis, SUNY at Buffalo, 1985.
[26] M. Fulk, Prudence and other conditions on formal language learning, Information and Computation 85 (1990) 1–11.
[27] E. Gold, Language identification in the limit, Information and Control 10 (1967) 447–474.
[28] Klaus Jantke, Monotonic and non-monotonic inductive inference of functions and patterns, in: J. Dix, K. Jantke, P. Schmitt (Eds.), Nonmonotonic and Inductive Logic, in: Lecture Notes in Computer Science, vol. 543, Springer, Berlin / Heidelberg, 1991, pp. 161–177.
[29] S. Jain, E.B. Kinber, Iterative learning from texts and counterexamples using additional information, ALT (2009) 308–322.
[30] S. Jain, S. Lange, S. Moelius, S. Zilles, Incremental learning with temporary memory, Theoretical Computer Science 411 (29–30) (2010) 2757–2772.
[31] S. Jain, D. Osherson, J. Royer, A. Sharma, Systems that Learn: An Introduction to Learning Theory, second edition, MIT Press, Cambridge, Massachusetts, 1999.
[32] T. Kötzing, Abstraction and Complexity in Computational Learning in the Limit, Ph.D. Thesis, University of Delaware, 2009. Available online at http://pqdtopen.proquest.com/#viewpdf?dispub=3373055.
[33] D. Kozen, Indexings of subrecursive classes, Theoretical Computer Science 11 (1980) 277–301.
[34] E. Kinber, F. Stephan, Language learning from texts: mind changes, limited memory and monotonicity, Information and Computation 123 (1995) 224–241.
[35] S. Lange, T. Zeugmann, Monotonic versus non-monotonic language learning, in: K. Jantke, G. Brewka, P. Schmitt (Eds.), Proceedings of the Second International Workshop on Nonmonotonic and Inductive Logic, in: Lecture Notes in Artificial Intelligence, vol. 659, Springer-Verlag, Berlin, 1993, pp. 254–269.
[36] S. Lange, T. Zeugmann, Incremental learning from positive data, Journal of Computer and System Sciences 53 (1996) 88–103.
[37] G. Marcus, S. Pinker, M. Ullman, M. Hollander, T.J. Rosen, F. Xu, Overregularization in Language Acquisition, in: Monographs of the Society for Research in Child Development, vol. 57, University of Chicago Press, 1992, Includes commentary by H. Clahsen.
[38] D. Osherson, M. Stob, S. Weinstein, Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists, MIT Press, Cambridge, Mass, 1986.
[39] D. Osherson, S. Weinstein, Criteria of language learning, Information and Control 52 (1982) 123–138.
[40] K. Plunkett, V. Marchman, U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition, Cognition 38 (1) (1991) 43–102.
[41] J. Royer, J. Case, Subrecursive Programming Systems: Complexity and Succinctness, in: Research Monograph in Progress in Theoretical Computer Science, Birkhäuser, Boston, 1994.
[42] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw Hill, New York, 1967, Reprinted by MIT Press, Cambridge, Massachusetts, 1987.
[43] G. Schäfer-Richter, Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien, Ph.D. Thesis, RWTH Aachen, 1984.
[44] S. Strauss, R. Stavy (Eds.), U-Shaped Behavioral Growth, in: Developmental Psychology Series, Academic Press, NY, 1982.
[45] S. Strauss, R. Stavy, N. Orpaz, The child's development of the concept of temperature, 1977. Unpublished manuscript, Tel-Aviv University.
[46] N. Taatgen, J. Anderson, Why do children learn to say broke? A model of learning the past tense without feedback, Cognition 86 (2) (2002) 123–155.
[47] K. Wexler, P. Culicover, Formal Principles of Language Acquisition, MIT Press, Cambridge, Massachusetts, 1980.
[48] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Elektronische Informationverarbeitung und Kybernetik 12 (1976) 93–99.
[49] R. Wiehagen, A thesis in inductive inference, in: J. Dix, K. Jantke, P. Schmitt (Eds.), Nonmonotonic and Inductive Logic, 1st International Workshop, in: Lecture Notes in Artificial Intelligence, vol. 543, Springer-Verlag, Karlsruhe, Germany, 1991, pp. 184–207.