Strongly Non-U-Shaped Learning Results by General Techniques

John Case
Department of Computer and Information Sciences,
University of Delaware, Newark, DE 19716-2586, USA.
case@cis.udel.edu

Timo Kötzing[*]
Max-Planck Institute for Informatics, 66123 Saarbrücken, Germany
koetzing@mpi-inf.mpg.de

Abstract

In learning, a semantic or behavioral U-shape occurs when a learner first learns, then unlearns, and, finally, relearns, some target concept (on the way to success). Within the framework of Inductive Inference, 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 focus more on syntactic U-shapes.

This paper introduces two general techniques and applies them especially to syntactic U-shapes in learning: one technique to show when they are necessary and one to show when they are unnecessary. The technique for the former is very general and applicable to a much wider range of learning criteria. It employs so-called *self-learning classes of languages* which are shown to *characterize* completely one criterion learning more than another.

We apply these techniques to show that, for set-driven and partially set-driven learning, any kind of U-shapes are unnecessary. Furthermore, we show that U-shapes are *not* unnecessary in a strong way for iterative learning, contrasting an earlier result by Case and Moelius that semantic U-shapes *are* unnecessary for iterative learning.

1 Introduction

In Section 1.1 we explain U-shaped learning. In Section 1.2 we briefly discuss the general techniques of the present paper and summarize in Section 1.3 our applications of these techniques regarding the necessity of U-shaped learning.

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 [SS82], understanding of temperature [SS82], weight conservation [SS82], object permanence [SS82] and face recognition [Car82]. The case of language acquisition is paradigmatic. For example, a child first uses *spoke*, the correct past tense of the irregular verb *speak*. Then the child overregularizes incorrectly using *speaked*. Lastly the child returns to using *spoke*. The language acquisition case of U-shaped learning behavior has figured prominently in cognitive science [MPU$^+$92, TA02].

While the prior cognitive science literature on U-shaped learning was typically concerned with modeling *how* humans achieve U-shaped behavior, [BCM$^+$08, CCJS07] are motivated by the question of *why* humans exhibit this seemingly inefficient behavior. Is it a mere harmless evolutionary 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, in effect, fed an infinite sequence consisting of the elements of a (formal) language L in arbitrary order with possibly some pause symbols $\#$ in between elements.

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

During this process h outputs a corresponding sequence $p(0), p(1), \dots$ 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 [Gol67]. 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* [CL82, OW82] 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* [Cas99, JORS99]. 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.

[BCM$^+$08] showed that every TxtEx-learnable class of languages is TxtEx-learnable by a non-U-shaped learner, that is, for TxtEx-learnability, U-shaped learning is *not* necessary. Furthermore, based on a proof in [FJO94], [BCM$^+$08] noted that, by contrast, for behaviorally correct learning [CL82, OW82], U-shaped learning *is* necessary for full learning power. In [CCJS07] it is shown that, for non-trivial vacillatory learning, U-shaped learning is again necessary (for full learning power). Thus, in many contexts, seemingly inefficient U-shaped learning can actually increase one's learning power.

What turns out to be a variant of non-U-shaped learning is *strongly non-U-shaped* learning essentially defined in [Wie91],[1] 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.[2] In our experience, for theoretical purposes, it is frequently easier to show non-U-shaped learnability by showing *strong* non-U-shaped learnability. Herein we especially study strong non-U-shaped learnability.

1.2 Presented Techniques

The present paper presents two general techniques to tackle problems regarding U-shaped learning.

The first general technique can be used to show the *necessity* of U-shapes and employs so-called *self-learning classes of languages*. These are explained in Section 3 below. These self-learning classes of languages provide a general way for finding classes of languages that separate two learning criteria, i.e., they give a general way of finding an *example* class of languages learnable with a given learning criterion, but not with another. Theorem 3.6 implies that its presented self-learning classes *necessarily* separate two learnability sets — *if any class does.* This technique is 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 3.

The second general technique is used to show that U-shapes are *unnecessary* and is phrased in terms of sufficient conditions for the non-U-shaped learnability of classes of languages. Section 4's Theorems 4.8 and 4.9 provide these sufficient conditions, each for a different kind of U-shapes. Theorem 4.8 actually *characterizes* strong non-U-shaped learnability.

1.3 Applications of General Techniques

A learning machine is *set-driven* [WC80, SR84, Ful90, JORS99] (respectively, *partially set-driven* [SR84, Ful90, JORS99]) iff, at any time, its output conjecture depends only on the *set* of numerical data it has seen (respectively, set and data-sequence length), not on the order of that data's presentation.[3] Child language learning may be insensitive to the order or timing of data presentation; set-drivenness and partial set-drivenness provide two *local* notions of such insensitivity [Cas99]. It is interesting, then, to see the interaction of these notions with forbidding U-shapes of one kind or another. As we shall see in Section 5, Theorems 5.1 and 5.2, proved with the aid of a general technique from Section 4, imply, for these local data order insensitivity notions, for TxtEx-learning, U-shapes, *even in the strong sense* are *un*necessary.

[1]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 [CM08a].

[2]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 see 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 can't *yet* see mere syntactic changes in people's heads.

[3]Note that partially set-driven learning is also known as *rearrangement independent learning*.

An *iterative* learner outputs its conjectures only on the basis of its immediately prior conjecture (if any) and its current datum. As we shall see in Section 5, iterative learning provides a (first) example of a setting in which non-U-shaped and strongly non-U-shaped learning are extensionally distinct: [CM08b] shows semantic U-shapes to be *unnecessary* in iterative learning, while Theorem 5.4 in the present paper implies that they are not *strongly* unnecessary. To prove this latter result, we actually employ a self-learning class of languages that is a bit easier to work with than the relevant one from Theorem 3.6 — although the latter *must* work too (by Theorems 3.6 and 5.4).[4]

Some of our proofs involve subtle infinitary program self-reference arguments employing (variants of) the Operator Recursion Theorem (ORT) from [Cas74, Cas94, JORS99].

1.4 Open Problems

Some problems regarding the necessity of U-shapes of one kind or another still remain open. An *iterative with counter learner* is an iterative learner which, in making its conjectures, also has access to the data-sequence length of the set of numerical data so far. For example, it is still open whether semantic U-shapes are necessary for iterative with counter learning — as asked in [CM08b]. If so, then the relevant self-learning class from Theorem 3.6 below *must* provide a separation.

See the end of Section 5 for some more open problems regarding the necessity of any one of the two kinds of U-shapes for learning criteria of the present paper.

2 Mathematical Preliminaries

Unintroduced computability-theoretic notions follow [Rog67].

Strings herein are finite and over the alphabet $\{0,1\}$. $\{0,1\}^*$ denotes the set of all such strings; ε denotes the empty string.

$\mathbb{N}$ denotes the set of natural numbers, $\{0,1,2,\ldots\}$. We do not distinguish between natural numbers and their *dyadic* representations as strings.[5]

We fix the 1-1 and onto pairing function $\langle\cdot,\cdot\rangle : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ from [RC94, Section 2.3]. In particular, for all x, y,

$$\langle x, y\rangle = \sum_{k=0}^{m} x_k 2^{2k+1} + \sum_{k=0}^{n} y_k 2^{2k}, \tag{1}$$

where $x = \sum_{k=0}^{m} x_k 2^k$, $y = \sum_{k=0}^{n} y_k 2^k$ and $x_0, \ldots, x_m, y_0, \ldots, y_n \in \{0,1\}$. The *binary* representation of $\langle x, y\rangle$ is an interleaving of the *binary* representations of x and y, where we alternate x's and y's digits and start on the right with the least most significant y digit. For example, $\langle 15, 2\rangle = 94$, since $15 = 1111$ (binary), $2 = 0010$ (binary), and $94 = 10101110$ (binary). Define π_1 and π_2 to be the functions such that, for all x and y,

$$\pi_1(\langle x, y\rangle) = x; \tag{2}$$

$$\pi_2(\langle x, y\rangle) = y. \tag{3}$$

π_1 and π_2 are, respectively, called the first and second projection functions.

The symbols $\subseteq, \subset, \supseteq, \supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets.

For sets A, B, we let $A \setminus B = \{a \in A \mid a \notin B\}$.

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, card(A) denotes the cardinality of A.

$\mathfrak{P}$ and $\mathfrak{R}$ denote, respectively, the set of all partial and of all total functions $\mathbb{N} \to \mathbb{N}$. With dom and range 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 $\lambda x_1, \ldots, x_n \centerdot f(x_1, \ldots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x \centerdot c$ is the constantly c function of one argument.

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.

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 $f(x)\downarrow = p$.

[4]Some recent papers [CK08, CK10] have also employed (different) self-learning classes for separations.

[5]The *dyadic* representation of a natural number x = the x-th finite string over $\{0,1\}$ *in length-lexicographical order*, where the counting of strings starts with zero [RC94]. Hence, unlike with binary representation, lead zeros matter.

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.[6]

Computability Notions

We let $\mathcal{P}$ and $\mathcal{R}$, respectively, denote the set of all partial computable and all total computable functions from $\mathbb{N}$ to $\mathbb{N}$; let $\varphi_0, \varphi_1, \ldots$ be any acceptable programming system (numbering) of $\mathcal{P}$ [Rog67]. φ_p is the element of $\mathcal{P}$ computed by the program in this system with numerical name p. Via numerical naming all general purpose real world programming systems are acceptable.

A set $L \subseteq \mathbb{N}$ is *computably enumerable (*ce*)* iff it is the domain of a partial 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. These programs e constitute a hypothesis space for learning in the present paper.

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), it outputs *some* enumeration of the graph of the output. Rogers [Rog67] extensively treats the one-ary case of these operators and calls them *recursive operators.*

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. $\#\mathrm{elets}(\sigma)$ denotes the number of elements in a finite sequence σ, that is, the cardinality of its domain.

Following [LV08], we define for all $x \in \mathbb{N}$: $\overline{x} = 1^{\mathrm{size}(x)}0x$. Using this notation we can define a function $\langle\cdot\rangle_{\mathbb{S}\mathrm{eq}}$ coding arbitrarily long finite sequences of natural numbers into $\mathbb{N}$ (represented dyadically) such that

$$\langle\sigma\rangle_{\mathbb{S}\mathrm{eq}} = \overline{\sigma(0)} \ldots \overline{\sigma(\#\mathrm{elets}(\sigma) - 1)}. \tag{4}$$

In particular, $\langle\emptyset\rangle_{\mathbb{S}\mathrm{eq}} = \varepsilon$.

For example the finite sequence $(4, 7, 10)_{decimal} = (01, 000, 011)_{dyadic}$ is coded as 11 0 01 111 0 000 111 0 011 (but without the spaces, which were added for ease of reading).[7]

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 $\#\mathrm{elets}(\sigma) > 0$, we let $\mathrm{last}(\sigma)$ be the last element of σ and σ^- be σ with its last element deleted. By convention, we set $\emptyset^- = \emptyset$.

Obviously, $\langle\cdot\rangle_{\mathbb{S}\mathrm{eq}}$ is 1-1 [LV08].

Henceforth, we will many times identify a finite sequence σ with its code number $\langle\sigma\rangle_{\mathbb{S}\mathrm{eq}}$. However, when we employ expressions such as $\sigma(x)$, $\sigma = f$ and $\sigma \subset f$, we consider σ as a *sequence*, not as a number.

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 $f(0), \ldots, f(i-1)$.

We fix the following 1-1 coding for all finite subsets of $\mathbb{N}$. For each non-empty finite set $D = \{x_0 < \ldots < x_n\}$, $\langle x_0, \ldots, x_n\rangle_{\mathbb{S}\mathrm{eq}}$ is the code for D and $\langle\rangle_{\mathbb{S}\mathrm{eq}}$ is the code for $\emptyset$.

Henceforth, we will many times identify a finite set D with its code number. However, when we employ expressions such as $x \in D$, $\mathrm{card}(D)$, $\max(D)$ and $D \subset D'$, we consider D and D' as *sets*, not as numbers.

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

Later, we will type infinite sequences as being in $\mathfrak{R}$, *but*, technically, *texts* (for languages $\subseteq \mathbb{N}$) *are infinite sequences*, but they may contain pauses (#s) which are not natural numbers. Also, finite initial segments of texts are example finite sequences which can contain pauses. In our coding above of finite sequences, we code only sequences of natural numbers (and not pauses). To get around this, we will assume from now on that $\mathbb{N} \cup \{\#\}$ is efficiently coded 1-1 onto $\mathbb{N}$, say, by coding # as 0 and $n \in \mathbb{N}$ as $(n+1)$. In this way texts can be thought of as $\in \mathfrak{R}$ and finite initial segments of texts can, then, be coded as sequences of natural numbers. However, for texts T and for finite initial segments

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

[7] 1100111100001110011 is of course the dyadic representation of some number $\in \mathbb{N}$.

of such T, we will, whenever we need to talk about the value of $T(m)$, ignore the coding and work, for example, with whether the actual (not coded) values of $T(m) = \#$ or $= n \in \mathbb{N}$. This ignoring of the coding will be useful from time to time.

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 [Rog67]. 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 monotonic increasing total computable function s such that

$$\forall a, b, c : \varphi_{s(a,b)}(c) = \varphi_a(b, c). \tag{5}$$

In (5), φ-program $s(a, b)$ is essentially φ-program a with datum b stored inside. We will also use a suitably padded version of Case's *Operator Recursion Theorem* (**ORT**), providing *infinitary* self (and other) reference [Cas74, Cas94, JORS99]. **ORT** itself states that, for all computable operators $\Theta : \mathfrak{P} \to \mathfrak{P}$,

$$\exists e \in \mathcal{R} \forall a, b : \varphi_{e(a)}(b) = \Theta(e)(a, b). \tag{6}$$

In the padded version we employ, the function e will also be strictly monotone increasing.

2.1 Computability-Theoretic Learning

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

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 content$(T) = L$. With $\mathbf{Txt}(L)$ we denote the set of all texts for L.

A *sequence generating operator* is a computable 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 can interact with 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 [Gol67], partially set-driven [SR84, Ful85, Ful90, JORS99], set-driven [WC80, JORS99] iterative with counter [CM08b] and iterative [WC80, Wie76], respectively. For all h, T, i,

$$\begin{aligned}
\mathbf{G}(h,T)(i) &= h(T[i]);\\
\mathbf{Psd}(h,T)(i) &= h(\mathrm{content}(T[i]), i);\\
\mathbf{Sd}(h,T)(i) &= h(\mathrm{content}(T[i]));\\
\mathbf{ItCtr}(h,T)(i) &= \begin{cases} ?, & \text{if } i = 0;\\ h(\mathbf{ItCtr}(h,T)(i-1), T(i-1), i-1), & \text{otherwise;}\end{cases}\\
\mathbf{It}(h,T)(i) &= \begin{cases} ?, & \text{if } i = 0;\\ h(\mathbf{It}(h,T)(i-1), T(i-1)), & \text{otherwise.}\end{cases}
\end{aligned}$$

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

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

- No restriction: The constantly true predicate of the appropriate type **T**.
- Total Learner: $\forall \beta, h, L : \mathcal{R}_\beta(h, L) \Leftrightarrow h \in \mathcal{R}$.
- Non-U-shaped: $\forall \beta, h, L : \mathbf{NU}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L), \forall i : (W_{\beta(h,T)(i)} = L \Rightarrow W_{\beta(h,T)(i+1)} = W_{\beta(h,T)(i)})]$.
- Strongly non-U-shaped: $\forall \beta, h, L : \mathbf{SNU}_\beta(h, L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \forall i : (W_{\beta(h,T)(i)} = L \Rightarrow \beta(h,T)(i+1) = \beta(h,T)(i))]$.

We combine any two learning restrictions by intersecting them, and we denote this combination by juxtaposition.

In order to motivate and define another group of learning restrictions, we now define the concept of a stabilizer sequence (called “stabilizer segment” in [Ful90]).

Let β be a sequence generating operator. Let L be a language and $h \in \mathcal{P}$ a learner. A sequence $\sigma \in \mathbb{S}\mathrm{eq}(L)$ is said to be a *β-stabilizer sequence of h on L* iff

$$(\forall T \in \mathbf{Txt}(L) | \sigma \subseteq T) \forall i \geq \#\mathrm{elets}(\sigma) : \beta(h,T)(\#\mathrm{elets}(\sigma)) = \beta(h,T)(i); \tag{7}$$

Intuitively, a stabilizer sequence σ of h on L is a sequence of elements from L such that h on any text extending σ will never make a change of conjecture after having seen σ.

It is well known that, if a learner h TxtEx-learns a language L, then there is a stabilizer sequence of h on L (see [JORS99]). However, texts don't necessarily contain such a sequence as an initial segment. Below, we define a learning restriction that requires a learner and a language to have stabilizer sequences as initial sequences of *all* texts for the language.

- Stabilizing: $\forall \beta, h, L$:

$$\mathbf{Stab}_\beta(h,L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \exists i_0 : T[i_0] \text{ is a } \beta\text{-stabilizer sequence of } h \text{ on } L].$$

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

$$\forall T \in \mathbf{Txt}(L) \forall i_0 : [\beta(h,T)(i_0) = e \Rightarrow (\forall i \geq i_0)(\beta(h,T)(i) = e)]. \tag{8}$$

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

We specialize the concept of a stabilizer sequence to a *sink-stabilizer sequence*. This will be technically helpful for some of our results.

- Sink-stabilizing: $\forall \beta, h, L$:

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

Clearly, for all β, h, L, $\mathbf{Sink}_\beta(h,L)$ implies $\mathbf{Stab}_\beta(h,L)$. Sink-stabilizing is of interest, as we show a characterization theorem (Theorem 4.8 below) of strong non-U-shaped learning in terms of sink-stabilizing learning.

In order to obtain at least a sufficient condition for (not necessarily strongly) non-U-shaped learning, we weaken the concept of a sink as follows. For now, let $f \in \mathcal{P}$. An *f-weak β-sink of h on L* is a conjecture e such that

$$\forall T \in \mathbf{Txt}(L) \forall i_0 : [\beta(h,T)(i_0) = e \Rightarrow (\forall i \geq i_0)(f(\beta(h,T)(i), e) = 1)]. \tag{9}$$

We would like to employ for f above $f_0 = \lambda e, e' \boldsymbol{.} W_e = W_{e'}$, in order to capture the notion of a “never semantically abandoned conjecture;” however, this function is not computable. Instead, we will use functions “approximating” f_0: Let

$$\mathcal{F} = \{f \in \mathcal{R} \mid \forall e, e' : (f(e,e') = 1 \Rightarrow W_e = W_{e'})\}.[8] \tag{10}$$

For all $f \in \mathcal{F}$, we define the following learning restriction.

- f-weak β-sink-stabilizing: $\forall \beta, h, L$:

$$\mathbf{Weaksink}^f_\beta(h,L) \Leftrightarrow [\forall T \in \mathbf{Txt}(L) \exists e : (\beta(h,T) \to e \ \wedge \ e \text{ is an } f\text{-weak } \beta\text{-sink of } h \text{ on } L].$$

We are now ready to give some formal definitions for successful learning.

Definition 2.1.

- For this paper, a *learning criterion* is a pair (α, β) such that α is a learning restriction and β a sequence generating operator. We also write $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ to denote the learning criterion (α, β).
- Let (α, β) be a learning criterion and h a learner. We say that h $\alpha\mathbf{Txt}\beta\mathbf{Ex}$-*learns* a language L iff $\alpha_\beta(h,L)$ and, for all texts T for L, $\beta(h,T)$ is total and there is e with $\beta(h,T) \to e$ and $W_e = L$.
- We denote the class of all languages $\alpha\mathbf{Txt}\beta\mathbf{Ex}$-learned by h with $\alpha\mathbf{Txt}\beta\mathbf{Ex}(h)$. Abusing notation, we use $\alpha\mathbf{Txt}\beta\mathbf{Ex}$ to denote the set of all classes of languages $\alpha\mathbf{Txt}\beta\mathbf{Ex}$-learnable by some learner (as well as the learning criterion).

[8]Intuitively, all $f \in \mathcal{F}$ only output 1 if the inputs are semantically equivalent.

- We omit α if $\alpha = \mathbf{T}$.

We let

$$\texttt{The30} = \begin{array}{l} \{(\alpha,\beta) \mid \alpha \in \{\mathbf{T},\mathbf{NU},\mathbf{SNU},\mathbf{Sink},\mathbf{Stab}\}, \beta \in \{\mathbf{G},\mathbf{Psd},\mathbf{Sd},\mathbf{ItCtr},\mathbf{It}\}\} \cup \\ \{\bigcup_{f\in\mathcal{F}}(\mathbf{Weaksink}^f,\beta) \mid \beta \in \{\mathbf{G},\mathbf{Psd},\mathbf{Sd},\mathbf{ItCtr},\mathbf{It}\}\}. \end{array} \tag{11}$$

This is a set of thirty *particular* learning criteria especially considered in this paper.

As noted above in Section 1.2, Theorem 3.6 below applies to learning criteria separations well beyond our concerns in the present paper with showing U-shapes do (or don't) make a difference in learning power. In particular we will see that Theorem 3.6 applies to *all* pairs of learning criteria from `The30`. Indirectly in Section 4 and directly in Section 5, though, we are mainly concerned with pairs of criteria (I_0, I_1), with each criterion from `The30`, *where* $I_0 = \mathbf{SNU}I_1$ (or $I_0 = \mathbf{NU}I_1$).

Starred Learners

For a learner h, possibly learning with restricted access to past data, we write $h^*(\sigma)$ for what the current output of h is after being fed the sequence σ of data items.

In particular, for $h \in \mathcal{P}$ and σ a sequence, we have the following.

- If h is a set-driven learner:
$$h^*(\sigma) = h(\text{content}(\sigma)). \tag{12}$$
- If h is a partially set-driven learner:
$$h^*(\sigma) = h(\text{content}(\sigma), \#\text{elets}(\sigma)). \tag{13}$$

3 Self-Learning Classes of Languages for Separations

In this section we discuss a way of showing U-shapes to be *necessary*. Formally, this is done via showing that a learnability class *separates* from its non-U-shaped variant.

The approach described below is very general and can be applied to show separation results in many other areas of computability-theoretic learning in the limit as well.

The key idea is that of *self-learning classes of languages*. In the previous literature, self-*describing* classes of languages have been used.[9] A particularly simple example class of self-describing languages, taken from [CL82, Theorem 1], is

$$\mathcal{L}_0 = \{L \text{ recursive} \mid L \neq \emptyset \ \wedge \ W_{\min L} = L\}. \tag{14}$$

Intuitively, each $L \in \mathcal{L}_0$ gives a complete *description* of itself, encoded (as a W-index) within only finitely many (in fact, one) of its elements. It is well-known, using standard computability theoretic arguments, that these kind of classes of languages are very big (for example, $\mathcal{L}_0$ contains a finite variant of any given `ce` language, which can easily be seen using Kleene's Recursion Theorem).

Many variants of self-describing classes of languages have been used for separation results within computability-theoretic learning (see, for example, [BB75, CL82, CS83, Cas99, JORS99]). Showing a separation with a complicated self-describing class of languages sometimes requires a non-trivial learner (see, for extreme examples, [CJLZ99]).

We now take the technique of self-describing one step further. 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. Thus, all a learner needs to do is to execute the instructions given by its latest datum. For example, an informal[10] learner h_1 can be defined such that

$$\forall \sigma, x : h_1(\sigma \diamond x) = \varphi_x(\sigma \diamond x). \tag{15}$$

Intuitively, h_1 interprets the latest datum as a program in the φ-system and runs this program on all known data. Variants of this h_1 can be defined to obtain learners with special additional properties, such as totality or set-drivenness (see Theorem 3.6).

In practice, the general scheme is as follows. Suppose we want to show, for two learning criteria I_0 and I_1, $I_1 \setminus I_0 \neq \emptyset$. Then, we define a simple learner, for example h_1 above, and let $\mathcal{L}_1$ be the class of all languages I_1-learned by h_1. All that would remain to be shown is that $\mathcal{L}_1$ is not I_0-learnable, which can often be done using **ORT**.

[9] See [JORS99]. In there, the term "self-describing" was used on page 71 in the context of function learning and extended on page 97 to language learning.

[10] Note that we ignore the possible input of $x = \#$.

Below, in Theorem 3.6, we give a *very general* result regarding some self-learning classes of languages *guaranteed* to witness separations when they exist. In order to do so, we proceed next by making some formal definitions.

For a function $e \in \mathcal{P}$ and a language L, we let $e(L) = \{e(x) \mid x \in L\}$; for a class of languages $\mathcal{L}$, we let $e(\mathcal{L}) = \{e(L) \mid L \in \mathcal{L}\}$.

Let $\mathcal{P}_{\text{c1-1}} \subseteq \mathcal{P}$ (repectively, $\mathcal{R}_{\text{c1-1}} \subseteq \mathcal{R}$), denote the set of all 1-1 partial (respectively, total) computable functions with computable domain and range.

Definition 3.1. A learning criterion I is called *computably $\mathcal{P}_{\text{c1-1}}$-robust* iff there is a computable operator $\Theta : \mathfrak{P}^2 \to \mathfrak{P}$ such that

$$\forall h \in \mathcal{P}, \forall e \in \mathcal{P}_{\text{c1-1}} : e(I(h)) \subseteq I(\Theta(h,e)). \tag{16}$$

Intuitively, if a learner h learns languages L, then $\Theta(h,e)$ learns $e(L)$.

Remark 3.2. Let I be computably $\mathcal{P}_{\text{c1-1}}$-robust. Then we have

$$\forall e \in \mathcal{P}_{\text{c1-1}}, \forall \mathcal{L} \subseteq \mathcal{E} : \mathcal{L} \in I \Leftrightarrow e(\mathcal{L}) \in I.$$

Remark 3.3. Let $(\alpha, \beta) \in \texttt{The30}$. Then (α, β) is computably $\mathcal{P}_{\text{c1-1}}$-robust.

Definition 3.4. Let $I = (\alpha, \beta)$ be a learning criterion. We call I *data normal* iff, for all p_0 such that $W_{p_0} = \emptyset$, there is a computable operator $\hat{\Theta} : \mathfrak{P} \to \mathfrak{P}$ such that

$$I(h) \subseteq I(\hat{\Theta}(h)) \tag{17}$$

and (a) – (d) below.

(a) There is $f_\beta \in \mathcal{R}$ such that

$$\forall h, T, \forall i > 0 : \beta(h,T)(i) = h(f_\beta(T[i], \beta(h,T)[i])).^{11} \tag{18}$$

(b) There is a function $d_\beta \in \mathcal{R}$ such that

$$\begin{aligned} \forall T \in \mathbf{Txt}, i \in \mathbb{N} : \quad & \beta(\hat{\Theta}(h), T)[i]\downarrow \Rightarrow \\ & d_\beta(f_\beta(T[i], \beta(\hat{\Theta}(h), T)[i])) \in \begin{cases} \{\#\} & \text{if content}(T[i]) = \emptyset; \\ \text{content}(T[i]), & \text{otherwise.}^{12} \end{cases} \end{aligned} \tag{19}$$

(c) For all $h \in \mathcal{P}$,

$$\forall \sigma, \tau : d_\beta(f_\beta(\sigma, \tau)) = \# \Rightarrow \hat{\Theta}(h)(f_\beta(\sigma, \tau)) = p_0.^{13} \tag{20}$$

(d) For all $h, h' \in \mathcal{P}$,

$$[\forall L \in I(h) \forall T \in \mathbf{Txt}(L) : \beta(h,T) = \beta(h',T)] \Rightarrow I(h) \subseteq I(h').^{14} \tag{21}$$

Remark 3.5. Let $(\alpha, \beta) \in \texttt{The30}$. Then (α, β) is data normal.

Next (Theorem 3.6) is the main result of this section, giving *sufficient* conditions for when a separation will necessarily be witnessed by a *specific* self-learning class of languages. As a corollary, we get that *for each pair of learning criteria from The30*, any separations are witnessed by such classes! In this sense, self-learning classes of languages capture the *essence* of separations (when they exist). Note that the proof of the theorem would simplify a lot, were one to suppose somewhat stronger properties of the learning criteria, in particular, excluding the use of **It** and **ItCtr** as sequence generating operators. Theorem 3.6 can be modified to cover other kinds of criteria, for example, those pertaining to learnability by total learners. In a future paper, we will analyze self learning-classes in more depth and will provide further theorems like Theorem 3.6.

[11] Intuitively, the i-th conjecture of h on T depends only on some information (as specified by f_β) about the first i datapoints and conjectures.

[12] Intuitively, from the information given by f_β, a datum (if any) that this datum is based on can be extracted.

[13] Intuitively, constantly outputting one and the same index for the empty language is a viable strategy as long as no numerical data has been presented.

[14] Intuitively, changing a learner on inputs that do not present data from a language learned does not harm learnability.

Theorem 3.6. Learning criteria are as in Definition 2.1. Let I_0 and I_1 be computably $\mathcal{P}_{\text{c1-1}}$-robust learning criteria. Suppose I_1 is data normal as witnessed by f_1 and d_1. Let p_0 be such that $W_{p_0} = \emptyset$ and h_1 be such that

$$\forall x : h_1(x) = \begin{cases} p_0, & \text{if } d_1(x) = \#; \\ \varphi_{d_1(x)}(x), & \text{otherwise.} \end{cases} \tag{22}$$

Further, let $\mathcal{L}_1 = I_1(h_1)$. Then we have

$$I_1 \setminus I_0 \neq \emptyset \Leftrightarrow \mathcal{L}_1 \notin I_0.$$

Proof. The implication "$\Leftarrow$" is obvious. Regarding "$\Rightarrow$", let $\mathcal{L} \in I_1$ as witnessed by h and suppose $\mathcal{L} \notin I_0$. Let Θ be as given by I_1 being computably $\mathcal{P}_{\text{c1-1}}$-robust. Let $\hat{\Theta}$ be as given by I_1 being data normal. By padded **ORT**, there is a strictly monotone increasing $e \in \mathcal{R}$ such that

$$\forall x, y : \varphi_{e(x)}(y) = (\hat{\Theta} \circ \Theta)(h, e)(y). \tag{23}$$

As $e \in \mathcal{P}_{\text{c1-1}}$ and I_0 is computably $\mathcal{P}_{\text{c1-1}}$-robust, we have, from Remark 3.2 with I_0 in the place of I, $e(\mathcal{L}) \notin I_0$. It now suffices to show $e(\mathcal{L}) \subseteq \mathcal{L}_1$, as this would imply $\mathcal{L}_1 \notin I_0$ as desired.

Suppose $I_1 = (\alpha_1, \beta_1)$. Let $L \in e(\mathcal{L})$ and $T \in \mathbf{Txt}(L)$. We show, by induction on i,

$$\forall i : \beta_1(h_1, T)(i) = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i). \tag{24}$$

Let $h' = \Theta(h, e)$. For all i with content$(T[i]) = \emptyset$, we have

$$\begin{aligned} \beta_1(h_1, T)[i] &\underset{(18)}{=} h_1(f_1(T[i], \beta_1(h_1, T)[i])) \underset{(22)}{=} p_0 \\ &\underset{(19)\ \&\ (20)}{=} \hat{\Theta}(h')(f_1(T[i], \beta_1(\hat{\Theta}(h'), T)[i])) \underset{(18)}{=} \beta_1(\hat{\Theta}(h'), T)(i). \end{aligned} \tag{25}$$

Let $i \in \mathbb{N}$, suppose content$(T[i]) \neq \emptyset$ and (inductively) $\beta_1(h_1, T)[i] = \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]$. Let

$$x = f_1(T[i], \beta_1(h_1, T)[i]) \underset{\text{IH}}{=} f_1(T[i], \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)[i]). \tag{26}$$

Note that $d_1(x) \underset{(19)}{\in}$ content$(T[i]) \subseteq L \subseteq$ range(e). We have

$$\beta_1(h_1, T)(i) \underset{(18)\ \&\ (26)}{=} h_1(x) \underset{(22)}{=} \varphi_{d_1(x)}(x) \underset{(23)}{=} (\hat{\Theta} \circ \Theta)(h, e)(x) \underset{(18)\ \&\ (26)}{=} \beta_1((\hat{\Theta} \circ \Theta)(h, e), T)(i). \tag{27}$$

This concludes the induction. Thus, h_1 on any text for a language from $e(\mathcal{L})$ makes the same conjectures as $(\hat{\Theta} \circ \Theta)(h, e)$ on T. By (16) and (17), $(\hat{\Theta} \circ \Theta)(h, e)$ I_1-learns $e(\mathcal{L})$; thus, using (d) of I_1 being data normal, $e(\mathcal{L}) \subseteq I_1(h_1) = \mathcal{L}_1$. □

4 Helping Remove U-Shapes

In this section we provide, in Theorem 4.8, a general technique for helping with the *removal* of U-shapes from a learner, *preserving what is learned*. When applicable, this shows U-shapes *un*necessary. Theorem 4.8 is technically a characterization theorem, and is applied in Section 5 to provide cases where *strong* non U-shaped learning makes no difference. Also, in this section is another result, Theorem 4.9, which could similarly be used to provide other cases where mere non U-shaped learning makes no difference — although we do not apply this theorem in the present paper.

Lemma 4.1. Let $h \in \mathcal{P}$. Then there is a infinite set $L \in \mathcal{E}$ such that h does not **TxtGEx**-learn any $L' \supseteq L$ such that $L' =^* L$.

Proof.[15] Trivial if $\mathbb{N} \notin \mathbf{TxtGEx}(h)$. Otherwise, let σ be a locking sequence for h on $\mathbb{N}$, $D =$ content(σ). Then, obviously, h does not learn any L such that $D \subseteq L \subset \mathbb{N}$. □

Definition 4.2. For each $h \in \mathcal{P}$, let L_h denote a set L corresponding to h and as shown existent in Lemma 4.1.

Definition 4.3. Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and let Q be a ce set. Then, using padded s-m-n, there is a strictly monotone increasing function $p_{h,Q} \in \mathcal{R}$

$$\forall e, x : W_{p_{h,Q}(e,x)} = \{y \in L_h \mid Q(e, x)\} \cup \{y \in W_e \mid \text{not } (Q(e, x) \text{ in } \leq y \text{ steps})\}. \tag{28}$$

[15]This version of the proof is due to Frank Stephan [Ste09].

Lemma 4.4. Let $h \in \mathcal{P}$, $\mathcal{L} = \mathbf{TxtGEx}(h)$ and Q be a ce set. For all e, x, we have

$$W_{p_{h,Q}(e,x)} \in \mathcal{L} \quad \Rightarrow \quad \neg Q(e,x) \tag{29}$$

$$\Rightarrow \quad W_{p_{h,Q}(e,x)} = W_e. \tag{30}$$

Proof. Immediate from the choice of L_h. □

Lemma 4.5. Let $\beta \in \{\mathbf{It}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{Psd}, \mathbf{G}\}$. We have that

$$\{\langle e_0, e_1, e_2\rangle \mid e_0 \text{ is } \textit{not} \text{ a } \beta\text{-sink of } \varphi_{e_1} \text{ on } W_{e_2}\} \text{ is } \mathsf{ce} \tag{31}$$

and, for all $f \in \mathcal{F}$ (where $\mathcal{F}$ is from (10)),

$$\{\langle e_0, e_1, e_2\rangle \mid e_0 \text{ is } \textit{not} \text{ an } f\text{-weak } \beta\text{-sink of } \varphi_{e_1} \text{ on } W_{e_2}\} \text{ is } \mathsf{ce}. \tag{32}$$

Proof. Immediate. □

Remember that $\mathcal{R}_{\text{c1-1}} \subseteq \mathcal{R}$ denotes the set of all 1-1 *total* computable functions with computable domain and range.

Definition 4.6. A sequence generating operator β is called *1-1 left-modifiable* iff

$$\forall r \in \mathcal{R}_{\text{c1-1}}\ \exists s_l, s_r \in \mathcal{R}\ \forall h \in \mathcal{P}, T \in \mathbf{Txt} : \beta(s_l \circ h \circ s_r, T) = r \circ \beta(h, T).$$

Remark 4.7. $\mathbf{It}, \mathbf{Sd}, \mathbf{ItCtr}, \mathbf{Psd}$ and $\mathbf{G}$ are 1-1 left-modifiable.

We now present the two main Theorems of this section, the first of which characterizes strong non-U-shaped learning; the second is a sufficient condition on (not necessarily strong) non-U-shaped learning.

Theorem 4.8. Let β be 1-1 left-modifiable and fulfill (a) of data normal. Let $\alpha \in \{\mathbf{T}, \mathcal{R}\}$. Then

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

Proof. "$\supseteq$": Let $h_0 \in \mathcal{P}$, $\mathcal{L} = \mathbf{SNUTxt}\beta\mathbf{Ex}(h_0)$. Let $L \in \mathcal{L}$ and let T be a text for L. Let k be such that h_0 has converged on T after $T[k]$ to some e. Then

$$W_e = L. \tag{33}$$

To show that e is a β-sink of h_0 on L: Let $T \in \mathbf{Txt}(L)$ and i_0 such that $\beta(h_0, T)(i_0) = e$. Let $i \geq i_0$. Then, as h_0 is strongly non-U-shaped on L and from (33), $\beta(h_0, T)(i) = e$.

"$\subseteq$": Let $h_0 \in \mathcal{P}$, $\mathcal{L} = \mathbf{SinkTxt}\beta\mathbf{Ex}(h_0)$. Let Q be a ce predicate such that

$$\forall e : Q(e) \Leftrightarrow e \text{ is } \textit{not} \text{ a } \beta\text{-sink of } h_0 \text{ on } W_e. \tag{34}$$

With f_β as given by (a) of data normal, let $h_0^* = h_0 \circ f_\beta$. Let $p = p_{h_0^*,Q}$ as in Definition 4.3. Let β's left-modifiability with respect to p be witnessed by s_l and $s_r \in \mathcal{R}$. Let $h \in \mathcal{R}$ be such that

$$h = s_l \circ h_0 \circ s_r. \tag{35}$$

Note that, if $h_0 \in \mathcal{R}$, then $h \in \mathcal{R}$.

Claim 1. h is strongly non-U-shaped.
Proof of Claim 1. Let $L \in \mathcal{L}$, $e \in \mathbb{N}$ and σ such that content$(\sigma) \subseteq L$. Suppose $h^*(\sigma) = p(e)$ and

$$W_{p(e)} = L \in \mathcal{L}. \tag{36}$$

From (29) we get $\neg Q(e)$. Hence, from the definition of Q in (34), for all texts T for L extending σ, we have that h_0 has syntactically converged after seeing σ. Thus, h has syntactically converged after seeing σ (and, thus, does not exhibit a U-shape). □ (FOR CLAIM 1)

Claim 2. $\mathcal{L} \subseteq \mathbf{Txt}\beta\mathbf{Ex}(h)$.
Proof of Claim 2. Let $L \in \mathcal{L}$ and let T be a text for L. As h_0 is sink-locking, there is k minimal such that, with $e = h_0^*(T[k])$,

$$e \text{ is a sink of } h \text{ on } L. \tag{37}$$

Thus, h_0 converges on T to e; therefore,

$$W_e = L. \tag{38}$$

Furthermore, h on T converges to $p(e)$ and, by (37), $\neg Q(e)$; hence,

$$W_{p(e)} \underset{(30)}{=} W_e \underset{(38)}{=} L. \tag{39}$$

$\square$ (for Claim 2)

$\square$

Theorem 4.9. Let $\mathcal{F}$ be as in (10). Let β be 1-1 left-modifiable and fulfill (a) of data normal. Let $\alpha \in \{\mathbf{T}, \mathcal{R}\}$. Then

$$\bigcup_{f \in \mathcal{F}} \mathbf{Weaksink}^f \alpha \mathbf{Txt} \beta \mathbf{Ex} \subseteq \mathbf{NU} \alpha \mathbf{Txt} \beta \mathbf{Ex}.$$

The proof is analogous to that of the proof of "$\subseteq$" of Theorem 4.8.

Theorem 4.9 gives a good way of showing the non-U-shaped learnability of a class of languages: To define a $\bigcup_{f \in \mathcal{F}} \mathbf{Weaksink}^f \alpha \mathbf{Txt} \beta \mathbf{Ex}$-learner to learn the class, one can use a strictly monotone increasing computable function $p \in \mathcal{R}$ (called a *padding function*) on two arguments such that $\forall e, x : W_{p(e,x)} = W_e$ and let $f \in \mathcal{F}$ be such that $f(p(e,x), p(e', x')) = 1$ iff $e = e'$.

5 Applications of the Techniques to The30

In this section we essentially apply general techniques presented in the two just previous sections to prove a number of results pertaining to the necessity of U-shapes in learning.

Note that [CM07] implies that $\mathcal{R}\mathbf{TxtSdEx} \subset \mathbf{TxtSdEx}$. This separation can also be shown using Theorem 3.6.

Theorem 5.1. We have

(i) $\mathbf{TxtSdEx} = \mathbf{SNUTxtSdEx}$ and
(ii) $\mathcal{R}\mathbf{TxtSdEx} = \mathbf{SNU}\mathcal{R}\mathbf{TxtSdEx}$.

Our proof of this theorem involves an application of Theorem 4.8.

Theorem 5.2.

$$\mathbf{SNU}\mathcal{R}\mathbf{TxtPsdEx} = \mathbf{TxtPsdEx}.$$

Our proof of this theorem involves an application of Theorem 4.8.

As $\mathbf{TxtGEx} = \mathbf{TxtPsdEx}$ [SR84, Ful85, Ful90, JORS99], we immediately get the following corollary, reproving a result from [BCM+08] and one from [CM08a].

Corollary 5.3.

$$\mathbf{NUTxtGEx} \underset{\text{[BCM}^+\text{08]}}{=} \mathbf{TxtGEx} \underset{\text{[CM08a]}}{=} \mathbf{SNUTxtGEx}.$$

From [CM08b, Theorem 2] we have $\mathbf{TxtItEx} = \mathbf{NUTxtItEx}$. Contrasting this result, we have the following theorem.

Theorem 5.4.

$$\mathbf{SNUTxtItEx} \subset \mathbf{NUTxtItEx}.$$

Our proof makes use of padded **ORT** and, as noted above, for convenience, a self-learning class simpler to work with than that from Theorem 3.6.

6 Summary and Open Problems

Summing up, the following main results within the focus of the present paper and regarding the necessity of syntactic or semantic U-shapes are shown or already known. Some of the justificatory remarks omit citing **TxtGEx** = **TxtPsdEx** [SR84, Ful90] (and trivial inclusions).

$$\begin{array}{rcccl}
\mathbf{SNUTxtGEx} & \underset{\text{[CM08a], Cor 5.3}}{=} & \mathbf{NUTxtGEx} & \underset{\text{[BCM}^{+}\text{08], Cor 5.3}}{=} & \mathbf{TxtGEx};\\
\mathbf{SNU}\mathcal{R}\mathbf{TxtGEx} & \underset{\text{Thm 5.2}}{=} & \mathbf{NU}\mathcal{R}\mathbf{TxtGEx} & \underset{\text{[BCM}^{+}\text{08], Thm 5.2}}{=} & \mathbf{Txt}\mathcal{R}\mathbf{GEx};\\
\mathbf{SNUTxtPsdEx} & \underset{\text{Thm 5.2}}{=} & \mathbf{NUTxtPsdEx} & \underset{\text{Thm 5.2}}{=} & \mathbf{TxtPsdEx};\\
\mathbf{SNU}\mathcal{R}\mathbf{TxtPsdEx} & \underset{\text{Thm 5.2}}{=} & \mathbf{NU}\mathcal{R}\mathbf{TxtPsdEx} & \underset{\text{Thm 5.2}}{=} & \mathbf{Txt}\mathcal{R}\mathbf{PsdEx};\\
\mathbf{SNUTxtSdEx} & \underset{\text{Thm 5.1}}{=} & \mathbf{NUTxtSdEx} & \underset{\text{Thm 5.1}}{=} & \mathbf{TxtSdEx};\\
\mathbf{SNU}\mathcal{R}\mathbf{TxtSdEx} & \underset{\text{Thm 5.1}}{=} & \mathbf{NU}\mathcal{R}\mathbf{TxtSdEx} & \underset{\text{Thm 5.1}}{=} & \mathbf{Txt}\mathcal{R}\mathbf{SdEx};\\
\mathbf{SNUTxtItEx} & \underset{\text{Thm 5.4}}{\subset} & \mathbf{NUTxtItEx} & \underset{\text{[CM08b]}}{=} & \mathbf{TxtItEx}.
\end{array}$$

Trivially, we have

$$\begin{array}{rcccl}
\mathbf{SNU}\mathcal{R}\mathbf{TxtItEx} & \subseteq & \mathbf{NU}\mathcal{R}\mathbf{TxtItEx} & \subseteq & \mathcal{R}\mathbf{TxtItEx};\\
\mathbf{SNUTxtItCtrEx} & \subseteq & \mathbf{NUTxtItCtrEx} & \subseteq & \mathbf{TxtItCtrEx};\\
\mathbf{SNU}\mathcal{R}\mathbf{TxtItCtrEx} & \subseteq & \mathbf{NU}\mathcal{R}\mathbf{TxtItCtrEx} & \subseteq & \mathcal{R}\mathbf{TxtItCtrEx}.
\end{array}$$

The other directions of inclusions are *open*. We think that $\mathbf{NU}\mathcal{R}\mathbf{TxtItEx} = \mathcal{R}\mathbf{TxtItEx}$ can be obtained as a corollary to the proof of **NUTxtItEx** = **TxtItEx** in [CM08b]. Furthermore, we suspect we can show $\mathbf{SNU}\mathcal{R}\mathbf{TxtItEx} \subset \mathbf{NU}\mathcal{R}\mathbf{TxtItEx}$ by a variant of the proof above of Theorem 5.4.

Acknowledgements

We would like to thank the anonymous referees for their comments and suggestions. Further, we want to thank Samuel E. Moelius and Thomas Zeugmann for discussions on various theorems and their proofs.

References

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

[BCM^{+}08] G. Baliga, J. Case, W. Merkle, F. Stephan, and W. Wiehagen. When unlearning helps. *Information and Computation*, 206:694–709, 2008.

[Car82] S. Carey. Face perception: Anomalies of development. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*, Developmental Psychology Series. Academic Press, NY, 1982.

[Cas74] J. Case. Periodicity in generations of automata. *Mathematical Systems Theory*, 8:15–32, 1974.

[Cas94] J. Case. Infinitary self-reference in learning theory. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:3–16, 1994.

[Cas99] J. Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.

[CCJS07] L. Carlucci, J. Case, S. Jain, and F. Stephan. Non-U-shaped vacillatory and team learning. *Journal of Computer and System Sciences*, 2007. Special issue in memory of Carl Smith.

[CJLZ99] J. Case, S. Jain, S. Lange, and T. Zeugmann. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110, 1999.

[CK08] J. Case and T. Kötzing. Dynamically delayed postdictive completeness and consistency in learning. In *19th International Conference on Algorithmic Learning Theory (ALT'08)*, volume 5254 of *Lecture Notes in Artificial Intelligence*, pages 389–403. Springer, 2008.

[CK10] J. Case and T. Kötzing. Solutions to open questions for non-U-shaped learning with memory limitations, 2010. Submitted to *ALT'10.*

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

[CM07] J. Case and S. Moelius. Parallelism increases iterative learning power. In *ALT '07: Proceedings of the 18th international conference on Algorithmic Learning Theory*, pages 49–63, Berlin, Heidelberg, 2007. Springer-Verlag.

[CM08a] J. Case and S. Moelius. Optimal language learning. In *ALT*, volume 5254 of *Lecture Notes in Computer Science*, pages 419–433. Springer, 2008.

[CM08b] J. Case and S. Moelius. U-shaped, iterative, and iterative-with-counter learning. *Machine Learning*, 72(1-2):63–88, 2008.

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

[FJO94] M. Fulk, S. Jain, and D. Osherson. Open problems in Systems That Learn. *Journal of Computer and System Sciences*, 49(3):589–604, December 1994.

[Ful85] M. Fulk. *A Study of Inductive Inference Machines.* PhD thesis, SUNY at Buffalo, 1985.

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

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

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

[LV08] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer Verlag, third edition, 2008.

[MPU$^+$92] G. Marcus, S. Pinker, M. Ullman, M. Hollander, T.J. Rosen, and F. Xu. *Overregularization in Language Acquisition.* Monographs of the Society for Research in Child Development, vol. 57, no. 4. University of Chicago Press, 1992. Includes commentary by H. Clahsen.

[OW82] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.

[RC94] J. Royer and J. Case. *Subrecursive Programming Systems: Complexity and Succinctness.* Research monograph in *Progress in Theoretical Computer Science.* Birkhäuser Boston, 1994.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.

[SR84] G. Schäfer-Richter. *Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien.* PhD thesis, RWTH Aachen, 1984.

[SS82] S. Strauss and R. Stavy, editors. *U-Shaped Behavioral Growth.* Developmental Psychology Series. Academic Press, NY, 1982.

[Ste09] F. Stephan, 2009. Private communication.

[TA02] N. Taatgen and J. Anderson. Why do children learn to say broke? A model of learning the past tense without feedback. *Cognition*, 86(2):123–155, 2002.

[WC80] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition.* MIT Press, Cambridge, Mass, 1980.

[Wie76] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationverarbeitung und Kybernetik*, 12:93–99, 1976.

[Wie91] R. Wiehagen. A thesis in inductive inference. In P. Schmitt J. Dix, K. Jantke, editor, *Nonmonotonic and Inductive Logic, 1st International Workshop*, volume 543 of *Lecture Notes in Artificial Intelligence*, pages 184–207. Springer-Verlag, Karlsruhe, Germany 1991.