Learning with ordinal-bounded memory from positive data

Lorenzo Carlucci[a,1], Sanjay Jain[b,2], Frank Stephan[c,2]

[a] *Department of Computer Science, University of Rome I, 00198, Roma, Italy*
[b] *Department of Computer Science, National University of Singapore, Singapore 117417*
[c] *Department of Mathematics and Department of Computer Science, National University of Singapore, Singapore 119076*

Abstract

A Bounded Example Memory learner operates incrementally and maintains a memory of finitely many data items. The paradigm is well-studied and known to coincide with set-driven learning. A hierarchy of stronger and stronger learning criteria had earlier been obtained when one considers, for each $k \in \mathbf{N}$, iterative learners that can maintain a memory of at most k previously processed data items. We investigate an extension of the paradigm into the constructive transfinite. For this purpose we use Kleene's universal ordinal notation system $\mathcal{O}$. To each ordinal notation in $\mathcal{O}$ one can associate a learning criterion in which the number of times a learner can extend its example memory is bounded by an algorithmic count-down from the notation. We prove a general hierarchy result: if b is larger than a in Kleene's system, then learners that extend their example memory "at most b times" can learn strictly more than learners that can extend their example memory "at most a times". For notations for ordinals below ω^2 the result only depends on the ordinals and is notation-independent. For higher ordinals it is notation-dependent. In the setting of learners with ordinal-bounded memory, we also study the impact of requiring that a learner cannot discard an element from memory without replacing it with a new one. A learner satisfying this condition is called cumulative.

Keywords: Inductive Inference, Bounded Example Memory, Constructive Ordinals, Kolmogorov Complexity

Email addresses: carlucci@di.uniroma1.it (Lorenzo Carlucci), sanjay@comp.nus.edu.sg (Sanjay Jain), fstephan@comp.nus.edu.sg (Frank Stephan)

[1] Lorenzo Carlucci was supported in part by grants "Complexity and compact representability of discrete structures" and "Graphs: Structures, Codes, and Complexity" by University of Rome I.

[2] Sanjay Jain was supported in part by NUS grants R252-000-308-112 and C252-000-087-001. Frank Stephan was supported in part by NUS grants R146-000-114-112 and R252-000-308-112.

1. Introduction

In many learning contexts a learner is confronted with the task of inductively forming hypotheses while being presented with an incoming stream of data. In such contexts the learning process can be said to be successful if, eventually, the hypotheses that the learner forms provide a correct description of the observed stream of data. Each step of the learning process in this scenario involves an observed data item and the formation of a new hypothesis. Each stage of the learning process is completely described by the flow of data seen so far, and the sequence of the learner's hypotheses so far.

It is very reasonable to assume that a real-world learner – be it artificial or human – has memory limitations. A learner with memory limitations is a learner that is unable to store complete information about the previous stages of the learning process. The action of a learner with memory limitations, at each step of the learning process, is completely determined by a *limited portion* of the previous stages of the learning process. Let us call *intensional* memory the learner's memory of its own previously issued hypotheses. Let us call *extensional memory* the learner's memory of previously observed data items.

In the context of Gold's formal theory of language learning [9], models with restrictions on intensional and on extensional memories have been studied. Lange and Zeugmann [16] introduced the paradigm of Bounded Example Memory. A bounded example memory learner is a learner whose intensional memory is, at each step of the learning process, limited to remembering its own previous hypothesis, and whose extensional memory is limited to a finite number of previously observed data items. At each step of the learning process, such a learner must decide, based on the contents of its intensional and extensional memory and on the currently observed data item, whether to change its hypothesis and whether to update the content of its extensional memory. For each non-negative integer k one can define a k-bounded example memory learner as a bounded example memory learner whose memory can never exceed size k. For $k = 0$ one obtains the paradigm of iterative learning [20], in which the learner has no extensional memory and can only remember its own previous conjecture. One of the main results of [16] is the following. For every k, there is a class of languages that can be learned by a $(k + 1)$-bounded example memory learner but not by any k-bounded example memory learner. Subsequent research [5, 14] obtained further results on this and related models.

In this paper we investigate a new model extending the bounded example memory paradigm. In this paradigm the learner is allowed to change its mind, during the learning process, on how many data items to store in memory. This mind-change is modelled by a count-down from a notation for a constructive ordinal.

Ordinals are canonical representatives of well-orderings. For example, the ordinal ω is isomorphic to the standard well-ordering of the natural numbers. The ordinal $\omega + n$ is isomorphic to the standard well-ordering of the natural numbers extended by n linearly ordered points at infinity. The ordinal ω^2 is isomorphic to the lexicographic ordering on pairs of natural numbers. A con-

structive ordinal can be defined as the order-type of a computable well-ordering of the natural numbers. Equivalently, constructive ordinals are those ordinals that have a program (a notation) that specifies how to build them from smaller ordinals using standard operations such as successor and constructive limit. Every constructive ordinal is countable and notations for constructive ordinals are numbers coding programs for building an ordinal from below. For each initial segment of the constructive ordinals a univalent system of notations can be defined. On the other hand, a universal (not univalent) system of notation containing at least one notation for every constructive ordinal has been defined by Kleene [11, 12, 13]. For more details, see, e.g., [18] Chapter 11.

Count-down from ordinal notations has been applied in a number of ways in algorithmic learning theory, starting with [7], where ordinal notations are used to bound the number of mind-changes that a learning machine is allowed to make on its way to convergence. We next describe how count-down from ordinal notations is used in the present paper.

For every constructive ordinal α, and every notation a for it in the fixed system of notation, the paradigm of a-bounded example memory is defined. Intuitively, a learner with example memory bounded by a must (algorithmically) count-down from the notation a for α each time a proper global memory extension occurs during the learning process (i.e., each time the size of the memory set becomes strictly larger than the size of all the previous memory sets). If and when the ordinal counter reaches value 0, no further proper global memory extension is allowed. Yet the learner is still allowed to modify the contents of its memory. To get an intuitive feeling of the new model, let us consider a few informal examples. A learner with ω-bounded memory can see an arbitrarily many input data before deciding to store an element in its example memory. When such a learner decides to make its first memory extension, the learner has to count down from the ordinal ω. That is, the learner declares a *finite* number n as the new bound on the number of *further* extensions of its memory size. From then on, such a learner behaves as a bounded example memory learner with memory bound $n + 1$. This is different from having a *fixed*, *finite* bound on the size of the example memory. In the new model, for each learning task, the learner can algorithmically compute a suitable bound on how many times a memory extension will be needed. The bound can be different from one task to another. Let us now consider a learner with memory bound ω^2. For the sake of this preliminary discussion we can think of the counter as starting at (ω, ω) and the count-down as respecting the lexicographical order on natural numbers extended by ω as a point at infinity. When the learner chooses to save a first data item in its example memory the first coordinate of the counter is decreased to a finite number, be it m. When the learner chooses to extend its example memory for the second time a finite bound for the second coordinate of the counter has to be declared, be it n_1. From this point on the learner can extend its memory n_1 times. The counter is then set to $(m - 1, \omega)$. When the next memory extension occurs, the learner can declare a new finite number n_2 and has to set its counter to $(m - 1, n_2)$. To sum up, a learner with a bound of ω^2 on its example memory can extend the size of its example memory by an

arbitrary finite amount an arbitrary finite number of times (where these finite numbers are in some sense algorithmically determined from input data). Note however that, as we will show, the concept of learning with a memory bound given by a notation for an ordinal larger than or equal to ω^2 depends on the choice of notation for that ordinal.

We show that, for every transfinite ordinal α, the new paradigm is strictly stronger than k-bounded example memory for every non-negative integer k, but strictly weaker than finitely-bounded example memory (with no form of *a priori* bound on memory size). This holds regardlessly of the notation chosen for α. The main result of the present paper is that the concept of ordinal-bounded example memory learning gives rise to a ramified hierarchy of learning criteria along paths in $\mathcal{O}$.

A natural concept that emerged as a by-product of this investigation is the following. Consider a learner that never discards an element from its example memory except when the element is replaced by a new one. We call such a learner a learner with cumulative memory. It is natural to ask whether imposing a cumulative memory policy does restrict learning power. We give a partial answer to this question, showing that learning with cumulative memory does restrict learning power for bounded example memory learners with memory size bounded by a notation for an ordinal of the form $\alpha + 2$, for some ordinal α. Consistently with the information-theoretic flavour of the problem, the proofs of these results make use of notions from Kolmogorov Complexity [17].

2. Preliminaries

This section contains notation for the rest of the paper, and a short presentation of Kleene's ordinal notation system $\mathcal{O}$. These technical tools are then used to formally define the criteria of learning with ordinal-bounded example memory.

2.1. Notation and terminology

Unexplained notation follows Rogers [18]. Let $\mathbf{N}$ denote the set of natural numbers $\{0, 1, 2, \ldots\}$ and let $\mathbf{N}^+$ denote the set of positive natural numbers. The set of finite subsets of $\mathbf{N}$ is denoted by $Fin(\mathbf{N})$. We use the following set-theoretic notations: $\emptyset$ (empty set), $\subseteq$ (subset), $\subset$ (proper subset), $\supseteq$ (superset), $\supset$ (proper superset). If X and Y are sets, then $X \cup Y$, $X \cap Y$, and $X - Y$ denote the union, the intersection, and the difference of X and Y, respectively. We use $Z = X \dot{\cup} Y$ to abbreviate $(Z = X \cup Y \wedge X \cap Y = \emptyset)$. The cardinality of a set X is denoted by $\operatorname{card} X$. By $\operatorname{card} X \leq *$ we indicate that the cardinality of X is finite. We let $\lambda x, y.\langle x, y\rangle$ stand for a standard computable and 1–1 pairing function. We extend the notation to coding of n-tuples of numbers in a straightforward way. We denote with π_i^n $(1 \leq i \leq n)$ the projection function of an n-tuple to its i-th component. We omit the superscript when it is clear from the context.

We fix an acceptable programming system $\varphi_0, \varphi_1, \ldots$ for the partial computable functions of type $\mathbf{N} \to \mathbf{N}$. We denote by W_i the domain of the i-th

partial computable function φ_i. We could equivalently (modulo isomorphism of numberings) define W_i as the set generated by grammar i. Every subset $L \subseteq \mathbf{N}$ is called a language. We are only interested in recursively enumerable languages, whose collection we denote by $\mathcal{E}$. The symbol L ranges over elements in $\mathcal{E}$. The symbol $\mathcal{L}$ ranges over subsets of $\mathcal{E}$, called language classes. Let $\lambda x, y.\mathsf{pad}(x, y)$ be an injective padding function (i.e., $W_{\mathsf{pad}(x,y)} = W_x$).

A finite sequence is a mapping from an initial segment of $\mathbf{N}^+$ into $\mathbf{N} \cup \{\#\}$, where $\#$ is a reserved symbol which we call pause symbol. We use $\mathbf{N}^\#$ to abbreviate $\mathbf{N} \cup \{\#\}$. The symbols σ, τ range over finite sequences. The range of σ minus the $\#$ symbol is denoted by content(σ). The length of σ is denoted by $|\sigma|$.

A text is a mapping from $\mathbf{N}^+$ into $\mathbf{N}^\#$. The symbol t ranges over texts. If $t = (x_i)_{i \in \mathbf{N}^+}$ is a text, $t[n]$ denotes the initial segment of t of length n, i.e., the sequence $(x_1 x_2 \dots x_n)$. We use $\cdot$ for concatenation. If the range of t minus the $\#$ symbol is equal to L, then we say that t is a text for L. If content(σ) $\subseteq L$, then we say that σ is in L.

A language learning machine is a partial computable function mapping finite sequences to natural numbers.

2.2. Constructive Ordinals and Kleene's $\mathcal{O}$

We proceed informally (for a detailed treatment see [18]). A *system of notation* S is a collection of programs (*S-notations*) each of which specifies a structured algorithmic description of some ordinal. In other words, the notations are programs for building, or laying down end-to-end, the denoted ordinal. An ordinal is called *constructive* when it has a notation in some system of notation.

A *system of notation* S consists of a subset N_S of $\mathbf{N}$ (the set of S-notations), and a mapping $S[\cdot]$ from N_S to an initial segment of the ordinals, such that:

- For $x \in N_S$, the properties of being a notation for 0, a notation for a successor ordinal and a notation for a limit ordinal are recursively decidable.
- There is a partial computable function pred (the *predecessor* function) such that, if $x \in N_S$ is a notation for a successor ordinal, then $\mathsf{pred}(x)$ is defined and is a notation for the immediate predecessor of the ordinal $S[x]$ denoted by x, that is, $S[\mathsf{pred}(x)] + 1 = S[x]$.
- There is a partial computable function $\mathsf{fundseq}$ (the *fundamental sequence* function) such that for every notation $x \in N_S$ for a limit ordinal, $\lambda y.\mathsf{fundseq}(x, y)$ is total and $S[\mathsf{fundseq}(x, 0)] < S[\mathsf{fundseq}(x, 1)] < \dots$ has limit $S[x]$, where $<$ is the natural well-ordering of the ordinals.

A system S of ordinal notation is *acceptable* if any other system of ordinal notation is recursively order-preservingly embeddable in it: for any other system of notation S', there exists a partial computable ψ such that, for all S'-notation x, $\psi(x)$ is defined and is an S-notation, and $S'[x] \leq S[\psi(x)]$, and $S'[x] \leq S'[y]$ implies $S[\psi(x)] \leq S[\psi(y)]$. Each acceptable system of notation assigns at least one notation to every constructive ordinal. A system of notation S is *univalent*

if $S[\cdot]$ is injective. It is known that every acceptable system *fails* to be univalent (see [18]).

Kleene [11, 12, 13] developed a general acceptable system of notation $\mathcal{O}$. The system is endowed with a partial relation $<_{\mathcal{O}}$ on notations that naturally embeds in the ordering of the corresponding constructive ordinals: for all $\mathcal{O}$-notations u, v, if $u <_{\mathcal{O}} v$ then $\mathcal{O}[u] < \mathcal{O}[v]$.

We will not need much of the particular features of $\mathcal{O}$ in what follows, but it is nonetheless necessary to refer to *some* system of notations to rigorously define the criteria of ordinal-bounded example memory learning. The use of $\mathcal{O}$ is advantageous in that it provides a single system containing at least one notation for each constructive ordinal. This will allow us to state our general Hierarchy Theorem with satisfactory generality and uniformity.

In Kleene's system $\mathcal{O}$, 2^0 is (by definition) the *notation* for the *ordinal* 0. If u is a notation for the immediate predecessor of a successor ordinal, then a notation for that successor ordinal is (by definition) 2^u. We omit the details of the definition of $<_{\mathcal{O}}$ (see, e.g., [18]). Suppose $\varphi_p(0), \varphi_p(1), \varphi_p(2), \ldots$ are notations in increasing $<_{\mathcal{O}}$ order. Suppose, then, that the corresponding ordinals are longer and longer initial segments of some limit ordinal which is their supremum. For example, some such p generates the respective notations for $0, 1, 2, \ldots$ in increasing $<_{\mathcal{O}}$ order, and ω is the supremum of this sequence. In general, then, p essentially describes how to build the limit ordinal which is the supremum of the ordinals with notations $\varphi_p(0), \varphi_p(1), \varphi_p(2), \ldots$. A notation for this limit ordinal is (by definition) $3 \cdot 5^p$. A sequence of notations for larger and larger ordinals is also called a *fundamental sequence* for their limit. The assignment of fundamental sequences to limit ordinals is an essential ingredient of an ordinal notation system. Clearly limit ordinals have infinitely many notations, different ones for different generating p's. Nothing else is a notation. We define '$x =_{\mathcal{O}} y$' to mean '$x, y \in \mathcal{O}$ and $x = y$'. As in the literature on constructive ordinals, we use '$x \leq_{\mathcal{O}} y$' for '$x <_{\mathcal{O}} y \vee x =_{\mathcal{O}} y$', '$x \geq_{\mathcal{O}} y$' to mean '$y \leq_{\mathcal{O}} x$' and '$x >_{\mathcal{O}} y$' to mean '$y <_{\mathcal{O}} x$'. We also recall that the mapping $\mathcal{O}[\cdot]$ from $\mathcal{O}$ to the set of (constructive) ordinals, is defined as follows [11, 13, 18]:

$$\mathcal{O}[1] = 0; \quad \mathcal{O}[2^u] = \mathcal{O}[u] + 1; \quad \mathcal{O}[3 \cdot 5^p] = \lim_{n \to \infty} \mathcal{O}[\varphi_p(n)].$$

For all $x, y \in \mathcal{O}$, it is true that, if $x <_{\mathcal{O}} y$ then $\mathcal{O}[x] < \mathcal{O}[y]$. It is also true that, for all $y \in \mathcal{O}$, if $\mathcal{O}[y] = \beta$, then for every $\alpha < \beta$, there is an x such that $x <_{\mathcal{O}} y$ and $\mathcal{O}[x] = \alpha$. If $u \in \mathcal{O}$ and $\mathcal{O}[u] = \alpha$, then we say that u is *for* α.

We shall use the following properties of $<_{\mathcal{O}}$ in later proofs.

Lemma 1 (Some properties of $\mathcal{O}$, [18]).

1. *For every $n \in \mathbf{N}$ there exists a unique $\mathcal{O}$-notation for n. This notation will be denoted by $\underline{n}$.*

2. *For every $v \in \mathcal{O}$, $\{u : u <_{\mathcal{O}} v\}$ is a univalent system of notations for the corresponding initial segment of the ordinals.*

3. *There exists an r.e. set Z such that $\{u : u <_{\mathcal{O}} v\} = \{u : \langle u, v\rangle \in Z\}$, for each $v \in \mathcal{O}$.*

4. *There exists a computable mapping $+_{\mathcal{O}} : \mathbf{N} \times \mathbf{N} \longrightarrow \mathbf{N}$ such that, for every $u, v \in \mathcal{O}$,* (i) *$u +_{\mathcal{O}} v \in \mathcal{O}$,* (ii) *$\mathcal{O}[u +_{\mathcal{O}} v] = \mathcal{O}[u] + \mathcal{O}[v]$, and* (iii) *if $v \neq \underline{0}$ then $u <_{\mathcal{O}} u +_{\mathcal{O}} v$.*

In the rest of this paper a, b, u, v, w denote elements in $\mathcal{O}$.

3. Ordinal Bounded Example Memory Learning

We define the learning criteria that are relevant for the present paper. We first define the fundamental paradigm of explanatory identification from text [8].

Definition 2 (Explanatory Learning, Gold [8]). *Let $\mathbf{M}$ be a language learning machine, let L be a language, and let $\mathcal{L}$ be a language class.*

1. **M TxtEx**-*identifies L if and only if, for every text $t = (x_i)_{i \in \mathbf{N}^+}$ for L, there exists an $n \in \mathbf{N}^+$ such that $W_{\mathbf{M}(t[n])} = L$ and, for all $n' \geq n$, $\mathbf{M}(t[n']) = \mathbf{M}(t[n])$.*
2. **M TxtEx**-*identifies $\mathcal{L}$ if and only if* **M TxtEx**-*identifies L for all $L \in \mathcal{L}$.*
3. $\mathbf{TxtEx}(\mathbf{M}) = \{L : \mathbf{M}$ **TxtEx**-*identifies* $L\}$.
4. $\mathbf{TxtEx} = \{\mathcal{L} : (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})]\}$.

We next define the paradigm of iterative learning. This is the basic paradigm of incremental learning upon which the paradigms of bounded example memory learning are built.

Definition 3 (Iterative Learning, Wiehagen [20]). *Let $M : \mathbf{N} \times \mathbf{N}^{\#} \to \mathbf{N}$ be a partial computable function, let $j_0 \in \mathbf{N}$, let L be a language.*

1. *(M, j_0)* **TxtIt**-*identifies L if and only if, for each text $t = (x_i)_{i \in \mathbf{N}^+}$ for L, the following hold:*
   i. *For each $n \in \mathbf{N}$, $M_n(t)$ is defined, where $M_0(t) = j_0$ and $M_{n+1}(t) = M(M_n(t), x_{n+1}) = j_{n+1}$.*
   ii. $(\exists n \in \mathbf{N})[W_{j_n} = L \wedge (\forall n' \geq n)[j_n = j_{n'}]]$.
2. *We say that (M, j_0)* **TxtIt**-*identifies $\mathcal{L}$ if and only if, (M, j_0)* **TxtIt**-*identifies each $L \in \mathcal{L}$.*
3. *For M, j_0 as above,* $\mathbf{TxtIt}(M, j_0) = \{L : (M, j_0)$ **TxtIt**-*identifies* $L\}$.
4. $\mathbf{TxtIt} = \{\mathcal{L} : (\exists M, j_0)[\mathcal{L} \subseteq \mathbf{TxtIt}(M, j_0)]\}$.

The parameter j_0 in the above definition represents an initial conjecture of the learner. Such an initial conjecture is used for technical convenience to ensure uniformity in the recursive equations defining the behaviour of machine M. It is possible to dispense with it at the cost of using a slightly less uniform but equivalent definition.

The learning criterion **TxtIt** is known to be strictly contained in **TxtEx** [15]. We observe that a function M as in the previous definition can be used

to define a language learning machine $\mathbf{M}$ as follows: $\mathbf{M}(t[0]) = M_0(t) = j_0$ and, for all $n \in \mathbf{N}$, $\mathbf{M}(t[n+1]) = M(\mathbf{M}(t[n]), x_{n+1})$. Note that $\mathbf{M}$ is uniquely determined by M and j_0. For every L such that (M, j_0) **TxtIt**-identifies L, we also say that $\mathbf{M}$ **TxtIt**-identifies L.

The paradigm of Bounded Example Memory was introduced in [16] and further investigated in [5] and in the recent [14]. For $k \in \mathbf{N}^+$, a k-bounded example memory learner is an iterative learner that is allowed to store at most k data items chosen from the input text. At the same time we define the paradigm of $*$-bounded example memory learning, where one only requires that the learner's memory be finite.

Definition 4 (Bounded Example Memory Learning, Lange & Zeugmann [16]). *Let $s \in \mathbf{N}^+ \cup \{*\}$.*

1. *Let $M : (\mathbf{N} \times Fin(\mathbf{N})) \times \mathbf{N}^\# \to \mathbf{N} \times Fin(\mathbf{N})$ be a partial computable function, let $j_0 \in \mathbf{N}$, let L be a language. (M, j_0) $\mathbf{Bem}_s$-identifies L if and only if, for each text $t = (x_i)_{i \in \mathbf{N}^+}$ for L, the following hold:*
   i. *For each $n \in \mathbf{N}$, $M_n(t)$ is defined, where $M_0(t) = (j_0, \emptyset)$ and $M_{n+1}(t) = M(M_n(t), x_{n+1}) = (j_{n+1}, S_{n+1})$;*
   ii. *$S_0 = \emptyset$ and $\forall n \in \mathbf{N}$, $S_{n+1} \subseteq S_n \cup \{x_{n+1}\}$;*
   iii. *$\forall n \in \mathbf{N}$, $\operatorname{card} S_{n+1} \leq s$;*
   iv. *$(\exists n \in \mathbf{N})[W_{j_n} = L \wedge (\forall n' \geq n)[j_n = j_{n'}]]$.*
2. *We say that (M, j_0) $\mathbf{Bem}_s$-identifies $\mathcal{L}$ if and only if (M, j_0) $\mathbf{Bem}_s$-identifies L for every $L \in \mathcal{L}$.*

A machine M of the appropriate type that satisfies points *i* through *iii* above, for some j_0, for all texts t is referred to as a $\mathbf{Bem}_s$-learner. By [10], $\mathbf{Bem}_*$ is known to coincide with set-driven learning [19]. With a slight abuse of notation we sometimes use $\mathbf{Bem}_0$ to denote **TxtIt**.

We now introduce an extension of the Bounded Example Memory model. The learner uses an ordinal to bound the number of times the example memory is properly extended. In fact, the bound is enforced by algorithmic counting-down from an ordinal notation each time the extensional memory is properly extended.

Definition 5 (Ordinal Bounded Example Memory Learning). *Let $a \in \mathcal{O}$. Let $M : (\mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}) \times \mathbf{N}^\# \to \mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}$ be a partial computable function. Let $j_0 \in \mathbf{N}$, let L be a language.*

1. *We say that (M, j_0) $\mathbf{OBem}_a$-identifies L if and only if for every text $t = (x_j)_{j \in \mathbf{N}^+}$ for L, points (i) to (v) below hold.*
   i. *for all $n \in \mathbf{N}$, $M_n(t)$ is defined, where $M_0(t) = (j_0, S_0, a_0)$, $S_0 = \emptyset$, $a = a_0$, $M_{n+1}(t) = M(M_n(t), x_{n+1}) = (j_{n+1}, S_{n+1}, a_{n+1})$.*
   ii. *$S_{n+1} \subseteq S_n \cup \{x_{n+1}\}$.*
   iii. *$a_n \geq_{\mathcal{O}} a_{n+1}$.*
   iv. *$a_n >_{\mathcal{O}} a_{n+1}$ if and only if $\operatorname{card} S_{n+1} > \max(\{\operatorname{card} S_i : i \leq n\})$.*
   v. *$(\exists n)(\forall n' \geq n)[j_{n'} = j_n \wedge W_{j_n} = L]$.*

2. *We say that* (M, j_0) $\mathbf{OBem}_a$*-identifies* $\mathcal{L}$ *if and only if* (M, j_0) $\mathbf{OBem}_a$*-identifies* L *for every* $L \in \mathcal{L}$.

A machine of the appropriate type that satisfies points i through iv above for some j_0 and for all texts is referred to as a $\mathbf{OBem}_a$-learner. $\mathbf{OBem}_a$-learning is a species of incremental learning: each new hypothesis depends only on the previous hypothesis, the current memory, and the current data item.

For ease of notation, we often just refer to learning by a learner M; j_0 as in the above definitions is then implicit. Furthemore, note that, for given **Bem** or $\mathbf{OBem}_a$ learner M (with corresponding j_0), $M_n(t)$ depends only on $t[n]$. Thus, for $n \leq |\sigma|$, we define $M_n(\sigma)$ as $M_n(t)$, for any text t such that $\sigma = t[|\sigma|]$.

Remark 6 (Cumulative Learners). *The above definition could be simplified in case the following were true: Call a bounded example memory learner a learner with cumulative memory if the learner never discards an element from memory without replacing it with a new one. For learners with cumulative memory (also called cumulative learners for brevity) it is sufficient to require in point (iv) of Definition 5 that* $\operatorname{card} S_{n+1} > \operatorname{card} S_n$*. Interestingly, Theorem 18 below shows that there are* $\mathbf{Bem}_2$*-learnable classes which cannot be* $\mathbf{Bem}_2$*-learned by a cumulative learner. The proof strategy generalizes to the case of* $\mathbf{OBem}_{\alpha+2}$*-learners, for any* $\alpha \in \mathcal{O}$.

For $\mathbf{I} \in \{\mathbf{Bem}_k, \mathbf{Bem}_*, \mathbf{OBem}_a\}$, M of the appropriate type and $j_0 \in \mathbf{N}$,

- we write $\mathbf{I}(M, j_0)$ for $\{L : (M, j_0)\, \mathbf{I}\text{-identifies } L\}$ and
- we write $\mathbf{I}$ for $\{\mathcal{L} : (\exists M, j_0)[\mathcal{L} \subseteq \mathbf{I}(M, j_0)]\}$.

We write $M(t)$ to indicate the conjecture to which M converges while processing text t (where j_0 as in the above definitions is implicit). We always assume that such a conjecture exists when we use this notation. We state some basic facts in the following Lemma.

Lemma 7. *For all* $k \in \mathbf{N}^+$*, for all* $a, b \in \mathcal{O}$*, the following hold:*

1. $\mathbf{OBem}_{\underline{k}} = \mathbf{Bem}_k$.
2. *If* $a <_{\mathcal{O}} b$*, then* $\mathbf{OBem}_a \subseteq \mathbf{OBem}_b$.
3. $\mathbf{OBem}_a \subseteq \mathbf{Bem}_*$.

Proof. Item 2. can easily be shown: A $\mathbf{OBem}_b$-learner just changes the ordinal notation a used by the $\mathbf{OBem}_a$-learner to b.

For item 1. and 3., one only has to take care of the rather technical point that the type of a machine that $\mathbf{Bem}_k$-identifies is different from the type of a machine that $\mathbf{OBem}_{\underline{k}}$-identifies a language. We observe the following. First, it is easy to realize that $\mathbf{Bem}_k \subseteq \mathbf{OBem}_{\underline{k}}$: if (M, j_0) witnesses $\mathcal{L} \in \mathbf{Bem}_k$, then one can define $\widehat{M}$ witnessing $\mathcal{L} \in \mathbf{OBem}_{\underline{k}}$ as follows. Suppose $t = (x_i)_{i \in \mathbf{N}^+}$. $\widehat{M}_0(t) = (\mathsf{pad}(j_0, b_0), S_0, a_0)$, $\widehat{M}_{n+1}(t) = (\mathsf{pad}(j_{n+1}, b_{n+1}), S_{n+1}, a_{n+1})$, where

$b_0 = 0$, $a_0 = \underline{k}$, $S_0 = \emptyset$, S_{n+1}, j_{n+1} are such that, $M(M_n(t), x_{n+1}) = (j_{n+1}, S_{n+1})$ and b_{n+1}, a_{n+1} are defined as follows:

$$b_{n+1} = \begin{cases} b_n + 1 & \text{if } b_n < |S_{n+1}|, \\ b_n & \text{otherwise.} \end{cases}$$

and

$$a_{n+1} = \begin{cases} 0 & \text{if } a_n = \underline{0}, \\ \mathsf{pred}(a_n) & \text{if } a_n >_{\mathcal{O}} \underline{0} \text{ and } b_n < b_{n+1}, \\ a_n & \text{otherwise.} \end{cases}$$

Essentially b_n records the *maximum* cardinality of a memory set of M after $(x_1, \ldots, x_n)$ has been processed. The ordinal counter of $\widehat{M}$ is decreased by 1 when and only when b_n increases. Since $\mathcal{L} \subseteq \mathbf{Bem}_k(M, j_0)$, then, for every t, for every $L \in \mathcal{L}$, $\lim_{n\to\infty} b_n = m$ for some m such that $m \leq k$. Thus a_n is well-defined, $\widehat{M}$'s conjectures $\widehat{j}_n$ stabilize to $\mathsf{pad}(j^*, b^*)$, where j^* is M's final conjecture on t and $b^* = m$. Thus $(\widehat{M}, \mathsf{pad}(j_0, b_0))$ witness $\mathcal{L} \in \mathbf{OBem}_{\underline{k}}$. $\widehat{M}$'s behaviour is pictured as follows, where we use $\mapsto_M$ (resp. $\mapsto_{\widehat{M}}$) informally to indicate the behaviour of M on the input and the corresponding simulation done by $\widehat{M}$:

$$x_{n+1} \mapsto_M (j_{n+1}, S_{n+1}) \mapsto_{\widehat{M}} (\mathsf{pad}(j_{n+1}, b_{n+1}), S_{n+1}, a_{n+1}).$$

To see that $\mathbf{OBem}_{\underline{k}} \subseteq \mathbf{Bem}_k$ and, by the same token, that 3. holds, observe the following. Suppose $\mathcal{L} \subseteq \mathbf{OBem}_a(M, j_0)$. Then for all $L \in \mathcal{L}$, for all t for L, the ordinal counter of M eventually stabilizes while M processes t. Thus one can define a $\mathbf{Bem}_*$-learner $\widehat{M}$ for $\mathcal{L}$ by coding the ordinal counter of M in the previous conjecture as follows. Suppose $M_0(t) = (j_0, S_0, a_0)$. Then, $\widehat{M}_0(t) = (\mathsf{pad}(j_0, a_0), S_0)$. Furthemore, if $M(j_n, S_n, a_n) = (j_{n+1}, S_{n+1}, a_{n+1})$, then $\widehat{M}_{n+1}(t) = (\mathsf{pad}(j_{n+1}, a_{n+1}), S_{n+1})$. It is easy to see that if $\mathcal{O}[a] = k$, then $\widehat{M}$ is a $\mathbf{Bem}_k$-learner for $\mathcal{L}$. $\widehat{M}$'s behaviour is pictured as follows:

$$x_{n+1} \mapsto_M (j_{n+1}, S_{n+1}, a_{n+1}) \mapsto_{\widehat{M}} (\mathsf{pad}(j_{n+1}, a_{n+1}), S_{n+1}).$$

This completes the proof. ∎

Similar relations as those expressed in the above Lemma also hold (with analogous proof) for the model of Temporary Bounded Example Memory as defined in [14] when the definition is extended to ordinals in the straightforward way.

We state a basic locking sequence lemma for $\mathbf{OBem}_a$-learning. Let (M, j_0) be an $\mathbf{OBem}_a$-learner. A finite sequence σ is a *locking-sequence of the first type for* (M, j_0) on L if and only if (1) content$(\sigma) \subseteq L$, (2) for every extension σ' of σ with content$(\sigma') \subseteq L$, if $M_{|\sigma|}(\sigma) = (j, S, b)$ and $M_{|\sigma'|}(\sigma') = (j', S', b')$, then $j = j'$, and (3) $W_j = L$, where $W_{M_{|\sigma|}(\sigma)} = (j, S, b)$ for some S, b. A finite sequence σ is a *locking-sequence of the second type for* (M, j_0) on L if and only if (a) σ is a locking sequence of the first type for (M, j_0) on L, and (b) for every extension σ' of σ with content$(\sigma') \subseteq L$, if $M_{|\sigma|}(\sigma) = (j, S, b)$ and $M_{|\sigma'|}(\sigma') = (j', S', b')$, then $b = b'$.

Lemma 8 (Locking Sequence Lemma). *Let $a \in \mathcal{O}$. Let $L \in \mathbf{OBem}_a(M, j_0)$, for some M and j_0. There exists a locking sequence of the second type for M on L.*

Proof. That M admits a locking sequence of the first type can be proven using the standard locking sequence argument [3, 9]. Suppose that for every locking sequence of the first type σ for M on L, there exists an extension σ' of σ with content$(\sigma') \subseteq L$ such that $M_{|\sigma'|}(\sigma') = (j, S, b)$, $M_{|\sigma|}(\sigma) = (j', S', b')$, where $b \neq b'$. Then σ' is also a locking sequence for M on L and one can iterate the argument. Then there exists an infinite strictly $<_{\mathcal{O}}$-decreasing sequence of distinct ordinal notations taken as values by M's counter on a text for L. But this is impossible, since the ordinals are well-ordered. ∎

4. Learning with ω-Bounded Example Memory

We prove that learners with w-bounded example memory, with w any notation for ω, can learn strictly more than learners with bounded finite memory. Still, $\mathbf{OBem}_w$ is strictly weaker than $\mathbf{Bem}_*$. The latter fact is actually true for all $\mathbf{OBem}_a$, $a \in \mathcal{O}$.

We start by recalling the definitions of the classes used in [16] to show that

$$\mathbf{TxtIt} \subset \mathbf{Bem}_1 \subset \mathbf{Bem}_2 \subset \ldots$$

We actually give a k-tagged version of Lange and Zeugmann's class $\mathcal{L}_k$, where the parameter k can be read off from the input. For $p \in \mathbf{N}$ we use $\{p\}^+$ to denote the set $\{p, p^2, p^3, \ldots\}$. Let $p_0, p_1, \ldots$ be an enumeration of the prime numbers in increasing order. We denote the set $\{p_i, p_i^2, p_i^3, \ldots\}$ by $\{p_i\}^+$.

Definition 9 (Class $\mathcal{C}_k$). *Let $k \in \mathbf{N}^+$. $\mathcal{C}_k$ is the class consisting of the following languages:*

- $L_k = \{p_k\}^+$,
- $L_{(j,\ell_1,\ldots,\ell_k)} = \{p_k^1, \ldots, p_k^j\} \cup \{p_0^j\} \cup \{p_k^{\ell_1}, \ldots, p_k^{\ell_k}\}$ *for all* $j, \ell_1, \ldots, \ell_k \in \mathbf{N}^+$.

One of the main results in [16] is the following Theorem.

Theorem 10 (Lange and Zeugmann, [16]). *For all $k \in \mathbf{N}$,*

$$\mathcal{C}_{k+1} \in (\mathbf{Bem}_{k+1} - \mathbf{Bem}_k).$$

We denote by $\mathcal{C}_{k,\geq d}$, for $d \in \mathbf{N}$, the subclass of $\mathcal{C}_k$ containing L_k and $L_{(j,\ell_1,\ldots,\ell_k)}$ with $j \geq d$. Since the proof of Theorem 10 in [16] is essentially asymptotic, the following proposition holds.

Proposition 11. *For all $d, k \in \mathbf{N}^+$,*

$$\mathcal{C}_{k+1,\geq d} \in (\mathbf{Bem}_{k+1} - \mathbf{Bem}_k).$$

Let $\mathcal{C}_\omega = \bigcup_{k\in\mathbf{N}^+} \mathcal{C}_k$.

Theorem 12. *Let w be a notation for ω. Then the following holds:*

$$\mathcal{C}_\omega \in (\mathbf{OBem}_w - \bigcup_{k\in\mathbf{N}^+} \mathbf{OBem}_{\underline{k}}).$$

Proof. Let $j, k \in \mathbf{N}^+$. For a set $X \subseteq \{p_k\}^+$ such that $\operatorname{card} X \leq k$, we write $L^k_{(j,X)}$ for the set $\{p^1_k, \ldots, p^j_k\} \cup \{p^j_0\} \cup X$.

We first show that $\mathcal{C}_\omega \in \mathbf{OBem}_w$. For this consider the learner M defined as follows:

The parameters used by M are its conjecture $\mathsf{pad}(c_n, \langle A_n, B_n\rangle)$, memory S_n and ordinal counter a_n.

Intuitively, A_n denotes the j, if any, such that p^j_0 (with $j > 0$) has appeared in the first n elements of the input (with $j > 0$). If no such j exists, then A_n is 0. B_n denotes the subscript k, if any, of the first p^e_k (with $k > 0, e > 0$) seen in the input. If there is no such k, then B_n is 0. S_n keeps track of the largest (up to) k elements p^e_k which have appeared in the input (where $k = B_n$). Parameter a starts with ω, goes down to $\underline{k}$ when the first p^e_k $(k > 0)$ is observed, and is then decreased by one each time the cardinality of S_n is increased.

Formally, initially, $M_0(t) = (j_0, S_0, a_0)$, where $A_0 = B_0 = 0$, $S_0 = \emptyset$, $a_0 = w$, c_0 is a grammar for $\emptyset$, and $j_0 = \mathsf{pad}(c_0, \mathsf{pad}(A_0, B_0))$.

Then, $M_{n+1}(t) = (j_{n+1}, S_{n+1}, a_{n+1})$, where

$$A_{n+1} = \begin{cases} j & \text{if } A_n = 0 \text{ and } x_{n+1} = p^j_0, \text{ for some } j > 0, \\ A_n & \text{otherwise.} \end{cases}$$

$$B_{n+1} = \begin{cases} k & \text{if } x_{n+1} = p^e_k, \text{ where } k > 0, e > 0, \text{ and } B_n = 0, \\ B_n & \text{otherwise.} \end{cases}$$

$$S_{n+1} = \begin{cases} max_k(S_n \cup \{x_{n+1}\}) & \text{if } A_{n+1} = k > 0 \text{ and} \\ & x_{n+1} = p^e_k \text{ for some } e > 0; \\ S_n & \text{otherwise.} \end{cases}$$

where $\max_k(X)$ denotes the k maximal elements of X (if X contains less than k elements then $\max_k(X) = X$).

$$a_{n+1} = \begin{cases} \underline{k}, & \text{if } B_n = 0 \text{ and } B_{n+1} = k > 0 \\ \mathsf{pred}(a_n), & \operatorname{card} S_{n+1} > \operatorname{card} S_n \text{ and } B_n > 0 \\ a_n, & \text{otherwise.} \end{cases}$$

Here

- c_{n+1} is a grammar for L_k, if $A_n = 0$ and $B_n = k > 0$.
- c_{n+1} is a grammar for $L^k_{j,S_{n+1}}$, if $A_{n+1} = j > 0$ and $B_{n+1} = k > 0$.
- $j_{n+1} = \mathsf{pad}(c_{n+1}, \langle A_{n+1}, B_{n+1}\rangle)$.

It is easy to verify that the above learner M witnesses $\mathcal{C}_\omega \in \mathbf{OBem}_w$.

We now prove that $\mathcal{C}_\omega \notin \bigcup_{k \in \mathbf{N}^+} \mathbf{OBem}_{\underline{k}}$. Suppose that $\mathcal{C}_\omega \in \mathbf{OBem}_{\underline{k}}$ for some k, as witnessed by (M, j_0). Then $\mathcal{C}_\omega \in \mathbf{Bem}_k$. But $\mathcal{C}_{k+1} \subseteq \mathcal{C}_\omega$, and $\mathcal{C}_{k+1}$ is not $\mathbf{Bem}_k$-identifiable. Contradiction. ∎

With a very minor change the above proof shows that $\mathcal{C}_\omega$ is learnable by an $\mathbf{OBem}_\omega$-learner with temporary memory as defined in [14].

We now observe that ordinal bounded example memory learning does *not* exhaust $\mathbf{Bem}_*$, i.e., set-driven learning.

Theorem 13. *For all $a \in \mathcal{O}$, $(\mathbf{Bem}_* - \mathbf{OBem}_a) \neq \emptyset$.*

Proof. Consider the following class from [16]: For each $j \in \mathbf{N}$, $L_j = \{2\}^+ - \{2^{j+1}\}$, $\mathcal{L}^- = \{L_j : j \in \mathbf{N}\}$. This class is obviously in $\mathbf{Bem}_*$ but it is shown in [16] not to be in $\bigcup_k \mathbf{Bem}_k$.

To show that $\mathcal{L}^- \notin \mathbf{OBem}_a$, we can then argue exactly as in the proof of Theorem 5, Claim 2 in [16]. Suppose otherwise as witnessed by (M, j_0). Let σ be a locking sequence of the second type for M on L_0. Let $M_{|\sigma|}(\sigma) = (j_{|\sigma|}, S_{|\sigma|}, a_{|\sigma|})$. Let $\beta = a_{|\sigma|} \leq_{\mathcal{O}} a$ such that M's ordinal counter is equal to β while M processes $(\sigma \cdot t)(|\sigma| + i)$, $\forall i \in \mathbf{N}$, for every t such that $\sigma \cdot t$ is a text for L_0. Then, on all such extensions of σ in L_0, M does not make any proper global memory extension. Thus, M's memory size on all such extensions is bounded by $b = \max(\{\operatorname{card} S_i \,:\, i \leq |\sigma|\})$, where S_i is the memory of M after having seen the initial segment of σ of length i.

For every non-empty subset D of $\{2\}^+$, let $\mathrm{rf}(D)$ denote the repetition-free enumeration of the elements of D in increasing order of magnitude. Let $m = 2 + \max(\{r : 2^r \in \mathrm{content}(\sigma)\})$. One can consider m as a constant for the following argument. On the one hand we have, for every $n \in \mathbf{N}$,

$$\operatorname{card}\{D \subseteq \{2^m, 2^{m+1}, \ldots, 2^{m+2n}\} : \operatorname{card} D = n\} = \Omega(2^n).$$

On the other hand, the possible values of M's memory after processing $\sigma \cdot \mathrm{rf}(D)$ are bounded by the number of subsets of $\mathrm{content}(\sigma) \cup \{2^m, 2^{m+1}, \ldots, 2^{m+2n}\}$ of cardinality at most b, i.e., by

$$\operatorname{card}\{D \,:\, D \subseteq \mathrm{content}(\sigma) \cup \{2^m, 2^{m+1}, \ldots, 2^{m+2n}\}, \operatorname{card} D \leq b\},$$

which is in $O(n^b)$. Therefore, for n sufficiently large, there must exist two distinct subsets D_1, D_2 of $\{2^m, 2^{m+1}, \ldots, 2^{m+2n}\}$ of size n such that M ends up with the same memory content after processing the extension of σ with the repetition-free enumerations of D_1 and of D_2, i.e., for some j, S, b, $M_{|\sigma|+n}(\sigma \cdot \mathrm{rf}(D_1)) = (j, S, b)$, and $M_{|\sigma|+n}(\sigma \cdot \mathrm{rf}(D_2)) = (j, S, b)$. Let $\tau_1 = \mathrm{rf}(D_1)$ and $\tau_2 = \mathrm{rf}(D_2)$. Note that τ_1 and τ_2 have the same length but different content. Let $w_1 \in \mathrm{content}(\tau_1) - \mathrm{content}(\tau_2)$ and $w_2 \in \mathrm{content}(\tau_2) - \mathrm{content}(\tau_1)$. Let $L = \{2\}^+ - \{w_2\}$ and $L' = \{2\}^+ - \{w_1\}$. Let t^* be a text for $L_0 - \{w_1, w_2\}$.

Then $t_1 = \sigma \cdot \tau_1 \cdot (2^1) \cdot t^*$ and $t_2 = \sigma \cdot \tau_2 \cdot (2^1) \cdot t^*$ are, respectively, texts for L and L', which are different from each other. Nevertheless we have that

$$M_{|\sigma|+n}(t_1) = M_{|\sigma|+n}(t_2),$$

since $t_1[|\sigma| + n] = \sigma \cdot \mathrm{rf}(D_1)$ and $t_2[|\sigma| + n] = \sigma \cdot \mathrm{rf}(D_2)$. Therefore, for every $z \geq 0$,

$$M_{|\sigma|+n+z}(t_1) = M_{|\sigma|+n+z}(t_2),$$

since past the initial portions of length $|\sigma| + n$, texts t_1 and t_2 are identical. Thus, M fails to identify at least one of L and L'. ∎

We prove a technical lemma that will be used in Section 5 below. A slightly more general version appears in [4] but is not needed here. Let $\mathbf{M} : (\mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}) \times \mathbf{N}^{\#} \to \mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}$ be a partial computable function. Let $j_0 \in \mathbf{N}$. Let $a \in \mathcal{O}$. We say that the pair (M, j_0, a) is *well-behaved* on a text $t = (x_i)_{i \in \mathbf{N}^+}$ if and only if (M, j_0) satisfies conditions (i)-(iv) of Definition 5. In other words, (M, j_0) is an $\mathbf{OBem}_a$-learner for t.

Lemma 14 (Extraction Lemma). *Let M, $\mathcal{C}$, σ, j_0, m, $a \geq_{\mathcal{O}} \underline{m}$ be such that the following conditions hold:*

1. *For all $L \in \mathcal{C}$, for all t for L, (M, j_0, a) is well-behaved on $\sigma \cdot t$,*
2. *For all t for $L \in \mathcal{C}$ such that* $\mathrm{content}(\sigma) \subseteq L$, *$W_{M(\sigma \cdot t)} = L$,*
3. *M counts down exactly m times while processing σ, and*
4. *For all $L \in \mathcal{C}$, for all t for L, after processing σ and while further processing $\sigma \cdot t$, M does no count-down of its ordinal.*

Then $\mathcal{C}^\sigma$ defined as $\{L \in \mathcal{C} : \mathrm{content}(\sigma) \subseteq L\}$ is $\mathbf{OBem}_{\underline{m}}$-learnable.

Proof. We define an $\mathbf{OBem}_m$-learner $\widehat{M}$ for $\mathcal{C}^\sigma$. The idea is that $\widehat{M}$ on t for $L \in \mathcal{C}^\sigma$ simulates M on $\sigma \cdot t$. There are a few technicalities to take care of. If we define a map $\widehat{M}$ as follows: $\widehat{M}_0(t) = M_{|\sigma|}(\sigma)$, $\widehat{M}_{n+1}(t) = M_{|\sigma|+n+1}(\sigma \cdot t)$, then we don't necessarily obtain a function satisfying point (ii) in Definition 5. This is so because M's memory after processing σ may be a non-empty subset of $\mathrm{content}(\sigma)$. In general $\pi_2(M_{|\sigma|}(\sigma)) = S_{|\sigma|}$ has cardinality k with $0 \leq k \leq m$. If $k = 0$, then the above definition of $\widehat{M}$ essentially works. In case $k > 0$, we need to be more careful.

We define $\widehat{M}$ as follows. Let $s = |\sigma|$. Then by definition $M_{|\sigma|}(\sigma) = (j_s, S_s, a_s)$, for some $a_s \leq_{\mathcal{O}} a$. Set

$$\widehat{M}_0(t) = (\mathsf{pad}(j_s, \langle S_s, a_s \rangle), \widehat{S}_0, \widehat{a}_0),$$

where $\widehat{S}_0 = \emptyset$ and $\widehat{a}_0 = \underline{m}$. We pad into $\widehat{M}$'s conjecture at $t(n+1)$ a set $R_{n+1} \subseteq S_s$ consisting of the portion of M's memory at $(\sigma \cdot t)(|\sigma| + n + 1)$ that has not already been transferred to $\widehat{M}$'s example memory. As soon as an element of the padded memory appears in t, we transfer it to $\widehat{M}$'s example memory. We want to keep the following invariants, for all $n \in \mathbf{N}$:

(i) $S_{s+n} = R_n \cup \widehat{S}_n$,
(ii) $R_n \cap \widehat{S}_n = \emptyset$,
(iii) $R_{n+1} \subseteq R_n$ and
(iv) $\widehat{j_n} = \mathsf{pad}(j_{s+n}, \langle R_n, a_{s+n}\rangle)$.

This allows us to simulate M's behaviour on $\sigma \cdot t[s+n+1]$. We now define $\widehat{M}$'s behaviour on new input $t(n+1) = x_{n+1}$. We set

$$\begin{aligned}\widehat{M}_{n+1}(t) &= \widehat{M}((\widehat{j}_n, \widehat{S}_n, \widehat{a}_n), x_{n+1}) \\ &= (\widehat{j}_{n+1}, \widehat{S}_{n+1}, \widehat{a}_{n+1}) \\ &= (\mathsf{pad}(j_{s+(n+1)}, \langle R_{n+1}, a_{s+(n+1)}\rangle), \widehat{S}_{n+1}, \widehat{a}_{n+1}),\end{aligned}$$

where R_{n+1}, $\widehat{S}_{n+1}$ and $\widehat{a}_{n+1}$ are defined by induction as follows. We will also prove inductively that the invariants (i)–(iv) above are satisfied.

We first set $R_0 = S_s$. Since (M, j_0, a) is well-behaved on $\sigma \cdot t$, S_{s+n+1} has the following form:

$$S_{s+n+1} = (S_{s+n} \cup \{x_{n+1}\}) - X,$$

where $X \subseteq S_{s+n} \cup \{x_{n+1}\}$. Inductively, by invariant (i),

$$S_{s+n+1} = (S_{s+n} \cup \{x_{n+1}\}) - (A \cup B),$$

where $A \subseteq R_n \cup \{x_{n+1}\}$ and $B \subseteq \widehat{S}_n - \{x_{n+1}\}$ are possibly empty sets. Necessarily $A \cap B = \emptyset$ holds (by invariant (ii)). We define

$$R_{n+1} = R_n - (A \cup \{x_{n+1}\})$$

and

$$\widehat{S}_{n+1} = S_{s+n+1} - R_{n+1}.$$

$\widehat{M}$'s counter $\widehat{a}_{n+1}$ is defined as follows: Suppose $\widehat{a}_n = \underline{c}$. If $\operatorname{card} \widehat{S}_{n+1} > m - c$, then $\widehat{a}_{n+1} = \underline{c-1}$; otherwise $\widehat{a}_{n+1} = \widehat{a}_n = \underline{c}$. As the cardinality of $\widehat{M}$'s memory is at most m, counter starting at $\underline{m}$ is sufficient.

The above invariants (i)–(iv) are clearly satisfied by definition of $\widehat{S}_{n+1}$ and R_{n+1}. For every $n \in \mathbf{N}$, $S_{s+n+1}, R_{n+1}, j_{s+n+1}$ can be computed from R_n, $\widehat{S}_n$, x_{n+1}, j_{s+n} and a_{s+n} by applying M. Thus, $\widehat{M}_{n+1}$ can be computed based on the information available to $\widehat{M}$ when processing $t(n+1) = x_{n+1}$.

Since for every $L \in \mathcal{C}^\sigma$, for every text t for L, $\operatorname{content}(\sigma) \subseteq \operatorname{content}(t)$, R_n is eventually empty. Furthermore, $\lim_{n\to\infty} a_{s+n+1}$ also converges by condition 4 of the present Lemma. Condition 2 of the Lemma then ensures that $\widehat{M}$ stabilizes on a correct conjecture for $L = \operatorname{content}(t)$ since by definition $W_{M(\sigma\cdot t)} = W_{\widehat{M}(t)}$. ∎

5. General Hierarchy Theorem

We prove a general hierarchy result, showing that the learning power of **OBem**-learners increases along paths in $\mathcal{O}$.

Theorem 15 (General Hierarchy Theorem). *For all $a, b \in \mathcal{O}$, if $a <_{\mathcal{O}} b$ then*

$$\mathbf{OBem}_a \subset \mathbf{OBem}_b.$$

Proof. Below a (with or without decorations) denotes an ordinal notation and $<, >, +$ stand for $<_{\mathcal{O}}, >_{\mathcal{O}}, +_{\mathcal{O}}$. For the sake of readability we also blur the distinction between a natural number and its unique notation in $\mathcal{O}$. It will always be clear from the context whether a number or its notation is meant.

Recall that the class $\mathcal{C}_k$ from Definition 9 can be learned using k-memory but not using smaller memory. The class itself is not critical – we can use any "uniform" class which witnesses the finite hierarchy. Let

$$X_{a,D,k,L} = \{\langle a, D, k, x\rangle : x \in L\},$$

where $k \in \mathbf{N}^+$, a is an ordinal notation and D is a (index for a) finite set. Now, for $a_1 > a_2 > \ldots > a_k$, let $\mathcal{S}_{a_1,a_2,\ldots,a_k}$ consist of languages of the form:

$$D \cup X_{a_k,D,k,L},$$

where $L \in \mathcal{C}_k$, and $D \subseteq \{\langle a_i, \cdot, i, \cdot\rangle : (1 \leq i < k)\}$.

Let $\mathbb{S}(a) = \{(a_1, \ldots, a_k) \ : \ a = a_1 > a_2 > \ldots > a_k, k \in \mathbf{N}\}$.

$$\mathcal{U}_a = \bigcup_{(a_1,\ldots,a_k)\in\mathbb{S}(a)} \mathcal{S}_{a_1,a_2,\ldots,a_k}.$$

We first prove $\mathcal{U}_{a+1} \in \mathbf{OBem}_{a+1}$. Let $L = D \cup X_{a_k,D,k,L'} \in \mathcal{U}_{a+1}$, for some $L' \in \mathcal{C}_k$, and let t be for L. It suffices for the learner to learn the part of t consisting of elements of the form $\langle a_i, \cdot, i, \cdot\rangle$, where i is the current maximal third component which has appeared in t so far, as the rest of the language can be derived from it. Thus, it suffices to learn the part $L^* = \{x : \langle a_i, D, i, x\rangle \in L\}$, as D, i, a_i, can be padded up in the hypothesis and an index for $D \cup X_{a_i,D,i,L^*}$ can be computed. Eventually the maximum $\pi_3(e)$ for e appearing in t is k and the learner has to learn $L' = \{x : \langle a_k, D, k, x\rangle \in L\}$.

To do the above M simulates a standard strategy for learning $\mathcal{C}_i$ where i is the largest $\pi_3(e)$ for e which has appeared in t so far. Note that this can be done as $\mathcal{C}_i$ can be learned by a i-memory bounded learner. We just have to make sure that M can extend its memory and update its ordinal counter as needed for this simulation. Let the current input element be $\langle a_j, \cdot, j, \cdot\rangle$, M's current memory size be m, and let *prev* be the currently smallest ordinal notation seen (that is $\pi_1(x)$, for inputs x seen so far) or $a + 1$ if no such ordinal notation has been seen so far. Note that m is the largest memory size up to the current point of the learning process.

We only care about the case $prev > a_j$. Let $d \leq j$ be such that $m = j - d$. Then – as soon as the first memory extension for simulating the standard $\mathcal{C}_j$-learner is needed – M's ordinal counter is decreased from the current value to $a_j + d - 1$. This allows d memory extensions from current to value a_j, so that M can increase its memory to size $m + d = j$, whenever needed for simulation of the $\mathcal{C}_j$–learner.

We have to show that the memory update is correct, i.e., that $a_j + d - 1$ is always strictly smaller than the previous ordinal counter value. The basic observation is that for $1 \leq j < i \leq k$, $a_j \geq a_i + (i - j)$. If $\langle a_j, \cdot, j, \cdot \rangle$ is the first non-trivial element seen, then $m = 0$, $prev = a + 1$, $d = j \geq 1$, and counter is set to $a_j + j - 1 \leq a_1 < a_1 + 1 = a + 1$ as soon as the first memory extension for simulating a $\mathcal{C}_j$–learner is needed. If and when M moves from simulating a $\mathcal{C}_j$–learner to simulating a $\mathcal{C}_i$–learner for $j < i$, M's current counter is $\geq a_j + j - m$ where $m \leq j$ is M's memory size at that moment. We have $a_j \geq a_i + i - j$ by the above observation, and therefore

$$a_j + j - m \geq a_i + i - j + j - m = a_i + i - m > a_i + i - m - 1,$$

and the latter is the next value of M's ordinal counter.

We now prove that $\mathcal{U}_{a+1} \notin \mathbf{OBem}_a$. Suppose by way of contradiction that $\mathcal{U}_{a+1} \in \mathbf{OBem}_a$, as witnessed by (M, j_0). Then (M, j_0) $\mathbf{OBem}_a$-identifies $\mathcal{S}_a$.

Claim: there exists $L \in \mathcal{S}_a$ and t for L such that M's ordinal counter is decreased while processing t.

This follows from Lemma 14, showing that otherwise $\mathcal{C}_1$ would be iteratively learnable. Let L^1 be such a language and t^1 such a text for L^1. Let M's counter be decreased for the first time at $t^1(i_1)$ and let $a_2 < a$ be the new value. Let $D_2 = \text{content}(t^1[i_1])$.

For every $L \in \mathcal{C}_2$, the language $D_2 \cup X_{a_2, D_2, 2, L}$ is in $\mathcal{S}_{a, a_2}$.

Claim: There exists $L_2 \in \mathcal{C}_2$ and t^2 for $D_2 \cup X_{a_2, D_2, 2, L_2}$ such that M's counter is decreased below a_2 while processing $t^1[i_1]t^2$. Otherwise, it easily follows from Lemma 14 that one could extract an $\mathbf{OBem}_1$-learner for the class $\{D_2 \cup X_{a_2, D_2, 2, L} \ : \ L \in \mathcal{C}_2\}$. But the latter class is obviously as hard as $\mathcal{C}_2$. Let L^2 be such a language and t^2 such a text for L^2. Let $i_2 > i_1$ be the first place where M's counter is decreased below a_2 while processing $t^1[i_1]t^2$ beyond $t^1[i_1]$, and let a_3 be the counter's value. Let D_3 be content$(t^1[i_1]t^2[i_2])$. For every $L \in \mathcal{C}_3$, $D_3 \cup X_{a_3, D_3, 3, L}$ is in $\mathcal{S}_{a, a_2, a_3}$. The argument is iterated according to the above pattern. The conditions for applying Lemma 14 are always met.

If a contradiction is not reached earlier, then eventually M's counter hits 0. Let this happen after $(k - 1)$ changes to the ordinal counter for some $k \in \mathbf{N}^+$ (i.e., M's ordinal counter went through the values: $a = a_1$, a_2, …, $a_k = 0$). Then M can make no further memory extension. But, for each $L \in \mathcal{C}_k$ we can extend the initial segment t_* of the text defined so far to a text for $D \cup X_{a_k, D, k, L}$, where $D = \text{content}(t_*)$. We can show as above that M needs to make at least one memory extension beyond t_* on some such text for some such language. But this is not possible. ∎

6. Dependence on notations

We prove that for ordinals below ω^2 the concept of ordinal-bounded example memory is notation-independent. This reflects the fact that $\mathcal{O}$-notations below ω^2 are essentially unique.

Proposition 16 (Notation Independence). *Suppose $a, a' \in \mathcal{O}$ are notations for the same ordinal which is smaller than ω^2. Then*

$$\mathbf{OBem}_a = \mathbf{OBem}_b.$$

Proof. The following holds in general. Let a, b be notations for the same ordinal $\omega \cdot m + n < \omega^2$.

Observe that, if a is for $\omega \cdot m + n$, then for every i with $1 \leq i \leq m$, there exists a unique notation a_i for $\omega \cdot i$ such that $a_i \leq_O a$. This follows from the fact that $\{a' : a' \leq_{\mathcal{O}} a\}$ is a univalent system of notations.

Let $a_1, a_2, \ldots, a_m$ be the notations for $\omega, \omega \cdot 2 \ldots, \omega \cdot m$ below a and $b_1, b_2, \ldots,$ b_m be the notations for $\omega, \omega \cdot 2 \ldots, \omega \cdot m$ below b. Using the map $a_i \mapsto b_i$ one can effectively map any notation below a to a notation below b.

Given an $\mathbf{OBem}_a$-learner, the above translation can be used to define a corresponding $\mathbf{OBem}_b$-learner identifying the language class identified by the $\mathbf{OBem}_a$-learner. ∎

We prove that, by contrast, for notations for ordinals above ω^2, the concepts are notation-dependent. A similar phenomenon occurs in other learning scenarios, e.g., [1, 2], and the following proof very much follows the idea from [1].

Proposition 17 (Notation Dependence). *For any ordinal notation a, there is a class of languages which cannot be learned using ordinal notation a, but can be learned using some ordinal notation for ω^2.*

Proof. We first define a class $\mathcal{U}^*_{a+1}$ and show that it is not in $\mathbf{OBem}_a$. We then define a notation b for ω^2 and show that $\mathcal{U}^*_{a+1}$ is in $\mathbf{OBem}_b$.

1. We fix an enumeration of all learning machines $M_0, M_1, \ldots$ [3]. We use the diagonalization used for showing $(\mathbf{OBem}_{a+1} - \mathbf{OBem}_a) \neq \varnothing$ in Theorem 15 to associate to each $r \in \mathbf{N}$ a limit-computable value $F(r)$ measuring the number of mind-changes done by M_r in the limit, and a corresponding language L_r.

We slightly modify the definition of the class $\mathcal{U}_a$ of Theorem 15 by using

$$X_{r,a,D,k,L} = \{\langle r, a, D, k, x\rangle : x \in L\},$$

instead of $X_{a,D,k,L}$. It is obvious that the addition of the extra parameter r is irrelevant, so that Theorem 15 holds with respect to the modified class. In particular, the diagonalization in the second part of Theorem 15 can be carried out in a strictly analogous way. Let $L_r = D \cup X_{r,a,D,k,L}$ be the language formed by doing the diagonalizing against M_r being $\mathbf{OBem}_a$-learner for $\mathcal{U}_{a+1}$ as in the second part of the proof of Theorem 15. The value k (depending on r) denotes the same quantity as in the latter proof. Thus L above is in $\mathcal{C}_k$. Let $\mathcal{U}^*_{a+1} = \{L_r : r \in \mathbf{N}\}$. Then clearly, $\mathcal{U}^*_{a+1}$ is not in $\mathbf{OBem}_a$ by construction.

[3] To be rigorous, in view of Definition 5 we should here say: Fix an enumeration $M^0, M^1, \ldots$ of all partial computable functions of type $(\mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}) \times \mathbf{N}^{\#} \to \mathbf{N} \times Fin(\mathbf{N}) \times \mathcal{O}$. $M_0, M_1, \ldots$ used in the proof is actually an enumeration of $(M^r, j)_{r \in \mathbf{N}, j \in \mathbf{N}}$.

Note that the diagonalization defines a sequence of increasing ℓ_{i_j}'s for $j = 1, 2, \ldots, k$, where ℓ_{i_j} measures the length of the text processed by M_r in step j of the diagonalization, as well as a corresponding decreasing sequence of a_j's. These sequences can be effectively enumerated given M_r. Thus it is easy to see that there exists a limit-computable function $F(x) = \lim_{t\to\infty} f(x,t)$ for some computable f, such that $F(r) = k$ and such that $F(r)$ converges to k in monotonically increasing fashion, starting with value 1.

2. Note that F as above is such that for each $r \in \mathbf{N}$, given $F(r)$, L_r can be $\mathbf{OBem}_{F(r)}$-identified, by a uniform learning strategy. This is so since, by the diagonalization in Theorem 15, it is easily seen that memory of size $k = F(r)$ is sufficient for learning L_r.

We now define a sequence of notations $b_{r,0}, b_{r,1}, \ldots$, satisfying the following property:

$$\mathcal{O}[b_{r,i}] = \begin{cases} \mathcal{O}[b_{r,i+1}] + \omega & \text{if } b_{r,i+1} \text{ is defined,} \\ \omega & \text{otherwise.} \end{cases}$$

Only finitely many such notations will be defined: more precisely, only $b_{r,0}, b_{r,1}, \ldots b_{r,k}$ will be defined, where $k = F(r)$. We let $b_{r,i}$ be defined if and only if $F(r) \geq i$. Then $b_{r,0}$ will be always defined, because F starts converging from 1. Also, $b_{r,k}$ will always be a notation for ω.

We define $b_{r,i}$ by defining a fundamental sequence of notations and taking $b_{r,i}$ to be their limit. The fundamental sequence is defined as follows. We run the computation of $F(r) = \lim_{t\to\infty} f(r,t)$. After discovering that $F(r) \geq i$, and while observing that $i \leq f(r,t) < i+1$, the fundamental sequence defining $b_{r,i}$ is defined as the standard fundamental sequence for ω, i.e., $\underline{1}, \underline{2}, \underline{3}, \underline{4}, \ldots$. When, if ever, for some t we have $f(r,t) \geq i+1$, we continue the fundamental sequence for $b_{r,i}$ as $\ldots b_{r,i+1} + \underline{1}, b_{r,i+1} + \underline{2}, \ldots$. In other words, whenever $b_{r,i}$ is defined, $b_{r,i}$ is obtained as the limit of the sequence of notations

$$\underline{1}, \underline{2}, \underline{3}, \ldots, \underline{t}, b_{r,i+1} + \underline{1}, b_{r,i+1} + \underline{2}, \ldots,$$

where $t+1$ is minimal such that $f(r,t) \geq i+1$ (if no such $t+1$ exists, then $b_{r,i}$ is just the limit of the sequence of notations $\underline{1}, \underline{2}, \underline{3}, \ldots,$). Thus, $b_{r,i}$ (if defined) is either a notation for ω or a notation for $\mathcal{O}[b_{r+i+1}] + \omega$.

3. Since $F(r)$ converges to some natural number, we immediately have that $b_{r,0}$ is a notation for $\omega \cdot (F(r)+1)$. It is easy to see that L_r can be $\mathbf{OBem}_{b_{r,0}}$-identified. The learner starts with counter at notation $b_{r,0}$ and then uses the ordinal notations $b_{r,1}, b_{r,2}, \ldots$ to decrement the counter, as follows. If and when the j-th mind-change in $F(r)$ (to a value $f(r,s_j)$) is observed, the ordinal value is updated to $b_{r,j} + \underline{f(r,s_j) - c}$, where c is the current size of memory. Note that only notations up through $b_{r,k}$ with $k = F(r)$ will be needed and that all such notations are defined. The learner can adopt a learning strategy similar to that of the learner from the positive part of Theorem 15. In other words, the learner can simulate the standard learning strategy for the class $\mathcal{C}_i$ for the largest parameter i (corresponding to $\pi_4(x)$ for the input datum x) seen so far in the input. At any point of the learning process, the current notation has the

form $b_{r,j}+z$, where z is used for countdown as long as the parameter i does not change. A new count-down is started each time a new maximal i is seen. At worst, this happens $k = F(r)$ times.

4. For every r, let b_r denote $b_{r,0}$ and let c_r be a notation for $b_0 +_{\mathcal{O}} b_1 +_{\mathcal{O}} b_2 +_{\mathcal{O}} \cdots +_{\mathcal{O}} b_r$. Obviously then $c_0 <_{\mathcal{O}} c_1 <_{\mathcal{O}} \ldots$. Let c be a notation for the limit of $\mathcal{O}[c_0], \mathcal{O}[c_1], \ldots$.

We now show that $\mathcal{U}^*_{a+1}$ can be $\mathbf{OBem}_c$-identified. The learner starts with notation c, and when it gets parameter r from the input, changes its notation to c_r. From now on the learner can behave as the $\mathbf{OBem}_{b_r}$-learner for L_r from part 3 above using b_r. Note that c_r is for $b_0 +_{\mathcal{O}} b_1 +_{\mathcal{O}} b_2 +_{\mathcal{O}} \cdots +_{\mathcal{O}} b_r$, so that the $\mathbf{OBem}_{b_r}$-learning strategy from part 3 can be implemented using c_r.

5. Since $\mathcal{O}[b_{r,0}] = \omega \cdot (F(r)+1)$ by point 3 above,

$$\mathcal{O}[c_r] = \omega \cdot \sum_{i=0}^{r} (F(i)+1) = \omega \cdot (F(0)+F(1)+\cdots+F(r)) + \omega \cdot (r+1).$$

Thus, c is a notation for $\omega \times \omega = \omega^2$. ∎

7. Cumulative Learners

Cumulative learners, for many cases, are weaker than the corresponding bounded example memory learners with the same memory bound. Thus, Definition 5 cannot be simplified as described in Remark 6. We give the proof for memory bound $k = 2$. The proof can be modified to work for larger constant $k \in \mathbf{N}^+$, by enlarging the class in Theorem 18 accordingly so that one uses the parameters $a_1, a_2, \ldots, a_k$ instead of a_1, a_2 below. The proof can also be modified to work for any ordinal memory bound of the form $\alpha + 2$. For that, one combines the technique below with the technique used to prove the Hierarchy Theorem 15. The result is consistent with the intuition that a learner that can discard elements from memory without replacing them is using the contents of its example memory to code more than just some of the examples seen. The proof uses notions from Kolmogorov Complexity [17]. In Kolmogorov Complexity one is interested in measuring the descriptive complexity of an object. The Kolmogorov complexity of a binary string relative to the Halting Problem is the length of the shortest program (in a suitable fixed universal Turing machine) that outputs the string using the characteristic function for the Halting Problem as an oracle.

Theorem 18. *There is a* $\mathbf{Bem}_2$*-learnable class which is not learnable by a cumulative* $\mathbf{Bem}_2$*-learner.*

Proof. Assume without loss of generality that the pairing function $\langle \cdot, \cdot \rangle$ is increasing in both its arguments; in particular $\langle 0,0 \rangle = 0$ and $\langle 0,1 \rangle = 1$. Let $\mathcal{L}$ contain the set $\{1, 2, \ldots\}$ and, for each of the following D, the corresponding set L_D, where $w > 1$, $x > 1$, $x > w$ and, in cases (b) below, $a_1 \neq a_2$ and, in case (c) below, $|W_x| = 3$.

(a) $D = \{\langle w, a_1\rangle, \langle x, a_2\rangle\}$
and $L_D = \{0, 1, 2, \ldots, x\} \cup \{\langle x, a_2\rangle\}$,

(b) $D = \{\langle x, a_1\rangle, \langle x, a_2\rangle\}$
and $L_D = \{0, 1, 2, \ldots, x\} \cup \{\langle x, a_1\rangle, \langle x, a_2\rangle\}$,

(c) $D = \{\langle x, a_1\rangle\}$
and $L_D = \{0, 1, 2, \ldots, x\} \cup \{\langle x, b\rangle : b \in W_x\}$.

The learning algorithm M maintains a hypothesis for the current conjecture plus some extra information and uses the bounded example memory to store the currently most likely hypothesis different from $\{1, 2, \ldots\}$, so that this hypothesis can be conjectured whenever a 0 is observed in the input. The algorithm updates the long term memory D and the hypothesis j based on the current datum u as follows, where e_0 is a fixed index for the emptyset.

- Initially $j = e_0$ and $D = \emptyset$.
- If $u = \#$ then no update is done and j, D remain unchanged.
- If $j = e_0$ and $|D \cup \{u\}| \leq 1$ then update $D = D \cup \{u\}$ and let $j = e_0$.
- If $j = e_0$ and $|D \cup \{u\}| = 2$ then update $D = D \cup \{u\}$. Furthermore, if $0 \notin D$ then choose j such that $W_j = \{1, 2, \ldots\}$ else choose j such that $W_j = D$.
- Otherwise, D is updated according to the table below. Furthermore, j is updated so that if 0 has been seen so far then $W_j = L_D$ else $W_j = \{1, 2, \ldots\}$. Here always a fixed index is used for $\{1, 2, \ldots\}$ so that 0 has been seen iff j is different from that fixed index. In the table assume that $v < x$, $w < x$ and $x < y$; furthermore, numbers are split into pairs.

Old value of D	Current Datum u	New Value of D
$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$	$\langle w, a_3\rangle$	$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$
$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$	$\langle x, a_3\rangle$ where $a_2 \neq a_3$	$\{\langle x, a_2\rangle, \langle x, a_3\rangle\}$
$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$	$\langle x, a_3\rangle$ where $a_2 = a_3$	$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$
$\{\langle v, a_1\rangle, \langle x, a_2\rangle\}$	$\langle y, a_3\rangle$	$\{\langle v, a_1\rangle, \langle y, a_3\rangle\}$
$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$ where $a_1 \neq a_2$	$\langle w, a_3\rangle$	$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$
$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$ where $a_1 \neq a_2$	$\langle x, a_3\rangle$ where $a_3 \notin \{a_1, a_2\}$	$\{\langle x, a_1\rangle\}$
$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$ where $a_1 \neq a_2$	$\langle x, a_3\rangle$ where $a_3 \in \{a_1, a_2\}$	$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$
$\{\langle x, a_1\rangle, \langle x, a_2\rangle\}$ where $a_1 \neq a_2$	$\langle y, a_3\rangle$	$\{\langle x, a_1\rangle, \langle y, a_3\rangle\}$
$\{\langle x, a_1\rangle\}$	$\langle w, a_2\rangle$	$\{\langle x, a_1\rangle\}$
$\{\langle x, a_1\rangle\}$	$\langle x, a_2\rangle$	$\{\langle x, a_1\rangle\}$
$\{\langle x, a_1\rangle\}$	$\langle y, a_2\rangle$	$\{\langle x, a_1\rangle, \langle y, a_2\rangle\}$

One can see that the bounded example memory of the learner when learning L_D is D', with $L_D = L_{D'}$, after the three largest members of L_D have been seen. Furthermore, the conjecture is $\{1, 2, \ldots\}$ after at least two elements of this set have been seen and if the input does not contain 0; the conjecture is L_D after the three largest elements of L_D as well as 0 have been seen in the input. This establishes the correctness of the learner M.

Assume now by way of contradiction that a cumulative $\mathbf{Bem}_2$-learner N also learns $\mathcal{L}$. Let σ be a locking sequence (of type 2) of N for the set $\{1, 2, \ldots\}$. Here we assume that the memory of the learner N after having seen input σ is of size 2 (otherwise, the following argument can be appropriately modified). For all sufficiently large x, let τ_x be an extension of σ with the range $\{1, 2, \ldots, x\}$, say $\tau_x = \sigma \cdot (1, 2, 3, \ldots, x)$. As long as N does not see 0 and has processed σ, N only updates the bounded example memory and does not change its hypothesis. Hence when the text is of the form $\tau_x \, \eta \, 0 \, \#^\infty$ then the learner will have to rely exclusively on the bounded example memory in order to extract the information contained in η when it finally sees the 0.

Given x, let $P_x(a_1, a_2, a_3)$ denote the property that holds if and only if for every distinct $b, c \in \{a_1, a_2, a_3\}$ the example memory of N after processing the input $\eta_{x,b,c} = \tau_x \, \langle x, b\rangle \, \langle x, c\rangle$ is equal to $\{\langle x, b\rangle, \langle x, c\rangle\}$. Given x, let $S_x = \{a_1, a_2, a_3\}$ for the first three distinct a_1, a_2, a_3 numbers found which satisfy $P_x(a_1, a_2, a_3)$; if no such numbers are found then S_x is empty. Note that an index for S_x can be obtained effectively from x. Thus, by the Recursion Theorem, there are infinitely many x such that $W_x = S_x$.

Now it is shown that S_x is non-empty when $x > \max(\text{content}(\sigma))$. To get an example of such a_1, a_2, a_3, let k be a sufficiently large integer with $k > x$ and take a_1, a_2, a_3 as three k-bit numbers such that the Kolmogorov complexity of $\langle a_1, a_2, a_3\rangle$ relative to the Halting Problem is at least $3k$. This implies that one cannot code any two numbers $b, c \in \{a_1, a_2, a_3\}$ relative to the Halting Problem with less than $2k$ bits. In particular, N cannot find b, c from processing the remaining portion $0 \, \#^\infty$ of the input text beyond $\eta_{x,b,c}$, and from a bounded example memory of size two where one member has Kolmogorov complexity up to $k + 2\log(k)$ and the other member has only Kolmogorov complexity up to $2\log(k)$, that is, where only one of b, c is remembered and the other entry is a number between 1 and x. Therefore N, when learning from the text $\eta_{x,b,c} \, 0 \, \#^\infty$, has to memorize two different numbers of Kolmogorov complexity at least k and the only candidates are $\langle x, b\rangle$ and $\langle x, c\rangle$. Hence the bounded example memory after processing $\eta_{x,b,c}$ is $\{\langle x, b\rangle, \langle x, c\rangle\}$.

As a consequence there are infinitely many x such that W_x consists of three numbers a_1, a_2, a_3 satisfying $P_x(a_1, a_2, a_3)$. For these x, a_1, a_2, a_3 and the input $\tau_x \, \langle x, a_1\rangle \, \langle x, a_2\rangle \, \langle x, a_3\rangle$, when processing the last item $\langle x, a_3\rangle$, N updates its bounded example memory to $D = \{\langle x, b\rangle, \langle x, c\rangle\} \subseteq \{\langle x, a_1\rangle, \langle x, a_2\rangle, \langle x, a_3\rangle\}$ and it follows from the choice of W_x that N has the same memory D after having seen $\tau_x \, \langle x, b\rangle \, \langle x, c\rangle$. Therefore N converges on the texts $\tau_x \, \langle x, b\rangle \, \langle x, c\rangle \, 0 \, \#^\infty$ and $\tau_x \, \langle x, a_1\rangle \, \langle x, a_2\rangle \, \langle x, a_3\rangle \, 0 \, \#^\infty$ to the same index although the texts are for different languages in $\mathcal{L}$. ∎

8. Conclusion

We have proven a number of results on a proper extension of the Bounded Example Memory model [16]. The extension was first introduced in [4] and features algorithmic count-down from constructive ordinals to bound the number of proper, global memory extensions an incremental learner is allowed on its way to convergence. We have shown that the concept gives rise to criteria that lie strictly between the finite Bounded Example Memory hierarchy $\bigcup_k \mathbf{Bem}_k$ and set-driven learning $\mathbf{Bem}_*$. We have exhibited a hierarchy of learning criteria: if a, b are constructive ordinal notations in Kleene's $\mathcal{O}$ such that $a >_{\mathcal{O}} b$, then a learner that can extend its example memory "a times" can learn strictly more than any learner that can extend its memory only "b times". Below the ordinal ω^2 the result is notation-independent and the involved learning criteria only depend on the order-type. From ω^2 up the notions become notation-dependent.

We wish to stress that the use of constructive ordinals does not necessarily make the concept of **OBem**-learning unrealistic from the point of view of modeling learners with memory limitations. The unrealistic features of this model are already shared by the model with finite memory limitations, in that the time the learner takes to converge to its final hypothesis is not necessarily computable, and the time needed to calculate each of the learner's conjectures is not necessarily feasible. Nevertheless, models of learning in the limit offer an appropriate framework where a number of cognitively-motivated questions can be formulated and answered. Also note that for every constructive ordinal α, there exist feasible systems of notations for the ordinals up to α, essentially meaning that the predecessor and fundamental sequence functions are polynomial-time computable (see [6]).

A possible topic of future investigation is to investigate ordinal versions of feedback learning from [5]; here a feedback learner does not remember the full history of the examples seen so far, but it can make queries on the past, that is, it can query a teacher whether certain data items had been seen before. It would also be interesting to know whether cumulative memory learning does restrict learning power for all ordinal bounds as well as to study the impact of cumulative learning in other memory-limited models.

Acknowledgments. Some of the results from the present paper have been presented at the conference ALT 2009 [4]. We would like to thank the referees of both the conference ALT 2009 and the Journal of Computer and System Sciences for useful comments.

[1] Andris Ambainis. Power of procrastination in inductive inference: How it depends on used ordinal notations. In, P. Vitányi, editor, *Computational Learning Theory, Second European Conference*, EuroCOLT 1995, Proceedings. Lecture Notes in Artificial Intelligence 904, Springer, pages 99–111, 1995.

[2] Andris Ambainis, Sanjay Jain and Arun Sharma. Ordinal mind change complexity of language identification. *Theoretical Computer Science*, 220(2):323–343, 1999.

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

[4] Lorenzo Carlucci. Incremental Learning with Ordinal Bounded Example Memory. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann and Sandra Zilles, editors, *Algorithmic Learning Theory, 20th International Conference*, ALT 2009, Proceedings. Lecture Notes in Artificial Intelligence 5809, Springer, pages 323–337, 2009.

[5] John Case, Sanjay Jain, Steffen Lange and Thomas Zeugmann. Incremental concept learning for bounded data mining. *Information and Computation*, 152(1):74–110, 1999.

[6] John Case, Timo Kötzing and Todd Paddock. Feasible Iteration of Feasible Learning Functionals. In Marcus Hutter, Rocco A. Servedio and Eiji Takimoto, editors, *Algorithmic Learning Theory, 18th International Conference*, ALT 2007, Proceedings. Lecture Notes in Artificial Intelligence 4754, Springer, pages 34–48, 2007.

[7] Rūsiņš Freivalds and Carl H. Smith. On the role of procrastination for Machine Learning. *Information and Computation*, 107(2):237–271, 1993.

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

[9] Sanjay Jain, Daniel Osherson, James Royer and Arun Sharma. Systems that learn: an introduction to learning theory. 2nd edition MIT Press, Cambridge, 1999.

[10] Efim Kinber and Frank Stephan. Language learning from texts: mind-changes, limited memory, and monotonicity. *Information and Computation*, 123(2):224–241, 1995.

[11] Stephen C. Kleene. On notation for ordinal numbers. *Journal of Symbolic Logic*, 3(4):150–155, 1938.

[12] Stephen C. Kleene. On the forms of predicates in the theory of constructive ordinals. *American Journal of Mathematics*, 66(1):41–58, 1944.

[13] Stephen C. Kleene. On the forms of predicates in the theory of constructive ordinals (second paper). *American Journal of Mathematics*, 77(3):405–428, 1955.

[14] Steffen Lange, Samuel Moelius III and Sandra Zilles. Learning with Temporary Memory. In Y. Freund and L. Györfi and G. Turán and T. Zeugmann, editors, *Algorithmic Learning Theory, 19th International Conference*, ALT 2008, Lecture Notes in Artificial Intelligences 5254, Springer, pages 449–463, 2008.

[15] Steffen Lange and Thomas Zeugmann. Types of monotonic language learning and their characterizations. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 377–390, ACM Press, New York, 1992.

[16] Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53(1):88–103, 1996.

[17] Ming Li and Paul B. Vitányi. *An introduction to Kolmogorov complexity and its applications*, (2. ed.). Springer 1997.

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

[19] Kenneth Wexler and Peter W. Culicover. *Formal principles of language acquisition*, MIT Press, Cambridge, 1980.

[20] Rolf Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik*, 12(1/2):93–99, 1976.