Applications of Arithmetic Complexity and Priority Arguments in Algorithmic Learning Theory

By

Achilles Athanasios Beros

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Mathematics)

at the

UNIVERSITY OF WISCONSIN – MADISON

2013

Date of final oral examination: May 13, 2013

Abstract

We consider the arithmetic complexity of index sets of uniformly computably enumerable families learnable under different learning criteria. We determine the exact complexity of these sets for the standard notions of finite learning, learning in the limit, behaviorally correct learning and anomalous learning in the limit. In proving the Σ^0_5-completeness result for behaviorally correct learning we prove a result of independent interest; if a uniformly computably enumerable family is not learnable, then for any computable learner there is a Δ^0_2 enumeration witnessing failure.

Using related techniques, we show that $\mathrm{TxtFex}^*_* \neq \mathrm{TxtFext}^*_*$, thereby answering a question posed by Osherson, Stob and Weinstein in 1986. We prove this in a strong way by exhibiting a family in $\mathrm{TxtFex}^*_2 \setminus \mathrm{TxtFext}^*_*$.

Acknowledgements

It would be remiss of me not to acknowledge those who have helped reach this landmark in my professional life. First of all, I would like to thank my advisor, Steffen Lempp, for guiding me through the process, helping me find research problems and teaching me so much of what I know about computability theory. In addition, I would like to thank Leo Harrington with who met with me several times during the Summer and Fall of 2011 and who mentored and advised me during the Spring and Summer of 2012 while I was in Berkeley. I learned the beautiful technique of infinite injury priority arguments from Professor Harrington and developed a great deal as a mathematician during my time working with him.

Through correspondence, I benefited greatly from Frank Stephan's encyclopedic knowledge of algorithmic learning theory. It is from him that I learned of the open question I answer in the final chapter of this thesis. Finally, I would like to thank Arnold Miller for numerous seminar classes in various subjects and for advising me for a period of time.

I would also thank my brother, mother and father. They have been integral to the process. My parents have provided me with advice and support, especially during the more difficult moments of graduate school. While both acquiring our degrees, my brother Kostas, has been my truest comrade.

Contents

Chapter 1

Introduction

1.1 Introduction

Algorithmic learning theory examines the process by which members of a class are identified from a finite amount of information. The classes to be learned are either classes of functions or classes of computably enumerable (*c.e.*) sets. Intuitively, a *c.e.* represents an infinite object that cannot be completely understood in a finite amount of time.

Since learning is not a mathematical concept, it is not endowed with an unambiguous definition. Like the concept of computability, learning has an intuitive meaning, but lends itself to a number of different formalizations. Whereas computability theory has produced one model, to which all accepted models are equivalent, learning theory has produced a plethora of non-equivalent models. In learning theory, we consider effective formalizations and call them models of learning. A model is principally defined by two factors: the type of information and the criterion for success. The information is read by a learning machine from an enumeration of the set to be learned. As the machine must be computable, it cannot consider the entirety of an enumeration and will only take a finite initial segment as input. On such an input, the learning machine outputs a natural number, interpreted as a Σ^0_1-code describing the content of the set being enumerated.

We call such outputs *hypotheses*. As the machine reads longer initial segments of the enumeration, it outputs a sequence of hypotheses that we call the *hypothesis stream*. Depending on the model of learning, the criterion for success is defined by the accuracy, consistency and frequency of correct information in the hypothesis stream. Additional limitations, such as bounds on the computational resources of the learning machine, are often considered.

The first model of learning is due to Gold [7]. According to his model, now commonly referred to as TxtEx-learning, a machine is deemed to have successfully identified an enumeration if, on cofinitely many initial segments, the machine outputs the same hypothesis and it is correct. A machine is said to have learned a set if it identifies every enumeration of the set, and has learned a family if it learns every member of the family. In this thesis, we take up several additional models of learning. Two of these, TxtEx^n- and TxtEx^*-learning, are the anomalous variants of TxtEx-learning. The other models we examine are the standard notions of TxtFin-learning, TxtBC-learning, TxtFex-learning, TxtFex^i_j-learning and $\mathrm{TxtFext}^i_j$-learning. All of these models differ from TxtEx-learning in what constitutes successful identification of an enumeration. TxtEx^n- and TxtEx^*-learning differ from TxtEx-learning in that the unique hypothesis appearing infinitely many times in the hypothesis stream need not code a set that is entirely correct. For TxtEx^*-learning, the hypothesis must code a set having finite symmetric difference with the content of the given enumeration and for TxtEx^n-learning, the cardinality of the symmetric difference must be bounded by n. In the case of TxtFin-learning, while a machine is permitted to abstain from making a hypothesis for a finite amount of time, it must eventually output a hypothesis and the first such hypothesis must be correct. A machine is said to have TxtBC-identified an enumeration if all but finitely many of the hypotheses in the hypothesis stream are correct; in contrast to TxtEx-learning, the hypotheses need not be the same. TxtFex-learning requires that the number of unique hypotheses in the hypothesis stream is finite and that those output infinitely many times must be correct. TxtFex^i_j-learning differs from TxtFex-learning in that there can be at most j hypotheses output infinitely many times and each of those hypotheses can have a symmetric difference with the enumerated set of cardinality at most i. Finally, TxtFext^i_j-learning is nearly identical to TxtFex^i_j-learning except that all hypotheses output infinitely often must code the same set.

In Chapter 2, we introduce a new line of inquiry to the field of learning theory. We examine the complexity of determining whether a family is learnable, given a code for an effective presentation of the family, thereby establishing a measure of the complexity of the learning process.

We prove that decision problems for learning under the standard notions of TxtFin-learning and TxtEx-learning are Σ^0_3-complete and Σ^0_4-complete, respectively, and that those for TxtBC-learning and TxtEx^*-learning are both Σ^0_5-complete, when certain natural limitations are placed on the complexity of the families considered. In proving the Σ^0_5-completeness of TxtBC-learning, we obtain a TxtBC-learning analog of a theorem of Blum and Blum [4]. Blum and Blum's work demonstrated that a set is TxtEx-learnable if it is TxtEx-learnable from computable enumerations. We show that, provided the family under consideration is uniformly computably enumerable (*u.c.e.*), one need only consider Δ^0_2 enumerations to decide if a family is TxtBC-learnable. Further, we prove that the decision problems for both TxtFex-learning and TxtEx^n-learning are Σ^0_4-complete.

In Chapter 3, we answer a question posed in 1986 about the relative strength of two learning criteria, TxtFex^*_* and TxtFext^*_*. In order to prove $\text{TxtFex}^*_* \neq \text{TxtFext}^*_*$, we explicitly construct a family that is TxtFex_2^*-learnable, but not $\mathrm{TxtFext}_*^*$-learnable. Because the family is TxtFex_2^*-learnable, the result is the strongest possible result. To obtain the result, we diagonalize against every attempt to $\mathrm{TxtFext}_*^*$-learn the family by including, for each machine, a subfamily that witnesses the machine's failure to $\mathrm{TxtFext}_*^*$-learn the family. Each subfamily is produced by means of an effective construction and the entire family is *u.c.e.*

We preface the results with a brief introduction to some of the concepts and notation of learning theory and computability theory. For a more in depth treatment, we refer the reader to Osherson *et al.* [8] and Soare [10].

1.2 Preliminaries

1.2.1 Notation

In this section, we give a brief overview of the computability theory and learning theory notation that will be used throughout.

We fix standard numberings of the *c.e.* sets and of the partial computable functions, $W_0, W_1, \ldots$ and $\psi_0, \psi_1, \ldots$, respectively. For the *c.e.* sets, we denote by $W_{e,s}$ the result of computing the set coded by e up to s stages and write $\mathcal{E}$ for the set of all *c.e.* sets. We write $\psi_{e,s}(x) \downarrow$ if the function coded by e has converged on input x, after s computation stages. Otherwise, we write $\psi_{e,s}(x) \uparrow$. We regard a set, A, as coding a family where the i^{th}-member of the family is $\{x : \langle i, x \rangle \in A\}$ and $\langle x, y \rangle$ is a fixed computable pairing function. Lower case Greek letters will typically refer to finite strings of natural numbers. Enumerations will be treated either as infinite strings, called texts in learning theory, or as functions on the natural numbers. Initial segments of enumerations will feature throughout this paper and will either be denoted by lowercase Greek letters, as mentioned above, or by initial segments of functions, e.g. $f\upharpoonright n$. Elements of an enumeration will be typically denoted by $f(n)$. To switch from ordered lists to unordered sets, we say that content$(\sigma) = \{x \in \mathbb{N} : (\exists n)(x = \sigma(n))\}$. For infinite enumerations, we extend the content notation to denote the set that is enumerated. If A and B are sets of natural numbers and their symmetric difference, $A \triangle B$, is finite, then we write $A =^* B$.

From learning theory, we adopt the term "learning machine" to describe a machine that takes a finite string as input and outputs a natural number. We also follow the convention of denoting such machines by either M or N, with subscripts and superscripts as needed to indicate parameters. When an enumeration of all learning machines is required, we denote the n^{th} such partial computable machine by M_n. As with the enumeration of the *c.e.* sets, the enumeration is effective and fixed.

1.2.2 Definitions and Basic Results

In this section we give formal definitions of all learning models considered here and provide key examples distinguishing the models that will feature in the hardness arguments in Chapter 2. These examples also serve to provide additional intuition about the differences between the learning models.

The simplest definition of learning is finite-learning, or TxtFin-learning.

Definition 1.2.1 (Bārzdiņš and Freivalds [3]). *Fix a symbol, '?', as a placeholder to indicate that a hypothesis has not yet been made. The definition of TxtFin-learning by a learner, M, is in four parts:*

1. *M* TxtFin-identifies *an enumeration f if and only if* $(\exists n)(\forall n' < n)\Big(M(f\restriction n') = ?\wedge M(f\restriction n) \neq ? \wedge W_{M(f\restriction n)} = \mathit{content}(f)\Big)$.

2. *M* TxtFin-learns *a c.e. set A if and only if M TxtFin-identifies every enumeration for A.*

3. *M* TxtFin-learns *a family of c.e. sets if and only if M TxtFin-identifies every member of the family.*

4. *A family, $\mathcal{F}$, is* TxtFin-learnable *(denoted $\mathcal{F} \in$ TxtFin) if and only if there is a machine, M, that TxtFin-learns $\mathcal{F}$.*

Definition 1.2.1 allows us to consider finite learning either of a single set or of a family of sets. Observe, however, that learning is trivial when learning a single set. For any *c.e.* set, W_e, the learning machine

$$N(\sigma) = e$$

TxtFin-learns W_e. This is true for all models of learning and for this reason we do not consider the learning problem associated with a single set.

Consider two examples, one of a TxtFin-learnable family, the other of a family that is not TxtFin-learnable.

Example 1.2.1. *Let $\mathcal{F} = \{F_0, F_1, \ldots\}$ be a u.c.e. family with the property that there is a computable function, g, such that $g(n) \in F_n$, but $g(n) \notin \cup_{i\neq n} F_i$. Since $\mathcal{F}$ is u.c.e., an application of the s-n-m Theorem (see Soare [10]) yields a computable function f such that $W_{f(n)} = F_n$. The family is TxtFin-learned by a machine, M, defined as follows:*

$$M(\sigma) = \begin{cases} f(n) & (\exists n)(g(n) \in \mathit{content}(\sigma)), \\ ? & \mathit{otherwise}. \end{cases}$$

Example 1.2.2. *Fix c.e. sets, W_a and W_b, such that $W_a \subsetneq W_b$. The family $\mathcal{F} = \{W_a, W_b\}$ is not TxtFin-learnable by any machine.*

Fix a learning machine M. We will demonstrate that M cannot TxtFin-learn $\mathcal{F}$ by building an enumeration of either W_a or W_b. As long as the machine makes no hypothesis, we enumerate W_a. As soon as M makes a hypothesis other than ?, we enumerate the other set in $\mathcal{F}$. Fix enumerations $\{a_n\}_{n\in\mathbb{N}}$ of W_a and $\{b_n\}_{n\in\mathbb{N}}$ of W_b. We will inductively define an enumeration, $\{c_n\}_{n\in\mathbb{N}}$, for either W_a or W_b that M cannot learn. During this construction we must maintain a pair of variables, x_0 and x_1.

Stage 0: *We set $c_0 = a_0$ and $x_0 = x_1 = \ ?$.*

Stage s+1: *First, suppose that $x_0 = x_1 = \ ?$. Let $h = M(c_0 c_1 \dots c_s)$. If $h = \ ?$ then we set $c_{s+1} = a_{s+1}$ and proceed to stage $s+2$. If $W_h = W_a$ then set $c_{s+1} = x$, $x_0 = a$ and $x_1 = s+1$. If $W_h = W_b$ then set $c_{s+1} = a_{s+1}$, $x_0 = b$ and $x_1 = s+1$.*

If $x_0 = a$ and $x_1 = n$ then set $c_{s+1} = b_{s+1-n}$. Finally, if $x_0 = b$ and $x_1 = n$ then set $c_{s+1} = a_{s+1-n}$. Observe that each of these conditions, once true, are true cofinitely.

If $h = \ ?$ at every stage of the construction, then $\{c_n\}_{n\in\mathbb{N}}$ enumerates W_a. Since M hypothesizes ? on every initial segment of $\{c_n\}_{n\in\mathbb{N}}$, M has failed to TxtFin-identify the enumeration. Suppose h is the least hypothesis M outputs on $\{c_n\}_{n\in\mathbb{N}}$. If $W_h = W_a$, then $\{c_n\}_{n\in\mathbb{N}}$ enumerates W_b and M has failed. Finally, if $W_h = W_b$ at some stage, $\{c_n\}_{n\in\mathbb{N}}$ enumerates W_a and again M has failed. Thus, the construction produces an enumeration that M cannot TxtFin-identify. Since M is arbitrary, we conclude that $\mathcal{F}$

is not TxtFin-learnable.

The family $\mathcal{G} = \{\mathbb{N} \setminus \{x\} : x \in \mathbb{N}\}$ provides a very different example of a family no machine can TxtFin-learn. Unlike the family in Example 1.2.2, $\mathcal{G}$ is infinite and every finite subset of $\mathcal{G}$ is TxtFin-learnable. The proof, however, that the family is not TxtFin-learnable is nearly the same. A family similar to $\mathcal{G}$ is used in the proof of Theorem 2.1.2.

Next, we define TxtEx-learning. Like Definition 1.2.1, the definition of TxtEx-learning describes identification of enumerations, learning of sets and learning of families.

Definition 1.2.2 (Gold [7]). *The definition of TxtEx-learning is in four parts:*

1. M TxtEx-identifies *an enumeration f if and only if* $(\exists n)\Big(\lim_{i\to\infty} M(f{\upharpoonright}i) = n \wedge W_n = \mathit{content}(f)\Big)$.

2. M TxtEx-learns *a c.e. set A if and only if M TxtEx-identifies every enumeration for A.*

3. M TxtEx-learns *a family of c.e. sets if and only if M TxtEx-identifies every member of the family.*

4. *A family, $\mathcal{F}$, is* TxtEx-learnable *(denoted $\mathcal{F} \in$ TxtEx) if and only if there is a machine, M, that TxtEx-learns $\mathcal{F}$.*

From the definition it is clear that every TxtFin-learnable family is also TxtEx-learnable. Observe that the family described in Example 1.2.2 is TxtEx-learnable, therefore TxtEx $\supsetneq$ TxtFin.

Example 1.2.3. *The family $\mathcal{F} = \{A : A \text{ is finite}\} \cup \{\mathbb{N}\}$ is not TxtEx-learnable by any machine.*

Fix M, a learning machine. We construct an enumeration, f, for a set in $\mathcal{F}$ that M fails to TxtEx-identify. The construction proceeds in stages.

Stage 0: *Set $f(0) = 0$ and $k = 0$.*

Stage s+1: *Let $M(f \upharpoonright s+1) = h$. If $W_h = \text{content}(f \upharpoonright s+1)$, then set $f(s+1) = k+1$ and increment k. If $W_h \neq \text{content}(f \upharpoonright s+1)$, then set $f(s+1) = k$.*

Either f is an enumeration of $\mathbb{N}$ or of a finite set. In the former case, there must be an infinite sequence, $\{n(i)\}_{i \in \mathbb{N}}$, such that $W_{M(f \upharpoonright n(i))}$ is finite. If f enumerates a finite set, then there is an n such that, for all $m \geq n$, $W_{M(f \upharpoonright m)} \neq \text{content}(f)$. In both cases M has failed to TxtEx-learn $\mathcal{F}$.

The model of learning we consider next is that of TxtBC-learning.

Definition 1.2.3 (Bārzdiņš [2])**.** *The definition of TxtBC-learning is in four parts:*

1. *M* TxtBC-identifies *an enumeration f if and only if $(\exists n)(\forall i > n)\Big(W_{M(f \upharpoonright i)} = \text{content}(f)\Big)$.*

2. *M* TxtBC-learns *a c.e. set A if and only if M TxtBC-identifies every enumeration for A.*

3. *M* TxtBC-learns *a family of c.e. sets if and only if M TxtBC-identifies every member of the family.*

4. *A family, $\mathcal{F}$, is* TxtBC-learnable *(denoted $\mathcal{F} \in TxtBC$) if and only if there is a machine, M, that TxtBC-learns $\mathcal{F}$.*

To show that Definition 1.2.3 is not a trivial extension of Definition 1.2.2 we demonstrate the existence of a family that is not TxtBC-learnable as well as a family that is TxtBC-learnable, but not TxtEx-learnable. Example 1.2.3 already provides us with the first family. Constructing the second family requires a little more work. We begin with a theorem.

Theorem 1.2.4 (Blum and Blum [4]). *Suppose A is a c.e. set and M is a computable learner that TxtEx-learns A. There exists a string σ such that:*

- *content(σ)$\subseteq A$*
- $W_{M(\sigma)} = A$
- *if content(τ)$\subseteq A$, then $M(\sigma\tau) = M(\sigma)$*

Proof. Suppose M TxtEx-learns A, but no such σ exists. Either there is an enumeration of A on which the learner converges to an incorrect hypothesis or, for any string σ of elements of A, there is an extension τ of elements of A such that $M(\sigma) \neq M(\tau)$. The former contradicts the assumption that M TxtEx-learns A thus, we need only consider the latter. Let $a_0, a_1, \ldots$ be a fixed computable enumeration of A. We now construct a new enumeration in stages.

Stage 0: Set $f(0) = a_0$.

Stage s+1: Suppose $f\restriction n$ has been defined so far. Search for the least string, τ, with content(τ) $\subseteq A$ such that $M(f\restriction n\hat{\ }\tau) \neq M(f\restriction n)$. By assumption, such a τ exists thus, the search must terminate. Define $f(n+i) = \tau(i)$ for $0 \leq i < |\tau|$.

The learner changes hypothesis infinitely many times on f, which is a contradiction since M TxtEx-learns A. Hence, the desired string, σ, must exist.

□

A string satisfying the conditions above is called a locking sequence. With this in hand, we can now construct the desired family.

Example 1.2.4. *Fix a non-computable c.e. set, A, and $a_0, a_1, \ldots$ a computable enumeration of A. Let $\mathcal{F} = \{A \cup \{n\} : n \in \mathbb{N}\}$. We will prove that $\mathcal{F}$ is TxtBC-learnable, but not TxtEx-learnable.*

Suppose M TxtEx-learns $\mathcal{F}$. By Theorem 1.2.4, M has a locking sequence for each set in $\mathcal{F}$. In particular, let σ be a locking sequence for A. Define

$$S = \{x : \exists n(M(\sigma\hat{}x\hat{}a_0\hat{}a_1\hat{}\ldots\hat{}a_n) \neq M(\sigma))\}.$$

By the definition of a locking sequence, $A \cap S = \emptyset$. The set, S, has a Σ^0_1 description and hence is c.e. Since A is not computable and $A \cap S = \emptyset$, the complement of $A \cup S$ must be nonempty. Fix $x_0 \in \mathbb{N} \setminus (A \cup S)$. As $x_0 \notin S$, $M(\sigma\hat{}x_0\hat{}a_0\hat{}a_1\hat{}\ldots\hat{}a_n) = M(\sigma\hat{}x_0)$ for all $n \in \mathbb{N}$. Furthermore, $W_{M(\sigma)} = A$ and $A \cup \{x_0\} \neq A$. By extending $\sigma\hat{}x_0$ we can produce an enumeration on which M fails to TxtEx-identify a set in $\mathcal{F}$.

To see that $\mathcal{F}$ is TxtBC-learnable, fix a code e such that $W_e = A$ and let M be a computable function which, on input σ, outputs a code for $W_e \cup$ content(σ). Note that M is computable by the s-n-m Theorem. Even though the learner may never output the same hypothesis twice, the hypothesis stream is cofinitely often correct.

Taking Example 1.2.4 together with the definitions, it is clear that TxtEx $\subsetneq$ TxtBC.

We now define the remaining models of learning. First, we consider vacillatory learning, which permits a machine to vacillate between a finite number of hypotheses infinitely. We give a more general definition, that of anomalous vacillatory learning, of which vacillatory learning is a special case. There are two forms of anomalous vacillatory learning, TxtFex^i_j and TxtFext^i_j.

Definition 1.2.5 (Case [5]). *Let $i, j \in \mathbb{N} \cup \{*\}$. The definition of TxtFex^i_j-learning is in four parts.*

1. *M TxtFex^i_j-identifies an enumeration f if and only if there is a set S with $|S| \leq j$ (or merely finite S if $j = *$) such that $(\forall^\infty n)(\forall a \in S)\Big(M(f{\restriction}n) \in S \wedge W_a =^i \mathit{content}(f)\Big)$.*

2. *M TxtFex^i_j-identifies a c.e. set A if and only if M TxtFex^i_j-identifies every enumeration for A.*

3. *M TxtFex^i_j-learns a family of c.e. sets if and only if it TxtFex^i_j-identifies every member of the family.*

4. *$\mathcal{F}$ is TxtFex^i_j-learnable (denoted $\mathcal{F} \in \mathit{TxtFex}^i_j$) if and only if there is a machine M that TxtFex^i_j-learns $\mathcal{F}$.*

For simplicity, we write TxtFex to denote TxtFex^0_* and TxtFex_j to denote TxtFex^0_j when $j \in \mathbb{N}$.

Definition 1.2.6 (Osherson, Stob and Weinstein [8]). *Let $i, j \in \mathbb{N} \cup \{*\}$. The definition of $\mathit{TxtFext}^i_j$-learning is in four parts.*

1. *M $\mathit{TxtFext}^i_j$-identifies an enumeration f if and only if there is a set S with $|S| \leq j$ (or merely finite S if $j = *$) such that $(\forall^\infty n)(\forall a, b \in S)(M(f{\restriction}n) \in S \wedge W_a = W_b =^i \mathit{content}(f))$.*

2. *M TxtFext^i_j-identifies a c.e. set A if and only if M TxtFext^i_j-identifies every enumeration for A.*

3. *M TxtFext^i_j-learns a family of c.e. sets if and only if it TxtFext^i_j-identifies every member of the family.*

4. *$\mathcal{F}$ is TxtFext^i_j-learnable (denoted $\mathcal{F} \in$ TxtFext^i_j) if and only if there is a machine M that TxtFext^i_j-learns $\mathcal{F}$.*

Observe that $\text{TxtFex}^0_j = \text{TxtFext}^0_j$ for all $j \in \mathbb{N} \cup \{*\}$. The differences between the two variants of anomalous learning are explored by Case [5], by Osherson, Stob and Weinstein [8] and in Chapter 3.

Finally, we define the anomalous variant of TxtEx-learning.

Definition 1.2.7 (Osherson and Weinstein [9]). *Let $i \in \mathbb{N} \cup \{*\}$. The definition of TxtEx^i-learning is in four parts:*

1. *M* TxtEx^i-identifies *an enumeration f if and only if $(\exists n)\Big(\lim_{j\to\infty} M(f\restriction j) = n \wedge W_n =^i content(f)\Big)$.*

2. *M TxtEx^i-identifies a c.e. set A if and only if M TxtEx^i-identifies every text for A.*

3. *M TxtEx^i-learns a family of c.e. sets if and only if it TxtEx^i-identifies every member of the family.*

4. *$\mathcal{F}$ is TxtEx^i-learnable (denoted $\mathcal{F} \in$ TxtEx^i) if and only if there is a machine M that TxtEx^i-learns $\mathcal{F}$.*

We now give a final example, one that distinguishes TxtEx-learning and TxtFex-learning. A theorem, due to Angluin, is required.

Theorem 1.2.8 (Angluin's Theorem [1])**.** *Let $\mathcal{L} = \{L_0, L_1, \ldots\}$ be a uniformly computable family. $\mathcal{L}$ is TxtEx-learnable if and only if there is a u.c.e. family of finite sets, $\mathcal{F} = \{F_0, F_1, \ldots\}$, such that*

1. *$F_i \subseteq L_i$ for all $i \in \mathbb{N}$*
2. *If $F_i \subseteq L_j \subseteq L_i$, then $L_i = L_j$*

Proof. To prove necessity, suppose M TxtEx-learns $\mathcal{L}$ and fix $\{\sigma_i^n\}_{i\in\mathbb{N}}$, a recursive enumeration of all strings of elements of L_n. Define

$$F_n = \{x : (\exists i)(\forall j \leq i)(\exists k)\Big((x \in \text{content}(\sigma_i^n)) \wedge (M(\sigma_j^n \sigma_k^n) \neq M(\sigma_j^n))\Big)\}$$

Note that F_n is finite because for each L_n there is a locking sequence, σ_k^n, of least index. Each $x \in F_n$ must appear in a string with index less that k. Fix $i, j \in \mathbb{N}$ and suppose $F_i \subseteq L_j \subseteq L_i$. The content of the locking sequence for L_i is contained in L_j and extending to an enumeration of L_j is an extension by elements of L_i and hence would not change the learners hypothesis. Thus, we have an enumeration of L_j on which M converges to an index for L_i. Since M learns the family, $L_i = L_j$. If $\mathcal{L}$ is *u.c.e.*, then $\{F_0, F_1, \ldots\}$ is also *u.c.e.* hence, this direction of the equivalence holds even if $\mathcal{L}$ is merely *u.c.e.*

Now suppose we are given a *u.c.e.* sequence of finite sets as above. We will define a learner, M, that *TxtEx*-learns the family. On input σ, M outputs the least $i \leq |\sigma|$ such that $i = |\sigma|$ or $F_{i,n} \subseteq \text{content}(\sigma) \subseteq L_i$. Observe that the statement "content$(\sigma) \subseteq L_i$" is decidable because $\mathcal{L}$ is uniformly computable. Fix L_i and assume that, for all

$j \leq i$, $L_j \neq L_i$. Suppose $\{a_j\}_{j \in \mathbb{N}}$ is an enumeration of L_i. Fix k large enough that $\{a_0, a_1, \ldots a_k\} \nsubseteq L_n$ for $n \leq i$ with $L_i \nsubseteq L_n$ and $F_{n,k} = F_n$ for all $n \leq i$. On any initial segment of $\{a_j\}_{j \in \mathbb{N}}$ longer than k the learner will output i, which is a correct hypothesis.

□

As noted in the proof, the existence of $\mathcal{F}$ does not require $\mathcal{L}$ to be uniformly computable, merely *u.c.e.* The following example uses this observation to show that TxtFex-learning and TxtEx-learning are different.

Example 1.2.5. *Let $H_x = \{x + n : n \leq |W_x|\}$, and $L_x = \{x + n : n \in \mathbb{N}\}$. Define $\mathcal{F} = \{H_0, L_0, H_1, L_1, \ldots\}$. We claim that $\mathcal{F}$ is TxtFex-learnable, but not TxtEx-learnable.*

$\mathcal{F}$ is clearly u.c.e. and can be enumerated so that H_e is the $(2e)^{th}$ column and L_e is the $(2e+1)^{st}$ column. We must verify that $\mathcal{F}$ is TxtFex-learnable, but that no machine can TxtEx-learn $\mathcal{F}$. Consider a machine that, on input σ, sets x_0 and x_1 to be the least element and greatest element, respectively, of content(σ) and sets y_1 equal to $|W_{x_0,|\sigma|}|$. If $y_1 > x_1 - x_0$, the machine outputs $2x_0$; if $x_1 - x_0 \geq y_1$, it outputs $2x_0 + 1$. Thus, for enumerations of infinite intervals, the machine may vacillate between two different correct codes. For finite intervals, eventually only one correct code will be output. We conclude that $\mathcal{F}$ is TxtFex-learnable.

Now, we wish to show that $\mathcal{F}$ is not TxtEx-learnable. To obtain a contradiction, assume that we have a machine, M, that TxtEx-learns $\mathcal{F}$. As we observed at the outset, this means there is a u.c.e. family $\{G_0, G_1, \ldots\}$ such that each G_i is finite, $G_{2i} \subseteq H_i$, $G_{2i+1} \subseteq L_i$ and, if $G_i \subseteq A \subseteq B$ where B is the i^{th} set in $\mathcal{F}$ and $A \in \mathcal{F}$, then $A = B$. Specifically, $G_{2i+1} \subseteq L_i$ and $G_{2i+1} \subseteq H_i$ if, and only if, $H_i = L_i$ – exactly when W_i is an infinite set. Let m be the maximum number in G_{2i+1}. If card(W_i) $\geq m$, then

$H_{2e+1} = L_{2e+1}$. *In other words, we can decide in the limit whether or not W_i is infinite. Since we cannot actually decide in the limit whether or not a number codes an infinite set, we have obtained the desired contradiction.*

1.2.3 Enumerations

All the learning criteria described in Section 1.2.2 use enumerations as the source of information. Other sources of information are possible, e.g. informant and oracle, but we do not examine them here. We present a few basic results that will be used to place complexity bounds on the learning criteria.

While the definitions of each of the learning criteria are stated in terms of arbitrary enumerations, it can be shown that, in certain cases, learnability may be determined by examining a restricted class of enumerations. We present three theorems in this section, the first of which is due to Blum and Blum [4] and serves as a model for the other two. Theorem 2.3.1 continues in this vein, but the techniques are very different.

Theorem 1.2.9 (Blum and Blum [4])**.** *If $\mathcal{F}$ is TxtEx-learned from computable enumerations by a computable machine M, then it is TxtEx-learnable from arbitrary enumerations by a computable machine $\hat{M}$.*

Proof. We fix a family, $\mathcal{F}$, and a machine, M, that TxtEx-learns $\mathcal{F}$ from computable enumerations. We will define a new machine, effectively produced from M, that searches the least string within computational bounds that appears to be a locking sequence of the set for which it is receiving an enumeration. The new machine then outputs the hypothesis that M outputs on the string that currently appears to be a locking sequence. If at some stage the string is shown not to be a locking sequence, then the learner will update its hypothesis.

Given M, we will define $\hat{M}$ on an input of σ of length n. Let $A_n(\sigma) = \{\tau : \text{content}(\tau) \subseteq \text{content}(\sigma) \wedge |\tau| \leq n\}$. Define $\hat{M}(\sigma) = M(\alpha)$ where α is the string of least length in $A_n(\sigma)$ that is lexicographically least such that for any $\beta \in A_n(\sigma)$, $M(\alpha\beta) = M(\alpha)$.

For the sake of a contradiction, suppose $\hat{M}$ fails to TxtEx-learn $\mathcal{F}$. Let g be an enumeration for a set, F with computable enumeration f, in $\mathcal{F}$ on which $\hat{M}$ fails to identify F. We consider two cases.

Case 1: Suppose that the choice of α changes infinitely many times on initial segments of g. In this case, we can contruct in stages a computable enumeration for F on which M fails to TxtEx-learn. At each stage, search for an extension of what has been defined on which M changes its hypothesis. By assumption, for every n and every σ with content(σ) contained in F, there is an extension on which the learner changes hypothesis, thus the given search will always terminate and the resulting enumeration is computable.

Case 2: Suppose the choice of α stabilizes. Let α be the final state, suppose $M(\alpha) = h$ and $W_h \neq F$. Once again, our goal is to create a computable enumeration on which M must exhibit the same behaviour. Define a computable enumeration, g', for F: $g'(i) = \alpha(i)$ if $i < |\alpha|$ and $g'(i) = f(i - |\alpha|)$ if $i \geq |\alpha|$. On g' M must converge to h, which is an incorrect hypothesis.

$\square$

We follow with analogous results and proofs for TxtFin-learning and TxtEx^j-learning.

Theorem 1.2.10. *If $\mathcal{F}$ is TxtEx^j-learned from computable enumerations by a computable machine M, then it is TxtEx^j-learnable from arbitrary enumerations by a computable machine $\hat{M}$.*

Proof. The proof is identical to the proof of Theorem 1.2.9 with TxtEx everywhere changed to TxtEx^j. We omit the details of the proof.

□

Theorem 1.2.11. *If $\mathcal{F}$ is TxtFin-learned from computable enumerations by a computable machine M, then it is TxtFin-learnable from arbitrary enumerations by some computable machine $\hat{M}$.*

Proof. First note that if there is an enumeration for a set, $F \in \mathcal{F}$, on which the first hypothesis output by M is incorrect, then there is a computable computable enumeration of F with the same property. To see this, consider the finite string on which M has output its first hypothesis. Extend it by a computable enumeration of F. The result is computable and M converges, in the sense of finite learning, to an incorrect code. Since this is a contradiction, we may assume that M does not converge to incorrect codes on any enumeration in addition to identifying computable enumerations for sets in $\mathcal{F}$. We must still prove that there are no enumerations on which the learner fails to output a hypothesis.

For a string σ, define $A_\sigma = \{\tau : \text{content}(\tau) \subseteq \text{content}(\sigma) \wedge |\tau| \leq |\sigma|\}$. Order A_σ by $\sigma < \tau$ if either $|\sigma| < |\tau|$ or $|\sigma| = |\tau|$ and σ is below τ in the lexicographical order on $\omega^{|\sigma|}$. Define $\hat{M}(\sigma)$ to be $M(\tau)$ where τ is the least element of A_σ on which M outputs a hypothesis other than ?. If no such τ exists $\hat{M}(\sigma) =?$. On no enumeration will the least hypothesis of $\hat{M}$ be incorrect since that would imply the existence of such an enumeration for M. Fix an arbitrary enumeration, f, and a computable enumeration, f', for $F \in \mathcal{F}$. Let σ be an initial segment of f' on which M outputs a least (and hence correct) hypothesis. Every element of content(σ) appears in f eventually, thus there is an n such that $\sigma \in A_{f\restriction n}$. For some $m \leq n$, $\hat{M}(f\restriction m)$ will be a least and correct hypothesis.

$\square$

The existence of a computable enumeration on which a machine fails to learn a set does not mean that such an enumeration can be found via an effective procedure based on a code for the machine. Consider Example 1.2.3. Clearly the process described in the example is not computable. To find a computable enumeration witnessing that failure of a machine M, we must follow a different strategy. Instead we construct a collection of enumerations on at least one of which M must fail to successfully TxtEx-learn.

Given learner M, suppose that $T[n]$ has already been defined. We divide what follows into two cases.

If every string, σ, is extended by a string, τ, such that $M(\sigma) \neq M(\tau)$, then an inductive procedure based on a search that is guaranteed to terminate produces a computable enumeration on which M fails. If, on the other hand, there is a string, σ, on every extension of which M hypothesizes $M(\sigma)$, then M fails to learn either

$$\sigma(0)\ \sigma(1)\ \ldots\ \sigma(|\sigma|-1)\ \sigma(0)\ \sigma(0)\ \ldots$$

or

$$\sigma(0)\ \sigma(1)\ \ldots\ \sigma(|\sigma|-1)\ 0\ 1\ 2 \ldots$$

All the given texts are computable, but deciding which M fails to TxtEx-identify is not decidable.

1.2.4 Index Sets

We will make use of several standard index sets in the results that follow, as well as proving results about a collection of novel index sets defined in terms of learning models. We provide definitions of the standard index sets for reference.

Definition 1.2.12. *1. Define FIN* $= \{x : W_e$ *is finite*$\}$.

2. Define INF $= \{x : W_e$ *is infinite*$\}$.

3. Define COF $= \{x : W_e$ *is cofinite*$\}$.

4. Define COINF $= \{x : W_e$ *is coinfinite*$\}$.

In the following theorem, we summarize the well known complexities of the predicates in Definition 1.2.12.

Theorem 1.2.13. *1. FIN is* Σ^0_2*-complete.*

2. INF is Π^0_2*-complete.*

3. COF is Σ^0_3*-complete.*

4. COINF is Π^0_3*-complete.*

Based on the learning models defined in Section 1.2.2, we define the following index sets of codes for families.

Definition 1.2.14. *Define the following four index sets of* Σ^0_1 *codes for u.c.e. families learnable according to each given criterion.*

1. Define FINL to be the index set for TxtFin-learning.

2. *Define EXL to be the index set for TxtEx-learning.*

3. *Define BCL to be the index set for TxtBC-learning.*

4. *Define EXL* to be the index set for TxtEx*-learning.*

Chapter 2

Completeness Results

2.1 TxtFin-Learning

We begin by demonstrating that TxtFin-learning is Σ^0_3-complete. This is accomplished in two steps. With Theorem 2.1.1 we place a Σ^0_3 upper bound on the complexity of FINL. Next, Theorem 2.1.2 reduces an arbitrary Σ^0_3 predicate to FINL.

Theorem 2.1.1. *FINL has a Σ^0_3 description.*

Proof. Suppose e codes a *u.c.e.* family $\mathcal{L} = \{L_0, L_1, \ldots\}$. We will show that $e \in$ FINL if and only if

$$(\exists k)(\forall i)\Big((\exists\sigma)\Big((\text{content}(\sigma) \subseteq L_i) \wedge (M_k(\sigma) \neq ?)\Big) \wedge \psi(k)\Big)$$

where $\psi(k)$ denotes

$$(\forall\alpha, j)(\exists\tau \prec \alpha)\Big((M_k(\tau) \neq ?) \vee (\text{content}(\alpha) \not\subseteq L_j) \vee (M_k(\alpha) = ?) \vee (W_{M_k(\alpha)} = L_j)\Big).$$

Observe that the formula mandates the existence of a learning machine, M_k, such that for every set in the family there is a string of elements from that set on which the learner outputs a hypothesis. Furthermore, if it outputs a least hypothesis, in the sense that the only hypothesis it makes on proper initial segments of the given data is ?, then that hypothesis is correct. We now build a new machine, $\hat{M}$, based on M_k, which TxtFin-learns the family $\mathcal{L}$.

For a string σ, define $A_\sigma = \{\tau : \text{content}(\tau) \subseteq \text{content}(\sigma) \wedge |\tau| \leq |\sigma|\}$. Order A_σ by $\sigma < \tau$ if either $|\sigma| < |\tau|$ or else $|\sigma| = |\tau|$ and σ is below τ in the lexicographical order on $\mathbb{N}^{|\sigma|}$. Define $\hat{M}(\sigma)$ to be $M_k(\tau)$ where τ is the least element of A_σ on which M_k outputs a hypothesis other than ?. If no such τ exists, $\hat{M}(\sigma) =?$. On no enumeration will the least hypothesis of $\hat{M}$ be incorrect since that would imply the existence of such an enumeration for M_k. Fix an arbitrary enumeration, f, for $L_i \in \mathcal{L}$. Let σ be a string, with content$(\sigma) \subseteq L_i$, on which M_k outputs a least (and hence correct) hypothesis. Every element of content(σ) appears in f, thus there is an n such that $\sigma \in A_{f\upharpoonright n}$. For some $m \leq n$, $\hat{M}(f\upharpoonright m)$ will be a least and correct hypothesis.

Since the given formula is Σ^0_3, we have produced a predicate with the desired properties.

□

Theorem 2.1.2. *FINL is* Σ^0_3*-hard.*

Proof. First, we provide an informal description of the proof. The construction consists of two conflicting processes. One is an attempt to place a unique marker in every set hence, permitting easy identification. The other is an attempt to create, for each marker, an infinite number of sets that contain the marker each of which lacks one element all the other sets have. During the construction, the interplay between the two processes is controlled by a finite amount of information about the outcome of a Σ^0_3 predicate. If the predicate is true, then there is a computable collection of markers with a unique marker in each set. Agreement between sets with the same marker is enforced. If the predicate is false, then for every marker there is a subfamily which is not TxtFin-learnable and every member of which contains the marker.

Consider a Σ_3^0 predicate $P(e) \leftrightarrow (\exists x)(\forall y)(\exists z)(R(e,x,y,z))$, where R is a computable predicate. We will reduce P to FINL by means of a computable function such that the image of e is a code for a *u.c.e.* family that is TxtFin-learnable if $P(e)$ and not TxtFin-learnable if $\neg P(e)$. We now fix e and proceed with the construction of a family based on that particular e. The family under construction is denoted $\mathcal{G} = \{G_0, G_1, \ldots\}$. While $\mathcal{G}$ depends on e, we omit the parameter for the sake of simplicity as we are only concerned with the fixed value of e during the construction below.

Each $G \in \mathcal{G}$ will consist of ordered pairs and can thus be partitioned into columns $C(G,i) = \{x : \langle i, x\rangle \in G\}$. For convenience, we will index the columns starting with -1. Let $C(i) = \{\langle i, x\rangle : x \in \mathbb{N}\}$. For the remainder of the construction we will adhere to the notation defined in the following list.

- Let $\langle x_s, y_s, z_s\rangle$ be a computable enumeration of all triples of natural numbers.
- For $x \geq 1$, let h_x^s be the number of stages, up to s, at which the largest j, such that $(\forall y \leq j)(\exists i \leq s)(x_i = x \wedge y_i = y \wedge R(e, x-1, y, z_i))$, has increased.
- If it exists, let $h_x = \lim_{s\to\infty} h_x^s$.
- For $x \in \mathbb{N}$, an *x-label* is a number in the x^{th}-column, $C(x)$, used to distinguish sets in $\mathcal{G}$. Labels may be enumerated into any column of any $G \in \mathcal{G}$ except $C(G,-1)$ during the construction. We say G has an x-label k when $k \in C(G,x)$. Equivalently, when $\langle x, k\rangle \in G$.
- Define $\mathcal{S}_x^k = \{G \in \mathcal{G} : G \text{ has an } x\text{-label } k\}$. The family depends on the stage, but we do not include any notation to indicate the stage as it will be clear from context. When we wish to reference the i^{th}-member of $\mathcal{S}_x^k$, we will write $S_x^k(i)$.

- Let $n_x^k = \mathrm{card}(\mathcal{S}_x^k)$.
- Define a function, p_x^k, used to record numbers associated with each member of $\mathcal{S}_x^k$. In particular, $p_x^k(i)$ will be a number withheld from $S_x^k(i)$. Denote by P_x^k the set $\{p_x^k(0), \ldots, p_x^k(n_x^k)\}$. At each stage, we will ensure that $P_x^k \setminus \{p_x^k(i)\} \subseteq C(S_x^k(i), -1)$ and $p_x^k(i) \notin C(S_x^k(i), -1)$. At certain stages, the values of the p_x^k will change.

Next, we describe the actions taken at a given stage of the construction. The construction consists of using the predicate, P, to resolve two opposing forces. One is the attempt to label all sets in a unique way, and the other is to create an infinite family that mirrors the structure of $\{\mathbb{N} \setminus \{x\} : x \in \mathbb{N}\}$ every set of which has the same label.

Stage s: The triple under consideration is $\langle x_s, y_s, z_s \rangle$. Let t be the most recent previous stage at which $x_t = x_s$. We examine two cases: $h_{x_s}^s > h_{x_s}^t$ and $h_{x_s}^s = h_{x_s}^t$.

First, suppose that $h_{x_s}^s > h_{x_s}^t$. We interpret this increase as progress toward verifying $P(e)$. We enumerate elements as needed to ensure that, for each x_s-label, k, and $i \leq \langle x_s, k \rangle + h_{x_s}^s$, if $G, G' \in \mathcal{S}_{x_s}^k$, then $C(G, i) \cap [0, \langle x_s, k \rangle + h_{x_s}^s] = C(G', i) \cap [0, \langle x_s, k \rangle + h_{x_s}^s]$. If $p_{x_s}^k(i) \leq \langle x_s, k \rangle + h_{x_s}^s$, we pick a member of $C(-1)$ greater than $\langle x_s, k \rangle + h_{x_s}^s$ and every number used in the construction so far, and set $p_{x_s}^k(i)$ equal to the chosen number. We enumerate $p_{x_s}^k(i)$ into every member of $\mathcal{S}_{x_s}^k \setminus \{S_{x_s}^k(i)\}$. Thus, for each set with x_s-label k, there is a particular natural number the set does not contain, but which is contained in all other sets with x_s-label k.

Finally, we pick the set of least index in $\mathcal{G}$ that has not yet been assigned an x_s-label and assign it a unique, and previously unused, x_s-label.

In the second case, suppose that $h_{x_s}^s = h_{x_s}^t$. This stability suggests that the outcome will be $\neg P(e)$. For each k currently in use as an x_s-label, we create a new set, $S^k_{x_s}(n^k_{x_s} + 1) \in \mathcal{S}^k_{x_s}$, such that $C(S^k_{x_s}(n^k_{x_s}+1), j) \cap [0, \langle x_s, k\rangle + h^s_{x_s}] = C(S^k_{x_s}(n^k_{x_s}), j) \cap [0, \langle x_s, k\rangle + h^s_{x_s}]$, for $j \leq k + h^s_{x_s}$. We enumerate $P^k_{x_s}$ into $S^k_{x_s}(n^k_{x_s}+1)$ and set $p^k_{x_s}(n^k_{x_s}+1)$ equal to the least member of $C(-1)$ not used during the construction so far and greater than $\langle x_s, k\rangle + h^s_{x_s}$. Finally, we enumerate $p^k_{x_s}(n^k_{x_s} + 1)$ into every member of $\mathcal{S}^k_{x_s} \setminus \{S^k_{x_s}(n^k_{x_s} + 1)\}$.

Verification: If $P(e)$, then $(\exists x)(\forall y)(\exists z)(R(e, x, y, z))$. Hence, for some x, $h^s_x \to \infty$. For infinitely many s, $x_s = x$, thus every set in $\mathcal{G}$ will eventually receive an x-label. At such stages, agreement between sets with the same label is also increased. Consequently, any two sets in $\mathcal{G}$ with the same x-label are equal. The family is learned by a machine that searches for the least x-label and outputs a code for the first set in $\mathcal{G}$ that receives the same x-label.

If $\neg P(e)$, then $(\forall x)(\exists y)(\forall z)(\neg R(e, x, y, z))$. Fix any machine, M. If $M(\sigma) = ?$ for every string σ with content contained in a member of $\mathcal{G}$, M has failed to learn $\mathcal{G}$ and we are done. Otherwise, we may pick a string, σ, such that

- $M(\sigma) \neq ?$;
- for all $\tau \prec \sigma$, $M(\tau) = ?$;
- for some $G \in \mathcal{G}$, content$(\sigma) \subseteq G$.

Let k be an x-label with which G is marked such that $\max(\text{content}(\sigma)) < \langle x_s, k\rangle + h^s_{x_s}$. Pick a stage, s, at which $h^s_y = h_y$ for all y such that G contains a y-label less than or equal to $\langle x, k\rangle + h_x$. At a subsequent stage, t, $x_t = x$ and a new set, G', is created containing content(σ). Labels contained in G' are either y-labels, k, such that h^s_y will never increase or labels enumerated into G' after stage t. The latter are only shared with sets created at subsequent stages. As a consequence, there is a member of P^k_x never enumerated into G', but contained in G. We conclude that M, an arbitrarily chosen learning machine, has failed to TxtFin-learn $\mathcal{G}$.

The computable function that maps e to a code for the *u.c.e.* family $\mathcal{G}$ constructed above is a reduction of P to FINL.

□

2.2 TxtEx-Learning

We now proceed to describe the arithmetic complexity of TxtEx-learning. The first Σ^0_4 description of EXL of which we are aware is due to Sanjay Jain. Here we present a different formula, but one which explicitly illustrates the underlying structure and serves as a model for the Σ^0_5 description of BCL given in Section 2.3.

Theorem 2.2.1. *EXL has a Σ^0_4 description.*

Proof. By Theorem 1.2.9 we need only consider computable enumerations when analyzing the complexity of EXL. Further, observe that if there is a machine, M, that TxtEx-learns a family, there is a total machine, $\hat{M}$, that TxtEx-learns the same family. Specifically, define $\hat{M}(\sigma)$ to be $M(\sigma\restriction n)$ for the greatest n such that $M(\sigma\restriction n)$ converges within $|\sigma|$ computation stages and define $\hat{M}(\sigma) = 0$ if no such initial segment exists. Suppose e codes a *u.c.e.* family $\{L_0, L_1, \ldots\}$.

We will define a formula which states that there is a learner such that for every enumeration and every set in the family, if the enumeration is total and enumerates the set, then eventually the hypotheses stabilize and a given hypothesis is either correct or the hypotheses have not yet stabilized. Syntactically, this can be stated as follows:

$$(\exists a)(\forall k,i)\Big((M_a \text{ is total}) \wedge (\phi_k \text{ is total}) \wedge (\phi_k \text{ enumerates } L_i) \to \psi(k,a,i)\Big) \tag{2.1}$$

where we define $\psi(k,a,i)$ to be

$$(\exists s)(\forall t > s)\Big(M_a(\phi_k{\restriction}t) = M_a(\phi_k{\restriction}s)\Big) \wedge (\forall n)\Big(W_{M_a(\phi_k\restriction n)} = L_i \\ \vee (\exists m > n)\big(M_a(\phi_k{\restriction}m) \neq M_a(\phi_k{\restriction}n)\big)\Big).$$

The last formula is Δ^0_3. Thus, (2.1) is Σ^0_4 and characterizes TxtEx-learning because, for any family coded by a number e which satisfies formula (2.1), there is a learner whose hypotheses converge to correct hypotheses on every computable enumeration and if e fails to satisfy formula (2.1), every learner must fail on some hypothesis for some set in the family.

We have, therefore, exhibited a Σ^0_4 description of EXL.

□

To achieve the desired completeness result, we now prove that an arbitrary Σ^0_4 predicate can be reduced to EXL. The proof utilizes the family described in Example 1.2.5. While not TxtEx-learnable, the family is learnable under more liberal descriptions of learning. It is, in a sense, just barely not TxtEx-learnable.

Theorem 2.2.2. *EXL is Σ^0_4-hard.*

Proof. Let COINF be the index set of all codes for *c.e.* sets which are coinfinite. Since COINF is Π^0_3-complete, it suffices to prove that any predicate of the form $(\exists x)(f(e,x) \in$ COINF) for a computable function f can be reduced to EXL. As in Example 1.2.5 from the preliminary section, let $H_e = \{e + x : x \leq |W_e|\}$, $L_e = \{e + x : x \in \omega\}$, and $\mathcal{F} = \{H_0, L_0, H_1, L_1, \ldots\}$. Fix a uniformly computable enumeration of $\mathcal{F}$ where H_e is the $2e+1^{st}$ column of $\mathcal{F}$ and L_e is the $2e^{th}$ column. For notational convenience, we denote the e^{th} column of $\mathcal{F}$ by F_e. We will define a sequence of *u.c.e.* families, $\mathcal{R}_{n,e}$, and choose a computable map g so that $g(e, x)$ is a Σ^0_1 code for the *u.c.e.* family $\mathcal{G}_{e,x}$ where, for $x \leq e$

$$\mathcal{G}_{e,x} = \bigcup_{n \in [x,e]} \mathcal{R}_{n,e}.$$

We construct the *u.c.e.* families $\mathcal{R}_{n,e}$ simultaneously for $x \leq n \leq e$. The family, $\mathcal{R}_{n,e}$, will consist of an infinite number of partial enumerations of F_n. How complete the enumerations are will depend on whether $e \in$ COF or $e \in$ COINF.

Stage 0: Let $\mathcal{R}_{n,e}$ be the empty set.

Stage s: Suppose that $[i, i+j] \subseteq W_{e,s}$. In this case, enumerate $F_{n,j}$, a finite partial enumeration of F_n, and the least natural number greater than or equal to $n/2$ into the i^{th} column of $\mathcal{R}_{n,e}$. Denote this last number by n_0. We include n_0 in order to guarantee that the set is nonempty and, if it is a partial enumeration of H_a or L_a, it will contain a. Also, for all $k \in [i, i+j]$, enumerate all the elements in the k^{th} column into the i^{th} column and vice versa so that all the columns with indices between i and $i+j$ are identical.

There are two cases. First, suppose $e \in$ COF, then there are only finitely many distinct sets in $\mathcal{R}_{n,e}$; cofinitely many columns of $\mathcal{R}_{n,e}$ will be identical to F_n and the rest will be finite subsets of F_n. Thus $\mathcal{G}_{e,x}$ will consist of F_n for $x \leq n \leq e$ together with some finite subsets of these sets. When $e \in$ COINF, $\mathcal{R}_{n,e}$ will contain only finite subsets of F_n and so $\mathcal{G}_{e,x}$ will consist of a collection of finite sets, possibly infinitely many.

Based on $g(e, x)$ and given an arbitrary Σ_4^0 unary predicate P, we define a new map, h, which witnesses the reduction of P to EXL. Since P is Σ_4^0, it is of the form $(\exists y)(Q(x, y))$ where Q is a Π_3^0 predicate. Let r be a one-to-one and computable map witnessing the reduction of Q to COINF. In other words, $P(x) \leftrightarrow (\exists y)(r(x, y) \in \text{COINF})$. Furthermore, we may assume that

$$P(e) \to (\forall^\infty y)(r(e, y) \in \text{COINF})$$

and

$$\neg P(e) \to (\forall y)(r(e, y) \in \text{COF})$$

Let s be such that for fixed e, $\{s(e, y)\}_{y \in \mathbb{N}}$ is a computable, strictly increasing, subsequence of $\{r(e, y)\}_{y \in \mathbb{N}}$. The existence of such a subsequence is guaranteed by the fact that r is one-to-one, implying that $\{r(e, y)\}_{y \in \mathbb{N}}$ is an unbounded sequence. For convenience, suppose that $s(e, -1) = 0$ for all $e \in \mathbb{N}$. Let h be a computable function such that, for $e \in \mathbb{N}$, $h(e)$ is a code for the *u.c.e.* family

$$\mathcal{H}_e = \bigcup_{y \in \omega} \mathcal{G}_{s(e,y), s(e,y-1)}.$$

If $\neg P(e)$, then $(\forall y)(r(x, y) \in \text{COF})$. For each $n \in \omega$ there is a y such that $s(e, y-1) \leq n \leq s(e, y)$. $F_n \in \mathcal{G}_{s(e,y+1), s(e,y)}$, hence $\mathcal{F} \subseteq \mathcal{H}_e$. Recalling that $\mathcal{F}$ is not TxtEx-learnable, we conclude that $\mathcal{H}_e$ is not TxtEx-learnable.

If $P(e)$, then $(\forall^\infty y)(r(e, y) \in \text{COINF})$ and hence $(\forall^\infty y)(s(e, y) \in \text{COINF})$. Pick an n_0 such that $(\forall n \geq n_0)(s(e, n) \in \text{COINF})$. For all $n \geq n_0$ $\mathcal{G}_{s(e,n+1), s(e,n)}$ will consist entirely of finite sets. Furthermore, these sets will, by definition, contain no numbers less than n_0. On the other hand, every set in $\mathcal{G}_{s(e,y+1), s(e,y)}$ for $y \leq n_0$ will contain a number less than or equal to n_0 or be finite. Therefore the whole family is learnable as follows. Let M_0 be a computable function which learns the finite family $\bigcup_{y < n_0 + 1} \mathcal{G}_{s(e,y), s(e,y-1)}$

and let M_1 be a computable function which learns the collection of all finite sets – in other words, a function which interprets the input it receives as a string and outputs a code for the content of that string. Define

$$M(\sigma) = \begin{cases} M_0(\sigma) & n_0 \in \text{content}(\sigma), \\ \\ M_1(\sigma) & n_0 \notin \text{content}(\sigma). \end{cases}$$

If M is fed an enumeration for a set in the family, then either n_0 will eventually appear in the enumeration or it will not. In either case, the learner will eventually settle on a correct code for the set.

We have shown how to reduce an arbitrary Σ^0_4 predicate to EXL and we may conclude that EXL is Σ^0_4-hard.

$\square$

2.3 TxtBC-Learning

To prove the upper bound for BCL, we require a result of interest in its own right – independent of the arithmetic complexity of BCL.

Theorem 2.3.1. *Suppose that $\mathcal{G}$ is a u.c.e. family. Either $\mathcal{G}$ is TxtBC-learnable or, for each computable learner M, there is a Δ^0_2 enumeration of a set in $\mathcal{G}$ that M fails to TxtBC-identify.*

Proof. Fix a *u.c.e.* family $\mathcal{G} = \{G_0, G_1, \dots\}$. We must prove the following disjunction. Either:

1. there is a computable machine which TxtBC-learns $\mathcal{G}$ or

2. for any computable machine, M, either

   (a) there is a Δ^0_2 enumeration for a set $G \in \mathcal{G}$ on which M stabilizes to an incorrect answer or

   (b) there is a Δ^0_2 enumeration for a set $G \in \mathcal{G}$ on which M never stabilizes to codes for a single set.

Assume that statement (2) is false. We may then fix a learner, M, that fails to satisfy statements (2)(a) and (2)(b). We shall demonstrate that, under this assumption, $\mathcal{G}$ is TxtBC-learnable by some machine, i.e. statement (1) is true. To accomplish this, we perform a construction starting from a computable enumeration, $g(0), g(1), \ldots$, of $G \in \mathcal{G}$ uniformly obtained from an enumeration of the family. The construction will follow a strategy designed to produce a Δ^0_2 enumeration witnessing statement (2)(b). Our assumption that these constructions fail will ultimately yield a method we shall use to build a learner for the family.

We construct a Δ^0_2 enumeration, f, in stages. After stage s has completed, the state of the enumeration is a finite partial function, f_s. Let $k_s(0), \ldots, k_s(s)$ denote an increasing reordering of $g(0), \ldots, g(s)$. In addition, we define a restraint function, $r_s(i)$, and a counter, i_s, which monitor the length of the enumeration and the initial segments of f_s on which M exhibits key behavior. Specifically, i_s counts the number of times M appears to have output hypotheses coding distinct sets on f_s, $r_s(i_s)$ is the length of f_s and $r_s(j)$ for $1 < j < i_s$ is the length of the initial segment of f_s on which M outputs the j^{th} hypothesis believed to code a different set. For $1 \leq j < i_s$, define the hypothesis $h^s_j = M(f_s \upharpoonright r_s(j))$ and pick a least witness $x^s_j \in W_{h^s_j,s} \triangle W_{h^s_{j-1},s}$. We will call h^s_j and x^s_j the j^{th} hypothesis and witness chosen at stage s, respectively.

Stage s+1: Let f_s, r_s, i_s, k_s, $x_0^s, \ldots, x_{i_s}^s$ and $h_0^s, \ldots, h_{i_s}^s$ be as obtained from stage s. We shall refer to the preceding collectively as the variables. Let the finite sequence $k_{s+1}(0), \ldots, k_{s+1}(s+1)$ be an increasing reordering of $g(0), \ldots, g(s+1)$. Define a set of strings

$$S(s+1) = \{\alpha : (|\alpha|, y < s) \wedge (\text{content}(\alpha) \subseteq \{k_{s+1}(0), \ldots, k_{s+1}(s+1)\})\}.$$

We must consider two possible types of injury at the beginning of the stage.

First, suppose that $k_{s+1}\restriction(s+1) \neq k_s\restriction(s+1)$. Let $j \leq s$ be the least number such that $k_{s+1}(j) \neq k_s(j)$. Reset the variables to their states at the beginning of stage j (for example, define f_{s+1} to be f_j).

The second type of injury occurs when a witness is found either to be "wrong" or "not least". We call a witness, x_j^s, "wrong" if $x_j^s \notin W_{h_j^s,s+1} \triangle W_{h_{j-1}^s,s+1}$ and "not least" if the tuple $\langle x_j^s, f_s\restriction r_s(j)\rangle$ is not the least member of the set

$$\{\langle y, \alpha\rangle : (f_s\restriction r_s(j-1) \prec \alpha) \wedge (y \in W_{h_j^s,s+1} \triangle W_{M(\alpha),s+1}) \wedge (\alpha \in S(s+1))\},$$

where the set is ordered lexicographically and $\alpha <_{llex} \beta$ if $|\alpha| < |\beta|$ or $|\alpha| = |\beta|$ and α is lexicographically less than β. Let $j \in \mathbb{N}$ be least such that x_j^s is either "wrong" or "not least" and make the following changes to the variables.

1. $i_{s+1} = j$.

2. $r_{s+1}(m) = r_s(m)$ for $m < j$ and undefined for $m \geq j$.

3. $f_{s+1}\restriction r_s(j-1) = f_s\restriction r_s(j-1)$, and $f_{s+1}(x)$ is undefined for $x \geq r_s(j-1)$.

4. Discard $x_j^s, \ldots, x_{i_s}^s$ and $h_j^s, \ldots, h_{i_s}^s$.

Having dealt with all required injury, we proceed to the actions of the stage. In particular, we search for the $<_{llex}$-least pair in the set

$$\{\langle y, \alpha\rangle : (f_{s+1} \prec \alpha) \wedge (y \in W_{h^s_{i_s}, s+1} \triangle W_{M(\alpha), s+1}) \wedge (\alpha \in S(s+1))\}.$$

If a least such pair, $\langle y, \alpha\rangle$, is found, update the variables to reflect the successful search for an extension:

1. Increment i_{s+1}.
2. Extend f_{s+1} to α.
3. Define $r_{s+1}(i_{s+1})$ to equal the length of f_{s+1}.
4. Update $x^{s+1}_j = x^s_j$ and $h^{s+1}_j = h^s_j$ for $j < i_{s+1}$.
5. Define $x^{s+1}_{i_{s+1}} = y$ and $h_{i_{s+1}} = M(\alpha)$.

On the other hand, if no such pair can be found, end the stage with no further changes.

Now suppose that $\lim_{s\to\infty} i_s = \infty$. Then, for any n, the n^{th} hypothesis and witness will be changed at most finitely many times. Therefore, $\lim_{s\to\infty} x^s_n$ and $\lim_{s\to\infty} f_s \restriction n$ exist for every n, in which case our construction has produced an enumeration of G which is Δ^0_2 and on which the hypothesis stream generated by M includes hypotheses that code different sets infinitely often. This is, of course, impossible since, by assumption, M TxtBC-learns $\mathcal{G}$ from Δ^0_2-enumerations. Therefore, $\lim_{s\to\infty} i_s \neq \infty$.

The construction was performed using a computable and uniformly obtained enumeration $g(0), g(1), \ldots$ of G. The purpose of using a computable enumeration was to ensure that $\lim_{s\to\infty} f_s$ was Δ^0_2. Because each pair of extension and witness are chosen in a canonical manner that is independent of the enumeration, any two instances of the construction will eventually select the same pair despite using different enumerations of G. This can be proved inductively. Suppose two different enumerations have produced two finite partial functions that agree on an initial, possibly empty, segment. Take the first point of disagreement. The choices of extension made at the point when the functions disagree cannot both be $<_{llex}$-least, therefore one will change at a subsequent stage.

Define a computable function ψ such that $\psi(\sigma, s) = \tau$, where τ is the partial function that results from performing the construction on an initial segment, σ, of an enumeration after s stages of computation. Let $\hat{M}(\sigma) = M(\psi(\sigma, |\sigma|))$. Fix an arbitrary enumeration $q(0), q(1), \ldots$ of G. Since M TxtBC-learns $\mathcal{G}$ from Δ^0_2-enumerations, there must be a longest partial function, α, that is cofinitely often extended by $\psi(q\restriction s, s)$. For any β such that content$(\beta) \subseteq G$, we have $W_{\hat{M}(\alpha)} = W_{\hat{M}(\alpha\hat{\ }\beta)} = G$. Because G is an arbitrary member of $\mathcal{G}$, $\hat{M}$ succeeds in TxtBC-learning $\mathcal{G}$.

Since we have proved that $\mathcal{G}$ is TxtBC-learnable assuming only that $\mathcal{G}$ is TxtBC-learnable from Δ^0_2-enumerations, we have proved the desired claim.

$\square$

The above result allows us to place a bound on the complexity of the enumerations that must be considered when searching for an enumeration that witnesses a failure of TxtBC-learning. The next result applies Theorem 2.3.1 to obtain an upper bound on the complexity of BCL – the first half of the completeness result for BCL.

Theorem 2.3.2. *BCL has a Σ^0_5 description.*

Proof. Let M be an arbitrary learner and i an index for a set in the family $\mathcal{F} = \{F_0, F_1, \ldots\}$. From a computable function, f, define a sequence of functions, $\{f_s\}_{s\in\mathbb{N}}$, by $f_s(x) = f(s,x)$. Let $\phi(M, f, i)$ be the formula

$$\begin{aligned}(\forall n, s)(W_{M(f_s\restriction n)} = F_i \vee (\exists n' > n)(\exists s' > s)(W_{M(f_{s'}\restriction n')} \neq W_{M(f_s\restriction n)} \\ \wedge (\forall s'' > s')(f_{s''}\restriction n' = f_{s'}\restriction n'))).\end{aligned}$$

In words, $\phi(M, f, i)$ asserts that for any stage, s, and initial segment, $f_s\restriction n$, either the hypothesis $M(f_s\restriction n)$ is correct or there is a later stage and longer initial segment on which the Δ^0_2-enumeration has stabilized and on which M outputs a code for a different set.

Define $\psi(M, f)$ to be the formula

$$(\exists n)(\forall n' > n)(\forall s)(W_{M(f_s\restriction n)} = W_{M(f_s\restriction n')} \vee (\exists s' > s)(f_{s'}\restriction n' \neq f_s\restriction n'))$$

and $\xi(f, i)$ to be

$$\begin{aligned}&(\forall n)(\exists s)(\forall t > s)(f_s(n) = f_t(n)) \\ \wedge&(\forall n, s)(\exists u, t > s)((f_s(n) \in F_{i,u}) \vee (f_s(n) \neq f_t(n))) \\ \wedge&(\forall x, u)(\exists n, s)(\forall t > s)(x \in F_{i,u} \to f_s(n) = f_t(n) \wedge f_s(n) = x).\end{aligned}$$

If $\psi(M, f)$, then there is an initial segment of length n such that for any stage, s, and greater length, n', there are two possibilities. One, the hypotheses M outputs on $f_s\restriction n$ and $f_s\restriction n'$ code the same set. Two, there is a subsequent stage, s', at which the n' length initial segment changes: $f_{s'}\restriction n' \neq f_s\restriction n'$. The formula $\xi(f, i)$ asserts that f_s converges to an enumeration of F_i as s goes to infinity. In particular, $\lim_{s\to\infty} f_s(n)$ exists for all n and $\lim_{s\to\infty} f_s(n) = x$ if and only if $x \in F_i$.

We must prove that the following is Σ^0_5 and equivalent to $e \in \mathrm{BCL}$, where e codes a *u.c.e.* family $\{F_0, F_1, \ldots\}$.

$$(\exists M)(\forall f, i)(\xi(f,i) \to \psi(M,f) \wedge \phi(M,f,i)). \tag{2.2}$$

Observe that $\psi(M,f)$ is Σ^0_3. The formula $\phi(M,f,i)$ universally quantifies over the disjunction of a Π^0_2 formula and a Σ^0_2 formula. Thus, $\phi(M,f,i)$ is Π^0_3. Since $\xi(f,i)$ is the conjunction of three Π^0_3 formulas, $\xi(f,i)$ is Π^0_3. From this, we conclude that

$$\xi(f,i) \to \psi(M,f) \wedge \phi(M,f,i)$$

is Δ^0_4. Consequently, (2.2) is Σ^0_5.

To complete the proof, we must verify that any family that satisfies (2.2) is TxtBC-learnable. If $\xi(f,i)$, then f converges to a Δ^0_2 enumeration of F_i. From $\psi(M,f)$, we have that there is an initial segment of the enumeration given by f such that, on longer initial segments of f, either the output hypotheses code the same set, or the Δ^0_2 enumeration has not yet stabilized. Finally, $\phi(M,f,i)$ states that for any initial segment either the hypothesis output by the learner is correct or the learner will output a later hypothesis that is different on an initial segment of f that has stabilized.

Thus, if (2.2) is true, there is a computable learning machine M such that for any Δ^0_2 enumeration f, M converges to consistent hypotheses on f and, if it has not yet output a correct hypothesis, it will change the content of its hypothesis at a later stage. This is clearly equivalent to TxtBC-learning $\mathcal{F}$ from Δ^0_2-enumerations. By Theorem 2.3.1, TxtBC-learning from Δ^0_2-enumerations is equivalent to TxtBC-learning from arbitrary enumerations for *u.c.e.* families.

□

We present the lower bound in a modular fashion. The construction describes an attempt to diagonalize against every possible learner, which succeeds only if a given Σ^0_5 predicate is false. A single step of the diagonalization is proved as a lemma.

Lemma 2.3.3. *Let $M = M_m$ be a computable learning machine and W_e a c.e. set. There is a family $\mathcal{F}_{m,e}$, uniformly computable in m and e, such that:*

1. *If W_e is coinfinite, then $\mathcal{F}_{m,e}$ is not TxtBC-learnable by M, but the family is TxtBC-learnable.*

2. *If W_e is cofinite, then $\mathcal{F}_{m,e}$ is uniformly TxtBC-learnable in both m and e. Further, $\bigcup_{e \in COF, m \in \mathbb{N}} \mathcal{F}_{m,e}$ is TxtBC-learnable.*

Proof. The construction will be performed in stages. During the stages, steps of a diagonalization process will be attempted, although these steps may not be completed. The diagonalization is against the learner M and a step of the diagonalization will be complete when a string is found on which the learner outputs, as a hypothesis, a code for a set that includes an element not in the content of the enumeration it has been fed. Such an element will be called a speculation. To be explicit, we define a natural number, x, to be a *speculation* of M on input σ if for some $s \in \mathbb{N}$, $x \in W_{M(\sigma),s}$ and $x \notin \text{content}(\sigma)$.

We will build a family $\mathcal{F}_{m,e} = \{A, B_0, B_1, B_2, \ldots\}$. At each step i, the set B_i is initialized with the contents of the set A. A set, C, of speculations will be maintained. We reserve the 0th and 1st columns of each set for markers. If $\langle 0, j\rangle \in B_i$, then B_i is said to have been tagged with j. Every set in the construction will contain $\langle 1, \langle 0, m\rangle\rangle$ and $\langle 1, \langle 1, e\rangle\rangle$ where m is a code for M. The rest of the construction occurs off the

0th and 1st-columns and the 0th-column of A is left empty. We now proceed with the construction of $\mathcal{F}_{m,e}$.

Fix M and W_e.

Stage 0: $C, A, B_0, B_1, \ldots$ are all empty. Enumerate 0 into A. Set $\sigma_0 = 0$.

Stage s: Suppose the first i steps have been completed. By $C, A, B_0, \ldots, B_{i+1}$ we mean those sets in their current state. We are thus in the midst of step $(i+1)$. Let $w_0, w_1, \ldots, w_i$ enumerate the current members of C, where the index reflects the order in which they were chosen. Enumerate into each of the sets $A, B_0, \ldots, B_{i+1}$ those w_j having $j \in W_{e,s}$.

Next, we search for the least speculation, $x \leq s$, of M on input $\sigma_s\hat{\ }\alpha$, for some α with $|\alpha| \leq s$, $\max(\text{content}(\alpha)) \leq s$, and $\text{content}(\alpha) \cap (C \setminus A) = \emptyset$. If no speculation is found, pick the least number neither in $C \setminus A$ nor the marker columns and enumerate this number into B_{i+1}, after which we end the current stage of the construction. If a speculation, x witnessed by a string α, is found, enumerate x into C and enumerate the members of $\{y : y \notin (C \setminus A) \wedge y \leq \max(\text{content}(\alpha))\}$ into A. Enumerate $\langle 0, i+1 \rangle$ into B_{i+1}. From this point on, only W_e is allowed to enumerate anything further into B_{i+1}. Step $i+2$ is now initiated by enumerating every element of A into B_{i+2}. Finally, we set $\sigma_{s+1} = \sigma_s\hat{\ }\alpha\hat{\ }\beta$, where β is an increasing enumeration of $\{y : y \notin (C \setminus A) \wedge y \leq \max(\text{content}(\alpha))\}$. This ends the current stage of the construction.

Observe that $C \setminus A$ are the speculations that, at the current stage, have not been enumerated into A.

For coinfinite W_e there are two possibilities. If infinitely many steps complete, there is a subsequence $\{\tau_s\}_{n\in\mathbb{N}}$ of $\{\sigma_s\}_{n\in\mathbb{N}}$ such that $W_{M(\tau_s)} \neq A$, for each $s \in \mathbb{N}$. Since σ_s and

σ_t are compatible for all $s, t \in \mathbb{N}$, the computable function $f(n) = \sigma_n(n)$ enumerates A. Thus, we have an enumeration for a set in the family on which M fails to converge to the correct set. If only finitely many steps complete, then there is a string σ, equal to σ_s for some $s \in \mathbb{N}$, that has no extension witnessing speculation by M. The content of σ is contained in the last nonempty B_i, and B_i will become a cofinite set. Since M engages in no speculation beyond σ, M must only output codes for finite sets, thus on any enumeration of B_i that begins with the string σ, M fails to TxtBC-learn B_i.

Depending on the outcome of the construction, but independent of W_e, we can define a learning machine N_0 that succeeds in TxtBC-learning $\mathcal{F}_{m,e}$.

Case 1: Suppose infinitely many steps of the construction complete. Define N_0 to be a learner that outputs a code for A on any input string unless the string contains $\langle 0, i\rangle$ for some i, in which case it outputs a code for B_i. N_0 succeeds in TxtBC-learning $\mathcal{F}_{m,e}$.

Case 2: If the j^{th}-step is the last step initiated, define N_0 to be a learner that, on input σ, simulates the construction for A and outputs a code for one of A, $B_0, B_1, \ldots, B_j$. If content$(\sigma) \subseteq A$ and $\langle 0, i\rangle \notin$ content(σ) for any $i < j$, $N_0(\sigma)$ codes A. If $\langle 0, i\rangle \in$ content(σ), $N_0(\sigma)$ is a code for B_i. Otherwise, $N_0(\sigma)$ is a code for B_j and N_0 has TxtBC-learned $\mathcal{F}_{m,e}$.

Next, we define a machine that can learn $\bigcup_{e\in\text{COF}, m\in\mathbb{N}} \mathcal{F}_{m,e}$. Fix $e \in$ COF. To distinguish it from the completed set, let A_s denote a simulation of the construction of A at stage s.

$$N(\sigma) = \begin{cases} 0 & \text{if card}((\{1\} \oplus \mathbb{N}) \cap \text{content}(\sigma)) \leq 1, \\ N_{m,e}(\sigma) & \text{if } \langle 1, \langle 0, m\rangle\rangle, \langle 1, \langle 1, e\rangle\rangle \in \text{content}(\sigma). \end{cases}$$

The $N_{m,e}$ will be defined below. Each $N_{m,e}$ need only TxtBC-learn the family resulting from the construction based on M_m and W_e. For $i \in \mathbb{N}$, let A^*, B_i^* and $(A \cup B_i \setminus (\{0\} \oplus \mathbb{N}))^*$ denote Σ_1^0-codes for A, B_i and $A \cup B_i \setminus (\{0\} \oplus \mathbb{N})$, respectively. These codes can be computably derived from m and e. We define $N_{m,e}$ as follows:

$$N_{m,e}(\sigma) = \begin{cases} B_i^* & \text{if } \langle 0, i\rangle \in \text{content}(\sigma), \\ A^* & \text{if } \langle 0, i\rangle \notin \text{content}(\sigma) \wedge \text{content}(\sigma) \subseteq A_{|\sigma|}, \\ (A \cup B_k \setminus (\{0\} \oplus \mathbb{N}))^* & \text{otherwise,} \end{cases}$$

where k denotes the greatest index of a set that has been used in the simulated construction up to stage $|\sigma|$.

To determine if $N_{m,e}$ TxtBC-learns the family, we must consider four cases, depending on the outcome of the construction and which set, $D \in \mathcal{F}_{m,e}$, is enumerated to $N_{m,e}$.

Case 1: Suppose D has a tag on the 0^{th}-column; in other words, there exists $i \in \mathbb{N}$ such that $D = B_i$. If the construction completes $l < \infty$ steps, then $i < l$. Otherwise, D may be any of the B_i. Once $\langle 0, i\rangle$ has appeared in the enumeration, the learner will hypothesize B_i^* and never change hypothesis.

Case 2: Suppose $D = B_j$ where j is the index of the final, but incomplete, step of the construction. Since no $\langle 0, i\rangle$ will ever be enumerated into B_j, the first case of $N_{m,e}$ will never be satisfied. Cofinitely, since A is a finite set and B_j is not, the second case will not be satisfied either. (A is finite because only finitely many steps of the construction complete.) Thus, cofinitely, the learner will output $(A \cup B_k \setminus (0 \oplus \mathbb{N}))^*$, where k is updated at each stage to reflect the most recent addition to the family during the construction. Eventually k will stabilize to j, after which time the learner's hypotheses will always be correct.

Case 3: Suppose $D = A$, where only finitely many steps complete. Since A is finite, for all but finitely many s, $A_s = A$ and we may replace $A_{|\sigma|}$ with A in the second case of the definition of $N_{m,e}$. Since no tag will ever be enumerated into the 0^{th}-column of A, the first case will never be satisfied and eventually the second case will always be satisfied and $N_{m,e}$ outputs A^* cofinitely.

Case 4: Finally, suppose $D = A$, where infinitely many steps complete. Note that because the learner is receiving an arbitrary enumeration, there need not be any correlation between the enumeration given to the learner and the enumeration of the simulation A_s. It is quite possible that the second case will be true only infinitely often. The first case, however, is never satisfied. All that remains is to prove that eventually the third case only produces correct hypotheses. Since $e \in \mathrm{COF}$, we may choose s such that $[s, \infty) \subseteq W_e$. The set $C \setminus A$ is finite and $(C \setminus A) \cap B_i = \emptyset$. Thus, $A \cup B_i \setminus (0 \oplus \mathbb{N}) = A$ for $i \geq s$.

Thus N succeeds in TxtBC-learning the following, possibly non-*u.c.e.*, family

$$\bigcup_{e \in \mathrm{COF}, m \in \mathbb{N}} \mathcal{F}_{m,e}.$$

and thus can TxtBC-learn any subfamily.

□

Theorem 2.3.4. *BCL is Σ^0_5-hard*

Proof. We wish to reduce an arbitrary Σ^0_5 predicate $P(e)$ to BCL. For an arbitrary Σ^0_4 predicate $Q(e)$, there is a Σ^0_2 predicate, $R(e, x, y)$, such that the following representation can be made:

$$\begin{aligned}Q(e) &\leftrightarrow (\exists a)(\forall b)(R(e,a,b))\\ &\leftrightarrow (\exists\langle a,s\rangle)[((\forall b)(R(e,a,b))) \wedge ((\forall a' < a)(\exists s' \leq s)(\neg R(e,a',s')))\\ &\qquad \wedge ((\exists a' < a)(\forall s' < s)(R(e,a',s')))]\\ &\leftrightarrow (\exists!\langle a,s\rangle)[((\forall b)(R(e,a,b))) \wedge ((\forall a' < a)(\exists s' \leq s)(\neg R(e,a',s')))\\ &\qquad \wedge ((\exists a' < a)(\forall s' < s)(R(e,a',s')))].\end{aligned}$$

Since the predicate

$$((\forall b)(R(e,a,b))) \wedge ((\forall a' < a)(\exists s' \leq s)(\neg R(e,a',s'))) \wedge ((\exists a' < a)(\forall s' < s)(R(e,a',s')))$$

is Π^0_3, for a suitable computable function g,

$$Q(e) \rightarrow (\exists! x)(g(e,x) \in \text{COINF})$$

and

$$\neg Q(e) \rightarrow (\forall x)(g(e,x) \in \text{COF}).$$

Applying the above to $P(e)$, the arbitrary Σ^0_5 predicate under consideration, we may define a computable function f such that

$$\begin{aligned}P(e) \rightarrow (\exists x)[(\forall x' > x)(\forall y)(f(e,x',y) \in \text{COF})\\ \wedge (\forall x' \leq x)(\exists^{\leq 1} y)(f(e,x',y) \in \text{COINF})]\end{aligned}$$

and

$$\neg P(e) \rightarrow (\forall x)[(\exists! y)(f(e,x,y) \in \text{COINF})].$$

We will now define a family $\mathcal{G}_e$ from e such that $\mathcal{G}_e$ will be learnable if and only if $P(e)$. Define

$$\mathcal{G}_e = \bigcup_{x,y \in \mathbb{N}} \mathcal{F}_{x,f(e,x,y)}.$$

Case 1: Suppose $\neg P(e)$. Then for every x, there is a y for which $f(e,x,y) \in$ COINF. From this we conclude that for each computable learner, M coded by m, there is a y such that $f(e,m,y) \in$ COINF. $\mathcal{G}_e$ contains a subfamily, $\mathcal{F}_{m,f(e,m,y)}$, that M cannot TxtBC-learn. Thus, $\mathcal{G}_e$ is not TxtBC-learnable.

Case 2: Suppose $P(e)$ and let x_0 be such that $(\forall x \geq x_0)(\forall y)(f(e,x,y) \in \text{COF})$. Let $a_0, a_1, \ldots, a_k$ enumerate the numbers less than x_0 such that, for unique corresponding $b_0, b_1, \ldots, b_k$, we have $f(e,a_i,b_i) \in$ COINF and let K_i be a computable machine that learns $\mathcal{F}_{a_i,f(e,a,b_i)}$. The existence of such a machine is guaranteed by Lemma 2.3.3. Using the machine N from the proof of Lemma 2.3.3, define a computable machine M on input string σ by

$$M(\sigma) = \begin{cases} K_i(\sigma) & \text{if } \langle 1, \langle 0, a_i \rangle\rangle, \langle 1, \langle 1, b_i \rangle\rangle \in \text{content}(\sigma) \text{ for } i \leq k, \\ N(\sigma) & \text{otherwise.} \end{cases}$$

If an enumeration of a set in the subfamily $\mathcal{F}_{a_i,f(e,a_i,b_i)}$ is fed to M, then eventually a tag in the 1^{st}-column will appear identifying it as such. Cofinitely often, the appropriate K_i will be used to learn the enumeration. If the enumeration is for a set from $\mathcal{F}_{x,f(e,x,y)}$ with either $x \neq a_i$ or $y \neq b_i$ for any $i \leq k$, then N will be used. From Lemma 2.3.3, it is known that N is capable of TxtBC-learning $\mathcal{F}_{x,f(x,y)}$ for any x provided that $y \in$ COF.

We conclude that BCL is Σ^0_5-hard.

$\square$

2.4 TxtEx*-learning

Our final collection of results borrows from the BCL lower bound arguments as well as the EXL description, given in Section 2.3 and Section 2.2, respectively. We begin with a Σ^0_5 description of EXL*.

Theorem 2.4.1. *EXL* has a Σ^0_5 description.*

Proof. Suppose e is a code for a *u.c.e.* family $\{L_0, L_1, \ldots\}$. By Corollary **??**, a family that is TxtEx*-learnable from computable enumerations is TxtEx*-learnable from arbitrary enumerations. Further, observe that if there is a machine, M, that TxtEx*-learns a family, there is a total machine, $\hat{M}$, that TxtEx*-learns the same family. Specifically, define $\hat{M}(\sigma)$ to be $M(\sigma{\restriction}n)$ for the greatest n such that $M(\sigma{\restriction}n)$ converges within $|\sigma|$ computation stages and define $\hat{M}(\sigma) = 0$ if no such initial segment exists. We proceeed with a formula nearly identical to the description of TxtEx-learning. Let $D_0, D_1, \ldots$ be a canonical, computable enumeration of the finite sets. Consider the formula

$$(\exists a)(\forall k, i)(\exists \ell)\Big((M_a \text{ is total}) \wedge (\phi_k \text{ is total}) \wedge (\phi_k \text{ enumerates } L_i) \to \psi(k, a, i, \ell)\Big) \tag{2.3}$$

where we define $\psi(k, a, i, \ell)$ to be

$$(\exists s)(\forall t > s)\Big(M_a(\phi_k{\restriction}t) = M_a(\phi_k{\restriction}s)\Big) \wedge (\forall n)\Big(W_{M_a(\phi_k\restriction n)} \triangle L_i = D_\ell$$
$$\vee\, (\exists m > n)\big(M_a(\phi_k{\restriction}m) \neq M_a(\phi_k{\restriction}n)\big)\Big).$$

The only difference between the above formula and that of Theorem 2.2.1 is an additional existential quantifier over finite sets. Just as before, if a family satisfies formula (2.3) then there is a computable machine that identifies every computable enumeration for a set in the family.

□

Lemma 2.4.2. *Let $M = M_m$ be a computable learning machine and W_e a c.e. set. There is a family $\mathcal{F}_{m,e}$, uniformly computable in m and e, such that:*

1. *If W_e is coinfinite, then $\mathcal{F}_{m,e}$ is not TxtEx*-learnable by M, but the family is TxtEx*-learnable.*

2. *If W_e is cofinite, then $\mathcal{F}_{m,e}$ is uniformly TxtEx*-learnable in both M and e. Further, $\bigcup_{e \in COF, m \in \mathbb{N}} \mathcal{F}_{m,e}$ is TxtBC-learnable.*

Proof. Fix a machine $M = M_m$ and a *c.e.* set W_e. We will construct a family, $\mathcal{F}_{m,e} = \{A, L_1, R_1, L_2, R_2, \ldots\}$ in stages. Each set in $\mathcal{F}_{m,e}$ will have two columns ($\{\langle 0, x\rangle : x \in \mathbb{N}\}$ and $\{\langle 1, x\rangle : x \in \mathbb{N}\}$) reserved for markers. Every set in $\mathcal{F}_{m,e}$ contains $\langle 0, \langle m, 0\rangle\rangle$ and $\langle 0, \langle e, 1\rangle\rangle$ and A contains the marker $\langle 0, \langle 0, 3\rangle\rangle$ as well. Unless otherwise indicated, any action during the construction is performed on the complement of the reserved columns. We identify this complement with $\mathbb{N}$ as it is a computable copy. At any stage of the construction, at most one pair of sets, L_n and R_n, will be actively involved in the construction. When there is such a pair, we call it the active pair and maintain an associated function, r_n, which stores information about that pair.

Stage 0: Search for the least string, σ, on which M outputs a hypothesis. Set σ_0 equal to σ and enumerate content(σ_0) into A.

Stage s+1: Let $\sigma_0 \preceq \ldots \preceq \sigma_s$ be the sequence of strings passed to the current stage from stage s. If there is a currently active pair, L_n and R_n, then for $j \in W_{e,s+1}$, we enumerate $r_n(j)$ and $r_n(j) + 1$ into both L_n and R_n. We then consider four cases depending on the status of two parameters. First, the existence of an active pair of sets. Second, the availability, within computational bounds, of an extension, α, of σ_s on which M outputs a hypothesis different from its most recent hypothesis. We only consider the finite set of strings $S = \{\sigma_s\hat{}\tau : (|\tau| < s+1) \wedge (\text{content}(\tau) \subset s+1)\}$ in our search for α.

Case 1: Suppose there is no active pair, but there is an extension, $\alpha \in S$, of σ_s such that $M(\alpha) \neq M(\sigma_s)$. We pick the least such α. Set $\sigma_{s+1} = \alpha\hat{}\beta$, where β is an increasing enumeration of $\{x : x \leq \max(\text{content}(\alpha))\}$, and enumerate $\text{content}(\sigma_{s+1})$ into A.

Case 2: Next, consider the case where there is neither an active pair nor an $\alpha \in S$ such that $M(\alpha) \neq M(\sigma_s)$. Let $n \in \mathbb{N}$ be least such that L_n and R_n have not yet been used in the construction and set L_n and R_n to be the active pair. Set $\sigma_{s+1} = \sigma_s$ and enumerate $\text{content}(\sigma_{s+1})$ into both L_n and R_n. Pick the least even number, k, such that k and $k+1$ have not appeared in the construction so far. Enumerate k into L_n, $k+1$ into R_n and set $r_n(0) = k$.

Case 3: Let L_n and R_n be the active pair of sets, and suppose $\alpha \in S$ is least such that $M(\alpha) \neq M(\sigma_s)$. Set $\sigma_{s+1} = \alpha\hat{}\beta$, where β is an increasing enumeration of $\{x : x \leq \max(\text{content}(\alpha))\}$, and enumerate $\text{content}(\sigma_{s+1})$ into A. Next, we "cancel" the active pair. Specifically, we enumerate the marker elements $\langle 1, \langle n, 0\rangle\rangle$ and $\langle 1, \langle n, 1\rangle\rangle$ into L_n and R_n, respectively, and mark the pair as inactive.

Case 4: Finally, assume there is a currently active pair, L_n and R_n, but no $\alpha \in S$ such that $M(\alpha) \neq M(\sigma_s)$. Pick the least even number k larger than any number used in the construction so far, enumerate k into L_n, $k+1$ into R_n and set $r_n(i+1) = k$, where i is the greatest value for which $r_n(i)$ is defined.

To verify that the above construction produces a family with the desired properties, we must verify three statements:

1. $\mathcal{F}_{m,e}$ is TxtEx*-learnable for all M and e.

2. If W_e is coinfinite, then M does not TxtEx*-learn $\mathcal{F}_{m,e}$.

3. If W_e is cofinite, then there is a machine, computable from m and e, that TxtEx*-learns $\mathcal{F}_{m,e}$.

If there is a pair of sets that remains active cofinitely, then $\mathcal{F}_{m,e}$ is a finite family. If no such pair exists, then every finite set has a unique marker by which it can be identified and the only infinite set is A. In either case, the family is learnable and we conclude that the first statement is true.

To prove the second statement, we must again consider two cases. Suppose W_e is coinfinite. If a pair of sets remains active cofinitely, then M outputs the same hypothesis on all extensions of a finite partial enumeration whose content is contained in both members of the pair. Thus, there are two enumerations, one for each of L_n and R_n, on which M converges to the same hypothesis. The symmetric difference or L_n and R_n, however, is infinite as one is co-odd and the other co-even. If no pair remains active infinitely, then there must be an infinite number of stages during the construction at which $\sigma_0 \prec \sigma_1 \prec \ldots$ are found such that $M(\sigma_s) \neq M(\sigma_{s+1})$ and A is enumerated by $f(n) = \sigma_n(n)$. In either case, M fails to TxtEx*-learn $\mathcal{F}_{m,e}$.

Finally, we must exhibit a machine that can TxtEx*-learn all possible families $\mathcal{F}_{m,e}$ where $e \in \mathrm{COF}$. In particular, a machine that can TxtEx*-learn the following possibly non-*u.c.e.* family as well as every subfamily:

$$\mathcal{G} = \bigcup_{e \in \mathrm{COF}, m \in \mathbb{N}} \mathcal{F}_{m,e}\,.$$

Since W_e is cofinite for all the families under consideration, observe that $\mathcal{F}_{m,e}$ consists of a (possibly infinite) number of finite sets and either one or two sets (A or a pair L_n and R_n) that are cofinite in the complement of the marker columns. Fix codes a_0, a_1 and $a_{m,e}$ such that $W_{a_0} = \emptyset$, $W_{a_1} = \mathbb{N} \setminus \{\langle x, y\rangle : (x = 0 \vee x = 1) \wedge y \in \mathbb{N}\}$ and $W_{a_{m,e}}$ is the set $A \in \mathcal{F}_{m,e}$. For notational ease, let $C(k) = \{\langle k, x\rangle : x \in \mathbb{N}\}$. Define $N_{m,e}$ by

$$N_{m,e}(\sigma) = \begin{cases} a_0 & \text{if } \mathrm{content}(\sigma) \cap C(1) \neq \emptyset, \\ a_{m,e} & \text{if } \langle 0, \langle 0, 3\rangle\rangle \in \mathrm{content}(\sigma), \\ a_1 & \text{otherwise.} \end{cases}$$

Further, define a machine N by

$$N(\sigma) = \begin{cases} N_{m,e}(\sigma) & \text{if } \langle 0, \langle m, 0\rangle\rangle, \langle 0, \langle e, 1\rangle\rangle \in \mathrm{content}(\sigma), \\ 0 & \text{otherwise.} \end{cases}$$

To prove that N learns $\mathcal{G}$, select an arbitrary $D \in \mathcal{G}$. Let e and m be the codes such that $D \in \mathcal{F}_{m,e}$ for cofinite W_e.

Case 1: Suppose that, during the construction of $\mathcal{F}_{m,e}$, no pair of sets remains active infinitely. In this case, every member of $\mathcal{F}_{m,e}$ is marked, either with a marker in $C(1)$ or with $\langle 0, \langle 0, 3\rangle\rangle$. Thus, N succeeds in TxtEx*-learning $\mathcal{F}_{m,e}$.

Case 2: Suppose, on the other hand, a pair of sets remains active cofinitely during the construction. Let L_n and R_n be that unique pair of sets. If $D = A$, then $\langle 0, \langle 0, 3\rangle\rangle \in D$ and cofinitely often $N_{m,e}$ hypothesizes $a_{m,e}$. Every other finite set contains a unique marker and is hence TxtEx*-learnable by $N_{m,e}$. Finally, if $D = L_n$ or R_n then $D =^* \mathbb{N} \setminus (C(0) \cup C(1))$. No initial segment of any enumeration of D contains either a marker in $C(1)$ or the marker $\langle 0, \langle 0, 3\rangle\rangle$. Thus, $N_{m,e}$ again succeeds in TxtEx*-learning the set.

$\square$

Theorem 2.4.3. *EXL* is Σ^0_5-hard*

Proof. We wish to reduce an arbitrary Σ^0_5 predicate $P(e)$ to EXL*. Applying the technique from the proof of Theorem 2.3.4 to $P(e)$, the arbitrary Σ^0_5 predicate under consideration, we may define a computable function f such that

$$P(e) \to \exists x[\forall x' > x \forall y(f(e,x',y) \in \text{COF})$$

$$\wedge \forall x' \leq x \exists^{\leq 1} y(f(e,x',y) \in \text{COINF})]$$

$$\neg P(e) \to \forall x[\exists! y(f(e,x,y) \in \text{COINF})]$$

We will now define a family $\mathcal{G}_e$ from e using a computable function f such that $\mathcal{G}_e$ will be learnable if and only if $P(e)$.

$$\mathcal{G}_e = \bigcup_{x,y \in \mathbb{N}} \mathcal{F}_{x,f(e,x,y)}$$

Case 1: Suppose $\neg P(e)$. For every x, there is a y for which $f(e,x,y) \in$ COINF. We conclude that for each computable learner, M coded by m, there is a y such that $f(e,m,y) \in$ COINF. Thus $\mathcal{G}$ contains a subfamily, $\mathcal{F}_{m,f(e,m,y)}$, that M cannot learn and $\mathcal{G}_e$ is not TxtEx*-learnable in this case.

Case 2: Suppose $P(e)$. Let x_0 be such that $(\forall x \geq x_0)(\forall y)(f(e,x,y) \in$ COF). Denote by $a_0, a_1, \ldots, a_k$ the numbers less than x_0 such that, for unique corresponding $b_0, b_1, \ldots, b_k$, $f(e,a_i,b_i) \in$ COINF. Let K_i be a computable machine that learns $\mathcal{F}_{a_i,f(e,a,b_i)}$. On input string σ, define a computable machine M by

$$M(\sigma) = \begin{cases} K_i(\sigma) & \text{if } \langle t_1, \langle 0, a_i\rangle\rangle, \langle t_1, \langle 1, b_i\rangle\rangle \in \text{content}(\sigma) \text{ for } i \leq k \\ N(\sigma) & \text{otherwise} \end{cases}.$$

If an enumeration for a set in the subfamily $\mathcal{F}_{a_i, f(e,a_i,b_i)}$ is being fed to the learner, then eventually a tag in the t_1^{st}-column will appear identifying it as such. Cofinitely often, the appropriate machine will be used to learn such an enumeration. If the enumeration is for a set from $\mathcal{F}_{x,f(e,x,y)}$ with either $x \neq a_i$ or $y \neq b_i$ for any $i \leq k$, then N will be used to attempt learning. By Lemma 2.4.2, N succeeds in TxtEx*-learning $\mathcal{F}_{x,f(x,y)}$ for any x provided that $y \in$ COF.

□

2.5 Additional Results

In the following two subsections, we prove that FEXL and EXL^n, for $n \in \mathbb{N}$, are Σ^0_4-complete.

2.5.1 TxtFex-Learning

Theorem 2.5.1. *FEXL has a Σ^0_4 description.*

Proof. By a result due to Case [5] we need only consider computable enumerations when analyzing the complexity of FEXL. Further, observe that if there is a machine, M, that TxtFex-learns a family, there is a total machine, $\hat{M}$, that TxtFex-learns the same family. Specifically, define $\hat{M}(\sigma)$ to be $M(\sigma\restriction n)$ for the greatest n such that $M(\sigma\restriction n)$ converges within $|\sigma|$ computation stages and define $\hat{M}(\sigma) = 0$ if no such initial segment exists. Suppose e codes a *u.c.e.* family $\{L_0, L_1, \ldots\}$.

We will define a formula which states that there is a learner such that for every enumeration and every set in the family, if the enumeration is total and enumerates the set, then eventually the learner stabilizes to a finite collection of hypotheses and a given hypothesis is either correct or is not a member of the finite collection. Syntactically, this can be stated as follows:

$$(\exists a)(\forall k, i)\Big((M_a \text{ is total}) \wedge (\phi_k \text{ is total}) \wedge (\phi_k \text{ enumerates } L_i) \to \psi(k, a, i)\Big) \tag{2.4}$$

where we define $\psi(k, a, i)$ to be

$$(\exists s, c)(\forall t \geq s)\Big(M_a(\phi_k \restriction t) \in D_c\Big) \wedge (\forall n)\Big(W_{M_a(\phi_k \restriction n)} = L_i \\ \vee (\exists m)(\forall u \geq m)\big(M_a(\phi_k \restriction m) \neq M_a(\phi_k \restriction n)\big)\Big).$$

The last formula is Π^0_3. Thus, (2.4) is Σ^0_4 and characterizes TxtFex-learning because, for any family coded by a number e which satisfies formula (2.4), there is a learner whose hypotheses converge to correct hypotheses on every computable enumeration and if e fails to satisfy formula (2.4), every learner must fail on some hypothesis for some set in the family.

We have, therefore, exhibited a Σ^0_4 description of FEXL.

□

Next, we prove a Σ^0_4 lower bound on the complexity of FEXL, thereby establishing Σ^0_4-completeness. Before exhibiting a reduction, we present an example that serves as the basis for the construction. Using a construction similar to the proof of Theorem 2.2.2, we will use this example to reduce an arbitrary Σ^0_4 predicate to FEXL.

Example 2.5.1. *We build a family, $\mathcal{L}_e = \{L_0, L_1, \ldots\}$, in stages as follows.*

Stage 0: *We begin the construction with all sets in $\mathcal{L}_e$ empty. We set $L_0 = \{e\}$.*

Stage s+1: *We set $L_{s+1} = [e, e+s+1+card(W_{e,s})]$. If $W_{e,s+1} \setminus W_{e,s} \neq \emptyset$, then enumerate into L_i the least number greater than e not already in L_i for each $i \leq s+1$. Thus, $L_i = [e, e+i+card(W_{e,s+1})]$.*

If W_e is finite, then $\mathcal{L}_e$ consists of all intervals of the form $[e, e+s+card(W_e)]$ with $s \in \mathbb{N}$. If W_e is infinite, then $\mathcal{L}_e$ contains only one set, namely $[e, \infty)$. We have exhibited an effective construction for $\mathcal{L}_e = \{[e, e+s+1+card(W_e)] : s \in \mathbb{N}\}$ thereby illustrating that $\mathcal{L}_e$ is u.c.e.

Next, we define $\mathcal{F} = \bigcup_{e\in\mathbb{N}} \mathcal{L}_e$. We will demonstrate, using a complexity calculation, that $\mathcal{F}$ is not TxtFex-learnable. Suppose M TxtFex-learns $\mathcal{F}$. Without loss of generality, we may assume M is defined on all initial segments of enumerations of sets in $\mathcal{F}$. If $e \in$ INF, then

$$(\exists k)(\forall n)\Big(M\big(e\ldots(e+k+n)\big) \in \Big\{M\big(e\big), \ldots, M\big(e\ldots(e+k)\big)\Big\}\Big),$$

and if $e \in$ FIN, then

$$(\forall k)(\exists n)\Big(M\big(e\ldots(e+k+n)\big) \notin \Big\{M\big(e\big), \ldots, M\big(e\ldots(e+k)\big)\Big\}\Big).$$

We have, therefore, produced a Σ^0_2 description of INF. As no such description exists, no machine, M, that TxtFex-learns $\mathcal{F}$ exists.

In the proof that follows, $\mathcal{L}_x$ and $\mathcal{F}$ will be defined as in Example 2.5.1. The proof is based on ideas that arose during correspondence with Frank Stephan.

Theorem 2.5.2 (Beros and Stephan)**.** *FEXL is Σ^0_4-hard.*

Proof. We fix an arbitrary Σ^0_4 predicate, P. We choose f, a computable, one-to-one function of two variables such that $\{f(a,n)\}_{n\in\mathbb{N}}$ is an increasing sequence for all $a \in \mathbb{N}$, $P(e) \to (\forall^\infty x)[f(e,x) \in \text{COINF}]$ and $\neg P(e) \to (\forall x)[f(e,x) \in \text{COF}]$.

Fix $x, y, z \in \mathbb{N}$. We construct a family $\mathcal{R}^z_{x,y} = \{R_0, R_1, \ldots\}$ in stages.

Stage s+1: For each $i \leq s+1$, let k_i be the length of the longest interval contained in $W_{z,s}$ that contains i. We denote by ℓ_i the greatest element of the y^{th} member of $\mathcal{L}_x$ computed to k_i stages. If $\ell_i \geq k_i$, enumerate the contents of the interval $[x, x+k_i]$ into R_i. If $\ell_i < k_i$, enumerate the contents of the interval $[x, x+\ell_i]$ into R_i.

Based upon $\mathcal{R}^z_{x,y}$, we define the following families.

$$\mathcal{G}_{e,n} = \bigcup\{\mathcal{R}^{f(e,n)}_{i,y} : y \in \mathbb{N} \wedge f(e,n) \leq i < f(e,n+1)\},$$

and

$$\mathcal{H}_e = \bigcup_{n\in\mathbb{N}} \mathcal{G}_{e,n}.$$

If $\neg P(e)$, then $\mathcal{F} \subset \mathcal{G}_e$. Since $\mathcal{F}$ is not TxtFex-learnable, $\mathcal{H}_e$ is not TxtFex-learnable in this case. Now suppose that $P(e)$. Under this assumption, there is a finite list, $x_0, \ldots, x_t$, such that $\mathcal{L}_{x_i} \subset \mathcal{H}_e$ and $x_i \in \text{INF}$. We define a subfamily, $\mathcal{L}'_x$, of $\mathcal{H}_e$ by

$$\mathcal{L}'_x = \{H \in \mathcal{H}_e : \min(H) = x\}$$

For $i \in [0,t]$, the family $\mathcal{L}'_{x_i}$ is finite and there is a machine, N_i, that TxtFex-learns $\mathcal{L}'_{x_i}$. Fix a machine, N, that TxtFex-learns the collection of all finite sets and define a machine, M, as follows.

$$M(\sigma) = \begin{cases} N_i(\sigma) & x_i = \min(\text{content}(\sigma)), \\ N(\sigma) & \text{otherwise.} \end{cases}$$

If an enumeration for a set in $\mathcal{L}_{x_i}$ is given to M in stages, then cofinitely often $\min(\text{content}(\sigma)) = x_i$. Thus, M succeeds in TxtFex-learning $\bigcup_{0\leq i\leq t} \mathcal{L}'_{x_i}$. Every other set in $\mathcal{H}_e$ is finite. For enumerations of every such set, $\min(\text{content}(\sigma)) \neq x_i$ cofinitely often for all $i \leq t$. We conclude that M TxtFex-learns $\mathcal{H}_e$.

The computable map that takes e to a code for $\mathcal{H}_e$ reduces P to FEXL. Since P was an arbitrary Σ^0_4 predicate, we have shown that FEXL is Σ^0_4-hard.

□

Together, Theorems 2.5.1 and 2.5.2 prove that FEXL is Σ^0_4-complete.

2.5.2 TxtEx^n-learning

Theorem 2.5.3. *EXL^n has a Σ^0_4 description.*

Proof. By Theorem 1.2.9 we need only consider computable enumerations when analyzing the complexity of EXL^n. Further, observe that if there is a machine, M, that TxtEx-learns a family, there is a total machine, $\hat{M}$, that TxtEx^n-learns the same family. Specifically, define $\hat{M}(\sigma)$ to be $M(\sigma\restriction n)$ for the greatest n such that $M(\sigma\restriction n)$ converges within $|\sigma|$ computation stages and define $\hat{M}(\sigma) = 0$ if no such initial segment exists. Suppose e codes a *u.c.e.* family $\{L_0, L_1, \ldots\}$.

We will define a formula which states that there is a learner such that for every enumeration and every set in the family, if the enumeration is total and enumerates the set, then eventually the hypotheses stabilize and a given hypothesis is either nearly correct or the hypotheses have not yet stabilized. Syntactically, this can be stated as follows:

$$(\exists a)(\forall k, i)\Big((M_a \text{ is total}) \wedge (\phi_k \text{ is total}) \wedge (\phi_k \text{ enumerates } L_i) \to \psi(k, a, i)\Big) \tag{2.5}$$

where we define $\psi(k, a, i)$ to be

$$(\exists s)(\forall t > s)\Big(M_a(\phi_k{\restriction}t) = M_a(\phi_k{\restriction}s) \wedge \mathrm{card}(\{M_A(\phi_k{\restriction}q) : q \leq s\}) \leq n\Big)$$
$$\wedge(\forall n)\Big(W_{M_a(\phi_k{\restriction}n)} = L_i \vee (\exists m > n)\big(M_a(\phi_k{\restriction}m) \neq M_a(\phi_k{\restriction}n)\big)\Big).$$

The last formula is Δ^0_3. Thus, (2.5) is Σ^0_4 and characterizes TxtEx-learning because, for any family coded by a number e which satisfies formula (2.5), there is a learner whose hypotheses converge to correct hypotheses on every computable enumeration and if e fails to satisfy formula (2.5), every learner must fail on some hypothesis for some set in the family.

We have, therefore, exhibited a Σ^0_4 description of EXL^n.

□

The next theorem is due to Frank Stephan. The proof reduces the decision problem EXL to EXL^n. EXL was proved to be Σ_4-complete in Theorem 2.2.2.

Theorem 2.5.4 (Stephan). *EXL^n is Σ^0_4-hard.*

Proof. Fix an effective eumeration, $\mathcal{G}_0, \mathcal{G}_1, \ldots$, of all *u.c.e.* families. We can define computable function, f and g, such that given e coding a *u.c.e.* family,

$$W_{f(e)} = \{\{\langle x, y\rangle : x \in L \wedge y \in \mathbb{N}\} : L \in \mathcal{G}_e\}$$

and

$$W_{g(e)} = \{\langle x, y\rangle : (\exists^{n+1} z)(\langle x, z\rangle \in W_e)\}$$

A computable machine, M, TxtEx^n-learns $\mathcal{G}_{f(e)}$ if and only if $g \circ M$ TxtEx-learns $\mathcal{G}_{f(e)}$. Note that one direction of this equivalence is obvious. For the other, observe that if $\mathrm{card}(W_a \triangle L) \leq n$ and $(\exists^{n+1} z)(\langle x, z\rangle \in W_a)$, then $x \in L$.

Next, we prove that $\mathcal{G}_{f(e)}$ is TxtEx-learnable if and only if $\mathcal{G}_e$ is TxtEx-learnable. If M TxtEx-learns $\mathcal{G}_{f(e)}$, then define h_0 to be a computable function such that

$$W_{h_0(e)} = \{x : (\exists y)(\langle x, y\rangle \in W_e)\}.$$

The machine $h_0 \circ M$ TxtEx-learns $\mathcal{G}_e$. If M TxtEx-learns $\mathcal{G}_e$, then define a computable function, h_1, such that

$$W_{h_1(e)} = \{\langle x, y\rangle : y \in \mathbb{N} \land x \in W_e\}.$$

Since $h_1 \circ M$ TxtEx-learns $\mathcal{G}_{f(e)}$, we have proved the equivalence.

Thus, $\mathcal{G}_e$ is TxtEx-learnable if and only if $\mathcal{G}_{f(e)}$ is TxtEx^n-learnable. Fix an arbitrary Σ^0_4 predicate, P, and let g be a computable function reducing P to EXL. The existence of such a g is proved by Theorem 2.2.2. The function $f \circ g$ witnesses the reduction of P to EXL^n.

□

2.6 Conclusion

In summary, we have proved Σ^0_3-completeness for FINL, Σ^0_4-completeness for EXL and Σ^0_5-completeness for both BCL and EXL*. Together with Frank Stephan, we have obtained Σ^0_4-completeness results for EXL^n and FEXL. Numerous other learning criteria are known, but their arithmetic complexities remain to be determined, including the complexity for the function learning criteria corresponding to the language learning criteria already considered.

One question stemming from the above work is to ask if there are natural classes other than *u.c.e.* families for which these complexity questions can be answered. Any candidate would have to provide a framework within which an upper bound could be placed on complexity. If more general classes are considered, observe that complexity can only increase, whereas for more restrictive classes complexity can only decrease.

Chapter 3

Anomalous Vacillatory Learning

Inspection of the definitions reveals that TxtFext^i_j is a weaker learning criterion than TxtFex^i_j in the sense that every TxtFext^i_j-learnable family is also TxtFex^i_j-learnable, i.e. $\text{TxtFext}^i_j \subseteq \text{TxtFex}^i_j$.

The following two theorems tantalizingly hinted that TxtFext^*_* might be equivalent to TxtFex^*_*. As we shall see, this is not the case.

Theorem 3.0.1 (Fulk, Jain, Osherson). *$(\forall i, j \in \mathbb{N})(TxtFex^i_j \subseteq TxtFext^{ci}_j)$, where c depends only on j.*

Theorem 3.0.2 (Fulk, Jain, Osherson). *$(\forall i \in \mathbb{N})(TxtFex^i_* \subseteq TxtFext^*_*)$.*

For proofs of these theorems, see Fulk, Jain, and Osherson [6].

In our concluding remarks, we shall make use of Theorem 3.0.2 together with our own Theorem 3.1.1 to describe an interesting relationship between the two notions of anomalous vacillatory learning.

3.1 $\text{TxtFex}^*_2 \neq \text{TxtFext}^*_*$

We begin with a heuristic overview of the diagonalization process. Intuitively, we are searching for a string, σ, on which the learner commits to hypothesizing a finite number of different codes for the same set on all extensions of σ. Such a string may not exist, but the construction will be such that if no string can be found, then the family under construction will include a set, on some enumeration of which, the machine never commits to output only hypotheses that code a single set. On the other hand, if σ does exist, the construction will produce two sets in the family that contain content(σ) and have infinite symmetric difference.

We treat each step of the diagonalization as indexed by e and consider the learner, M_e. That step of the diagonalization will produce a family, $\mathcal{L}_e$, that M_e cannot TxtFext$^*_*$-learn.

Theorem 3.1.1. *There is a u.c.e. family, $\mathcal{L}$, that is TxtFex$^*_2$-learnable, but is not TxtFext$^*_*$-learnable.*

Proof. Fix a learner, M_e. We begin by describing what is needed to prevent M_e from TxtFext$^*_*$-learning. The result of this step will be a family, $\mathcal{L}_e$. Let $L_e = \{e, e+1, \ldots\}$. Depending on the course of the construction, L_e may or may not be included in $\mathcal{L}_e$. Motivated by our interest in strings on which M_e has committed to a finite collection of hypotheses, we make the following definition.

Definition 3.1.2. *A string σ is said to be an (e,k)-stabilizing sequence if and only if the following conditions are met for all $\tau \succeq \sigma$ such that content$(\tau) \subseteq L_e$ and for all $t \in \mathbb{N}$:*

1. $\{e, e+1, \ldots, e+k\} \subseteq$ *content*(σ)
2. $M_e(\tau) \leq |\sigma|$
3. $W_{M_e(\sigma),|\sigma|+t} \cap [0,k) = W_{M_e(\tau),|\sigma|+t} \cap [0,k)$.

In essence, Definition 3.1.2 describes strings that adhere to a certain form, that define cones in $\{\tau : \text{content}(\tau) \subseteq L_e\}$ on which M_e outputs no new hypotheses, and on extensions of which M_e outputs hypotheses for sets that are equal. Since this last claim cannot be verified in the limit (it is a Π^0_2 predicate), the above definition describes a finite approximation.

The predicate "σ is not an (e,k)-stabilizing sequence" is Σ^0_1 as it requires only a witnessing string and natural number to verify. Thus, we may define a sequence of strings that converges in the limit to an $(e,0)$-stabilizing sequence, $\sigma_{e,0}$, if such a string exists. Extending this strategy, we will construct $\sigma_{e,n,s}$ for all $n, s \in \mathbb{N}$, such that

- $\sigma_{e,n,s} \preceq \sigma_{e,n+1,s}$ for all $n, s \in \mathbb{N}$, if both strings are defined.
- $\sigma_{e,0,0}, \sigma_{e,0,1}, \ldots$ converges to an $(e,0)$-stabilizing sequence, if one exists.
- If $\sigma_{e,k,0}, \sigma_{e,k,1}, \ldots$ converges to a string $\sigma_{e,k}$ for all $k < n$, then $\sigma_{e,n,0}, \sigma_{e,n,1}, \ldots$ converges to an (e,n)-stabilizing sequence, $\sigma_{e,n}$, that extends $\sigma_{e,k}$ for all $k < n$, if such a $\sigma_{e,n}$ exists.

Before constructing $\sigma_{e,n,s}$, we introduce some notation. First, define the following finite set of strings.

$$A(\sigma, s) = \{\tau : (\text{content}(\tau) \subseteq L_e) \wedge (\max(\text{content}(\tau)) \leq s) \wedge (|\tau| \leq s) \wedge (\tau \succeq \sigma)\}$$

Next, let $Q(e, n, \sigma, s)$ be the computable predicate "there is no string in $A(\sigma, s)$ and natural number less than or equal to s witnessing that σ is not an (e,n)-stabilizing sequence". Last, fix a symbol, ?, which will be used to indicate that a string is undefined. We now give an effective algorithm for computing $\sigma_{e,n,s}$.

Stage 0: At this stage, no strings have yet been defined. We set $\sigma_{e,0,0}$ to be the empty string.

Stage s+1: We set $\sigma_{e,s+1,i} = \ ?$ for $i \leq s$. We perform the following actions for each n, starting with $n = 0$, up to $n = s + 1$.

1. If $\sigma_{e,n,s} \neq \ ?$, $\sigma_{e,i,s+1} \neq \ ?$ for all $i < n$, and $Q(e, n, \sigma_{e,n,s}, s + 1)$, then we set $\sigma_{e,n,s+1} = \sigma_{e,n,s}$.

2. If $\sigma_{e,i,s+1} \neq \ ?$ for all $i < n$, but the above case is not satisfied, then we consider a string $\tau \in A(\sigma_{e,n-1,s+1}, s+1)$ (where we replace $\sigma_{e,n-1,s+1}$ with the empty string if $n = 0$) which is least such that $Q(e, n, \tau, s+1)$. Such a τ is certain to exist as there are strings in $A(\sigma_{e,n-1,s+1}, s + 1)$ which have no extensions in $A(\sigma_{e,n-1,s+1}, s + 1)$. Having selected τ, we set $\sigma_{e,n,s+1} = \tau$ and $\sigma_{e,k,s+1} = \ ?$ for all $n < k \leq s + 1$. We now terminate the process.

Once the process above terminates, either when Case (2) above occurs or $n = s + 1$, we end the stage of the construction.

Observe that $\{\sigma_{e,n,s}\}_{s\in\mathbb{N}}$ converges if and only if $\{\sigma_{e,k,s}\}_{s\in\mathbb{N}}$ converges for $k < n$ and there is an (e, n)-stabilizing sequence extending the string to which $\{\sigma_{e,n-1,s}\}_{s\in\mathbb{N}}$ converges. Furthermore, if $\{\sigma_{e,n,s}\}_{s\in\mathbb{N}}$ converges, it converges to such an (e, n)-stabilizing sequence.

Define $a_{e,\ell}$ to be the least even number greater than $e + \ell + 1$ such that $\sigma_{e,h,s} = \sigma_{e,h,s+1} \neq \ ?$ for all $h \leq \ell$ and $s \geq a_{e,\ell}$, if it exists. Let $b_{e,\ell} = a_{e,\ell} + 1$. These numbers will allow us to monitor the convergence of the sequences, $\{\sigma_{e,\ell,s}\}_{s\in\mathbb{N}}$, and control the construction as appropriate. If $\{\sigma_{e,k,s}\}_{s\in\mathbb{N}}$ does not converge for some $k \in \mathbb{N}$, then $a_{e,\ell}$ will be undefined for $\ell \geq k$.

Define two sets

$$R_e = \{x \in L_e : \forall \ell (x \neq a_{e,\ell})\}$$
$$\hat{R}_e = \{x \in L_e : \forall \ell (x \neq b_{e,\ell})\}.$$

Observe that R_e is *c.e.* Because $a_{e,0} < a_{e,1} < \dots$ and $a_{e,\ell} \geq \ell$, we see that $x \in R_e$ if and only if $x \neq a_{e,\ell}$ for all $\ell \leq x$. Although $a_{e,\ell}$ is not computable, $x \neq a_{e,\ell}$ is Σ^0_1.

$$\begin{aligned} x \neq a_{e,\ell} \leftrightarrow & (x \text{ is odd}) \vee (x \leq e + \ell + 1) \vee (\sigma_{e,\ell,x} = \ ?) \\ & \vee (\sigma_{e,\ell,x-1} = \sigma_{e,\ell,x}) \vee (\exists s \geq x)(\sigma_{e,\ell,s} \neq \sigma_{e,\ell,x}). \end{aligned}$$

Thus, $x \neq a_{e,\ell}$ is Σ^0_1 and $x \in R_e$ is a finite conjunction of Σ^0_1 statements. Similarly, substituting $b_{e,\ell}$ for $a_{e,\ell}$, we see that $\hat{R}_e$ is also *c.e.* We are now in a position to define $\mathcal{L}_e$. Recall that $D_0, D_1, \dots$ enumerates all finite sets.

$$\mathcal{L}_e = \{R_e \cup (D_n \cap [e, \infty)) : n \in \mathbb{N}\} \cup \{\hat{R}_e \cup (D_n \cap [e, \infty)) : n \in \mathbb{N}\}$$

We now return to M_e, the learner against which we are currently diagonalizing. We must prove that M_e is incapable of TxtFext^*_*-learning $\mathcal{L}_e$.

Case 1: Suppose there is a minimal $\ell \neq 0$ such that $\sigma_{e,\ell}$ is undefined. By definition, this implies there is no σ extending $\sigma_{e,\ell-1}$ such that $e + \ell \in \text{content}(\sigma) \subset L_e$ and $W_{M_e(\sigma),|\sigma|+s} \cap [0, \ell) = W_{M_e(\tau),|\sigma|+s} \cap [0, \ell)$, for all τ such that $\sigma \prec \tau$ with $\text{content}(\tau) \subset L_e$ and for all $s \in \mathbb{N}$. Furthermore, since $\sigma_{e,\ell}$ is undefined, $\sigma_{e,i}$ is undefined for all $i \geq \ell$. Consequently, $a_{e,i}$ is undefined for all $i \geq \ell$ and both R_e and $\hat{R}_e$ are cofinite subsets of L_e. For a suitable finite set, D_n, we have $R_e \cup D_n = L_e$, and hence, $L_e \in \mathcal{L}_e$. Since no σ exists on which the content of the hypothesis stabilizes, we can repeatedly select extensions on which M_e outputs hypotheses coding distinct sets and inductively build an enumeration of L_e on which M_e infinitely often outputs codes for at least two sets that are not equal. If $\ell = 0$, there is the additional possibility that M_e cannot be made to select a finite collection of hypotheses and restrict its output to that finite list. The machine has failed to TxtFext^*_*-learn $\mathcal{L}_e$.

Case 2: Suppose that $\sigma_{e,\ell}$ is defined for all ℓ. Both R_e and $\hat{R}_e$ are coinfinite sets and have infinite symmetric difference. By the definition of $\sigma_{e,0}$, for any τ such that $\sigma_{e,o} \prec \tau$ and $\text{content}(\tau) \subset L_e$, $M_e(\tau) \leq |\sigma_{e,0}|$. We may therefore define a finite list, $h_0, h_1, \ldots, h_n$, of all distinct hypotheses that M_e outputs on extensions of $\sigma_{e,0}$. Pick ℓ sufficiently large so that, for each $i, j \leq n$ for which $W_{h_i} \neq W_{h_j}$, there is an $x \in W_{h_i} \triangle W_{h_j}$ such that $x < \ell$.

All hypotheses made by M_e on extensions of $\sigma_{e,\ell}$ contained in L_e must have the same intersection with $[0, \ell - 1]$ as $W_{M_e(\sigma_{e,\ell})}$. By the choice of ℓ, all such hypotheses must code the same set, yet $\mathcal{L}_e$ contains two sets that extend $\sigma_{e,\ell}$ and have infinite symmetric difference: $\text{content}(\sigma_{e,\ell}) \cup R_e$ and $\text{content}(\sigma_{e,\ell}) \cup \hat{R}_e$. Again, we witness failure by M_e to TxtFext^*_*-learn $\mathcal{L}_e$.

For each $e \in \mathbb{N}$, we have shown that $\mathcal{L}_e$ is not TxtFext^*_*-learnable by M_e. Consequently, $\mathcal{L} = \bigcup_{e \in \mathbb{N}} \mathcal{L}_e$ is not TxtFext^*_*-learnable. We must now verify that $\mathcal{L}$ is indeed TxtFex^*_2-learnable.

Every set in $\mathcal{L}$ is a finite variant of R_e or $\hat{R}_e$ for some $e \in \mathbb{N}$. Therefore, a learner need only identify the appropriate e and determine to which of R_e and $\hat{R}_e$ the input enumeration is most similar. Since R_e is co-even and $\hat{R}_e$ is co-odd, they are identifiable by the numbers not in them. Let x_e and $\hat{x}_e$ be codes for R_e and $\hat{R}_e$, respectively. For notational ease, let $m_\sigma = \min(\text{content}(\sigma))$ and $n_\sigma = \min(\{y > m_\sigma : y \notin \text{content}(\sigma)\})$.

Given σ, an intial segment of an enumeration for a set in $\mathcal{L}$, m_σ is the current guess at the least member of the set (hence the e for which the set is in $\mathcal{L}_e$) and n_σ is the current guess at the least element of L_e not in the set. Define a machine M as follows:

$$M(\sigma) = \begin{cases} x_e & \text{if } e = m_\sigma \wedge (n_\sigma \text{ is even}), \\ \hat{x}_e & \text{if } e = m_\sigma \wedge (n_\sigma \text{ is odd}), \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that M is receiving an enumeration for $L \in \mathcal{L}$. Every set in $\mathcal{L}$ is either of the form $R_e \cup D_n$ or $\hat{R}_e \cup D_n$, for some $e, n \in \mathbb{N}$. By the symmetric relationship between R_e and $\hat{R}_e$, we may assume that $L = R_e \cup D_n$ for some specific e and n. We must consider two cases: R_e is either cofinite or coinfinite.

If R_e is cofinite, $\hat{R}_e$ is also cofinite. As a consequence, $R_e =^* \hat{R}_e$. Eventually, the enumeration will exhibit the least element of the set being enumerated. After that stage, M will either output x_e or $\hat{x}_e$. Given the model of learning, both are correct hypotheses. If R_e is coinfinite, then $L_e \setminus R_e$ is an unbounded set of even numbers. The target set is a finite variant of R_e. Hence, the least element not in content(σ) and greater than e will be even for cofinitely many initial segments of any enumeration. In other words, for cofinitely many initial segments, σ, of any enumeration of L, n_σ is even and $M(\sigma) = x_e$, a code for a finite variant of the enumerated set.

We have constructed a family $\mathcal{L}$ such that, for each computable machine, $\mathcal{L}$ contains a subfamily that the machine cannot TxtFext$^*_*$-learn, and we have exhibited a specific machine that TxtFex$^*_2$-learns the whole family. This completes the proof.

$\square$

3.2 Conclusion

Recall the statement of Theorem 3.0.2 from the introduction:

$$(\forall j)(\mathrm{TxtFex}_*^j \subseteq \mathrm{TxtFext}_*^*).$$

Combining this with Theorem 3.1.1, we observe the following intriguing relationship between the anomalous versions of the two learning criteria

$$(\forall j)(\mathrm{TxtFex}_*^j \subseteq \mathrm{TxtFext}_*^* \subsetneq \mathrm{TxtFex}_*^*).$$

Other results on vacillatory learning can be found in a Case's paper [5] and in Osherson, Stob and Weinstein's book [8].

Bibliography

[1] Dana Angluin. Inductive inference of formal languages from positive data. *Theory of Algorithm and Programs*, 45:117–135, 1980.

[2] Janis Bārzdiņš. Two theorems on the limit synthesis of functions. *Theory of Algorithm and Programs*, 1:82–88, 1974.

[3] Janis Bārzdiņš and Rūsiņš Freivalds. Prediction of general recursive functions. *Doklady Akademii Nauk SSSR*, 206:521–524, 1972.

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

[5] John Case. The power of vacillation. *SIAM Journal of Computation*, 49(6):1941–1969, 1999.

[6] Mark Fulk, Sanjay Jain, and Daniel Osherson. Open problems in "systems that learn". *Journal of Computer and System Sciences*, 49:589–604, 1994.

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

[8] Daniel Osherson, Michael Stob, and Scott Weinstein. *Systems That Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists.* MIT Press, Cambridge, MA, 1986.

[9] Daniel Osherson and Scott Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.

[10] Robert I. Soare. *Recursively enumerable sets and degrees: a study of computable functions and computably generated sets*. Springer-Verlag, 1987.