Learning without Coding

Samuel E. Moelius III[1] and Sandra Zilles[2]

[1] IDA Center for Computing Sciences
17100 Science Drive, Bowie, MD 20715-4300
semoeli@super.org
[2] Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
zilles@cs.uregina.ca

Abstract. Iterative learning is a model of language learning from positive data, due to Wiehagen. When compared to a learner in Gold's original model of language learning from positive data, an iterative learner can be thought of as *memory-limited*. However, an iterative learner can memorize *some* input elements by *coding* them into the syntax of its hypotheses. A main concern of this paper is: to what extent are such coding tricks *necessary*?

One means of preventing *some* such coding tricks is to require that the hypothesis space used be free of redundancy, i.e., that it be 1-1. By extending a result of Lange & Zeugmann, we show that many interesting and non-trivial classes of languages can be iteratively identified in this manner. On the other hand, we show that there exists a class of languages that can*not* be iteratively identified using any 1-1 effective numbering as the hypothesis space.

We also consider an iterative-like learning model in which the computational component of the learner is modeled as an *enumeration operator*, as opposed to a partial computable function. In this new model, there are no hypotheses, and, thus, no syntax in which the learner can encode what elements it has or has not yet seen. We show that there exists a class of languages that *can* be identified under this new model, but that can*not* be iteratively identified. On the other hand, we show that there exists a class of languages that can*not* be identified under this new model, but that *can* be iteratively identified using a Friedberg numbering as the hypothesis space.

Keywords: coding tricks, inductive inference, iterative learning.

1 Introduction

Iterative learning (**It**-learning, Definition 1(a)) is a model of language learning from positive data, due to Wiehagen [Wie76]. Like many models based on positive data, the **It**-learning model involves a learner that is repeatedly fed elements drawn from $\{\#\}$ and from some unknown target language $L \subseteq \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, $\{0, 1, 2, ...\}$.[3] After being fed each such element, the learner outputs a hypothesis (provided that the learner does not diverge). The learner is said to *identify* the target language L iff there is some point from whence on the learner outputs only one hypothesis, and that hypothesis corresponds to L. Furthermore, the learner is said to identify a *class* of languages $\mathcal{L}$ iff the learner identifies *each* $L \in \mathcal{L}$ when fed the elements of L (and possibly #).

In the **It**-learning model, the learner itself is modeled as a triple.

- The first element of the triple is a two-place partial computable function, whose arguments are, respectively, the learner's most recently output hypothesis, and the next input element.
- The second element of the triple is a preliminary hypothesis, i.e., the hypothesis output by the learner before being fed any input.
- The third element of the triple is a *hypothesis space*. The *hypothesis space* determines the language that corresponds to each of the learner's hypotheses. Formally, a hypothesis space is a numbering $(X_j)_{j\in\mathbb{N}}$ of some collection of subsets of $\mathbb{N}$, and that is *effective* in the sense that the two-place predicate $\lambda j, x \centerdot [x \in X_j]$ is partial computable.[4]

It-learning is a special case of Gold's original model of language learning from positive data [Gol67]. In Gold's original model, the learner is provided access to all previously seen input elements, in addition to the next input element. In this sense, a learner in Gold's model can be thought of as *memorizing* all previously seen input elements. When compared to learners in Gold's model, iterative learners are restricted in terms of the classes of languages that they can identify.[5] In this sense, the *memory-limited* aspect of iterative learners is a *true* restriction, and *not* a mere superficial difference in definitions.

This does not however mean that iterative learners are *memory-less*. In particular, an iterative learner can memorize *some* input elements by employing *coding tricks*, which we define (informally) as follows.

- A *coding trick* is any use by an iterative learner of the syntax of a hypothesis to determine what elements that learner has or has not yet seen.

The following is an example. Suppose that an iterative learner $(M, p, (X_j)_{j\in\mathbb{N}})$ identifies a class of languages $\mathcal{L}$. Further suppose that one desires a learner that identifies the class $\mathcal{L}'$, where

$$\mathcal{L}' = \mathcal{L} \cup \{L \cup \{0\} \mid L \in \mathcal{L}\}. \tag{1}$$

Such a learner $(M', p', (Y_k)_{k\in\mathbb{N}})$ may be obtained as follows. Let $(Y_k)_{k\in\mathbb{N}}$ be such that, for each j:

[3] The symbol '#' is pronounced "pause". The inclusion of # in the model allows the target language L to be empty, i.e., in such a case, the learner is repeatedly fed #.

[4] *Not-necessarily-effective* hypothesis spaces have also been considered [dBY10]. However, such hypothesis spaces are not needed herein. For the remainder, we use the terms *hypothesis space* and *effective numbering* interchangeably.

[5] Many variants of the **It**-learning model have been considered, and have also been shown to be restricted in this sense [LZ96,CCJS07,JLMZ10].

$$Y_{2j} = X_j; \qquad\qquad Y_{2j+1} = X_j \cup \{0\}.$$

Then, let M' be such that, for each $x \in (\mathbb{N} \cup \{\#\}) - \{0\}$:

$$\begin{aligned} M'(2j, \quad x) &= 2M(j,x); & M'(2j, \quad 0) &= 2M(j,0)+1; \\ M'(2j+1,x) &= 2M(j,x)+1; & M'(2j+1,0) &= 2M(j,0)+1. \end{aligned}$$

It is easily seen that $(M', 2p, (Y_k)_{k\in\mathbb{N}})$ iteratively identifies $\mathcal{L}'$. Intuitively, M' simulates M, while using the least-significant bit of each hypothesis to *encode* whether or not M' has seen a 0 (e.g., M' switches from an even to an odd hypothesis when it sees a 0). Further note that, if $\mathcal{L}$ already contains languages for which 0 is a member, then there is *redundancy* in the hypothesis space $(Y_k)_{k\in\mathbb{N}}$. In particular, if $0 \in X_j$, then $Y_{2j} = Y_{2j+1}$. For such hypotheses, the least-significant bit affects *only* their syntax, and *not* their semantics.

This example demonstrates how coding tricks can at least *facilitate* the identification of a class of languages. A main concern of this paper is: to what extent are such coding tricks *necessary*?

One approach to preventing *some* such coding tricks is to require that the hypothesis space be free of redundancy, i.e., that it be 1-1. One means of doing this is to require that the hypothesis space be a *Friedberg numbering* [Fri58,Kum90]. A *Friedberg numbering* is a 1-1 effective numbering of all computably enumerable (ce) subsets of $\mathbb{N}$. The use of such numberings as hypothesis spaces was considered by Jain & Stephan [JS08].[6] They observed, for example, that $\mathcal{F}in$, the collection of all finite subsets of $\mathbb{N}$, can*not* be iteratively identified using a Friedberg numbering as the hypothesis space [JS08, Remark 28]. For the remainder, to **FrIt**-*identify* a class of languages $\mathcal{L}$ shall mean to iteratively identify $\mathcal{L}$ using a Friedberg numbering as the hypothesis space (see Definition 1(b)).

Our first main result is to show that, despite this observation of Jain & Stephan, many interesting and non-trivial classes can be **FrIt**-identified. More specifically, we extend a result of Lange & Zeugmann [LZ96, Theorem 12] by showing that, for each class $\mathcal{L}$, if there exists a single hypothesis space witnessing that $\mathcal{L}$ is both uniformly decidable and computably finitely thick (see Definition 3 below), then $\mathcal{L}$ can be **FrIt**-identified (Theorem 6). By comparison, Lange & Zeugmann showed that such a class can be **It**-identified. One significant application of this result is the following. A *pattern language* [Ang80] is a type of language with applications to molecular biology (see, e.g., [SSS$^+$94]). Furthermore, the pattern languages naturally form classes that are **It**-identifiable by Lange & Zeugmann's result,[7] and, thus, are **FrIt**-identifiable, by ours.

As the reader may have already noticed, if one's intent is simply to eliminate redundancy in the hypothesis space, then to require that the hypothesis space be a Friedberg numbering is really overkill. That is because to require that the hypothesis space be a Friedberg numbering is to require that it be free of redundancy *and* that it represent all of the ce sets.

[6] Freivalds, et al. [FKW82] considered the use of Friedberg numberings as hypothesis spaces in the context of *function* learning.

[7] The pattern languages were first shown to be **It**-identifiable by Lange & Wiehagen [LW91].

Thus, we consider a milder variant of **FrIt**-learning, which we call *injective iterative learning* (**InjIt**-learning, Definition 1(c)). In this variant, the hypothesis space is required to be free of redundancy (i.e., be 1-1), but need not represent all of the ce sets.[8] Clearly, for each class $\mathcal{L}$, if $\mathcal{L}$ can be **FrIt**-identified, then $\mathcal{L}$ can be **InjIt**-identified. On the other hand, $\mathcal{F}in$ can be **InjIt**-identified, but, as per Jain & Stephan's observation mentioned above, $\mathcal{F}in$ can*not* be **FrIt**-identified.

Going further, if one's intent is to *prevent coding tricks*, then to require that the hypothesis space be free of redundancy may still be overkill. In particular, one might allow that there be redundancy in the hypothesis space, but require that the learner not benefit from this redundancy. This idea is captured in our next model, which is called *extensional iterative learning* (**ExtIt**-learning, Definition 1(d)).

For a learner to **ExtIt**-identify a class of languages, it is required that, when presented with equivalent hypotheses and identical input elements, the learner must produce equivalent hypotheses. More formally: suppose that $\mathcal{L}$ is a class of languages, that σ_0 and σ_1 are two non-empty sequences of elements drawn from $\{\#\}$ and from two (possibly distinct) languages in $\mathcal{L}$, and that the following conditions are satisfied.

- When fed all but the last elements of σ_0 and σ_1, the learner outputs hypotheses for the same language (though those hypotheses may differ syntactically).
- The last elements of σ_0 and σ_1 are identical.

Then, for the learner to **ExtIt**-identify $\mathcal{L}$, it is required that:

- When fed *all* of σ_0 and σ_1, the learner outputs hypotheses for the same language (though those hypotheses may differ syntactically).

Clearly, if a learner identifies a class of languages using a 1-1 hypothesis space, then that learner satisfies the just above requirement. Thus, every class of languages that can be **InjIt**-identified can be **ExtIt**-identified. On the other hand, we show that there exists a class of languages that *can* be **ExtIt**-identified, but that can*not* be **InjIt**-identified (Theorem 9).

Before introducing our final model, let us recall the definition of an *enumeration operator* [Rog67, §9.7].[9] Let $\mathcal{P}(\mathbb{N})$ be the powerset of $\mathbb{N}$, i.e., the collection of all subsets of $\mathbb{N}$. Let $\langle\cdot,\cdot\rangle$ be any pairing function, i.e., a computable, 1-1, onto function of type $\mathbb{N}^2 \to \mathbb{N}$ [Rog67, page 64]. Let $\hat{\#} = 0$, and, for each $x \in \mathbb{N}$, let $\hat{x} = x + 1$. Let $(D_j)_{j\in\mathbb{N}}$ be any canonical enumeration of $\mathcal{F}in$.

An *enumeration operator* of type $\mathcal{P}(\mathbb{N}) \times (\mathbb{N} \cup \{\#\}) \to \mathcal{P}(\mathbb{N})$ is a mapping that is algorithmic in the following precise sense. To each enumeration operator Θ (of the given type), there corresponds a ce set H, such that, for each $X \subseteq \mathbb{N}$ and $x \in \mathbb{N} \cup \{\#\}$,

$$\Theta(X, x) = \big\{ y \mid \big\langle j, \langle \hat{x}, y\rangle \big\rangle \in H \ \wedge \ D_j \subseteq X \big\}. \tag{2}$$

[8] The use of 1-1 hypothesis spaces was also considered in [BBCJS10] in the context of learning certain *specific* classes of languages.

[9] Herein, we focus on the enumeration operators of a particular type. A more general definition can be found in [MZ10].

Thus, given an *enumeration of* X, and given x, one can enumerate $\Theta(X, x)$ in the following manner.

- Enumerate H. For each element of the form $\langle j, \langle \hat{x}, y\rangle\rangle \in H$, if ever the finite set D_j appears in the enumeration of X, then list y into $\Theta(X, x)$.

Enumeration operators exhibit certain notable properties, including *monotonicity* [Rog67, Theorem 9-XXI]: for each enumeration operator $\mathcal{M}$ of type $\mathcal{P}(\mathbb{N}) \times (\mathbb{N} \cup \{\#\}) \to \mathcal{P}(\mathbb{N})$, each $X, Y \subseteq \mathbb{N}$, and each $x \in \mathbb{N} \cup \{\#\}$,

$$X \subseteq Y \;\Rightarrow\; \mathcal{M}(X, x) \subseteq \mathcal{M}(Y, x). \tag{3}$$

Intuitively, this means that an enumeration operator can tell from its set argument X what elements are in X, but it cannot tell from X what elements are in the *complement of* X.

The final model that we consider is called *iterative learning by enumeration operator* (**EOIt**-learning, Definition 1(e)). As the name suggests, the computational component of the learner is modeled as an enumeration operator, as opposed to a partial computable function. Specifically, the learner is modeled as a *pair*, where:

- The first element of the pair is an enumeration operator of type $\mathcal{P}(\mathbb{N}) \times (\mathbb{N} \cup \{\#\}) \to \mathcal{P}(\mathbb{N})$, whose arguments are, respectively, the learner's most recently output *language*, and the next input element.
- The second element of the pair is the learner's preliminarily output *language*, i.e., the language output by the learner before being fed any input. (We require that this preliminary language be ce.)

Thus, there are no hypotheses in this model. Since there are no hypotheses, there is no *syntax* in which the learner can encode what elements it has or has not yet seen.

The expulsion of hypotheses from the model has an additional consequence, and that is that the success criterion has to be adjusted. Specifically, we say that a learner in this model *identifies* a language L iff when fed the elements of L (and possibly $\#$), there is some point from whence on the learner outputs only the *language* L. The success criterion for identifying a *class* of languages is adjusted similarly. This more liberal approach to language identification, in some sense, gives an advantage to learners in this model. In particular, there exists a class of languages that *can* be **EOIt**-identified, but that can*not* be **It**-identified (Corollary 13).

Interestingly, there also exists a class of languages that can*not* be **EOIt**-identified, but that *can* be **FrIt**-identified (Theorem 14). To help to see why, consider the following two scenarios. First, suppose that $(\mathcal{M}, X)$ is a learner in the enumeration operator model, and that Y is its most recently output language. Then, since $\mathcal{M}$ is an enumeration operator, $\mathcal{M}$ can tell from Y what elements are in Y, but it cannot tell from Y what elements are in the *complement of* Y. Next, consider the analogous situation for a conventional iterative learner. That is, suppose that $(M, p, (X_j)_{j\in\mathbb{N}})$ is such a learner, and that j is its most recently

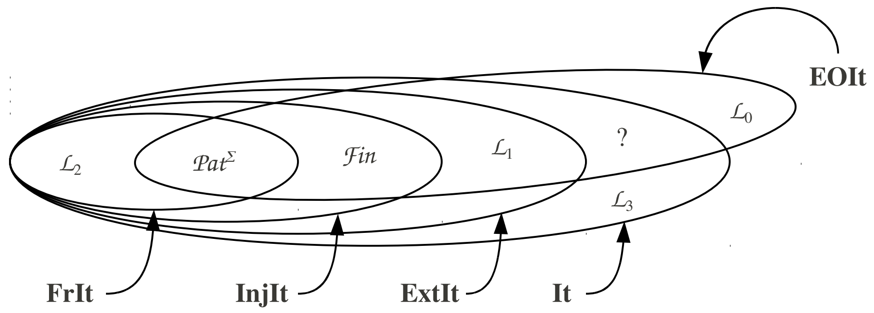

Fig. 1. A summary of main results and open problems. $\mathcal{P}at^{\Sigma}$ is the collection of all pattern languages over Σ, where Σ is an arbitrary alphabet. $\mathcal{F}in$ is the collection of all finite subsets of $\mathbb{N}$. The classes $\mathcal{L}_0$, $\mathcal{L}_1$, and $\mathcal{L}_2$ are defined in the proofs of Theorems 7, 9, and 14, respectively. The existence of the class $\mathcal{L}_3$ was shown by Jain (see Theorem 10).

output hypothesis. Then, in many cases, M *can* tell from j what elements are in the complement of X_j. In this sense, one could say that a hypothesis implicitly encodes *negative* information about the language that it represents. (In fact, this phenomenon can clearly be seen in the full proof of Theorem 14. See [MZ10, Theorem 20].)

A question to then ask is: is this a *coding trick*, i.e., is it the case that *every* learner that operates on hypotheses (as opposed to languages) is employing coding tricks? At present, we do not see a clear answer to this question. Thus, we leave it as a subject for further study.

The main points of the preceding paragraphs are summarized in Figure 1. The remainder of this paper is organized as follows. Section 2 covers preliminaries. Section 3 presents our results concerning uniformly decidable and computably finitely thick classes of languages. Section 4 presents our results concerning Friedberg, injective, and extensional iterative learning (**FrIt**, **InjIt**, and **ExtIt**-learning, respectively). Section 5 presents our results concerning iterative learning by enumeration operator (**EOIt**-learning).

Due to space constraints, only partial proofs are given. Complete proofs of all of our results can be found in the associated tech-report [MZ10].

2 Preliminaries

Computability-theoretic concepts not covered below are treated in [Rog67].

Lowercase math-italic letters (e.g., a, j, x), with or without decorations, range over elements of $\mathbb{N}$, unless stated otherwise. Uppercase italicized letters (e.g., A, J, X), with or without decorations, range over subsets of $\mathbb{N}$, unless stated otherwise. For each non-empty X, $\min X$ denotes the minimum element of X. $\min \emptyset \stackrel{\text{def}}{=} \infty$. For each non-empty, finite X, $\max X$ denotes the maximum element of X. $\max \emptyset \stackrel{\text{def}}{=} -1$. The symbol $\mathcal{L}$, with or without decorations, ranges over subsets of $\mathcal{P}(\mathbb{N})$, unless stated otherwise.

For each x, $\langle x\rangle \overset{\text{def}}{=} x$. For each $x_0, ..., x_{n-1}$, where $n > 2$, $\langle x_0, ..., x_{n-1}\rangle \overset{\text{def}}{=} \big\langle x_0, \langle x_1, ..., x_{n-1}\rangle\big\rangle$.

$\mathbb{N}_{\#} \overset{\text{def}}{=} \mathbb{N} \cup \{\#\}$. A *text* is a total function of type $\mathbb{N} \to \mathbb{N}_{\#}$. For each text t and $i \in \mathbb{N}$, $t[i]$ denotes the initial segment of t of length i. For each text t, $\text{content}(t) \overset{\text{def}}{=} \{t(i) \mid i \in \mathbb{N}\} - \{\#\}$. For each text t and $L \subseteq \mathbb{N}$, t is a text *for* L $\overset{\text{def}}{\Leftrightarrow}$ $\text{content}(t) = L$.

Seq denotes the set of all initial segments of texts. Lowercase Greek letters (e.g., ρ, σ, τ), with or without decorations, range over elements of Seq, unless stated otherwise. λ denotes the empty initial segment (equivalently, the everywhere divergent function). For each σ, $|\sigma|$ denotes the length of σ (equivalently, the size of the domain of σ). For each σ and $i \leq |\sigma|$, $\sigma[i]$ denotes the initial segment of σ of length i. For each σ, $\text{content}(\sigma) \overset{\text{def}}{=} \{\sigma(i) \mid i < |\sigma|\} - \{\#\}$. For each σ and τ, $\sigma \cdot \tau$ denotes the concatenation of σ and τ. For each $\sigma \in \text{Seq} - \{\lambda\}$:

$$\sigma^{-} \overset{\text{def}}{=} \sigma[|\sigma| - 1]; \qquad\qquad \text{last}(\sigma) \overset{\text{def}}{=} \sigma(|\sigma| - 1).$$

For each L and $\mathcal{L}$, $\text{Txt}(L)$, $\text{Txt}(\mathcal{L})$, $\text{Seq}(L)$, and $\text{Seq}(\mathcal{L})$ are defined as follows.

$$\begin{array}{ll} \text{Txt}(L) = \{\, t \mid t \text{ is a text for } L\}. & \text{Seq}(L) = \{\sigma \mid \text{content}(\sigma) \subseteq L\}. \\ \text{Txt}(\mathcal{L}) = \{\, t \mid (\exists L \in \mathcal{L})[t \in \text{Txt}(L)]\}. & \text{Seq}(\mathcal{L}) = \{\sigma \mid (\exists L \in \mathcal{L})[\sigma \in \text{Seq}(L)]\}. \end{array}$$

For each one-argument partial function ψ and $x \in \mathbb{N}$, $\psi(x){\downarrow}$ denotes that $\psi(x)$ converges; $\psi(x){\uparrow}$ denotes that $\psi(x)$ diverges. We use $\uparrow$ to denote the value of a divergent computation.

$\mathcal{EN}$ denotes the collection of all effective numberings. $\mathcal{CE}$ denotes the collection of all computably enumerable (ce) subsets of $\mathbb{N}$. For each m and n, $\mathcal{PC}_{m,n}$ denotes the collection of partial computable functions mapping $\mathbb{N}^m \times \mathbb{N}_{\#}^n$ to $\mathbb{N}$. We shall be concerned primarily with $\mathcal{PC}_{1,0}$ and $\mathcal{PC}_{1,1}$. $(\varphi_p)_{p\in\mathbb{N}}$ denotes any fixed, acceptable numbering of $\mathcal{PC}_{1,0}$. For each i, $W_i \overset{\text{def}}{=} \{x \mid \varphi_i(x){\downarrow}\}$. Thus, $(W_i)_{i\in\mathbb{N}}$ is an effective numbering of $\mathcal{CE}$.

$\mathcal{Iter} \overset{\text{def}}{=} \mathcal{PC}_{1,1} \times \mathbb{N} \times \mathcal{EN}$. For each $M \in \mathcal{PC}_{1,1}$ and p, the partial function M_p^* is such that, for each $\sigma \in \text{Seq}$ and $x \in \mathbb{N}_{\#}$:

$$M_p^*(\lambda) = p; \qquad\qquad M_p^*(\sigma \cdot x) = \begin{cases} M\big(M_p^*(\sigma), x\big), & \text{if } M_p^*(\sigma){\downarrow}; \\ \uparrow, & \text{otherwise.} \end{cases}$$

$\mathcal{EO}_{1,1}$ denotes the collection of all enumeration operators of type $\mathcal{P}(\mathbb{N}) \times \mathbb{N}_{\#} \to \mathcal{P}(\mathbb{N})$. For each $\mathcal{M} \in \mathcal{EO}_{1,1}$ and X, the function $\mathcal{M}_X^* : \text{Seq} \to \mathcal{P}(\mathbb{N})$ is such that, for each $\sigma \in \text{Seq}$ and $x \in \mathbb{N}_{\#}$:

$$\mathcal{M}_X^*(\lambda) = X; \qquad\qquad \mathcal{M}_X^*(\sigma \cdot x) = \mathcal{M}\big(\mathcal{M}_X^*(\sigma), x\big).$$

The following are the formal definitions of the learning models described in Section 1. The symbols **Fr**, **Inj**, **Ext**, and **EO** are mnemonic for *Friedberg*, *injective*, *extensional*, and *enumeration operator*, respectively.

Definition 1 For each $\mathcal{L}$, (a)-(e) below. In parts (a)-(d), $(M, p, (X_j)_{j\in\mathbb{N}}) \in \mathcal{Iter}$. In part (e), $(\mathcal{M}, X) \in \mathcal{EO}_{1,1} \times \mathcal{CE}$.

(a) **(Wiehagen [Wie76])** $(M, p, (X_j)_{j \in \mathbb{N}})$ **It**-*identifies* $\mathcal{L} \Leftrightarrow$ for each $t \in \text{Txt}(\mathcal{L})$, there exists $i_0 \in \mathbb{N}$ such that $X_{M_p^*(t[i_0])} = \text{content}(t)$, and, for each $i \geq i_0$, $\big[M_p^*(t[i]) = M_p^*(t[i_0])\big]$.
(b) **(Jain & Stephan [JS08])** $(M, p, (X_j)_{j \in \mathbb{N}})$ **FrIt**-*identifies* $\mathcal{L} \Leftrightarrow (M, p, (X_j)_{j \in \mathbb{N}})$ **It**-identifies $\mathcal{L}$, and $(X_j)_{j \in \mathbb{N}}$ is a Friedberg numbering.
(c) $(M, p, (X_j)_{j \in \mathbb{N}})$ **InjIt**-*identifies* $\mathcal{L} \Leftrightarrow (M, p, (X_j)_{j \in \mathbb{N}})$ **It**-identifies $\mathcal{L}$, and $(X_j)_{j \in \mathbb{N}}$ is 1-1.
(d) $(M, p, (X_j)_{j \in \mathbb{N}})$ **ExtIt**-*identifies* $\mathcal{L} \Leftrightarrow (M, p, (X_j)_{j \in \mathbb{N}})$ **It**-identifies $\mathcal{L}$, and, for each $\sigma_0, \sigma_1 \in \text{Seq}(\mathcal{L}) - \{\lambda\}$,

$$[X_{M_p^*(\sigma_0^-)} = X_{M_p^*(\sigma_1^-)} \ \wedge\ \text{last}(\sigma_0) = \text{last}(\sigma_1)] \ \Rightarrow\ X_{M_p^*(\sigma_0)} = X_{M_p^*(\sigma_1)}. \quad (4)$$

(e) $(\mathcal{M}, X)$ **EOIt**-*identifies* $\mathcal{L} \Leftrightarrow$ for each $t \in \text{Txt}(\mathcal{L})$, there exists $i_0 \in \mathbb{N}$ such that $(\forall i \geq i_0)\big[\mathcal{M}_X^*(t[i]) = \text{content}(t)\big]$.

Definition 2 Let **It** be as follows.

$$\mathbf{It} = \big\{\mathcal{L} \mid \big(\exists (M, p, (X_j)_{j \in \mathbb{N}}) \in \mathit{Iter}\big)[(M, p, (X_j)_{j \in \mathbb{N}}) \ \mathbf{It}\text{-identifies } \mathcal{L}]\big\}. \quad (5)$$

Let **FrIt**, **InjIt**, **ExtIt**, and **EOIt** be defined similarly.

3 Uniform Decidability and Computable Finite Thickness

In this section, we extend a result of Lange & Zeugmann by showing that, for each class $\mathcal{L}$, if there exists a single hypothesis space witnessing that $\mathcal{L}$ is both uniformly decidable and computably finitely thick, then $\mathcal{L}$ can be **FrIt**-identified (Theorem 6). We also show that there exists a class of languages that *is* uniformly decidable and computably finitely thick, but that is *not* in **It**, let alone **FrIt** (Theorem 7). Thus, one could not arrive at the conclusion of the just mentioned Theorem 6 if one were to merely require that: there exists a uniformly decidable effective numbering of $\mathcal{L}$, and a possibly *distinct* computably finitely thick effective numbering of $\mathcal{L}$.

The following are the formal definitions of the terms *uniformly decidable* and *computably finitely thick*. For additional background, see [LZZ08].

Definition 3

(a) An effective numbering $(X_j)_{j \in \mathbb{N}}$ is *uniformly decidable* $\Leftrightarrow$ the predicate $\lambda j, x\,.\,[x \in X_j]$ is decidable.
(b) A class of languages $\mathcal{L}$ is *uniformly decidable* $\Leftrightarrow$ there exists a uniformly decidable effective numbering of $\mathcal{L}$.
(c) An effective numbering $(X_j)_{j \in \mathbb{N}}$ is *computably finitely thick* $\Leftrightarrow$ there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for each x,

$$\{X_j \mid j \in D_{f(x)}\} = \{L \mid x \in L \ \wedge\ (\exists j)[X_j = L]\}. \quad (6)$$

(d) **(Lange & Zeugmann [LZ96, Definition 9])** A class of languages $\mathcal{L}$ is *computably finitely thick* $\Leftrightarrow$ there exists a computably finitely thick effective numbering of $\mathcal{L}$.

N.B. In part (c) just above, the function f need *not* satisfy $D_{f(x)} = \{j \mid x \in X_j\}$. However, see the proof of Theorem 6 below.

Example 4

(a) $\mathcal{F}in$ *is* uniformly decidable, but is *not* computably finitely thick.
(b) $\mathcal{CE}$ is *neither* uniformly decidable *nor* computably finitely thick.
(c) The class $\big\{\{e\}, \{e\} \cup (W_e + e + 1) \mid e \in \mathbb{N}\big\}$ is *not* uniformly decidable, but *is* computably finitely thick.[10]
(d) The class $\{\mathbb{N} + e \mid e \in \mathbb{N}\}$ is both uniformly decidable and computably finitely thick. Moreover, there exists a single effective numbering witnessing both properties simultaneously.
(e) Let $\mathcal{L}$ be as follows.

$$\mathcal{L} = \big\{\{e\} \mid e \in \mathbb{N}\big\} \cup \big\{\{e, \varphi_e(0) + e + 1\} \mid e \in \mathbb{N} \ \wedge\ \varphi_e(0){\downarrow}\big\}. \qquad (7)$$

Then, $\mathcal{L}$ is both uniformly decidable and computably finitely thick,[11] but there is *no* effective numbering of $\mathcal{L}$ witnessing both properties simultaneously. In fact, no such numbering exists for any class containing $\mathcal{L}$.

The following result, due to Lange & Zeugmann, gives a sufficient condition for a class of languages to be **It**-identifiable.

Theorem 5 (Lange & Zeugmann [LZ96, Theorem 12]). For each $\mathcal{L}$, if there exists an effective numbering of $\mathcal{L}$ that is both uniformly decidable and computably finitely thick, then $\mathcal{L} \in \mathbf{It}$.[12]

The following result strengthens Theorem 5 (Lange & Zeugmann) just above.

Theorem 6. For each $\mathcal{L}$, if there exists an effective numbering of $\mathcal{L}$ that is both uniformly decidable and computably finitely thick, then $\mathcal{L} \in \mathbf{FrIt}$.

Proof (Sketch). Suppose that $\mathcal{L}$ satisfies the conditions of the theorem. Then it can be shown that there exists an effective numbering $(X_j)_{j\in\mathbb{N}}$ of $\mathcal{L}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for each x,

$$D_{f(x)} = \{j \mid x \in X_j\}. \qquad (8)$$

For each x and J, say that x *narrows* J $\Leftrightarrow$ by letting $J' = \{j \in J \mid x \in X_j\}$,

$$\emptyset \neq J' \subset J \ \wedge\ \{w \in (\textstyle\bigcap_{j\in J'} X_j) \mid w < x\} = \{w \in (\textstyle\bigcap_{j\in J} X_j) \mid w < x\}. \qquad (9)$$

[10] One can construct computably finitely thick effective numberings of the classes given in parts (c) and (e) of Example 4 using a technique similar to that used in Figure 3(b) below.
[11] See footnote 10.
[12] In [LZ96], Theorem 12 is not stated exactly as Theorem 5 is stated here. However, based on the proof of this result, we believe that what is stated here is what is meant.

List $\emptyset$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once. Then, for each x, run $\mathsf{survey}(x)$.

$\mathsf{survey}(x)$: Act according the following conditions.

- Cond. (a) $[D_{f(x)} = \emptyset \ \vee \ \min(\bigcap_{j\in D_{f(x)}} X_j) \neq x]$. For each k, list $\{x\}\cup(Y_k+x+1)$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once.
- Cond. (b) $[D_{f(x)} \neq \emptyset \ \wedge \ \min(\bigcap_{j\in D_{f(x)}} X_j) = x]$. List $(\bigcap_{j\in D_{f(x)}} X_j)$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once, set $\zeta(D_{f(x)})$ to the index used to list this set, and run $\mathsf{descend}(D_{f(x)}, x+1)$.

$\mathsf{descend}(J,x)$: Let J', x_0, and A be such that:

$$J' = \{j \in J \mid x \in X_j\}; \quad x_0 = \min(\textstyle\bigcap_{j\in J} X_j); \quad A = \{w \in (\bigcap_{j\in J} X_j) \mid w < x\}.$$

Act according to the following conditions.

- Cond. (i) $\big[J' = J\big]$. For each k, list $A\cup(Y_k+x+1)$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once, and run $\mathsf{descend}(J, x+1)$.
- Cond. (ii) $\big[x \le x_0 \ \vee \ [J' \subset J \ \wedge \ x \text{ does } \textit{not} \text{ narrow } J]\big]$. For each k, list $A\cup\{x\}\cup(Y_k+x+1)$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once, and run $\mathsf{descend}(J, x+1)$.
- Cond. (iii) $\big[x > x_0 \ \wedge \ x \text{ narrows } J\big]$. List $(\bigcap_{j\in J'} X_j)$ into $(Z_\ell)_{\ell\in\mathbb{N}}$ exactly once, set $\zeta(J')$ to the index used to list this set, and run both $\mathsf{descend}(J, x+1)$ and $\mathsf{descend}(J', x+1)$.

Fig. 2. The construction of $(Z_\ell)_{\ell\in\mathbb{N}}$ in the proof of Theorem 6.

Let $(Y_k)_{k\in\mathbb{N}}$ be any Friedberg numbering.

An effective numbering $(Z_\ell)_{\ell\in\mathbb{N}}$ is constructed in Figure 2. The construction makes use of two procedures: survey and descend. The procedure survey takes one argument: an element of $\mathbb{N}$. The procedure descend takes two arguments: a finite subset of $\mathbb{N}$, and an element of $\mathbb{N}$.

In conjunction with $(Z_\ell)_{\ell\in\mathbb{N}}$, a partial computable function ζ from $\mathcal{Fin}$ to $\mathbb{N}$ is constructed. It is clear from the construction that ζ is 1-1, i.e., for each $J, J' \in \mathcal{Fin}$, $[\zeta(J){\downarrow} = \zeta(J') \ \Rightarrow \ J = J']$.

It can be shown that $(Z_\ell)_{\ell\in\mathbb{N}}$ is a Friedberg numbering. For ease of presentation, suppose that $Z_0 = \emptyset$. Let M be such that, for each $\ell > 0$ and x:

$$M(0,\#) = 0; \quad M(0,x) = \begin{cases} \zeta(D_{f(x)}), & \text{if } \zeta(D_{f(x)}){\downarrow}; \\ \uparrow, & \text{otherwise}; \end{cases}$$

$$M(\ell,\#) = \ell; \quad M(\ell,x) = \begin{cases} \zeta(J'), & \text{where } J \text{ and } J' \text{ are such that } \zeta(J) = \ell \text{ and} \\ & J' = \{j \in J \mid x \in X_j\}, \text{ if such a } J \text{ exists} \\ & \text{and } \zeta(J'){\downarrow}; \\ \uparrow, & \text{otherwise.} \end{cases}$$

It can be shown that $(M, 0, (Z_\ell)_{\ell\in\mathbb{N}})$ **FrIt**-identifies $\mathcal{L}$. $\approx\Box$ (**Theorem 6**)

(a) For each i, execute stage 0 below.

- Stage 0. For each i, include $(\mathbb{N}+i)$ and $\{i\}$ in $\mathcal{L}_0$. Go to stage 1.
- Stage 1. Let (M,p) be the ith pair in $((M,p)_i)_{i\in\mathbb{N}}$. Search for a $k \geq i$ such that

$$M_p^*\big((i\cdot\dots\cdot k)\cdot(k+1)\big)\downarrow \;=\; M_p^*\big((i\cdot\dots\cdot k)\cdot k\big) \;=\; M_p^*(i\cdot\dots\cdot k).$$

If such a k is found, then include $\{i,...,k\}$ and $\{i,...,k{+}1\}$ in $\mathcal{L}_0$, and terminate the construction (for i). If *no* such k is found, then search indefinitely.

(b) For each i, execute stage 0 below.

- Stage 0. Set $X_{\mathrm{start}(i)} = \mathbb{N}{+}i$, and, for each $j \in \{\mathrm{start}(i){+}1,...,\mathrm{start}(i{+}1){-}1\}$, set $X_j = \{i\}$. Go to stage 1.
- Stage 1. In a dovetailing manner, monitor and act according to the following conditions.
  - Cond. [in the construction of $\mathcal{L}_0$ above, a k is found for i].
    Set $X_{\mathrm{start}(i)+2} = \{i,...,k\}$ and $X_{\mathrm{start}(i)+3} = \{i,...,k+1\}$.
  - Cond. [$i \in X_j,\;$ where $j < \mathrm{start}(i)$]. Set $X_{\mathrm{start}(i)+j+4} = X_j$.

Fig. 3. (a) The construction of $\mathcal{L}_0$ in the proof of Theorem 7. **(b)** The construction of $(X_j)_{j\in\mathbb{N}}$ in the proof of Theorem 7. The function start is defined in (10).

The proof of Theorem 7 below exhibits a class of languages $\mathcal{L}_0$ that is uniformly decidable and computably finitely thick, but $\mathcal{L}_0 \notin \mathbf{It}$. Thus, one could not arrive at the conclusion of Theorem 6 if one were to merely require that: there exists a uniformly decidable effective numbering of $\mathcal{L}$, and a possibly *distinct* computably finitely thick effective numbering of $\mathcal{L}$.

Theorem 7. There exists a class of languages $\mathcal{L}_0$ that is uniformly decidable and computably finitely thick, but $\mathcal{L}_0 \notin \mathbf{It}$.

Proof (Sketch). Let $((M,p)_i)_{i\in\mathbb{N}}$ be an algorithmic enumeration of all pairs of type $\mathcal{PC}_{1,1}\times\mathbb{N}$. Let start : $\mathbb{N}\to\mathbb{N}$ be such that, for each i,

$$\mathrm{start}(i) = 2^{i+1} - 4. \tag{10}$$

Note that, for each i, $\mathrm{start}(i+1) - \mathrm{start}(i) = \mathrm{start}(i) + 4$. The class $\mathcal{L}_0$ is constructed in Figure 3(a). An effective numbering $(X_j)_{j\in\mathbb{N}}$, which is easily seen to be of $\mathcal{L}_0$, is constructed in Figure 3(b). Let $f:\mathbb{N}\to\mathbb{N}$ be such that, for each i,

$$D_{f(i)} = \{\mathrm{start}(i),...,\mathrm{start}(i+1)-1\}, \tag{11}$$

It can be shown that $(X_j)_{j\in\mathbb{N}}$ and f witness that $\mathcal{L}_0$ is computably finitely thick. It is straightforward to construct an effective numbering witnessing that $\mathcal{L}_0$ is uniformly decidable. Finally, it can be shown that $\mathcal{L}_0 \notin \mathbf{It}$. $\approx\Box$ (**Theorem 7**)

4 Friedberg, Injective, and Extensional Iterative Learning

This section examines the Friedberg, injective, and extensional iterative learning models (**FrIt**, **InjIt**, and **ExtIt**, respectively). In terms of the classes of languages learnable by these models and by **It**, they are clearly related as follows.

$$\mathbf{FrIt} \subseteq \mathbf{InjIt} \subseteq \mathbf{ExtIt} \subseteq \mathbf{It}. \tag{12}$$

In this section we establish that **InjIt** $\not\subseteq$ **FrIt** (Proposition 8), and that **ExtIt** $\not\subseteq$ **InjIt** (Theorem 9). The fact that **It** $\not\subseteq$ **ExtIt** is due to Jain (Theorem 10).

Proposition 8 just below establishes that **InjIt** $\not\subseteq$ **FrIt**.

Proposition 8 **InjIt** $\not\subseteq$ **FrIt**.

Proof. Recall that $\mathcal{F}in$ is the collection of all finite subsets of $\mathbb{N}$. Jain & Stephan observed that $\mathcal{F}in \notin$ **FrIt** [JS08, Remark 28]. However, it is easily seen that $\mathcal{F}in \in$ **InjIt**. □ (**Proposition 8**)

Theorem 9 just below establishes that **ExtIt** $\not\subseteq$ **InjIt**.

Theorem 9. **ExtIt** $\not\subseteq$ **InjIt**.

Proof (Sketch). Let $\mathcal{L}_1$ be as follows.

$$\begin{aligned}\mathcal{L}_1 = \big\{2\mathbb{N}\big\} \cup \ \big\{ & \{0,2,...,2e\} \cup \{2e+1\} \cup 2W_e, \\ & \{0,2,...,2e\} \cup \{2e+1\} \cup 2X \quad \mid e \in \mathbb{N} \ \wedge \ \text{(a)-(c) below}\big\}. \\ & \text{(a) } (\forall e' \in X)[W_{e'} = X]. \\ & \text{(b) If } W_e = X\text{, then } W_e \text{ and } X \text{ are finite.} \\ & \text{(c) If } W_e \neq X\text{, then } 2W_e \subseteq \{0,2,...,2e\} \text{ and } X \text{ is infinite.}\end{aligned}$$

It can be shown that $\mathcal{L}_1 \in$ **ExtIt**. To complete the sketch of the proof: by way of contradiction, suppose that $\mathcal{L}_1 \in$ **InjIt**, as witnessed by $(M, p, (X_j)_{j\in\mathbb{N}}) \in \mathit{Iter}$. Then, there exists a k_0 such that

$$(\forall e \geq k_0)[M_p^*(0 \cdot 2 \cdot \cdots \cdot 2k_0 \cdot 2e)\!\downarrow = M_p^*(0 \cdot 2 \cdot \cdots \cdot 2k_0)]. \tag{13}$$

By Case's 1-1 Operator Recursion Theorem [Cas74,Cas94],[13] there exists a computably enumerable sequence of pairwise-distinct φ-programs $(e_i)_{i\in\mathbb{N}}$ such that $(\forall i)[e_i \geq k_0]$, and such that the behavior of $(e_i)_{i\in\mathbb{N}}$ is as in Figure 4. It can be shown that $\{W_{e_0}, W_{e_1}\} \subseteq \mathcal{L}_1$, but that $(M, p, (X_j)_{j\in\mathbb{N}})$ does *not* **InjIt**-identify at least one of W_{e_0} and W_{e_1} (a contradiction). ≈ □ (**Theorem 9**)

As mentioned above, the fact that **It** $\not\subseteq$ **ExtIt** is due to Jain.

Theorem 10 (Jain [Jai10]). **It** $\not\subseteq$ **ExtIt**.

We conclude this section with the following remark.

[13] Intuitively, the 1-1 Operator Recursion Theorem allows one to construct a computably enumerable sequence of pairwise-distinct φ-programs $(e_i)_{i\in\mathbb{N}}$ such that each program e_i *knows* all programs in the sequence and its own index i.

– STAGE 0. Search for an $m \geq 1$ such that

$$M_p^*\big((0 \cdot 2 \cdot \cdots \cdot 2e_0) \cdot (2e_0+1)^{m+1}\big)\!\downarrow = M_p^*\big((0 \cdot 2 \cdot \cdots \cdot 2e_0) \cdot (2e_0+1)^{m}\big).$$

If such an m is found, then set $\sigma_1 = (0 \cdot 2 \cdot \cdots \cdot 2e_0) \cdot (2e_0+1)^m$, and go to stage 1. If *no* such m is found, then search indefinitely.

– STAGE 1. For larger and larger values of n, make it the case that $W_{e_1} = \cdots = W_{e_n} = \{e_1, ..., e_n\}$. Simultaneously, search for an $i \in \{1, ..., n\}$ such that

$$M_p^*(\sigma_1 \cdot 2e_i)\!\downarrow \neq M_p^*(\sigma_1).$$

If such i and n are found, then make it the case that $W_{e_0} = \{e_1, ..., e_n\}$, and terminate the construction. If *no* such i and n are found, then search indefinitely, while making it the case that $(\forall i \geq 1)[W_{e_i} = \{e_i \mid i \geq 1\}]$.

Fig. 4. The construction of $(e_i)_{i\in\mathbb{N}}$ in the proof of Theorem 9.

Remark 11 The fact **It** $\not\subseteq$ **InjIt** (as opposed to **It** $\not\subseteq$ **ExtIt** or **ExtIt** $\not\subseteq$ **InjIt**) can be shown directly using either of the next two pre-existing results.

- There exists a class of languages that *can* be **It**-identified, but that can*not* be so identified *order-independently* (in the sense of [BB75,Ful90]).[14]
- There exists a class of languages that *can* be **It**-identified, but that can*not* be so identified *strongly non-U-shapedly* [CK10, Theorem 5.4] (see also [Bei84,Wie91,CM08]).

5 Iterative Learning by Enumeration Operator

This section examines the iterative learning by enumeration operator model (**EOIt**). Recall that **EOIt** is similar to **It**, except that the computational component of the learner is modeled as an enumeration operator, as opposed to a partial computable function. Our main results of this section are the following.

- Every computably finitely thick class of languages (see Definition 3) can be **EOIt**-identified (Theorem 12).
- **EOIt** $\not\subseteq$ **It** (Corollary 13).
- **FrIt** $\not\subseteq$ **EOIt** (Theorem 14).

An open problem that remains is whether **It** $\cap$ **EOIt** $\subseteq$ **ExtIt**, i.e., whether every class of languages that can be **It**-identified *and* **EOIt**-identified can be **ExtIt**-identified (Problem 15).

Recall that $\mathcal{F}in \in$ **InjIt** $-$ **FrIt** (Proposition 8). Further recall that the class $\mathcal{L}_1$ from the proof Theorem 9 satisfies: $\mathcal{L}_1 \in$ **ExtIt** $-$ **InjIt**. It is straightforward to show that $\{\mathcal{F}in, \mathcal{L}_1\} \subseteq$ **EOIt**.

Theorem 12 just below is our first main result of this section.

[14] An anonymous referee attributes this result to Liepe & Wiehagen.

Theorem 12. Suppose that $\mathcal{L}$ is computably finitely thick. Then, $\mathcal{L} \in \mathbf{EOIt}$.

Proof (Sketch). Suppose that $\mathcal{L}$ is computably finitely thick. Let $\psi : \mathcal{Fin} \to \mathcal{CE}$ be such that, for each $A \in \mathcal{Fin} - \{\emptyset\}$:

$$\psi(\emptyset) = \emptyset; \qquad\qquad \psi(A) = \bigcap\{L \in \mathcal{L} \mid A \subseteq L\}.$$

Let $\mathcal{M} : \mathcal{P}(\mathbb{N}) \times \mathbb{N}_\# \to \mathcal{P}(\mathbb{N})$ be such that, for each $X \subseteq \mathbb{N}$ and x:

$$\mathcal{M}(X, \#) = X; \qquad \mathcal{M}(X, x) = \bigcup\big\{\psi(A) \mid A \text{ is finite } \wedge\ A \subseteq X \cup \{x\}\big\}.$$

It can be shown that $(\mathcal{M}, \emptyset)$ **EOIt**-identifies $\mathcal{L}$. $\approx\Box$ (**Theorem 12**)

Recall that the proof of Theorem 7 exhibited a computably finitely thick class of languages $\mathcal{L}_0 \notin \mathbf{It}$. By Theorem 12, $\mathcal{L}_0 \in \mathbf{EOIt}$. Thus, one has the following.

Corollary 13 (of Theorems 7 and 12) $\mathbf{EOIt} \not\subseteq \mathbf{It}$.

The proof of Theorem 14 below exhibits a class $\mathcal{L}_2 \in \mathbf{FrIt} - \mathbf{EOIt}$.

Theorem 14. $\mathbf{FrIt} \not\subseteq \mathbf{EOIt}$.

Proof (Sketch). It is straightforward to construct a Friedberg numbering $(X_j)_{j\in\mathbb{N}}$ satisfying:

$$(\forall j)[1 \leq |D_j| \leq 3 \ \Rightarrow\ X_j = D_j]. \tag{14}$$

Let $(Y_k)_{k\in\mathbb{N}}$ be any Friedberg numbering satisfying: $Y_0 = \emptyset$. Let $(Z_\ell)_{\ell\in\mathbb{N}}$ be such that, for each j and k, $Z_{\langle j,k\rangle} = (2X_j) \cup (2Y_k + 1)$. It is straightforward to show that $(Z_\ell)_{\ell\in\mathbb{N}}$ is a Friedberg numbering. Let $\mathcal{L}_2$ be the following class of languages.

$$\mathcal{L}_2 = \{Z_{\langle j, \max D_j\rangle} \mid 1 \leq |D_j| \leq 3\}. \tag{15}$$

Note that, for each j such that $1 \leq |D_j| \leq 3$,

$$Z_{\langle j,\max D_j\rangle} \ =\ (2X_j) \cup (2Y_{\max D_j} + 1) \ =\ (2D_j) \cup (2Y_{\max D_j} + 1). \tag{16}$$

It is straightforward to show that $\mathcal{L}_2 \in \mathbf{FrIt}$ (e.g., using $(Z_\ell)_{\ell\in\mathbb{N}}$ as the hypothesis space). On the other hand, it can be shown that $\mathcal{L}_2 \notin \mathbf{EOIt}$. $\approx\Box$ (**Theorem 14**)

As mentioned above, the following problem remains open.

Problem 15 Is it the case that $\mathbf{It} \cap \mathbf{EOIt} \subseteq \mathbf{ExtIt}$?

References

[Ang80] D. Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980.

[BB75] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Inform. Control*, 28(2):125–155, 1975.

[BBCJS10] L. Becerra-Bonache, J. Case, S. Jain, and F. Stephan. Iterative learning of simple external contextual languages. *Theor. Comput. Sci.*, 411(29–30):2741–2756, 2010.

[Bei84] H-R. Beick. *Induktive Inferenz mit höchster Konvergenzgeschwindigkeit.* PhD thesis, Sektion Mathematik, Humboldt-Universität Berlin, 1984.

[Cas74] J. Case. Periodicity in generations of automata. *Math. Syst. Theory*, 8(1):15–32, 1974.

[Cas94] J. Case. Infinitary self-reference in learning theory. *J. Exp. Theor. Artif. In.*, 6(1):3–16, 1994.

[CCJS07] L. Carlucci, J. Case, S. Jain, and F. Stephan. Results on memory-limited U-shaped learning. *Inform. Comput.*, 205(10):1551–1573, 2007.

[CK10] J. Case and T. Kötzing. Strongly non-U-shaped learning results by general techniques. In *Proc. of COLT'2010*, pages 181–193, 2010.

[CM08] J. Case and S. Moelius. Optimal language learning. In *Proc. of ALT'2008*, volume 5254 of *LNAI*, pages 419–433, 2008.

[dBY10] M. de Brecht and A. Yamamoto. Topological properties of concept spaces (full version). *Inform. Comput.*, 208(4):327–340, 2010.

[FKW82] R. Freivalds, E. Kinber, and R. Wiehagen. Inductive inference and computable one-one numberings. *Z. Math. Logik*, 28(27):463–479, 1982.

[Fri58] R. Friedberg. Three theorems on recursive enumeration. I. Decomposition. II. Maximal set. III. Enumeration without duplication. *J. Symbolic Logic*, 23(3):309–316, 1958.

[Ful90] M. Fulk. Prudence and other conditions on formal language learning. *Inform. Comput.*, 85(1):1–11, 1990.

[Gol67] E. Mark Gold. Language identification in the limit. *Inform. Control*, 10(5):447–474, 1967.

[Jai10] S. Jain. Private communcation, 2010.

[JLMZ10] S. Jain, S. Lange, S. Moelius, and S. Zilles. Incremental learning with temporary memory. *Theor. Comput. Sci.*, 411(29–30):2757–2772, 2010.

[JS08] S. Jain and F. Stephan. Learning in Friedberg numberings. *Inform. Comput.*, 206(6):776–790, 2008.

[Kum90] M. Kummer. An easy priority-free proof of a theorem of Friedberg. *Theor. Comput. Sci.*, 74(2):249–251, 1990.

[LW91] S. Lange and R. Wiehagen. Polynomial time inference of arbitrary pattern languages. *New Generat. Comput.*, 8(4):361–370, 1991.

[LZ96] S. Lange and T. Zeugmann. Incremental learning from positive data. *J. Comput. Syst. Sci.*, 53(1):88–103, 1996.

[LZZ08] S. Lange, T. Zeugmann, and S. Zilles. Learning indexed families of recursive languages from positive data: A survey. *Theor. Comput. Sci.*, 397(1–3):194–232, 2008.

[MZ10] S. Moelius and S. Zilles. Learning without coding. Technical report, 2010. `http://www2.cs.uregina.ca/~zilles/moeliusZ10TR.pdf` .

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

[SSS$^+$94] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, and S. Arikawa. Knowledge acquisition from amino acid sequences by machine learning system BONSAI. *Trans. Inform. Process. Soc. Jpn.*, 35(10):2009–2018, 1994.

[Wie76] R. Wiehagen. Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *J. Inform. Process. Cybern. (EIK)*, 12(1/2):93–99, 1976.

[Wie91] R. Wiehagen. A thesis in inductive inference. In *Proc. of 1st International Workshop on Nonmonotonic and Inductive Logic (1990)*, volume 543 of *LNAI*, pages 184–207, 1991.