Parallelism Increases Iterative Learning Power

John Case and Samuel E. Moelius III

Department of Computer & Information Sciences
University of Delaware
103 Smith Hall
Newark, DE 19716
{case,moelius}@cis.udel.edu

Abstract. *Iterative learning* (**It**-learning) is a Gold-style learning model in which each of a learner's output conjectures may depend *only* upon the learner's *current* conjecture and the *current* input element. Two extensions of the **It**-learning model are considered, each of which involves parallelism. The first is to run, in parallel, distinct instantiations of a single learner on each input element. The second is to run, in parallel, n individual learners *incorporating the first extension*, and to allow the n learners to communicate their results. In most contexts, parallelism is only a means of improving efficiency. However, as shown herein, learners incorporating the first extension are more powerful than **It**-learners, and, *collective* learners resulting from the second extension increase in learning power as n increases. Attention is paid to how one would actually implement a learner incorporating each extension. Parallelism is the underlying mechanism employed.

1 Introduction

Iterative learning (**It**-learning) [Wie76, LZ96, CJLZ99, CCJS06, CM07a] is a mathematical model of language learning in the style of Gold [Gol67].[1] In this model, the learner (commonly denoted by **M**, for *machine*) is an algorithmic device that is repeatedly fed elements from an infinite sequence. The elements of the sequence consist of numbers and, possibly, pauses (#). The set of all such numbers represents a *language*. After being fed each element, the learner either: outputs a *conjecture*, or diverges.[2] A conjecture may be either: a *grammar*, possibly for the language represented by the sequence, or '?'.[3] Most importantly, the learner may *only* consider its *current* conjecture and the *current* input element when forming a new conjecture.

For the remainder of this section, let **M** be a fixed learner. For now, **M** may be thought of as an **It**-learner. Later in this section, we will treat **M** as a instance of

[1] In this paper, we focus exclusively on language learning, as opposed to, say, function learning [JORS99].

[2] Intuitively, if a learner **M** diverges, then **M** *goes into an infinite loop*.

[3] N.B. Outputting '?' is *not* the same as diverging. Outputting '?' requires only *finitely many* steps; whereas, diverging requires *infinitely many* steps.

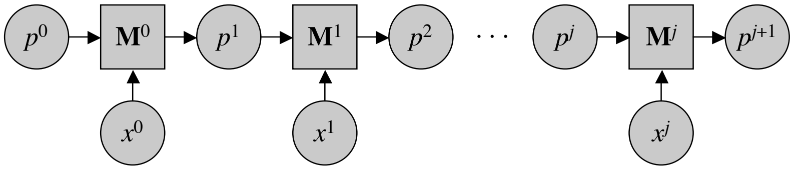

Fig. 1. The iterative learning process. The jth instantiation of learner $\mathbf{M}$, $\mathbf{M}^j$, is fed the current conjecture p^j and current input element x^j. From these, $\mathbf{M}^j$ produces a new conjecture p^{j+1}.

a more general type of learner. Let $x^0, x^1, ...$ be an arbitrary input sequence. Let p^0 be $\mathbf{M}$'s initial conjecture (i.e., $\mathbf{M}$'s conjecture having been fed no data), and, for all j, let p^{j+1} be the result of $\mathbf{M}^j$, where $\mathbf{M}^j$ is the computation performed by running $\mathbf{M}$ on inputs p^j and x^j. In the event that $\mathbf{M}^j$ diverges, we let $p^{j+1} = \perp$. (By convention, p^0 can*not* be $\perp$.) We shall refer to $\mathbf{M}^j$ as the *jth instantiation of* $\mathbf{M}$. See Figure 1.

An **It**-learner $\mathbf{M}$ is *successful* at learning the language represented by $x^0, x^1, ...$ $\overset{\text{def}}{\Leftrightarrow}$

- *none* of $\mathbf{M}^0, \mathbf{M}^1, ...$ diverge (i.e., *none* of $p^1, p^2, ...$ is $\perp$);
- for some index j_0, each of $\mathbf{M}^{j_0}, \mathbf{M}^{j_0+1}, ...$ results in p^{j_0+1}; *and*,
- p^{j_0+1} correctly describes the language represented by $x^0, x^1, ...$.

We say that $\mathbf{M}$ *identifies* a language L, or, L is *identifiable* by $\mathbf{M}$ $\overset{\text{def}}{\Leftrightarrow}$ $\mathbf{M}$ is successful at learning L from any input sequence representing L.

The *pattern languages* are an example of a class of languages that are **It**-learnable, i.e., there exists an **It**-learner capable of identifying every language in the class. A pattern language is (by definition) the language generated by all positive length substitution instances in a *pattern* (e.g., `abXYcbbZXa`, where the variables/*non*terminals are depicted in uppercase, and the constants/terminals are depicted in lowercase). The pattern languages and their learnability were first considered by Angluin [Ang80]. Since then, much work has been done on the learnability of pattern languages [Sal94a, Sal94b, CJK$^+$01] and finite unions thereof [Shi83, Wri89, KMU95, BUV96]. The class of pattern languages, itself, was shown to be **It**-learnable by Lange and Wiehagen [LW91]. Subsequently, this result was extended by Case, *et al.* [CJLZ99] who showed that, for each k, the class formed by taking the union of all choices of k pattern languages is **It**-learnable. Nix [Nix83], as well as Shinohara and Arikawa [SA95], outline interesting applications of pattern inference algorithms.

It-learning is a *memory limited* special case of the more general *explanatory learning* (**Ex**-learning) [Gol67, JORS99][4] and *behaviorally correct learning* (**Bc**-learning) [CL82, JORS99].[5] **Ex** and **Bc**-learners are *not*, in general, limited

[4] **Ex**-learning is the model that was actually studied by Gold [Gol67].

[5] Other memory limited learning models are considered in [OSW86, FJO94, CJLZ99, CCJS06].

to just the current conjecture and current input element when forming a new conjecture. Rather, such learners can refer to conjectures and/or input elements arbitrarily far into the past.[6]

Many **It**-learnable classes of languages are of practical interest. For example, the pattern languages, mentioned above, are a class whose learnability has applications to problems in molecular biology [AMS$^+$93, SSS$^+$94, SA95]. There is benefit in knowing that a class of languages is **It**-learnable, in that **It**-learners satisfy the following informal property.

Property 1. Each element of an input sequence may be discarded (and any associated resources freed) immediately after the element is fed to the learner.

Clearly, **Ex** and **Bc**-learners do *not* satisfy Property 1. In general, an implementation of an **Ex** or **Bc**-learner would have to store each element of an input sequence indefinitely. Thus, from a practical perspective, showing a class of languages to be **It**-learnable is far more desirable than showing it to be merely **Ex** or **Bc**-learnable.

Herein, we consider two extensions of the **It**-learning model, each of which involves parallelism. The first is to run, in parallel, distinct instantiations of a single learner on each input element (see Section 1.1 below). We call a learner incorporating this extension a 1-**ParIt**-*learner*. Our second extension is to run, in parallel, n distinct learners *incorporating the first extension*, and to allow the n learners to communicate their results (see Section 1.2 below).[7] We call a *collective* learner resulting from this latter extension, an n-**ParIt**-*learner*.

Each extension is described in further detail below.

1.1 First Extension

As mentioned previously, for an **It**-learner $\mathbf{M}$ to be *successful* at learning a language, *none* of its instantiations $\mathbf{M}^0, \mathbf{M}^1, ...$ may diverge. Thus, a most obvious implementation of $\mathbf{M}$ would run $\mathbf{M}^j$ only after $\mathbf{M}^{j-1}$ has converged. We can put each such $\mathbf{M}^j$ squarely into one of two categories: those that *need* p^j to compute p^{j+1}, and those that do *not*. For those that do *not*, there is *no* reason to wait until $\mathbf{M}^{j-1}$ has converged, *nor* is there reason to require that $\mathbf{M}^{j-1}$ converge at all.

Thus, our first extension is to allow $\mathbf{M}^0, \mathbf{M}^1, ...$ to run in parallel. We do *not* require that each of $\mathbf{M}^0, \mathbf{M}^1, ...$ converge, as is required by **It**-learning. However, we do require that if $\mathbf{M}^j$ *needs* p^j to compute p^{j+1}, *and*, $\mathbf{M}^{j-1}$ diverges, then $\mathbf{M}^j$ also diverges. This is an informal way of saying that $\mathbf{M}$ must be *monotonic* [Win93]. This issue is discussed further in Section 1.2.

We call a learner incorporating our first extension a 1-**ParIt**-*learner*. We say that such a learner is *successful* at learning the language represented by $x^0, x^1, ...$ $\overset{\text{def}}{\Leftrightarrow}$ for some index j_0,

[6] **Bc**-learners differ from **Ex**-learners in that, beyond some point, all of the conjectures output by a **Bc**-learner must correctly (semantically) describe the input language, but those conjectures need *not* be (syntactically) identical.

[7] The reader should *not* confuse this idea with *team learning* [JORS99].

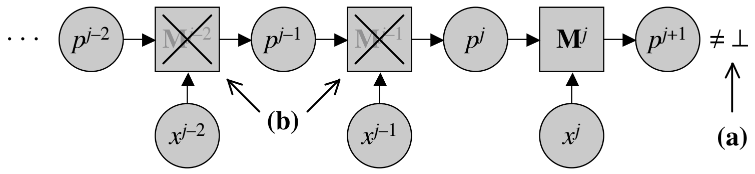

Fig. 2. How a 1-**ParIt**-learner **M** may be implemented. Once $\mathbf{M}^j$ has converged (i.e., has resulted in something other than $\perp$) (a), any previous instantiations of **M** that are still running may be forcibly terminated (b).

– each of $\mathbf{M}^{j_0}, \mathbf{M}^{j_0+1}, ...$ converges;
– each of $\mathbf{M}^{j_0}, \mathbf{M}^{j_0+1}, ...$ results in p^{j_0+1}; *and*,
– p^{j_0+1} correctly describes the language represented by $x^0, x^1, ...$.

A 1-**ParIt**-learner may be implemented in the following manner. Successively, for each j, start running $\mathbf{M}^j$. Simultaneously, watch for each $\mathbf{M}^j$ that is currently running to converge. Whenever j is such that $\mathbf{M}^j$ converges, forcibly terminate any currently running instantiations of the form $\mathbf{M}^0, ..., \mathbf{M}^{j-1}$. (The idea is that once $\mathbf{M}^j$ has converged, the results of any previous instantiations of **M** are no longer needed. See Figure 2.)

Clearly, a learner implemented in this way will *not* satisfy Property 1. However, if $x^0, x^1, ...$ represents a language *identifiable* by **M**, then, for some index j_0, each of $\mathbf{M}^{j_0}, \mathbf{M}^{j_0+1}, ...$ will converge. Thus, on such an input sequence, each instantiation $\mathbf{M}^j$ will eventually either: converge or be forced to terminate. Once either has occurred, the inputs of $\mathbf{M}^j$ may be discarded. As such, every 1-**ParIt**-learner satisfies the following weakened version of Property 1.

Property 2. If an input sequence represents a language identifiable by the learner, then each element of the sequence may be discarded *eventually*.

Clearly, **Ex** and **Bc**-learners do *not* satisfy even the weaker Property 2. Thus, from a practical perspective, 1-**ParIt**-learners are more attractive than **Ex** or **Bc**-learners.

Our first main result, Theorem 1 in Section 3, is that 1-**ParIt**-learners are strictly more powerful than **It**-learners.

1.2 Second Extension

An obvious parallel generalization of the preceding ideas is to run, in parallel, distinct, individual learners incorporating the first extension. Clearly, nothing is gained if each such learner runs in isolation. But, if the learners are allowed to communicate their results, then the resulting *collective* learner can actually be more powerful than each of its individual learners.

For the remainder of this section, let $n \geq 1$ be fixed, and let $\mathbf{M}_0, ..., \mathbf{M}_{n-1}$ be n learners incorporating the first extension. Let p_i^0 be $\mathbf{M}_i$'s initial conjecture, and, for each $i < n$, and each j, let p_i^{j+1} be the result of $\mathbf{M}_i^j$.

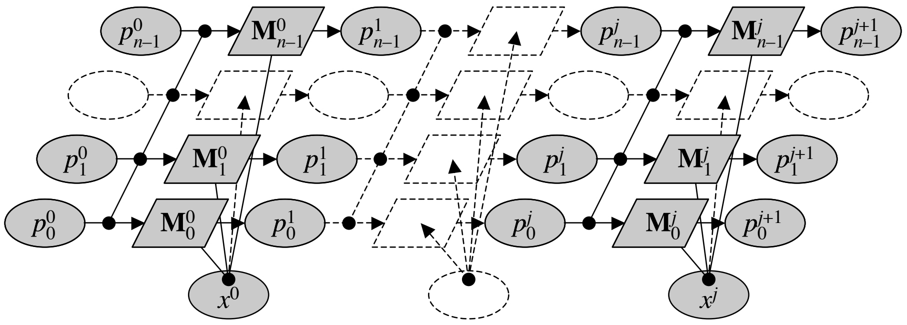

Fig. 3. A *collective* learner resulting from our second extension. For each $i < n$, and each j, individual learner $\mathbf{M}_i$ may consider conjectures $p^j_0, ..., p^j_{n-1}$ and input element x^j when forming conjecture p^{j+1}_i.

Our second extension is to allow $\mathbf{M}_0, ..., \mathbf{M}_{n-1}$ to run in parallel. For each $i < n$, and each j, we allow $\mathbf{M}^j_i$ to consider $p^j_0, ..., p^j_{n-1}$ and x^j when forming conjecture p^{j+1}_i (see Figure 3). However, as in the 1-ary case, we require that each $\mathbf{M}_i$ be monotonic.[8] So, if $\mathbf{M}^j_i$ *needs* $p^j_{i'}$ to compute p^{j+1}_i, *and*, $\mathbf{M}^{j-1}_{i'}$ diverges, then $\mathbf{M}^j_i$ also diverges. The following examples give some intuition as to which strategies $\mathbf{M}^j_i$ may employ, and which strategies $\mathbf{M}^j_i$ may *not* employ, in considering $p^j_0, ..., p^j_{n-1}$. *Exactly* which such strategies $\mathbf{M}^j_i$ may employ is made formal by Definition 2 in Section 3.

Example 1. $\mathbf{M}^j_i$ *may* employ any of the following strategies in considering $p^j_0, ...,$ p^j_{n-1}.

(a) $\mathbf{M}^j_i$ does *not* wait for any of $\mathbf{M}^{j-1}_0, ..., \mathbf{M}^{j-1}_{n-1}$ to converge; $\mathbf{M}^j_i$ uses just x^j to compute p^{j+1}_i.
(b) $\mathbf{M}^j_i$ waits for $\mathbf{M}^{j-1}_{i'}$ to converge. Then, $\mathbf{M}^j_i$ uses $p^j_{i'}$ to compute p^{j+1}_i.
(c) $\mathbf{M}^j_i$ waits for $\mathbf{M}^{j-1}_{i'}$ to converge. Then, $\mathbf{M}^j_i$ performs some computable test on $p^j_{i'}$, and, based on the outcome, either: uses just $p^j_{i'}$ to compute p^{j+1}_i; or, waits for $\mathbf{M}^{j-1}_{i''}$ to converge, and uses both $p^j_{i'}$ and $p^j_{i''}$ to compute p^{j+1}_i.
(d) $\mathbf{M}^j_i$ waits for each of $\mathbf{M}^{j-1}_0, ..., \mathbf{M}^{j-1}_{n-1}$ to converge, in some predetermined order. Then, $\mathbf{M}^j_i$ uses each of $p^j_0, ..., p^j_{n-1}$ to compute p^{j+1}_i.

Example 2. In general, $\mathbf{M}^j_i$ may *not* employ the following strategy in considering $p^j_0, ..., p^j_{n-1}$ when $n \geq 2$.

($*$) $\mathbf{M}^j_i$ waits for *any* of $\mathbf{M}^{j-1}_0, ..., \mathbf{M}^{j-1}_{n-1}$ to converge. Then, for that $i' < n$ such that $\mathbf{M}^{j-1}_{i'}$ converges *first*, $\mathbf{M}^j_i$ uses $p^{j+1}_{i'}$ to compute p^{j+1}_i.

Example 2 is revisited following Definition 2 in Section 3.

[8] In this context, monotonicity is equivalent to *continuity* [Win93], since each $\mathbf{M}^j_i$ operates on only *finitely much* data.

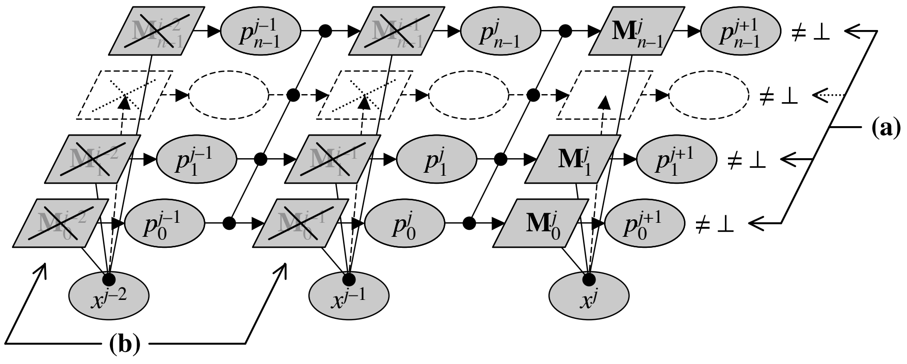

Fig. 4. How an n-**ParIt**-learner $\mathbf{M} = (\mathbf{M}_0, ..., \mathbf{M}_{n-1})$ may be implemented. Once *each* of $\mathbf{M}_0^j, ..., \mathbf{M}_{n-1}^j$ has converged (a), any previous instantiations of $\mathbf{M}_0, ..., \mathbf{M}_{n-1}$ that are still running may be forcibly terminated (b).

Let $\mathbf{M} = (\mathbf{M}_0, ..., \mathbf{M}_{n-1})$. We call such a *collective* learner $\mathbf{M}$ an n-**ParIt**-*learner*. We say that such a learner is *successful* at learning the language represented by $x^0, x^1, ... \overset{\text{def}}{\Leftrightarrow}$ for some index j_0, and each $i < n$,

- each of $\mathbf{M}_i^{j_0}, \mathbf{M}_i^{j_0+1}, ...$ converges;
- each of $\mathbf{M}_i^{j_0}, \mathbf{M}_i^{j_0+1}, ...$ results in $p_i^{j_0+1}$; *and*,
- $p_i^{j_0+1}$ correctly describes the language represented by $x^0, x^1, ...$.

A strategy for running instantiations of an n-**ParIt**-learner can easily be generalized from the 1-ary case. Instantiations may be terminated using the following strategy. Whenever j is such that *each* of $\mathbf{M}_0^j, ..., \mathbf{M}_{n-1}^j$ converges, forcibly terminate any currently running instantiations of the form $\mathbf{M}_i^0, ..., \mathbf{M}_i^{j-1}$, where $i < n$. (The idea is that once *each* of $\mathbf{M}_0^j, ..., \mathbf{M}_{n-1}^j$ has converged, the results of any previous instantiations of $\mathbf{M}_0, ..., \mathbf{M}_{n-1}$ are no longer needed. See Figure 4.)

Clearly, if $x^0, x^1, ...$ represents a language *identifiable* by $\mathbf{M}$, then, for all but finitely many j, each of $\mathbf{M}_0^j, ..., \mathbf{M}_{n-1}^j$ will converge. It follows that an n-**ParIt**-learner implemented as described in the just previous paragraph satisfies Property 2. Thus, from a practical perspective, n-**ParIt**-learners are more attractive than **Ex** or **Bc**-learners.

Our second main result, Theorem 2 in Section 3, is that, for all $n \geq 1$, $(n+1)$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners.

1.3 Summary of Results

Our results are summarized by the following diagram, where the arrows represent proper inclusions.

$$\mathbf{It} \longrightarrow 1\text{-}\mathbf{ParIt} \longrightarrow 2\text{-}\mathbf{ParIt} \longrightarrow \cdots$$

That is, 1-**ParIt**-learners are strictly more powerful than **It**-learners (Theorem 1). Furthermore, for all $n \geq 1$, $(n+1)$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners (Theorem 2). Thus, we think it fair to say: parallelism increases iterative learning power.

The remainder of this paper is organized as follows. Section 2 covers notation and preliminaries. Section 3 gives the formal definition of n-**ParIt**-learning and presents our results.

2 Notation and Preliminaries

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

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. $\mathbb{N}_? \overset{\text{def}}{=} \mathbb{N} \cup \{?\}$. $\mathbb{N}_{?,\perp} \overset{\text{def}}{=} \mathbb{N}_? \cup \{\perp\}$. $\mathbb{N}_\# \overset{\text{def}}{=} \mathbb{N} \cup \{\#\}$. Lowercase Roman letters *other than f, g, p, and q*, with or without decorations, range over elements of $\mathbb{N}$. f and g will be used to denote (possibly partial) functions of various types. The exact type of f and g will be made clear whenever they are introduced. p and q, with or without decorations, range over $\mathbb{N}_{?,\perp}$. $\boldsymbol{p}$ and $\boldsymbol{q}$ will be used to denote tuples whose elements are drawn from $\mathbb{N}_{?,\perp}$. The *size* of $\boldsymbol{p}$ and $\boldsymbol{q}$ will be made clear whenever they are introduced. For all n, all $\boldsymbol{p} \in \mathbb{N}^n_{?,\perp}$, and all $i < n$, $(\boldsymbol{p})_i$ denotes the ith element of $\boldsymbol{p}$, where the first element is considered the 0th. $D_0, D_1, \ldots$ denotes a fixed, canonical enumeration of all finite subsets of $\mathbb{N}$ such that $D_0 = \emptyset$ [Rog67]. Uppercase Roman letters, with or without decorations, range over *all* (finite and infinite) subsets of $\mathbb{N}$. $\mathcal{L}$ ranges over collections of subsets of $\mathbb{N}$.

$\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ denotes any fixed, 1-1, onto, computable function. In some cases, we will write $A \times B$ for $\{\langle a, b\rangle : a \in A \ \wedge\ b \in B\}$.

For all p and q, $p \sqsubseteq q \ \overset{\text{def}}{\Leftrightarrow}\ [p = \perp \ \vee\ p = q]$. For all n, and all $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{N}^n_{?,\perp}$, $\boldsymbol{p} \sqsubseteq \boldsymbol{q} \ \overset{\text{def}}{\Leftrightarrow}\ (\forall i < n)[(\boldsymbol{p})_i \sqsubseteq (\boldsymbol{q})_i]$. For all n, and all $\boldsymbol{p} \in \mathbb{N}^n_{?,\perp}$, $|\boldsymbol{p}|_{\neq\perp} \overset{\text{def}}{=} |\{i < n : (\boldsymbol{p})_i \neq \perp\}|$. So, for example, $|(0, 1, \perp, \perp, ?)|_{\neq\perp} = 3$.

$\varphi_0, \varphi_1, \ldots$ denotes any fixed, acceptable numbering of all *partial* computable functions of type $\mathbb{N} \rightharpoonup \mathbb{N}$ [Rog67]. For each i, we will treat φ_i as a *total* function of type $\mathbb{N} \to \mathbb{N}_\perp$, where $\perp$ denotes the value of a divergent computation.[9] For all i, $W_i \overset{\text{def}}{=} \{x \in \mathbb{N} : \varphi_i(x) \neq \perp\}$. Thus, for all i, W_i is the ith recursively enumerable set [Rog67].

$\mathbb{N}^*_\#$ denotes the set of all finite initial segments of total functions of type $\mathbb{N} \to \mathbb{N}_\#$. $\mathbb{N}^{\leq\omega}_\#$ denotes the set of *all* (finite and infinite) initial segments of total functions of type $\mathbb{N} \to \mathbb{N}_\#$. λ denotes the empty initial segment. ρ, σ, and τ, with or without decorations, range over elements of $\mathbb{N}^*_\#$.

For all $f \in \mathbb{N}^{\leq\omega}_\#$, $\text{content}(f) \overset{\text{def}}{=} \{y \in \mathbb{N} : (\exists x)[f(x) = y]\}$. For all $f \in \mathbb{N}^{\leq\omega}_\#$ and L, *f represents L* $\overset{\text{def}}{\Leftrightarrow}$ f is total and $\text{content}(f) = L$.[10] For all σ, $|\sigma|$ denotes the length of σ, i.e., the number of elements in σ. For all $f \in \mathbb{N}^{\leq\omega}_\#$, and all n, $f[n]$ denotes the initial segment of f of length n, if it exists; f, otherwise. For all σ, all $f \in \mathbb{N}^{\leq\omega}_\#$, and all i,

[9] N.B. It can*not*, in general, be determined whether $\varphi_i(x) = \perp$, for arbitrary i and x.

[10] Such an f is often called a *text* (for L) [JORS99].

$$(\sigma \diamond f)(i) \stackrel{\text{def}}{=} \begin{cases} \sigma(i), & \text{if } i < |\sigma|; \\ f(i - |\sigma|), & \text{otherwise.} \end{cases} \tag{1}$$

M will be used to denote *partial* computable functions of type $\mathbb{N}_{\#}^{*} \rightharpoonup \mathbb{N}_{?}^{n}$, for various n. However, as with $\varphi_0, \varphi_1, ...$, we will treat each **M** as a *total* function of type $\mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot}^{n}$. The exact type of **M** will be made clear whenever it is introduced. For all n, all $\mathbf{M} : \mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot}^{n}$, all $i < n$, and all ρ, $\mathbf{M}_i(\rho) \stackrel{\text{def}}{=} \big(\mathbf{M}(\rho)\big)_i$.

The following is the formal definition of **It**-learning.[11]

Definition 1

(a) For all $\mathbf{M} : \mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot}$ and L, **M** **It**-*identifies* L $\Leftrightarrow$ (i) and (ii) below.
  (i) For all f representing L, there exist j and $p \in \mathbb{N}$ such that $(\forall j' \geq j)$ $\big[\mathbf{M}(f[j']) = p\big]$ and $W_p = L$.[12]
  (ii) For all ρ, σ, and τ such that $\text{content}(\rho) \cup \text{content}(\sigma) \cup \text{content}(\tau) \subseteq L$, (α) and (β) below.
    (α) $\mathbf{M}(\rho) \neq \bot$.
    (β) $\mathbf{M}(\rho) = \mathbf{M}(\sigma) \Rightarrow \mathbf{M}(\rho \diamond \tau) = \mathbf{M}(\sigma \diamond \tau)$.

(b) For all $\mathbf{M} : \mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot}$, $\mathbf{It}(\mathbf{M}) = \{L : \mathbf{M}\ \mathbf{It}\text{-identifies } L\}$.

(c) $\mathbf{It} = \{\mathcal{L} : (\exists \mathbf{M} : \mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot})[\mathcal{L} \subseteq \mathbf{It}(\mathbf{M})]\}$.

Some of our proofs make use of the **Operator Recursion Theorem (ORT)** [Cas74]. **ORT** represents a form of infinitary self-reference, similar to the way in which Kleene's Recursion Theorem [Rog67, page 214, problem 11-4] represents a form of individual self-reference. That is, **ORT** provides a means of forming an infinite computable sequence of programs $e_0, e_1, ...$ such that each program e_i *knows all* programs in the sequence *and* its own index i. The sequence can also be assumed monotone increasing. The first author gives a thorough explanation of **ORT** in [Cas94].

3 Results

This section gives the formal definition of n-**ParIt**-learning and presents our results. Namely, this section shows that 1-**ParIt**-learners are strictly more powerful than **It**-learners (Theorem 1). It also shows that, for all $n \geq 1$, $(n+1)$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners (Theorem 2).

Definition 2. For all $n \geq 1$, (a)-(c) below.

(a) For all $\mathbf{M} : \mathbb{N}_{\#}^{*} \to \mathbb{N}_{?,\bot}^{n}$ and L, **M** n-**ParIt**-*identifies* L $\Leftrightarrow$ (i) and (ii) below.
  (i) For all f representing L, there exist j and $\boldsymbol{p} \in \mathbb{N}^n$ such that $(\forall j' \geq j)$ $[\mathbf{M}(f[j']) = \boldsymbol{p}]$ and $(\forall i < n)[W_{(\boldsymbol{p})_i} = L]$.

[11] **It**-learners are often given a formal definition more in line with their description in Section 1. The definition given herein was inspired, in part, by the Myhill-Nerode Theorem [DSW94]. A proof that this definition is equivalent to the more common definition can be found in [CM07b].

[12] Condition (a)(i) in Definition 1 is equivalent to: **M** **Ex**-identifies L [Gol67, JORS99].

(ii) $(\forall\rho,\sigma,\tau)[\mathbf{M}(\rho)\sqsubseteq\mathbf{M}(\sigma) \;\Rightarrow\; \mathbf{M}(\rho\diamond\tau)\sqsubseteq\mathbf{M}(\sigma\diamond\tau)]$.

(b) For all $\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp}^{n}$, n-$\mathbf{ParIt}(\mathbf{M})=\{L:\mathbf{M}\ n\text{-}\mathbf{ParIt}\text{-identifies } L\}$.

(c) n-$\mathbf{ParIt}=\{\mathcal{L}:(\exists\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp}^{n})[\mathcal{L}\subseteq n\text{-}\mathbf{ParIt}(\mathbf{M})]\}$.

Example 3 (Example 2 revisited). Suppose that $\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp}^{2}$, ρ, σ, $\boldsymbol{p}\in\mathbb{N}_{?,\perp}^{2}$, and x are such that (a)-(e) below.

(a) $\mathbf{M}(\rho)=\mathbf{M}(\sigma)=\boldsymbol{p}$.

(b) $|\boldsymbol{p}|_{\neq\perp}=2$.

(c) $(\boldsymbol{p})_0\neq(\boldsymbol{p})_1$.

(d) In the computation of $\mathbf{M}_0(\rho\diamond x)$, $\mathbf{M}_0$ waits for *either* of $\mathbf{M}_0(\rho)$ or $\mathbf{M}_1(\rho)$ to converge. Then, for the $i\leq 1$ such that $\mathbf{M}_i(\rho)$ converges *first*, $\mathbf{M}_0(\rho\diamond x)=\mathbf{M}_i(\rho)$. Similarly, in the computation of $\mathbf{M}_0(\sigma\diamond x)$, $\mathbf{M}_0$ waits for either of $\mathbf{M}_0(\sigma)$ or $\mathbf{M}_1(\sigma)$ to converge. Then, for the $i\leq 1$ such that $\mathbf{M}_i(\sigma)$ converges *first*, $\mathbf{M}_0(\sigma\diamond x)=\mathbf{M}_i(\sigma)$.

(e) In the computation of $\mathbf{M}(\rho)$, $\mathbf{M}_0(\rho)$ converges before $\mathbf{M}_1(\rho)$; in computation of $\mathbf{M}(\sigma)$, $\mathbf{M}_1(\sigma)$ converges before $\mathbf{M}_0(\sigma)$.

Then, for all L, $\mathbf{M}$ does *not* 2-**ParIt**-identify L, i.e., $\mathbf{M}$ is *not* a 2-**ParIt**-learner.

Proof. By (a) above, $\mathbf{M}(\rho)\sqsubseteq\mathbf{M}(\sigma)$. By (c)-(e) above, $\mathbf{M}_0(\rho\diamond x)=(\boldsymbol{p})_0\neq(\boldsymbol{p})_1=\mathbf{M}_0(\sigma\diamond x)$. Thus, by (b) above, $\mathbf{M}(\rho\diamond x)\not\sqsubseteq\mathbf{M}(\sigma\diamond x)$. But this contradicts condition (a)(ii) in Definition 2. □ (*Example 3*)

Intuitively, the $\mathbf{M}$ described in Example 3 violates Definition 2 because: (1) $\mathbf{M}$ makes use of, not just the *value* of a conjecture, but also the *time* used to compute it; and, (2) the elements of $\mathbb{N}_{?,\perp}$ do *not* capture this information. To overcome this difficulty would require that a learner be defined as object with a more complex range than $\mathbb{N}_{?,\perp}^{n}$. It would be interesting to explore generalizations of Definition 2 that do this.

The following straightforward variant of **It**-learning is used in the proof of Theorem 1.

Definition 3

(a) For all $\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp}$ and L, $\mathbf{M}$ **TotIt**-*identifies* L $\Leftrightarrow$ $\mathbf{M}$ **It**-identifies L, *and*, for *all* ρ, $\mathbf{M}(\rho)\neq\perp$.

(b) For all $\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp}$, $\mathbf{TotIt}(\mathbf{M})=\{L:\mathbf{M}\ \mathbf{TotIt}\text{-identifies } L\}$.

(c) $\mathbf{TotIt}=\{\mathcal{L}:(\exists\mathbf{M}:\mathbb{N}_{\#}^{*}\to\mathbb{N}_{?,\perp})[\mathcal{L}\subseteq\mathbf{TotIt}(\mathbf{M})]\}$.

Recall that if a learner $\mathbf{M}$ **It**-identifies language L, then it is only required that $\mathbf{M}(\rho)\neq\perp$ for those ρ such that $\mathrm{content}(\rho)\subseteq L$. However, if $\mathbf{M}$ **TotIt**-identifies L, then, for *all* ρ, $\mathbf{M}(\rho)\neq\perp$.

The following is a basic fact relating **It** and **TotIt**.

Proposition 1. For all $\mathcal{L}\in\mathbf{It}$, if $\mathbb{N}\in\mathcal{L}$, then $\mathcal{L}\in\mathbf{TotIt}$.

Proof. Straightforward. □ (*Proposition 1*)

The following lemma is used in the proof of Theorem 1.

Lemma 1. Let $\mathcal{L}$ be the class of languages consisting of each L satisfying (a)-(c) below.

(a) $(\forall e \in L)[\varphi_e(0) \neq \bot]$.
(b) $\{\varphi_e(0) : e \in L\}$ is finite.
(c) $L = \bigcup_{e \in L} W_{\varphi_e(0)}$.

Then, $\mathcal{L} \in \mathbf{It} - \mathbf{TotIt}$.

Proof that $\mathcal{L} \in \mathbf{It}$. Let $f : \mathbb{N} \to \mathbb{N}$ be a 1-1, computable function such that, for all a,

$$W_{f(a)} = \bigcup_{e \in D_a} W_e. \tag{2}$$

Let $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\bot}$ be such that $\mathbf{M}(\lambda) = f(0)$, and, for all ρ, a, and e, if $\mathbf{M}(\rho) = f(a)$, then

$$\mathbf{M}(\rho \diamond e) = \begin{cases} f(a), \text{ if } \varphi_e(0) \in D_a; \\ f(b), \text{ if } \varphi_e(0) \in (\mathbb{N} - D_a), \\ \qquad \text{where } b \text{ is such that } D_b = D_a \cup \{\varphi_e(0)\}; \\ \bot, \quad \text{ if } \varphi_e(0) = \bot. \end{cases} \tag{3}$$

Clearly, $\mathcal{L} \subseteq \mathbf{It}(\mathbf{M})$.

Proof that $\mathcal{L} \notin \mathbf{TotIt}$. By way of contradiction, suppose that $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\bot}$ is such that $\mathcal{L} \subseteq \mathbf{TotIt}(\mathbf{M})$. By **ORT**, there exist distinct φ-programs $e_0, e_1, ...$ such that, for all i and x,

$$W_{e_0} = \{e_{j+2} : \varphi_{e_{j+2}}(0) = e_0\}; \tag{4}$$

$$W_{e_1} = \{e_{j+2} : \varphi_{e_{j+2}}(0) = e_1\}; \tag{5}$$

$$\varphi_{e_{i+2}}(x) = \begin{cases} e_1, \text{ if } i \text{ is } \textit{least} \text{ such that} \\ \qquad \mathbf{M}(e_2 \diamond \cdots \diamond e_{i+2}) = \mathbf{M}(e_2 \diamond \cdots \diamond e_{i+1}); \\ e_0, \text{ otherwise.} \end{cases} \tag{6}$$

Consider the following cases.

Case $(\forall i)[\varphi_{e_{i+2}}(0) = e_0]$. Then, clearly, $W_{e_0} = \{e_{j+2} : j \in \mathbb{N}\}$ and $W_{e_0} \in \mathcal{L}$. By the case, for all i, $\mathbf{M}(e_2 \diamond \cdots \diamond e_{i+2}) \neq \mathbf{M}(e_2 \diamond \cdots \diamond e_{i+1})$. But then, clearly, $W_{e_0} \notin \mathbf{It}(\mathbf{M})$.

Case $(\exists i)[\varphi_{e_{i+2}}(0) = e_1]$. Then, clearly, $W_{e_0} = \{e_{j+2} : j \neq i\}$ and $(\forall j \neq i)$ $[\varphi_{e_{j+2}}(0) = e_0]$. Furthermore, $W_{e_1} = \{e_{i+2}\}$ and $\varphi_{e_{i+2}}(0) = e_1$. Thus, $W_{e_0} \cup W_{e_1}$ and W_{e_0} are *distinct* languages in $\mathcal{L}$. Let f and f^- be as follows.

$$f = e_2 \diamond e_3 \diamond \cdots \ . \tag{7}$$

$$f^- = e_2 \diamond e_3 \diamond \cdots \diamond e_{i+1} \diamond e_{i+3} \diamond e_{i+4} \diamond \cdots \ . \tag{8}$$

Clearly, f represents $W_{e_0} \cup W_{e_1}$, and f^- represents W_{e_0}. Let k be such that $\mathbf{M}(f[k]) \in \mathbb{N}$, $\mathbf{M}(f^-[k]) \in \mathbb{N}$, and

$$(\forall k' \geq k)\big[\mathbf{M}(f[k']) = \mathbf{M}(f[k]) \ \wedge \ \mathbf{M}(f^-[k']) = \mathbf{M}(f^-[k])\big]. \tag{9}$$

From the case, it follows that $\mathbf{M}(f[k]) = \mathbf{M}(f^-[k])$. But, clearly, this is a contradiction. □ (*Lemma 1*)

Theorem 1. Let $\mathcal{L}$ be as in Lemma 1. Let $\mathcal{L}'$ be such that $\mathcal{L}' = \mathcal{L} \cup \{\mathbb{N}\}$. Then, $\mathcal{L}' \in 1\text{-}\mathbf{ParIt} - \mathbf{It}$.

Proof (Sketch) that $\mathcal{L}' \in 1\text{-}\mathbf{ParIt}$. By Lemma 1, there exists $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ such that $\mathcal{L} \subseteq \mathbf{It}(\mathbf{M})$. Let z_0 be such that $\varphi_{z_0}(0) = \perp$. Clearly, for all $L \in \mathcal{L}$, $z_0 \notin L$. Consider an $\mathbf{M}' : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ described informally as follows. On any given input sequence, $\mathbf{M}'$ simulates $\mathbf{M}$ until, if ever, $\mathbf{M}'$ is fed z_0. Upon being fed z_0, $\mathbf{M}'$ stops simulating $\mathbf{M}$, and starts outputting a conjecture for $\mathbb{N}$. Clearly, for such an $\mathbf{M}'$, $\mathcal{L}' \subseteq 1\text{-}\mathbf{ParIt}(\mathbf{M}')$.

Proof that $\mathcal{L}' \notin \mathbf{It}$. By way of contradiction, suppose that $\mathcal{L}' \in \mathbf{It}$. Then, by Proposition 1, $\mathcal{L}' \in \mathbf{TotIt}$. Let $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ be such that $\mathcal{L}' \subseteq \mathbf{TotIt}(\mathbf{M})$. Then, $\mathcal{L} \subset \mathcal{L}' \subseteq \mathbf{TotIt}(\mathbf{M})$. But this contradicts Lemma 1. $\approx\Box$ (*Theorem 1*)

Theorem 2. Let $n \geq 1$ be fixed. For each $i < n$, let z_i be any fixed φ-program such that $W_{\varphi_{z_i}(0)} = \{\langle i, z_i \rangle\}$. Let $\mathcal{L}_n$ be the class of languages consisting of each $L \subseteq \{0, ..., n-1\} \times \mathbb{N}$ satisfying *either* (a) *or* (b) below.

(a) (i) and (ii) below.
  (i) $L \cap (\{0, ..., n-1\} \times \{z_0, ..., z_{n-1}\}) = \emptyset$.
  (ii) For each $i < n$, if E is such that $E = \{e \in \mathbb{N} : \langle i, e \rangle \in L\}$, then ($\alpha$)-($\gamma$) below.
    (α) $(\forall e \in E)[\varphi_e(0) \in \mathbb{N}]$.
    (β) $\{\varphi_e(0) : e \in E\}$ is finite.
    (γ) $L = \bigcup_{e \in E} W_{\varphi_e(0)}$.
(b) There exists $i < n$ such that (i) and (ii) below.
  (i) $L \cap (\{0, ..., n-1\} \times \{z_0, ..., z_{n-1}\}) = \{\langle i, z_i \rangle\}$.
  (ii) If E is such that $E = \{e \in \mathbb{N} : \langle i, e \rangle \in L\}$, then ($\alpha$)-($\gamma$) as in (a)(ii) above *for this E*.

Then, for all $n \geq 1$, $\mathcal{L}_{n+1} \in (n+1)\text{-}\mathbf{ParIt} - n\text{-}\mathbf{ParIt}$.

Proof (Sketch) that $\mathcal{L}_{n+1} \in (n+1)\text{-}\mathbf{ParIt}$. Let $n \geq 1$ be fixed. Let $f : \mathbb{N}^2 \to \mathbb{N}$ be a 1-1, computable function such that, for all j and a,

$$W_{f(j,a)} = \bigcup_{e \in D_a} W_e. \tag{10}$$

Let $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}^{n+1}$ be such that, for each $i \leq n$, $\mathbf{M}_i(\lambda) = f(i, 0)$, and, for all ρ, k, and e, $\mathbf{M}_i(\rho \diamond \langle k, e \rangle)$ is

$$\begin{cases} \mathbf{M}_k(\rho), & \text{if } e = z_k; \\ \mathbf{M}_j(\rho), & \text{if } e \neq z_k \ \wedge\ [j \neq i \ \vee\ k \neq i \ \vee\ \varphi_e(0) \in D_a], \\ & \quad \text{where } j \text{ and } a \text{ are such that } \mathbf{M}_i(\rho) = f(j,a); \\ f(i,b), & \text{if } e \neq z_k \ \wedge\ j = i \ \wedge\ k = i \ \wedge\ \varphi_e(0) \in (\mathbb{N} - D_a), \\ & \quad \text{where } j, a, \text{ and } b \text{ are such that } \mathbf{M}_i(\rho) = f(j,a) \\ & \quad \text{and } D_b = D_a \cup \{\varphi_e(0)\}; \\ \perp, & \text{if } e \neq z_k \ \wedge\ \big[[j = i \ \wedge\ k = i \ \wedge\ \varphi_e(0) = \perp] \ \vee\ \mathbf{M}_i(\rho) = \perp\big], \\ & \quad \text{where } j \text{ is such that } \mathbf{M}_i(\rho) = f(j,a), \text{ for some } a. \end{cases} \tag{11}$$

```
STAGE $s = 0$.
    1. For each $i \le n$, set $\varphi_{e_{i+1}}(0) = e_0$.
    2. Set $W_{e_0}^1 = \{\langle 0, e_1\rangle, ..., \langle n, e_{n+1}\rangle\}$.
    3. Set $\rho^1 = \langle 0, e_1\rangle \diamond \cdots \diamond \langle n, e_{n+1}\rangle$.
STAGE $s \ge 1$.
    1. Find $\rho'$, *if any*, such that $\rho^s \subseteq \rho' \subset \rho^s \diamond \#^\omega$ and $|\mathbf{M}(\rho')|_{\ne\perp} = n$.
    2. For $k$ from $n$ down through $-1$, do:
      Wait until, *if ever*, it is discovered that one of the following two conditions applies.
      COND. ($\alpha$): $(\exists \boldsymbol{q} : |\boldsymbol{q}|_{\ne\perp} = n - k)(\forall \sigma \in \{\langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle, \lambda\})$
                  $[\boldsymbol{q} \sqsubseteq \mathbf{M}(\rho' \diamond \sigma \diamond \langle k+1, e_{f(s)+k+1}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)]$.
        a. Set $\varphi_{e_{f(s)+k}}(0) =$ any $\varphi$-program $p$ such that $W_p = \{\langle k, e_{f(s)+k}\rangle\}$.
        b. Proceed to the next value of $k$.
      COND. ($\beta$): $\mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle) \not\sqsubseteq \mathbf{M}(\rho')$.
        a. For each $i \le k$, set $\varphi_{e_{f(s)+i}}(0) = e_0$.
        b. Set $W_{e_0}^{s+1} = W_{e_0}^s \cup \{\langle 0, e_{f(s)}\rangle, ..., \langle k, e_{f(s)+k}\rangle\}$.
        c. Set $\rho^{s+1} = \rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle$.
        d. Go to stage $s + 1$.
      (Note that the iteration of the loop in which $k = n$ is always exited. Also, note
      that if $k$ reaches the value $-1$, then the construction *intentionally* goes into an
      infinite loop.)
```

Fig. 5. The behavior of φ-programs $e_0, e_1, ...$ in the proof of Theorem 2

It can be shown that $\mathcal{L}_{n+1} \subseteq (n+1)$-**ParIt**$(\mathbf{M})$ (details omitted).

Proof that $\mathcal{L}_{n+1} \notin n$-**ParIt**. By way of contradiction, suppose that $n \ge 1$ and $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}^n$ are such that $\mathcal{L}_{n+1} \subseteq n$-**ParIt**$(\mathbf{M})$. Let $f : \mathbb{N} \to \mathbb{N}$ be such that, for all s,

$$f(s) = s \cdot (n+1) + 1. \tag{12}$$

By **ORT**, there exist distinct φ-programs $e_0, e_1, ...$, *none* of which are $z_0, ..., z_n$, and whose behavior is as in Figure 5.

Claim 1. For all $s \ge 1$, if stage s is entered, then (a)-(c) below.
(a) $(\forall \langle i, e\rangle \in W_{e_0}^s)[i \le n \ \wedge \ e \notin \{z_0, ..., z_n\} \ \wedge \ \varphi_e(0) = e_0]$.
(b) content$(\rho^s) = W_{e_0}^s$.
(c) $\rho^s \diamond \#^\omega$ represents $W_{e_0}^s$.

Proof of Claim. (a) is clear by construction. (b) is proven by a straightforward induction. (c) follows immediately from (b). □ (*Claim 1*)

Claim 2. For all $s \ge 1$, if stage s is exited, then there exist ρ' and ρ'' such that $\rho^s \subseteq \rho' \subset \rho'' \subseteq \rho^{s+1}$ and $\mathbf{M}(\rho'') \not\sqsubseteq \mathbf{M}(\rho')$.

Proof of Claim. Clear by construction. □ (*Claim 2*)

If every stage s is exited, then, by Claim 1(a) and Claim 2, $W_{e_0} \in \mathcal{L}_{n+1} - n\text{-}\mathbf{ParIt}(\mathbf{M})$ (a contradiction). So, for the remainder of the proof, suppose that stage s is entered but *never* exited.

If stage s is never exited because there is *no* ρ' such that $\rho^s \subseteq \rho' \subset \rho^s \diamond \#^\omega$ and $|\mathbf{M}(\rho')|_{\neq\perp} = n$, then, by (a) and (c) of Claim 1, $W_{e_0}^s \in \mathcal{L}_{n+1} - n\text{-}\mathbf{ParIt}(\mathbf{M})$ (a contradiction). So, suppose that stage s is never exited because there exists k such that $-1 \leq k < n$ and $(\neg\alpha)$ and $(\neg\beta)$ below.

$(\neg\alpha)$ $(\forall \boldsymbol{q} : |\boldsymbol{q}|_{\neq\perp} = n-k)(\exists \sigma \in \{\langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle, \lambda\})$
$[\boldsymbol{q} \not\sqsubseteq \mathbf{M}(\rho' \diamond \sigma \diamond \langle k+1, e_{f(s)+k+1}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)]$.

$(\neg\beta)$ $\mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle) \sqsubseteq \mathbf{M}(\rho')$.

Claim 3.
$\mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle) \sqsubseteq \mathbf{M}(\rho' \diamond \langle k+1, e_{f(s)+k+1}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)$.

Proof of Claim. Follows from $(\neg\beta)$. □ (*Claim 3*)

By the choice of k, there exists $\boldsymbol{p}$ such that $|\boldsymbol{p}|_{\neq\perp} = n-k-1$ and

$$(\forall \sigma \in \{\langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k+1, e_{f(s)+k+1}\rangle, \lambda\}) \\ [\boldsymbol{p} \sqsubseteq \mathbf{M}(\rho' \diamond \sigma \diamond \langle k+2, e_{f(s)+k+2}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)]. \tag{13}$$

Claim 4. $\boldsymbol{p} = \mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)$.

Proof of Claim. By way of contradiction, suppose otherwise. By (13), it must be the case that

$$\boldsymbol{p} \sqsubset \mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle). \tag{14}$$

But (14) together with Claim 3 contradicts $(\neg\alpha)$. □ (*Claim 4*)

Claim 5.
$\mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle) \sqsubseteq \mathbf{M}(\rho' \diamond \langle k+2, e_{f(s)+k+2}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)$.

Proof of Claim. Immediate by Claim 4 and (13). □ (*Claim 5*)

Let $p = \varphi_{e_{f(s)+k+1}}(0)$. Thus, by construction, $W_p = \{\langle k+1, e_{f(s)+k+1}\rangle\}$. Let $e' \notin \{z_0, ..., z_n, e_0, e_1, ...\}$ and p' be as follows.

$$\varphi_{e'}(0) = p'. \tag{15}$$

$$W_{p'} = \begin{array}{l} \{ \langle k+2, e_{f(s)+k+2}\rangle, ..., \langle n, e_{f(s)+n}\rangle, \\ \quad \langle 0, e_{f(s)}\rangle, ..., \langle k, e_{f(s)+k}\rangle, \langle k+1, e'\rangle \}. \end{array} \tag{16}$$

Let L and L^- be as follows.

$$L = W_{e_0}^s \cup W_p \cup W_{p'} \cup \{\langle k+1, z_{k+1}\rangle\}. \tag{17}$$

$$L^- = W_{e_0}^s \cup W_{p'} \cup \{\langle k+1, z_{k+1}\rangle\}. \tag{18}$$

Clearly, L and L^- are *distinct* languages in $\mathcal{L}_{n+1}$. Let g and g^- be as follows.

$$g = \rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle \\ \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle \diamond \langle k+1, e'\rangle \diamond \langle k+1, z_{k+1}\rangle \diamond \#^\omega. \tag{19}$$

$$g^- = \rho' \diamond \langle k+2, e_{f(s)+k+2}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle \\ \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle \diamond \langle k+1, e'\rangle \diamond \langle k+1, z_{k+1}\rangle \diamond \#^\omega. \tag{20}$$

Clearly, g represents L, and g^- represents L^-. Let ℓ be such that $\mathbf{M}(g[\ell]) \in \mathbb{N}^n$, $\mathbf{M}(g^-[\ell]) \in \mathbb{N}^n$, and

$$(\forall \ell' \geq \ell)\big[\mathbf{M}(g[\ell']) = \mathbf{M}(g[\ell]) \ \wedge\ \mathbf{M}(g^-[\ell']) = \mathbf{M}(g^-[\ell])\big]. \qquad (21)$$

From Claim 5, and the fact that $\mathbf{M}(g[\ell]) \in \mathbb{N}^n$ and $\mathbf{M}(g^-[\ell]) \in \mathbb{N}^n$, it follows that $\mathbf{M}(g[\ell]) = \mathbf{M}(g^-[\ell])$. But, clearly, this is a contradiction. $\approx\Box$ (*Theorem 2*)

Acknowledgments. We are grateful to several anonymous referees for their *meticulous* reading of an earlier draft of this paper.

References

[AMS$^+$93] Arikawa, S., Miyano, S., Shinohara, A., Kuhara, S., Mukouchi, Y., Shinohara, T.: A machine discovery from amino-acid-sequences by decision trees over regular patterns. New Generation Computing 11, 361–375 (1993)

[Ang80] Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62 (1980)

[BUV96] Brazma, A., Ukkonen, E., Vilo, J.: Discovering unbounded unions of regular pattern languages from positive examples. In: Nagamochi, H., Suri, S., Igarashi, Y., Miyano, S., Asano, T. (eds.) ISAAC 1996. LNCS, vol. 1178, Springer, Heidelberg (1996)

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

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

[CCJS06] Carlucci, L., Case, J., Jain, S., Stephan, F.: Memory-limited U-shaped learning. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 244–258. Springer, Heidelberg (2006)

[CJK$^+$01] Case, J., Jain, S., Kaufmann, S., Sharma, A., Stephan, F.: Predictive learning models for concept drift. Theoretical Computer Science, Special Issue for ALT'98, 268, 323–349 (2001)

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

[CL82] Case, J., Lynes, C.: Machine inductive inference and language identification. In: Nielsen, M., Schmidt, E.M. (eds.) Automata, Languages, and Programming. LNCS, vol. 140, pp. 107–115. Springer, Heidelberg (1982)

[CM07a] Case, J., Moelius, S.E.: U-shaped, iterative, and iterative-with-counter learning. In: COLT 2007. LNCS(LNAI), vol. 4539, pp. 172–186. Springer, Berlin (2007)

[CM07b] Case, J., Moelius, S.E.: U-shaped, iterative, and iterative-with-counter learning (expanded version). Technical report, University of Delaware (2007), Available at `http://www.cis.udel.edu/~moelius/publications`

[DSW94] Davis, M., Sigal, R., Weyuker, E.: Computability, Complexity, and Languages, 2nd edn. Academic Press, London (1994)

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

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

[JORS99] Jain, S., Osherson, D., Royer, J., Sharma, A.: Systems that Learn: An Introduction to Learning Theory, 2nd edn. MIT Press, Cambridge (1999)

[KMU95] Kilpeläinen, P., Mannila, H., Ukkonen, E.: MDL learning of unions of simple pattern languages from positive examples. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 252–260. Springer, Heidelberg (1995)

[LW91] Lange, S., Wiehagen, R.: Polynomial time inference of arbitrary pattern languages. New Generation Computing 8, 361–370 (1991)

[LZ96] Lange, S., Zeugmann, T.: Incremental learning from positive data. Journal of Computer and System Sciences 53, 88–103 (1996)

[Nix83] Nix, R.: Editing by examples. Technical Report 280, Department of Computer Science, Yale University, New Haven, CT, USA (1983)

[OSW86] Osherson, D., Stob, M., Weinstein, S.: Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists. MIT Press, Cambridge (1986)

[Rog67] Rogers, H.: Theory of Recursive Functions and Effective Computability. MIT Press, Cambridge (1967) (Reprinted, MIT Press, 1987)

[SA95] Shinohara, T., Arikawa, A.: Pattern inference. In: Lange, S., Jantke, K.P. (eds.) Algorithmic Learning for Knowledge-Based Systems. LNCS, vol. 961, pp. 259–291. Springer, Heidelberg (1995)

[Sal94a] Salomaa, A.: Patterns (The Formal Language Theory Column). EATCS Bulletin 54, 46–62 (1994)

[Sal94b] Salomaa, A.: Return to patterns (The Formal Language Theory Column). EATCS Bulletin 55, 144–157 (1994)

[Shi83] Shinohara, T.: Inferring unions of two pattern languages. Bulletin of Informatics and Cybernetics 20, 83–88 (1983)

[SSS$^+$94] Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S., Arikawa, S.: Knowledge acquisition from amino acid sequences by machine learning system BONSAI. Trans. Information Processing Society of Japan 35, 2009–2018 (1994)

[Wie76] Wiehagen, R.: Limes-erkennung rekursiver funktionen durch spezielle strategien. Electronische Informationverarbeitung und Kybernetik 12, 93–99 (1976)

[Win93] Winskel, G.: The Formal Semantics of Programming Languages: An Introduction. In: Foundations of Computing Series, MIT Press, Cambridge (1993)

[Wri89] Wright, K.: Identification of unions of languages drawn from an identifiable class. In: Rivest, R., Haussler, D., Warmuth, M. (eds.) Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, California, pp. 328–333. Morgan Kaufmann, San Francisco (1989)