Theoretical Computer Science 410 (2009) 1863–1875

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Parallelism increases iterative learning power[☆]

John Case, Samuel E. Moelius III [*]

Department of Computer & Information Sciences, University of Delaware, 103 Smith Hall, Newark, DE 19716, United States

ARTICLE INFO

Keywords:
Computational learning theory
Gold-style learning
Inductive inference
Iterative learning
Language learning
Memory limited learning
Parallel learning
Parallelism

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) [34,22,9,7,12] is a mathematical model of language learning in the style of Gold [15].[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 an instance of a more general type of learner. Let $x^0, x^1, \ldots$ be an arbitrary input sequence. Let p^0 be **M**'s initial conjecture (i.e., **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 **M** on inputs p^j and x^j. In the event that $\mathbf{M}^j$ diverges, we let $p^{j+1} = \bot$. (By convention, p^0 can*not* be $\bot$.) We shall refer to $\mathbf{M}^j$ as the *jth instantiation of* **M**. See Fig. 1.

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

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

☆ This paper is an expanded version of [J. Case, S.E. Moelius, Parallelism increases iterative learning power, in: Proceedings of the Eighteenth Annual Conference on Algorithmic Learning Theory, ALT'07, in: Lecture Notes in Artificial Intelligence, vol. 4754, Springer, Heidelberg, 2007, pp. 49–63].

* Corresponding author.
E-mail addresses: case@cis.udel.edu (J. Case), moelius@cis.udel.edu (S.E. Moelius III).

1 In this paper, we focus exclusively on language learning, as opposed to, say, function learning [18].

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 (in the sense of the just previous footnote).

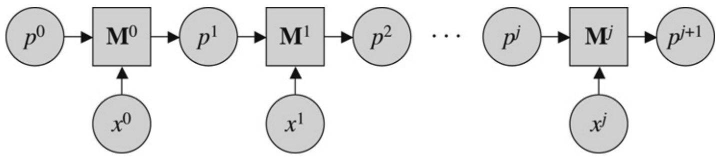

Fig. 1. The iterative learning process. The jth instantiation of learner $\mathbf{M}$, $\mathbf{M}^j$, is fed the current conjecture p^j and current input element x^j. From these, $\mathbf{M}^j$ produces a new conjecture p^{j+1}.

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 [2]. Since then, much work has been done on the learnability of pattern languages [28,29,8] and finite unions thereof [30,36,19,4]. The class of pattern languages, itself, was shown to be **It**-learnable by Lange and Wiehagen [21]. Subsequently, this result was extended by Case, *et al.* [9] who showed that, for each k, the class formed by taking the union of all choices of k pattern languages is **It**-learnable. Nix [24], as well as Shinohara and Arikawa [27], outline interesting applications of pattern inference algorithms.

It-learning is a *memory limited* special case of the more general *explanatory learning* (**Ex**-learning) [15,18][4] and *behaviorally correct learning* (**Bc**-learning) [10,18].[5] **Ex**- and **Bc**-learners are *not*, in general, limited 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 [1,32,27]. Furthermore, there is benefit in knowing that a class of languages is **It**-learnable, as **It**-learners satisfy the following informal property.

Property 1. *If an input sequence represents a language identifiable by the learner, then each element of the sequence may be discarded before the next 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). 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).[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 $\mathbf{M}^0, \mathbf{M}^1, \ldots$ 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 $\mathbf{M}^0, \mathbf{M}^1, \ldots$ to run in parallel. We do *not* require that each of $\mathbf{M}^0, \mathbf{M}^1, \ldots$ 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* [35]. 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 $x^0, x^1, \ldots \overset{\text{def}}{\Leftrightarrow}$ for some index j_0,

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

[4] **Ex**-learning is the model that was actually studied by Gold [15].

[5] Other memory limited learning models are considered in [25,14,9,7].

[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* [31,18].

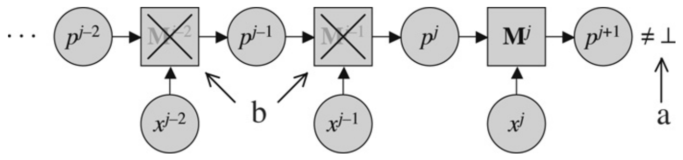

Fig. 2. How a 1-**ParIt**-learner **M** may be implemented. Once $\mathbf{M}^j$ has converged (i.e., has resulted in something other than $\perp$) (a), any previous instantiations of **M** that are still running may be forcibly terminated (b).

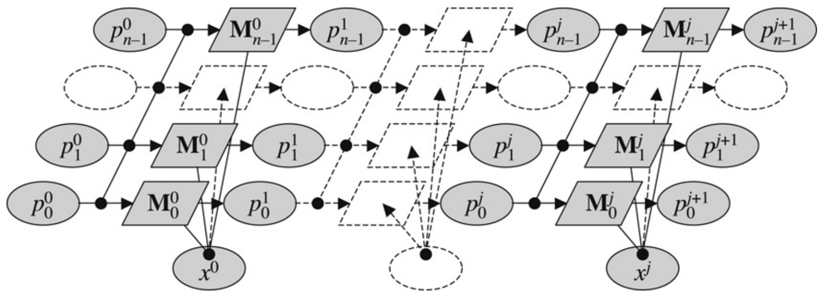

Fig. 3. A *collective* learner resulting from our second extension. For each $i < n$, and each j, ith constituent learner $\mathbf{M}_i$ may consider conjectures $p_0^j, \ldots, p_{n-1}^j$ and input element x^j when forming conjecture p_i^{j+1}.

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 $\mathbf{M}^0, \ldots, \mathbf{M}^{j-1}$. (The idea is that once $\mathbf{M}^j$ has converged, the results of any previous instantiations of **M** are no longer needed. See Fig. 2.)

Clearly, a learner implemented in this way will *not* satisfy Property 1. However, if $x^0, x^1, \ldots$ represents a language identifiable by **M**, then, for some index j_0, each of $\mathbf{M}^{j_0}, \mathbf{M}^{j_0+1}, \ldots$ 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, Corollary 16 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 constituent learners.

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

Our second extension is to allow $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ to run in parallel. For each $i < n$, and each j, we allow $\mathbf{M}_i^j$ to consider $p_0^j, \ldots, p_{n-1}^j$ and x^j when forming conjecture p_i^{j+1} (see Fig. 3). However, as in the 1-ary case, we require that each $\mathbf{M}_i$ be monotonic.[8] So, if $\mathbf{M}_i^j$ *needs* $p_{i'}^j$ to compute p_i^{j+1}, *and*, $\mathbf{M}_{i'}^{j-1}$ diverges, then $\mathbf{M}_i^j$ also diverges. The following examples give some intuition as to which strategies $\mathbf{M}_i^j$ may employ, and which strategies $\mathbf{M}_i^j$ may *not* employ, in considering $p_0^j, \ldots, p_{n-1}^j$. *Exactly* which such strategies $\mathbf{M}_i^j$ may employ is made formal by Definition 8 in Section 3.

Example 3. $\mathbf{M}_i^j$ *may* employ any of the following strategies in considering $p_0^j, \ldots, p_{n-1}^j$.

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

8 In this context, monotonicity is equivalent to *continuity* [35], since each $\mathbf{M}_i^j$ operates on only *finitely much* data.

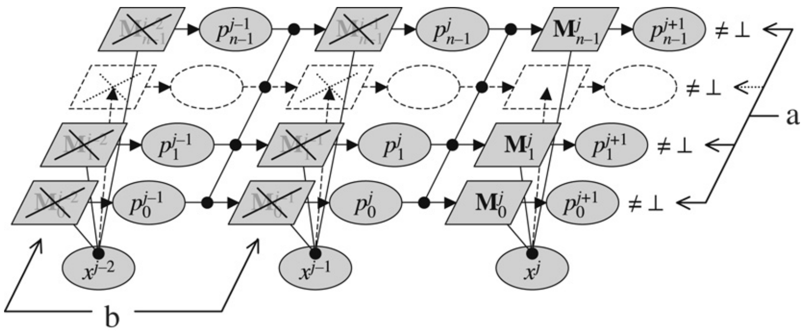

Fig. 4. How an n-**ParIt**-learner $\mathbf{M} = (\mathbf{M}_0, \ldots, \mathbf{M}_{n-1})$ may be implemented. Once *each* of $\mathbf{M}_0^j, \ldots, \mathbf{M}_{n-1}^j$ has converged (a), any previous instantiations of $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ that are still running may be forcibly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

(d) $\mathbf{M}_i^j$ waits for each of $\mathbf{M}_0^{j-1}, \ldots, \mathbf{M}_{n-1}^{j-1}$ to converge, in some predetermined order. Then, $\mathbf{M}_i^j$ uses each of $p_0^j, \ldots, p_{n-1}^j$ to compute p_i^{j+1}.

Example 4. In general, $\mathbf{M}_i^j$ may *not* employ the following strategy in considering $p_0^j, \ldots, p_{n-1}^j$ when $n \geq 2$.

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

Example 4 is revisited following Definition 8 in Section 3.

Let $\mathbf{M} = (\mathbf{M}_0, \ldots, \mathbf{M}_{n-1})$. 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 $x^0, x^1, \ldots \overset{\text{def}}{\Leftrightarrow}$ for some index j_0, and each $i < n$,

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

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 $\mathbf{M}_0^j, \ldots, \mathbf{M}_{n-1}^j$ converges, forcibly terminate any currently running instantiations of the form $\mathbf{M}_i^0, \ldots, \mathbf{M}_i^{j-1}$, where $i < n$. (The idea is that once *each* of $\mathbf{M}_0^j, \ldots, \mathbf{M}_{n-1}^j$ has converged, the results of any previous instantiations of $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ are no longer needed. See Fig. 4.)

Clearly, if $x^0, x^1, \ldots$ represents a language identifiable by $\mathbf{M}$, then, for all but finitely many j, each of $\mathbf{M}_0^j, \ldots, \mathbf{M}_{n-1}^j$ 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.[9]

Our second main result, Theorem 17 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.

$$\begin{tikzcd}
1\text{-}\mathbf{ParIt} \arrow[r] & 2\text{-}\mathbf{ParIt} \arrow[r] & \cdots \arrow[r] & \mathbf{SD}' \arrow[r] & \mathbf{Ex} \\
\mathbf{It} \arrow[u] \arrow[rrr] & & & \mathbf{SD} \arrow[u] &
\end{tikzcd}$$

That is, 1-**ParIt**-learners are strictly more powerful than **It**-learners (Corollary 16). Furthermore, for all $n \geq 1$, $(n+1)$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners (Theorem 17). Thus, we think it is fair to say: parallelism increases iterative learning power.

We also show that, for all $n \geq 1$, n-**ParIt**-learners and set-driven learners (**SD**-learners, Definition 6) are incomparable in learning power (Theorem 15 and Proposition 18). However, by lessening the restrictions of **SD**-learning only slightly, one obtains a form of learning that, while still strictly weaker than **Ex**, completely subsumes the n-**ParIt** hierarchy (Theorem 19).

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. Section 4 concludes.

[9] In addition to satisfying this intuitive property, there is some chance that n-**ParIt**-learners might be used to model cognitive processes. In particular, even human language processing appears to exploit parallelism [16,17].

2. Notation and preliminaries

Computability-theoretical concepts not explained below are treated in [26].

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. $\mathbb{N}_? \stackrel{\text{def}}{=} \mathbb{N} \cup \{?\}$. $\mathbb{N}_{?,\bot} \stackrel{\text{def}}{=} \mathbb{N}_? \cup \{\bot\}$. $\mathbb{N}_\# \stackrel{\text{def}}{=} \mathbb{N} \cup \{\#\}$. 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 $\mathbb{N}_{?,\bot}$. $\vec{p}$ and $\vec{q}$ will be used to denote tuples whose elements are drawn from $\mathbb{N}_{?,\bot}$. The *size* of $\vec{p}$ and $\vec{q}$ will be made clear whenever they are introduced. For all n, all $\vec{p} \in \mathbb{N}^n_{?,\bot}$, and all $i < n$, $(\vec{p})_i$ denotes the ith element of $\vec{p}$, 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 $D_0 = \emptyset$ [26]. 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. For the remainder, we will write $A \times B$ for $\{\langle a, b\rangle : a \in A \wedge b \in B\}$.

For all p and q, $p \sqsubseteq q \stackrel{\text{def}}{\Leftrightarrow} [p = \bot \vee p = q]$. The relation $\sqsubseteq$ is pronounced *approximates* or *is less or the same information as*.[10] For all n, and all $\vec{p}, \vec{q} \in \mathbb{N}^n_{?,\bot}$, $\vec{p} \sqsubseteq \vec{q} \stackrel{\text{def}}{\Leftrightarrow} (\forall i < n)[(\vec{p})_i \sqsubseteq (\vec{q})_i]$. For all n, and all $\vec{p} \in \mathbb{N}^n_{?,\bot}$, $|\vec{p}|_{\neq\bot} \stackrel{\text{def}}{=} |\{i < n : (\vec{p})_i \neq \bot\}|$. So, for example, $|(0, 1, \bot, \bot, ?)|_{\neq\bot} = 3$.

$\varphi_0, \varphi_1, \ldots$ denotes any fixed, acceptable numbering of all *partial* computable functions of type $\mathbb{N} \rightharpoonup \mathbb{N}$ [26]. For each i, we will treat φ_i as a *total* function of type $\mathbb{N} \to \mathbb{N}_\bot$, where $\bot$ denotes the value of a divergent computation.[11] For all i, $W_i \stackrel{\text{def}}{=} \{x \in \mathbb{N} : \varphi_i(x) \neq \bot\}$. Thus, for all i, W_i is the ith recursively enumerable set [26].

$\mathbb{N}^*_\#$ 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 $f \in \mathbb{N}^{\leq\omega}_\#$, $\text{content}(f) \stackrel{\text{def}}{=} \{y \in \mathbb{N} : (\exists x)[f(x) = y]\}$. For all $f \in \mathbb{N}^{\leq\omega}_\#$ and L, f *represents* $L \stackrel{\text{def}}{\Leftrightarrow} f$ is total and $\text{content}(f) = L$.[12] 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,

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

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

The following are the formal definitions of **It**- and **SD**-learning.[13]

Definition 5 (*Wiehagen [34]*). (a) For all $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\bot}$ and L, $\mathbf{M}$ **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 $W_p = L$.[14]

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

Definition 6 (*Wexler and Culicover [33]*). (a) For all $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\bot}$ and L, $\mathbf{M}$ **SD**-*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 $W_p = L$.

(ii) For all ρ and σ, if $\text{content}(\rho) = \text{content}(\sigma)$, then $\mathbf{M}(\rho) = \mathbf{M}(\sigma)$.

(iii) For all ρ, if $\text{content}(\rho) \subseteq L$, then $\mathbf{M}(\rho) \neq \bot$.

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

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

The formal definition of $\mathbf{SD}'$-learning is obtained from Definition 6 by dropping condition (a)(iii). That $\mathbf{SD}' \subset \mathbf{Ex}$ is shown by the proof of Proposition 5.25 in [18].

The following result, due to Kinber and Stephan [20], will be of relevance.

[10] Quoting Winskel [35, page 72]: "The elements [of $\mathbb{N}_{?,\bot}$] are thought of as points of information and the ordering $p \sqsubseteq q$ as meaning p approximates q (or, p is less or the same information as q) — so $\bot$ is the point of least information." (Note: the variables in the preceding quote were changed in order to adhere to the conventions of this paper.)

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

[12] Such an f is often called a *text* (for L) [18].

[13] **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 [13]. A proof that this definition is equivalent to the more common definition can be found in [12].

[14] Condition (a)(i) in Definition 5 is equivalent to: **M Ex**-identifies L [15,18].

Theorem 7 *([20, Theorem 7.7(c)]).* **It** $\subseteq$ **SD**.

Some of our proofs make use of the **Operator Recursion Theorem (ORT)** [5]. **ORT** represents a form of infinitary self-reference, similar to the way in which Kleene's Recursion Theorem [26, 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 $e_0, e_1, \ldots$ 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 [6].

3. Results

This section gives the formal definition of n-**ParIt**-learning and presents our results. To recap, our main results are:

- 1-**ParIt**-learners are strictly more powerful than **It**-learners (Corollary 16);
- for all $n \geq 1$, $(n+1)$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners (Theorem 17);
- for all $n \geq 1$, n-**ParIt**-learners and **SD**-learners are incomparable in learning power (Theorem 15 and Proposition 18); and,
- for all $n \geq 1$, **SD**$'$-learners are strictly more powerful than n-**ParIt**-learners (Theorem 19).

Definition 8. For all $n \geq 1$, (a)–(c) below.

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

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

Example 9 (*Example 4 Revisited*). Suppose that $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}^2_{?,\perp}$, $\rho, \sigma, \vec{p} \in \mathbb{N}^2_{?,\perp}$, and x are such that (a)–(e) below.

(a) $\mathbf{M}(\rho) = \mathbf{M}(\sigma) = \vec{p}$.
(b) $|\vec{p}|_{\neq\perp} = 2$.
(c) $(\vec{p})_0 \neq (\vec{p})_1$.
(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 the 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, $\mathbf{M}_0(\rho \diamond x) = (\vec{p})_0 \neq (\vec{p})_1 = \mathbf{M}_0(\sigma \diamond x)$. Finally, by (b) above, $\mathbf{M}(\rho \diamond x) \not\sqsubseteq \mathbf{M}(\sigma \diamond x)$. Thus, $\mathbf{M}$ does *not* satisfy condition (a)(ii) in Definition 8. □ (Example 9)

Though the $\mathbf{M}$ of Example 9 violates Definition 8, such a learner could certainly exist in the real world. For example, the reader familiar with the Message Passing Interface (MPI) [23] (see also [3]) can likely imagine an implementation of $\mathbf{M}$ where the constituent learners, $\mathbf{M}_0$ and $\mathbf{M}_1$, are distinct processes that communicate by passing messages. So, for example, since $\mathbf{M}_0$ needs $(\vec{p})_1$ to compute $\mathbf{M}_0(\sigma \diamond x)$, $\mathbf{M}_1$ could send a message to $\mathbf{M}_0$ with $(\vec{p})_1$ as its contents. Note, however, that in such an implementation, $\mathbf{M}_0$ would need to be able to wait for messages from multiple sources *simultaneously*. That is, since $\mathbf{M}_0$ does *not* know beforehand *which* of $(\vec{p})_0$ and $(\vec{p})_1$ should be the result of $\mathbf{M}_0(\sigma \diamond x)$, $\mathbf{M}_0$ would need to be able to receive a message from *either* $\mathbf{M}_0$ *or* $\mathbf{M}_1$. MPI provides constructs for handling exactly this sort of situation.[15] Very roughly, it is the fact that an implementation of $\mathbf{M}$ would require such constructs that causes $\mathbf{M}$ to violate Definition 8.

More generally, the $\mathbf{M}$ of Example 9 makes use of, not just the *value* of a conjecture, but also the *time* used to compute it. However, 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 $\mathbb{N}^n_{?,\perp}$. It would be interesting to explore generalizations of Definition 8 that do this.

The following straightforward variant of **SD**-learning is used in the proof of Theorem 15.

Definition 10. (a) For all $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\perp}$ and L, $\mathbf{M}$ **TotSD**-*identifies* L $\Leftrightarrow$ $\mathbf{M}$ **SD**-identifies L, *and*, for *all* ρ, $\mathbf{M}(\rho) \neq \perp$.
(b) For all $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\perp}$, **TotSD**$(\mathbf{M}) = \{L : \mathbf{M}$ **TotSD**-identifies $L\}$.
(c) **TotSD** $= \{\mathcal{L} : (\exists \mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}_{?,\perp})[\mathcal{L} \subseteq$ **TotSD**$(\mathbf{M})]\}$.

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

[15] For example, in a call to MPI_Recv (to receive a message), the calling process must indicate *from whom* it is willing to receive a message. This argument of the call may uniquely identify some process, or it may be the constant MPI_ANY_SOURCE, which indicates that the calling process is willing to receive a message from anyone.

The following is a basic fact relating **SD** and **TotSD**.

Proposition 11. *For all* $\mathcal{L} \in \textbf{SD}$*, if* $\mathbb{N} \in \mathcal{L}$*, then* $\mathcal{L} \in \textbf{TotSD}$*.*

Proof of Proposition. Straightforward. □ (Proposition 11)

[11] employs a form of *total iterative* learning, **TotIt**, analogous to the form of total set-driven learning, **TotSD**, given by Definition 10. The following proposition relates these two criteria.

Proposition 12. $\textbf{TotIt} \subseteq \textbf{TotSD}$.

Proof of Proposition. It is easily seen that if one applies the technique of [20, Theorem 7.7(c)] (Theorem 7) to a total iterative learner, then a total set-driven learner results. □ (Proposition 12)

The following lemma is used in the proof of Theorem 15.

Lemma 13. *Let* $\mathcal{L}$ *be the class of languages consisting of each* L *satisfying* (a)–(c) *below.*

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

Then, $\mathcal{L} \in \textbf{It} - \textbf{TotSD}$*.*

Proof that $\mathcal{L} \in \textbf{It}$. 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 $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ be such that $\mathbf{M}(\lambda) = f(0)$, and, for all ρ, a, and e, if $\mathbf{M}(\rho) = f(a)$, then

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

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

Proof that $\mathcal{L} \notin \textbf{TotSD}$. By way of contradiction, suppose that $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ is such that $\mathcal{L} \subseteq \textbf{TotSD}(\mathbf{M})$. By **ORT**, there exists a computable sequence of pairwise distinct φ-programs $e_0, e_1, \ldots$ such that, for all i and x,

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

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

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

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, $W_{e_0} \notin \textbf{SD}(\mathbf{M})$.

CASE $(\exists i)[\varphi_{e_{i+2}}(0) = e_1]$. Then, clearly, $W_{e_0} = \{e_{j+2} : j < i\}$ and $(\forall j < i)\ [\varphi_{e_{j+2}}(0) = e_0]$. 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.

$$f = e_2 \diamond e_3 \diamond \cdots \diamond e_{i+2} \diamond \#^\omega. \tag{7}$$

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

Clearly, f represents $W_{e_0} \cup W_{e_1}$, and f^- represents W_{e_0}. Note that, for all $j \ge i$,

$$\text{content}(f[j+1]) = \text{content}(f[i+1]) \ \wedge\ \text{content}(f^-[j]) = \text{content}(f^-[i]). \tag{9}$$

Thus,

$$(\forall j \ge i)\big[\mathbf{M}(f[j+1]) = \mathbf{M}(f[i+1]) \ \wedge\ \mathbf{M}(f^-[j]) = \mathbf{M}(f^-[i])\big]. \tag{10}$$

Furthermore, by the case,

$$\mathbf{M}(f[i+1]) = \mathbf{M}(f^-[i]). \tag{11}$$

But, clearly, this is a contradiction. □ (Lemma 13)

Proposition 14 (*Folklore*). $\textbf{TotIt} \subset \textbf{It}$.

Proof of Proposition. Immediate by Proposition 12 and Lemma 13. □ (Proposition 14)

Theorem 15. *Let $\mathcal{L}$ be as in Lemma 13. Let $\mathcal{L}'$ be such that $\mathcal{L}' = \mathcal{L} \cup \{\mathbb{N}\}$. Then, $\mathcal{L}' \in$ 1-**ParIt** − **SD**.*

*Proof that $\mathcal{L}' \in$ 1-**ParIt**.* By Lemma 13, 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 $\mathcal{L}' \notin$ **SD**.* By way of contradiction, suppose that $\mathcal{L}' \in \mathbf{SD}$. Then, by Proposition 11, $\mathcal{L}' \in \mathbf{TotSD}$. Let $\mathbf{M} : \mathbb{N}_\#^* \to \mathbb{N}_{?,\perp}$ be such that $\mathcal{L}' \subseteq \mathbf{TotSD}(\mathbf{M})$. Then, $\mathcal{L} \subset \mathcal{L}' \subseteq \mathbf{TotSD}(\mathbf{M})$. But this contradicts Lemma 13. □ (Theorem 15)

Corollary 16. *Let $\mathcal{L}'$ be as in Theorem 15. Then, $\mathcal{L}' \in$ 1-**ParIt** − **It**.*

Proof of Corollary. Immediate by Theorems 7 and 15. □ (Corollary 16)

Theorem 17. *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 $L \subseteq \{0, \ldots, n-1\} \times \mathbb{N}$ satisfying* either *(a)* or *(b) below.*

(a) (i) *and* (ii) *below.*
  (i) $L \cap (\{0, \ldots, n-1\} \times \{z_0, \ldots, z_{n-1}\}) = \emptyset$.
  (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 $k < n$ such that* (i) *and* (ii) *below.*
  (i) $L \cap (\{0, \ldots, n-1\} \times \{z_0, \ldots, z_{n-1}\}) = \{\langle k, z_k\rangle\}$.
  (ii) *If E is such that $E = \{e \in \mathbb{N} : \langle k, e\rangle \in L\}$, then ($\alpha$)–($\gamma$) as in* (a)(ii) *above* for this E.

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

*Proof that $\mathcal{L}_{n+1} \in (n+1)$-**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,

$$W_{f(j,a)} = \bigcup_{e \in D_a} W_e. \tag{12}$$

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,

$$\mathbf{M}_i(\rho \diamond \langle k, e\rangle) = \begin{cases} \mathbf{M}_k(\rho), & \text{if } (*)\ [k \neq i \ \wedge\ e = z_k]; \\ \mathbf{M}_j(\rho), & \text{if } \neg(*) \ \wedge\ \mathbf{M}_i(\rho) \in \mathbb{N} \ \wedge\ [j \neq i \ \vee\ \varphi_e(0) \in D_a], \\ & \quad \text{where } j \text{ and } a \text{ are such that } \mathbf{M}_i(\rho) = f(j, a); \\ f(i, b), & \text{if } \neg(*) \ \wedge\ \mathbf{M}_i(\rho) \in \mathbb{N} \ \wedge\ j = i \ \wedge\ \varphi_e(0) \in (\mathbb{N} - D_a), \\ & \quad \text{where } j, a, \text{ and } b \text{ are such that } \mathbf{M}_i(\rho) = f(j, a) \\ & \quad \text{and } D_b = D_a \cup \{\varphi_e(0)\}; \\ \perp, & \text{if } \neg(*) \ \wedge\ \mathbf{M}_i(\rho) \in \mathbb{N} \ \wedge\ j = i \ \wedge\ \varphi_e(0) = \perp, \\ & \quad \text{where } j \text{ is such that } \mathbf{M}_i(\rho) = f(j, a), \text{ for some } a; \\ \perp, & \text{if } \neg(*) \ \wedge\ \mathbf{M}_i(\rho) = \perp. \end{cases} \tag{13}$$

Let $L \in \mathcal{L}_{n+1}$ be fixed. Clearly, if L is a language of type (a) in the statement of the theorem, then, for each $i \leq n$, $\mathbf{M}_i$ identifies L. So, suppose that L is a language of type (b) in the statement of the theorem. Let k be such that $\langle k, z_k\rangle \in L$. Then, clearly, $\mathbf{M}_k$ identifies L. Furthermore, by a straightforward induction, it can be shown that, for each $i \leq n$, and all ρ and σ such that $\text{content}(\rho) \cup \text{content}(\sigma) \subseteq L$, $\mathbf{M}_i(\rho \diamond \langle k, z_k\rangle \diamond \sigma) = \mathbf{M}_k(\rho \diamond \langle k, z_k\rangle \diamond \sigma)$. Thus, for each $i \leq n$ such that $i \neq k$, $\mathbf{M}_i$ identifies L as well.

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

$$f(s) = s \cdot (n+1) + 1. \tag{14}$$

By **ORT**, there exists a computable sequence of pairwise distinct φ-programs $e_0, e_1, \ldots$, *none* of which are $z_0, \ldots, z_n$, and whose behavior is as in Fig. 5.

Claim 1. *For all $s \geq 1$, if stage s is entered, then* (a)–(c) *below.*

(a) $(\forall \langle i, e\rangle \in W_{e_0}^s)[i \leq n \ \wedge\ e \notin \{z_0, \ldots, z_n\} \ \wedge\ \varphi_e(0) = e_0]$.
(b) *content*$(\rho^s) = W_{e_0}^s$.
(c) $\rho^s \diamond \#^\omega$ *represents* $W_{e_0}^s$.

```
STAGE $s = 0$.
  1. For each $i \le n$, set $\varphi_{e_{i+1}}(0) = e_0$.
  2. Set $W^1_{e_0} = \{\langle 0, e_1\rangle, \ldots, \langle n, e_{n+1}\rangle\}$.
  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 $|\mathbf{M}(\rho')|_{\neq\perp} = n$.
  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. $(\alpha)$: $(\exists \vec{q} : |\vec{q}|_{\neq\perp} = n - k)(\forall \sigma \in \{\langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle, \lambda\})$
                   $[\vec{q} \sqsubseteq \mathbf{M}(\rho' \diamond \sigma \diamond \langle k+1, e_{f(s)+k+1}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)]$.
       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 $W^{s+1}_{e_0} = W^s_{e_0} \cup \{\langle 0, e_{f(s)}\rangle, \ldots, \langle k, e_{f(s)+k}\rangle\}$.
       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 $e_0, e_1, \ldots$ in the proof of Theorem 17.

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, $W_{e_0} \in \mathcal{L}_{n+1} - n$-**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, $W^s_{e_0} \in \mathcal{L}_{n+1} - n$-**ParIt**($\mathbf{M}$) (a contradiction). So, suppose that stage s is never exited because there exists k such that $-1 \le k < n$ and $(\neg\alpha)$ and $(\neg\beta)$ below.

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

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 $\vec{p}$ such that $|\vec{p}|_{\neq\perp} = n - k - 1$ and

$$(\forall \sigma \in \{\langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k+1, e_{f(s)+k+1}\rangle, \lambda\}) \\ [\vec{p} \sqsubseteq \mathbf{M}(\rho' \diamond \sigma \diamond \langle k+2, e_{f(s)+k+2}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle)]. \tag{15}$$

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

Proof of Claim. By way of contradiction, suppose otherwise. By (15), it must be the case that

$$\vec{p} \sqsubset \mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle). \tag{16}$$

Thus, there must exist $\vec{q}$ such that $|\vec{q}|_{\neq\perp} = |\vec{p}|_{\neq\perp} + 1 = n - k$ and

$$\vec{q} \sqsubseteq \mathbf{M}(\rho' \diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle). \tag{17}$$

By (17) and Claim 3,

$$\vec{q} \sqsubseteq \mathbf{M}(\rho' \diamond \langle k+1, e_{f(s)+k+1}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle). \tag{18}$$

But this 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 (15). □ (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 $e' \notin \{z_0, \ldots, z_n, e_0, e_1, \ldots\}$ and p' be as follows.

$$\varphi_{e'}(0) = p'. \tag{19}$$

$$W_{p'} = \{\langle k+2, e_{f(s)+k+2}\rangle, \ldots, \langle n, e_{f(s)+n}\rangle, \langle 0, e_{f(s)}\rangle, \ldots, \langle k, e_{f(s)+k}\rangle, \langle k+1, e'\rangle\}. \tag{20}$$

Let L and L^- be as follows.

$$L = W^s_{e_0} \cup W_p \cup W_{p'} \cup \{\langle k+1, z_{k+1}\rangle\}. \tag{21}$$

$$L^- = W^s_{e_0} \cup W_{p'} \cup \{\langle k+1, z_{k+1}\rangle\}. \tag{22}$$

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

$$\begin{aligned} g = \rho' &\diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle \\ &\diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle \diamond \langle k+1, e'\rangle \diamond \langle k+1, z_{k+1}\rangle \diamond \#^\omega. \end{aligned} \tag{23}$$

$$\begin{aligned} g^- = \rho' &\diamond \langle k+2, e_{f(s)+k+2}\rangle \diamond \cdots \diamond \langle n, e_{f(s)+n}\rangle \\ &\diamond \langle 0, e_{f(s)}\rangle \diamond \cdots \diamond \langle k, e_{f(s)+k}\rangle \diamond \langle k+1, e'\rangle \diamond \langle k+1, z_{k+1}\rangle \diamond \#^\omega. \end{aligned} \tag{24}$$

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

$$(\forall \ell' \geq \ell)\big[\mathbf{M}(g[\ell']) = \mathbf{M}(g[\ell]) \ \wedge\ \mathbf{M}(g^-[\ell']) = \mathbf{M}(g^-[\ell])\big]. \tag{25}$$

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. □ (Theorem 17)

Proposition 18. *Let $\mathcal{L}$ be such that*

$$\mathcal{L} = \big\{\{0, \ldots, m\} : m \in \mathbb{N}\big\} \cup \big\{\mathbb{N} - \{0\}\big\}. \tag{26}$$

*Then, for all $n \geq 1$, $\mathcal{L} \in$ **SD** $-$ n-**ParIt**.*

Proof. It is already known that $\mathcal{L} \in \mathbf{SD}$ [20, remark on page 238]. So, by way of contradiction, let $n \geq 1$ and $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}^n_{?,\perp}$ be such that $\mathcal{L} \subseteq n\text{-}\mathbf{ParIt}(\mathbf{M})$. Let f be as follows.

$$f = 1 \diamond 2 \diamond \cdots. \tag{27}$$

Clearly, f represents $\mathbb{N} - \{0\}$. Thus, there exist j and $\vec{p} \in \mathbb{N}^n$ such that $(\forall j' \geq j)[\mathbf{M}(f[j']) = \vec{p}]$ and $(\forall i < n)[W_{(\vec{p})_i} = \mathbb{N} - \{0\}]$. Let g and g^- be as follows.

$$g = 1 \diamond 2 \diamond \cdots \diamond j+1 \diamond 0 \diamond \#^\omega. \tag{28}$$

$$g^- = 1 \diamond 2 \diamond \cdots \diamond j \diamond 0 \diamond \#^\omega. \tag{29}$$

Clearly, g represents $\{0, \ldots, j+1\}$, and g^- represents $\{0, \ldots, j\}$. Note that

$$g[j+1] = f[j+1] \ \wedge\ g^-[j] = f[j]. \tag{30}$$

Thus, by the choice of j,

$$\mathbf{M}(g[j+1]) = \mathbf{M}(g^-[j]) \ (= \vec{p}). \tag{31}$$

Let $k \geq j$ be such that $\mathbf{M}(g[k+1]) \in \mathbb{N}^n$, $\mathbf{M}(g^-[k]) \in \mathbb{N}^n$, and

$$(\forall k' \geq k)\big[\mathbf{M}(g[k'+1]) = \mathbf{M}(g[k+1]) \ \wedge\ \mathbf{M}(g^-[k']) = \mathbf{M}(g^-[k])\big]. \tag{32}$$

From (31), it follows that $\mathbf{M}(g[k+1]) = \mathbf{M}(g^-[k])$. But, clearly, this is a contradiction. □ (Proposition 18)

Theorem 19. *For all $n \geq 1$, n-**ParIt** $\subseteq$ **SD**$'$.*

Proof. This proof is similar to that of [20, Theorem 7.7(c)] (Theorem 7). Let $n \geq 1$ and $\mathcal{L} \in n\text{-}\mathbf{ParIt}$ be fixed. Let $\mathbf{M} : \mathbb{N}^*_\# \to \mathbb{N}^n_{?,\perp}$ be such that $\mathcal{L} \subseteq n\text{-}\mathbf{ParIt}(\mathbf{M})$. Let f be a computable function such that, for each finite set $A = \{x_0 < x_1 < \cdots < x_{\ell-1}\} \subseteq \mathbb{N}$,

$$f(A) = \# \diamond x_0 \diamond \# \diamond x_1 \diamond \cdots \diamond \# \diamond x_{\ell-1}. \tag{33}$$

Clearly, such an f exists. Let $\mathbf{M}' : \mathbb{N}^*_\# \to \mathbb{N}_{?,\perp}$ be such that, for all σ,

$$\mathbf{M}'(\sigma) = \begin{cases} \mathbf{M}_0((f \circ \text{content})(\sigma) \diamond \#^m), & \text{where } m \text{ is } \textit{first found} \text{ such that} \\ & \quad |\mathbf{M}((f \circ \text{content})(\sigma) \diamond \#^m)|_{\neq\perp} = n \\ & \quad \text{and } \mathbf{M}((f \circ \text{content})(\sigma) \diamond \#^{m+1}) \\ & \qquad = \mathbf{M}((f \circ \text{content})(\sigma) \diamond \#^m), \\ & \quad \text{if such an } m \text{ exists;} \\ \perp, & \text{otherwise.} \end{cases} \tag{34}$$

Clearly, $\mathbf{M}'$ is set-driven. Let $L \in \mathcal{L}$ be fixed, and let g be such that g represents L. To see that $\mathbf{M}'$ identifies L from g (in the $\mathbf{SD}'$ sense), consider the following cases.

CASE [L is finite]. Let j be such that $L \subseteq \text{content}(g[j])$. Let h be such that

$$h = (f \circ \text{content})(g[j]) \diamond \#^\omega. \tag{35}$$

Clearly, h represents L. Thus, there exists a *least* ℓ such that, for all $m \geq \ell$,

$$\begin{aligned} &|\mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^m)|_{\neq\perp} = n \\ &\wedge\ \mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^m) = \mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^\ell). \end{aligned} \tag{36}$$

It follows that ℓ is least such that, for all $m \geq \ell$,

$$\begin{aligned} &|\mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^m)|_{\neq\perp} = n \\ &\wedge\ \mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^{m+1}) = \mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^m). \end{aligned} \tag{37}$$

Let $\vec{p} = \mathbf{M}((f \circ \text{content})(g[j]) \diamond \#^\ell)$. Clearly, $\vec{p} \in \mathbb{N}^n$ and $(\forall i < n)[W_{(\vec{p})_i} = L]$. To complete the proof for this case, it suffices to show that $(\forall j' \geq j)\ [\mathbf{M}'(g[j']) = (\vec{p})_0]$. Let $j' \geq j$ be fixed. Clearly,

$$\text{content}(g[j']) = \text{content}(g[j])\ (= L). \tag{38}$$

Thus, in (34), some m is found in the computation of $\mathbf{M}'(g[j'])$. Moreover, by (37), $m \geq \ell$. Consequently,

$$\begin{aligned} \mathbf{M}'(g[j']) &= \mathbf{M}_0((f \circ \text{content})(g[j']) \diamond \#^m) && \{\text{by (34)}\} \\ &= \mathbf{M}_0((f \circ \text{content})(g[j]) \diamond \#^m) && \{\text{by (38)}\} \\ &= \mathbf{M}_0((f \circ \text{content})(g[j]) \diamond \#^\ell) && \{\text{by (36)}\} \\ &= (\vec{p})_0 && \{\text{by the choice of } \vec{p}\}. \end{aligned}$$

CASE [L is infinite]. Let $x_0 < x_1 < \cdots$ be such that $L = \{x_0, x_1, \ldots\}$. Let h be such that

$$h = \# \diamond x_0 \diamond \# \diamond x_1 \diamond \cdots. \tag{39}$$

Clearly, h represents L. Thus, there exists $k \in 2\mathbb{N}$ and $\vec{p} \in \mathbb{N}^n$ such that $(\forall k' \geq k)[\mathbf{M}(h[k']) = \vec{p}]$ and $(\forall i < n)[W_{(\vec{p})_i} = L]$. Note that

$$\begin{aligned} \mathbf{M}(h[k] \diamond \#) &= \mathbf{M}(h[k+1]) && \{\text{by (39) and } k \in 2\mathbb{N}\} \\ &= \vec{p} && \{\text{by the choice of } \vec{p}\}. \end{aligned} \tag{40}$$

It can similarly be shown that, for all $y \in L - \text{content}(h[k])$,

$$\mathbf{M}(h[k] \diamond y) = \vec{p}. \tag{41}$$

Let j be such that $\text{content}(h[k]) \subseteq \text{content}(g[j])$. To complete the proof for this case, it suffices to show that $(\forall j' \geq j)$ $[\mathbf{M}'(g[j']) = (\vec{p})_0]$. Let $j' \geq j$ be fixed. Clearly, $\text{content}(h[k]) \subseteq \text{content}(g[j'])$. Thus,

$$(f \circ \text{content})(g[j']) = h[k] \diamond \# \diamond y_0 \diamond \# \diamond y_1 \diamond \cdots \diamond \# \diamond y_{\ell-1}, \tag{42}$$

for some $\{y_0 < y_1 < \cdots < y_{\ell-1}\} \subseteq L - \text{content}(h[k])$. Note that

$$\begin{aligned} (\mathbf{M} \circ f \circ \text{content})(g[j']) &= \vec{p} && \{\text{by (40)–(42)}\} \\ &= \mathbf{M}(h[k]) && \{\text{by the choice of } \vec{p}\}. \end{aligned} \tag{43}$$

Furthermore, by (40) and (43), for all m,

$$(\mathbf{M} \circ f \circ \text{content})(g[j'] \diamond \#^m) = \vec{p}. \tag{44}$$

Thus, in (34), some m is found in the computation of $\mathbf{M}'(g[j'])$. Consequently,

$$\begin{aligned} \mathbf{M}'(g[j']) &= \mathbf{M}_0((f \circ \text{content})(g[j']) \diamond \#^m) && \{\text{by (34)}\} \\ &= (\vec{p})_0 && \{\text{by (44)}\}. \quad \square\ (\text{Theorem 19}) \end{aligned}$$

4. Conclusion

We presented two extensions to the **It**-learning model, each of which involves parallelism. We called learners incorporating the two extensions n-**ParIt**-learners. We showed that 1-**ParIt**-learners are strictly more powerful than **It**-learners (Corollary 16). We further showed that, for all $n \geq 1$, $n+1$-**ParIt**-learners are strictly more powerful than n-**ParIt**-learners (Theorem 17). Our results are somewhat surprising, since, in most contexts, parallelism is only a means of improving efficiency.

We paid particular attention to how one would actually implement an n-**ParIt**-learner. In this regard, we gave a scheme detailing: when new instantiations of an n-**ParIt**-learner could be run, and when running instantiations could be terminated. We further argued that an n-**ParIt**-learner implemented using our scheme would have nice properties in terms of when input elements could be discarded (see the discussion surrounding Property 2 in Section 1.1). As such, n-**ParIt**-learners, though still not as attractive as **It**-learners, are more attractive than an **Ex**- or **Bc**-learners.

There are several ways in which this work might be extended. First, as mentioned in Section 1.2, our current formulation (Definition 8) implicitly does not allow an instantiation $\mathbf{M}_i^j$ of an n-**ParIt**-learner **M** to wait for the *first* of $\mathbf{M}_0^{j-1}, \ldots, \mathbf{M}_{n-1}^{j-1}$ to converge. However, since such a learner could certainly exist in the real world (see the discussion following Example 9 in Section 3), it would be interesting to explore generalizations of Definition 8 that allow this sort of *polling* behavior. We hope to do so in future work.

Another interesting generalization would be to allow a collective learner to make use of a *finite but bounded* number of constituent learners, i.e., to allow the number of constituent learners to *grow*, provided that this occurs only finitely often. This idea raises a host of questions, however. For example:

- Under what conditions should new constituent learners be created?
- How should the *program* governing the behavior of a newly created constituent learner be determined? How should its initial conjecture be determined?
- Should the number of constituent learners be allowed to grow *without bound* on an input sequence representing a language *not* identifiable by the collective learner?

None of the above questions seem to have a clear right or wrong answer. Regardless of such specific choices, however, it would be interesting to see where such an idea leads.

Acknowledgements

We are grateful to the referees for their helpful suggestions.

References

[1] S. Arikawa, S. Miyano, A. Shinohara, S. Kuhara, Y. Mukouchi, T. Shinohara, A machine discovery from amino-acid-sequences by decision trees over regular patterns, New Generation Computing 11 (1993) 361–375.
[2] D. Angluin, Finding patterns common to a set of strings, Journal of Computer and System Sciences 21 (1980) 46–62.
[3] Argonne national laboratory, the message passing interface (MPI) standard, http://www-unix.mcs.anl.gov/mpi is the URL.
[4] A. Brazma, E. Ukkonen, J. Vilo, Discovering unbounded unions of regular pattern languages from positive examples, in: Proceedings of the Seventh International Symposium on Algorithms and Computation, ISAAC' 96, in: Lecture Notes in Computer Science, Osaka, Japan, 1996.
[5] J. Case, Periodicity in generations of automata, Mathematical Systems Theory 8 (1974) 15–32.
[6] J. Case, Infinitary self-reference in learning theory, Journal of Experimental and Theoretical Artificial Intelligence 6 (1994) 3–16.
[7] L. Carlucci, J. Case, S. Jain, F. Stephan, Results on memory-limited U-shaped learning, Information and Computation 205 (2007) 1551–1573.
[8] J. Case, S. Jain, S. Kaufmann, A. Sharma, F. Stephan, Predictive learning models for concept drift, Theoretical Computer Science 268 (2001) 323–349.
[9] J. Case, S. Jain, S. Lange, T. Zeugmann, Incremental concept learning for bounded data mining, Information and Computation 152 (1999) 74–110.
[10] J. Case, C. Lynes, Machine inductive inference and language identification, in: Proceedings of the Ninth International Colloquium on Automata, Languages and Programming, ICALP'82, in: Lecture Notes in Computer Science, vol. 140, Springer, Heidelberg, 1982, pp. 107–115.
[11] J. Case, S. E. Moelius, Parallelism increases iterative learning power, in: Proceedings of the Eighteenth Annual Conference on Algorithmic Learning Theory, ALT'07, in: Lecture Notes in Artificial Intelligence, vol. 4754, Springer, Heidelberg, 2007, pp. 49–63.
[12] J. Case, S.E. Moelius, U-shaped, iterative, and iterative-with-counter learning, Machine Learning 72 (2008) 63–88.
[13] M. Davis, R. Sigal, E. Weyuker, Computability, Complexity, and Languages, second ed., Morgan Kaufmann, San Francisco, 1994.
[14] M. Fulk, S. Jain, D. Osherson, Open problems in systems that learn, Journal of Computer and System Sciences 49 (1994) 589–604.
[15] E. Gold, Language identification in the limit, Information and Control 10 (1967) 447–474.
[16] J. Heinz, The inductive learning of phonotactic patterns, Ph.D. Thesis, University of California, Los Angeles, 2007.
[17] J. Heinz, Learning left-to-right and right-to-left iterative languages, in: Proceedings of the Ninth International International Colloquium on Grammatical Inference, ICGI'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 84–97.
[18] S. Jain, D. Osherson, J. Royer, A. Sharma, Systems that Learn: An Introduction to Learning Theory, second ed., MIT Press, Cambridge, MA, 1999.
[19] P. Kilpeläinen, H. Mannila, E. Ukkonen, MDL learning of unions of simple pattern languages from positive examples, in: Proceedings of the Second European Conference on Computational Learning Theory, in: Lecture Notes in Artificial Intelligence, vol. 904, Springer, Heidelberg, 1995, pp. 252–260.
[20] E. Kinber, F. Stephan, Language learning from texts: Mind changes, limited memory, and monotonicity, Information and Computation 123 (1995) 224–241.
[21] S. Lange, R. Wiehagen, Polynomial time inference of arbitrary pattern languages, New Generation Computing 8 (1991) 361–370.
[22] S. Lange, T. Zeugmann, Incremental learning from positive data, Journal of Computer and System Sciences 53 (1996) 88–103.
[23] Message passing interface (MPI) forum, http://www.mpi-forum.org is the URL.
[24] R. Nix, Editing by examples, Technical Report 280, Department of Computer Science, Yale University, New Haven, CT, USA, 1983.
[25] D. Osherson, M. Stob, S. Weinstein, Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists, MIT Press, Cambridge, MA, 1986.

[26] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw Hill, New York, 1967, Reprinted, MIT Press, 1987.
[27] T. Shinohara, A. Arikawa, Pattern inference, in: Algorithmic Learning for Knowledge-Based Systems, in: Lecture Notes in Artificial Intelligence, vol. 961, Springer, Heidelberg, 1995, pp. 259–291.
[28] A. Salomaa, Patterns (The formal Language theory column), EATCS Bulletin 54 (1994) 46–62.
[29] A. Salomaa, Return to patterns (The formal Language theory column), EATCS Bulletin 55 (1994) 144–157.
[30] T. Shinohara, Inferring unions of two pattern languages, Bulletin of Informatics and Cybernetics 20 (1983) 83–88.
[31] Carl H. Smith, Three decades of team learning, in: Proceedings of the Fourth International Workshop on Analogical and Inductive Inference, AII' 94, Springer, Heidelberg, 1994, pp. 211–228.
[32] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, S. Arikawa, Knowledge acquisition from amino acid sequences by machine learning system BONSAI, Transactions of Information Processing Society of Japan 35 (1994) 2009–2018.
[33] K. Wexler, P. Culicover, Formal Principles of Language Acquisition, MIT Press, Cambridge, MA, 1980.
[34] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Electronische Informationverarbeitung und Kybernetik 12 (1976) 93–99.
[35] G. Winskel, The Formal Semantics of Programming Languages: An Introduction, MIT Press, Cambridge, MA, 1993.
[36] K. Wright, Identification of unions of languages drawn from an identifiable class, in: Proceedings of the Second Annual Workshop on Computational Learning Theory, Morgan Kaufmann, San Francisco, 1989, pp. 328–333.