Program Size Restrictions in Computational Learning

Sanjay Jain
Institute of Systems Science
National University of Singapore
Singapore 0511
Republic of Singapore
Email: sanjay@iss.nus.sg

Arun Sharma
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2033, Australia
Email: arun@cs.unsw.oz.au

March 11, 2007

Abstract

A model for a subject S learning its environment E could be described thus. S, placed in E, receives data about E, and simultaneously conjectures a sequence of hypotheses. S is said to *learn* E just in case the sequence of hypotheses conjectured by S stabilize to a final hypothesis which correctly represents E. *Computational learning theory* provides a framework for studying problems of this nature when the subject is a machine.

A natural abstraction for the notion of hypothesis is a computer program. The present paper, in the above framework of learning, presents arguments for the final hypothesis to be *succinct*, and introduces a plethora of formulations of such succinctness. A revelation of this study is that some of the "natural" notions of succinctness may be uninteresting because learning capability of machines under these seemingly natural constraints is dependent on the choice of programming system used to interpret hypotheses.

1 Introduction

Consider the following description of a typical situation involving a subject S learning its environment E. At any given time, a finite piece of data about E is made available to S. S reacts to this finite information by conjecturing a hypothesis. Availability of additional data may cause the learner to revise old hypotheses. S is said to *learn* or *explain* E just in case the sequence of hypotheses conjectured by S eventually stabilizes to a final hypothesis which correctly explains E. *Computational learning theory* provides a framework for studying problems of this nature when the subject is a machine. This paradigm of learning originated in the work of Gold [10]. Klette and Wiehagen [14], Angluin and Smith [1], Case [3], and Osherson, Stob, and Weinstein [18] provide surveys of work in this area.

The present paper provides arguments in favor of placing "size" restrictions on the final hypothesis in the above learning situation. It is shown that for numerous 'natural' formulations of such size restrictions, the resulting learning models are dependent on the 'programming system' used to interpret the hypotheses. The arguments and results are presented in the context of two learning tasks that can be modeled in the above framework: scientific inquiry and first language acquisition. Section 1.1 contains a description of how scientific inquiry could be modeled as machine identification in the limit of computer programs for computable functions. In Section 1.2, we describe how first language acquisition can be modeled as machine identification in the limit of grammars for recursively enumerable languages. Section 1.3 contains motivation for studying size restrictions in both the models.

1.1 A Model for Scientific Inquiry

Consider a scientist S investigating a real world phenomenon F. S performs experiments on F, noting the result of each experiment, while simultaneously conjecturing a succession of candidate explanations for F. A criterion of success is for S to eventually conjecture an explanation which S never gives up and which explanation correctly explains F.

Since we never measure a continuum of possibilities, we could treat S as performing discrete experiments x on F and receiving back experimental results $f(x)$. By suitable Gödel numbering, we may treat the f, associated with F, as a function from N, the set of natural numbers, into N. Then, a complete and predictive explanation of F is just a computer program for computing f.

Thus, replacing the ever experimenting S with a machine in the above scenario yields a plausible model for scientific inquiry—algorithmic identification in the limit of programs for computable functions from their graphs. This is essentially the theme of inductive inference studied by Gold [10].

1.2 A Model for First Language Acquisition

Motivated by psycholinguistic studies which conclude that children are rarely, if ever, informed of grammatical errors, Gold [10] introduced the seminal notion of *identification* as a model for first language acquisition. According to this paradigm, a child C (modeled as a machine) receives (in arbitrary order) all the well-defined sentences, a *text*, of a language L, and simultaneously, conjectures a succession of grammars. A criterion of success is for C to eventually conjecture a grammar for L which grammar C never gives up thereafter.

Languages are sets of sentences and a sentence is a finite object; the set of all possible sentences can be coded into N. Hence, languages may be construed as subsets of N. A grammar for a language is a set of rules that generates (or equivalently accepts [11]) the language; such grammars are, in some cases, referred to as *type* 0 grammars. Languages for which a grammar exists are called *recursively enumerable*. Henceforth, we work under the assumption that natural languages fall into the class of recursively enumerable languages[1].

Thus, replacing the child machine by an arbitrary machine in the above language learning scenario, we have a plausible model for language acquisition—algorithmic identification in the limit of grammars for recursively enumerable languages from their texts. This is essentially Gold's influential language learning paradigm discussed, for example, by Pinker [19], Wexler and Culicover [29], Wexler [28], and Osherson, Stob, and Weinstein [18].

1.3 Motivations for Study of Program Size Restrictions

A drawback of the two learning models presented above is that there are no restrictions placed on the *size* of the programs/grammars inferred in the limit. We present, below, arguments, both for scientific inquiry and first language acquisition, motivating the desirability of placing size restrictions on the final explanations conjectured by learning machines.

1.3.1 Scientific Inquiry

To begin with, "small" size explanations satisfy a variant of *Occam's Razor*[2]—a heuristic about the desirability of parsimony. Succinct theories or explanations are many times likely to be

[1] See Langendoen and Postal [15] for an opposing viewpoint.

[2] *Entia non sunt multiplicanda, præter necessitatem*: attributed to the medieval philosopher William of Ockham. However, W. M. Thorburn [26] raises some doubts as to whether William of Ockham ever used the above expression to express his Critique of Entities, and writes in [27], "The Metaphysical (or Methodological) Law of Parsimony (or Logical Frugality), indicated but not very distinctly expressed by Aristotle, was fully and finally established, not by Ockham, but by his teacher Duns Scotus ...". According to Thorburn [27], the terminology, Occam's Razor, seems to have first appeared in 1852 in the work of Sir William Hamilton. Moody [17] provides a study of the philosophy of William of Ockham.

understood with less effort; thus, in these cases they facilitate smooth dissemination of scientific information—an important aspect of the practice of science.

A case for the desirability of succinctness in explanations of phenomena can also be made by arguing that useful explanations are such that the relevant features of the phenomena are brought out instead of being inundated in a sea of irrelevant information. This may be illustrated in the light of an anti-reductionist argument of Putnam [20]. Although, Putnam's example is in the context of his argument for the independent significance of higher level sciences like psychology and sociology, therein lies a strong case for a meaningful explanation to be succinct. Putnam argues that, although, laws of higher level sciences can be reduced to the laws of lower level sciences—biology, chemistry, and ultimately elementary particle physics, these reductions are often uninformative, intractable, and mostly not interesting. Clearly, the lack of succinctness in these reduced explanations is one important reason why they fail to provide any significant insight into the phenomena associated with the higher level sciences.

However, the desirability of learning succinct explanations has a sobering side to it. Most learning criteria with the additional requirement of inferring succinct explanations pay a price of decreased inferring power.

1.3.2 Language Acquisition

In the case of an algorithmic device inferring a grammar for a recursively enumerable language, the size of the final stabilized grammar can be very "large." This poses a difficulty for Gold's paradigm to be a model of language acquisition. We describe this problem in the context of a child modeled as a machine.[3] The human head is of bounded size. A simple result from computability theory tells us that any recursively enumerable language can be generated by infinitely many syntactically distinct grammars whose size is bigger than any prespecified bound on the size of a child's head. A child learning a language, hence, must converge to a grammar which fits in its finite size head. This of course assumes that human brain storage is not magic, admitting of infinite regress, etc. An interesting complexity restriction to make, then, on the final grammar converged to in the limit is that it be of "small" size. It should of course be noted that there exist recursively enumerable languages for which even the minimal size grammar is larger than any prespecified bound on the size of the human head.

Our main concern, in the present paper, is to study the dependence of these size restricted learning criteria on the underlying 'programming system' in which a learning machine's conjectures are interpreted. A few words on programming systems is in order. An *acceptable programming system* [23, 24, 16] is (by definition) one such that there are effective translations back and forth between it and a standard Turing Machine formalism. It is also referred to as acceptable numbering. Rogers [23] has shown that any two acceptable programming systems

[3]These motivations for succinctness in language learning are based on discussions with John Case.

are computably isomorphic; in other words, programs written in one acceptable programming system are simply an algorithmic renaming of programs written in another acceptable programming system. Case showed [21, 22, 25] that acceptable programming systems are characterized as those in which every control structure is implementable. All general purpose programming languages are essentially acceptable programming systems.

Our study builds on previous work by Freivalds [8], Chen [6, 7], Kinber [13, 12], and Case and Chi [4].

We now proceed formally. Section 2 introduces notation and relevant notions from recursive function theory. Preliminary concepts from theory of inductive inference are described in Section 3. Results occupy Sections 4, 5, 6, and 7.

2 Notation

Recursion-theoretic concepts not explained below are treated in [24].

N denotes the set of natural numbers, $\{0, 1, 2, 3, \ldots\}$, and N^+ denotes the set of positive integers, $\{1, 2, 3, \ldots\}$. $\in$, $\subseteq$, and $\subset$ denote, respectively, membership, containment, and proper containment for sets (including sets of ordered pairs).

$*$ denotes *unbounded* but *finite*; we assume $(\forall n \in N)[n < * < \infty]$. a and b, with or without decorations, range over $(N \cup \{*\})$ and $(N^+ \cup \{*\})$, respectively. Generally e, i, j, k, l, m, n, x, y, z range over N.

We let D, P, S, with or without decorations, range over subsets of N. $\mathrm{card}(P)$ denotes the cardinality of P. So then, '$\mathrm{card}(P) \leq *$' means that $\mathrm{card}(P)$ is finite. $\min(P)$ and $\max(P)$, respectively, denote the minimum and maximum element in P. We take $\min(\emptyset)$ to be undefined and $\max(\emptyset)$ to be 0.

f, g, h and sometimes p, q, r, and v range over total functions. On many occasions, we will use g and p to range over N, where g will be construed as a grammar and p will be construed as a program; such usage will be clear from context. η and ξ range over partial functions. $\mathcal{R}$ denotes the class of all *recursive* functions, i.e., total computable functions with arguments and values from N. $\mathcal{R}^+$ denotes the class of all total computable functions with arguments from N and values from N^+. $\mathcal{S}$ and $\mathcal{C}$ range over subsets of $\mathcal{R}$. For $a \in (N \cup \{*\})$, $\eta_1 =^a \eta_2$ means that $\mathrm{card}(\{x \mid \eta_1(x) \neq \eta_2(x)\}) \leq a$. $\mathrm{domain}(\eta)$ and $\mathrm{range}(\eta)$ respectively denote the domain and range of partial function η.

L, with or without decorations, ranges over subsets of N which subsets are usually construed as codings of formal languages. $\mathcal{E}$ denotes the class of all *recursively enumerable* (r. e.) languages. We let $\mathcal{L}$, with or without decorations, range over subsets of $\mathcal{E}$. $L_1 \mathbf{\Delta} L_2$ denotes $(L_1 - L_2) \cup (L_2 - L_1)$, the symmetric difference of L_1 and L_2. For $a \in N \cup \{*\}$, $L_1 =^a L_2$ means that $\mathrm{card}(L_1 \mathbf{\Delta} L_2) \leq a$.

We let ψ, ψ', and ψ'' range over acceptable programming systems for the partial recursive functions: $N \rightarrow N$. ψ_p denotes the partial recursive function computed by ψ-program p. W_p^ψ denotes domain(ψ_p). W_p^ψ is, then, the r. e. set/language ($\subseteq N$) accepted (or equivalently, generated) by the ψ-program p. We let Ψ be an arbitrary Blum Complexity measure [2] associated with acceptable programming system ψ; such measures exist for any acceptable programming system [2]. Then, $W_i^{\psi,s}$ denotes the set $\{x \mid x < s \wedge \Psi_i(x) \leq s\}$. In general given a computable function f and an r. e. language L, $\mathbf{minprogram}_\psi(f)$ denotes $\min(\{p \mid \psi_p = f\})$; $\mathbf{mingrammar}_\psi(L)$ denotes $\min(\{p \mid W_p^\psi = L\})$. For a large part of this document, we will use a fixed acceptable programming system, which we will denote by φ. We will sometimes drop the mention of φ from W_p^φ, $W_i^{\varphi,s}$, $\mathbf{minprogram}_\varphi(f)$, and $\mathbf{mingrammar}_\varphi(L)$.

Let $\lambda x, y.\langle x, y\rangle$ denote a fixed pairing function (a recursive, bijective mapping: $N \times N \rightarrow N$) [24]. $\lambda x, y.\langle x, y\rangle$ and its inverses are useful to simulate the effect of having multiple argument functions in an acceptable programming system ψ.

$S \subseteq N$ is said to *represent* the set $\{(x, y) \mid \langle x, y\rangle \in S\}$. $S \subseteq N$ is called *single-valued* just in case S represents a function. A single-valued set is said to be *single-valued total* (abbreviated: **svt**) just in case the function it represents is total.

For any predicate Q, $\mu n.[Q(n)]$ denotes the minimum integer n such that $Q(n)$ is true if such an n exists; it is 0 otherwise.

The quantifiers '$\overset{\infty}{\forall}$' and '$\overset{\infty}{\exists}$' mean 'for all but finitely many' and 'there exist infinitely many,' respectively. The quantifier '$\exists!$' means 'there exists a unique.'

3 Preliminaries

Our study is about machine inference of two kinds of objects: computable functions and recursively enumerable languages. In most of the exposition to follow, we will discuss a notion for function inference first, and then describe an analogous notion for language learning.

3.1 Learning Machines

In Definition 1, below, we formally introduce what we mean by a machine that learns a function and in Definition 3 we do the same for a machine that learns a language.

For any recursive function f and any natural number n, we let $f[n]$ denote the finite initial segment $\{(x, f(x)) \mid x < n\}$. Clearly, $f[0]$ denotes the empty sequence. Let SEG denote the set of all finite initial segments.

Definition 1 [10] A *function learning machine* is an algorithmic device, which computes a mapping from SEG into N.

We now consider language learning machines. Definition 2 below introduces a notion that facilitates discussion about elements of a language being fed to a learning machine.

Definition 2 A *sequence* σ is a mapping from an initial segment of N into $(N \cup \{\#\})$. The *content* of a sequence σ, denoted by content(σ), is the set of natural numbers in the range of σ. Length of σ, denoted by $|\sigma|$, is the number of elements in σ.

Intuitively, #'s represent pauses in the presentation of data. Let SEQ denote the set of all sequences. We let σ and τ, with or without decorations, range over sequences. $\sigma_1 \diamond \sigma_2$ denotes the *concatenation* of σ_1 and σ_2, where $\sigma = \sigma_1 \diamond \sigma_2$ is defined as follows:

$$\sigma(x) = \begin{cases} \sigma_1(x) & \text{if } x < |\sigma_1|; \\ \sigma_2(x - |\sigma_1|) & \text{if } x \geq |\sigma_1|. \end{cases}$$

Definition 3 A *language learning machine* is an algorithmic device, which computes a mapping from SEQ into N.

SEG can be coded onto N. Also, SEQ can be coded onto N. Thus, in both Definitions 1 and 3, we are essentially dealing with machines that take as input natural numbers at a time, and which from time to time, output natural numbers. Henceforth, we will refer to both function learning machines and language learning machines as just learning machines. We let $\mathbf{M}$, with or without superscripts, range over learning machines (we reserve $\mathbf{M}$ with subscripts to denote learning machines in a special kind of enumeration described at the end of Section 3.2).

3.2 Fundamental Learning Paradigms

3.2.1 Function Inference

In Definition 4, below, we spell out what it means for a learning machine to converge in the limit on a function.

Definition 4 Suppose $\mathbf{M}$ is a learning machine and f is a computable function. $\mathbf{M}(f)\downarrow$ (read: $\mathbf{M}(f)$ *converges*) $\iff (\exists p)(\overset{\infty}{\forall}\, n)\ [\mathbf{M}(f[n]) = p]$. If $\mathbf{M}(f)\downarrow$, then $\mathbf{M}(f)$ is defined = the unique p such that $(\overset{\infty}{\forall}\, n)[\mathbf{M}(f[n]) = p]$; otherwise $\mathbf{M}(f)$ is said to be undefined.

We now introduce a criterion for a learning machine to be successful on a function. Definition 5-(a) introduces **Ex**-identification, a criterion for successful inference of functions. The class **Ex**, referred to as the *inferring power* of **Ex**-identification, is described in Definition 5-(b).

Definition 5 [10]
(a) $\mathbf{M}$ **Ex**-*identifies* f (written: $f \in \mathbf{Ex}(\mathbf{M})$) $\iff (\exists p \mid \varphi_p = f)[\mathbf{M}(f)\downarrow = p]$.
(b) $\mathbf{Ex} = \{\mathcal{C} \mid (\exists \mathbf{M})[\mathcal{C} \subseteq \mathbf{Ex}(\mathbf{M})\}$.

Intuitively, **Ex** is a set theoretic summary of the capability of various learning machines to **Ex**-identify entire collections of recursive functions.

The criterion introduced in Definition 5, along with its inferring power, is *implicitly* parameterized by the choice of acceptable programming system in which programs conjectured by the learning machine are interpreted. A reason for not explicitly mentioning the acceptable programming system is that the class **Ex** is independent of the choice of the underlying acceptable programming system.

3.3 Language Learning

Definition 6 A *text* T for a language L is a mapping from N into $(N \cup \{\#\})$ such that L is the set of natural numbers in the range of T. The *content* of a text T, denoted content(T), is the set of natural numbers in the range of T.

Intuitively, a text for a language is an enumeration or sequential presentation of all the objects in the language with the #'s representing pauses in the listing or presentation of such objects. For example, the only text for the empty language is just an infinite sequence of #'s.

We let T, with or without superscripts, range over texts. $T[n]$ denotes the finite sequence of T with length n. Hence, domain$(T[n]) = \{x \mid x < n\}$.

In Definition 7 below we spell out what it means for a learning machine to converge in the limit on a text.

Definition 7 Suppose $\mathbf{M}$ is a learning machine and T is a text. $\mathbf{M}(T)\downarrow$ (read: $\mathbf{M}(T)$ *converges*) $\iff$ $(\exists p)(\overset{\infty}{\forall}\ n)$ $[\mathbf{M}(T[n]) = p]$. If $\mathbf{M}(T)\downarrow$, then $\mathbf{M}(T)$ is defined = the unique p such that $(\overset{\infty}{\forall}\ n)[\mathbf{M}(T[n]) = p]$; otherwise $\mathbf{M}(T)$ is said to be undefined.

We now introduce a criterion for a learning machine to be successful on a language. Based on psycholinguistic studies of first language acquisition in children, Gold [10] proposed a criterion of success called *identification* which we refer to as **TxtEx**-*identification* following Case and Lynes [5].

Definition 8 [10]
(a) $\mathbf{M}$ **TxtEx**-*identifies* L (written: $L \in \mathbf{TxtEx}(\mathbf{M})$) $\iff$ ($\forall$ texts T for L)$(\exists p \mid W_p = L)[\mathbf{M}(T)\downarrow = p]$.
(b) $\mathbf{TxtEx} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}(\mathbf{M})\}$.

The language learning criteria introduced in Definition 8, along with its inferring power, is *implicitly* parameterized by the choice of acceptable programming system in which grammars conjectured by the learning machine are interpreted. A reason for not explicitly mentioning

the acceptable programming system is that the class **TxtEx** is independent of the choice of the underlying acceptable programming system.

It is easy to observe that there exists a recursive enumeration $\mathbf{M}_0, \mathbf{M}_1, \ldots$ of learning machines such that, for all the criteria of inference, $\mathbf{I}$, discussed in this paper (including **Ex** and **TxtEx**) the following two properties hold.

(1) $\{\mathbf{I}(\mathbf{M}_i) \mid i \in N\} = \{\mathbf{I}(\mathbf{M}) \mid \mathbf{M} \text{ is a learning machine}\}$.
(2) For each i, there exist infinitely many j such that, $\mathbf{I}(\mathbf{M}_i) = \mathbf{I}(\mathbf{M}_j)$.

We assume and make use of such an enumeration, $\mathbf{M}_0, \mathbf{M}_1, \ldots$, in several of the proofs in the paper without explicitly mentioning it.

4 Strictly Minimal Identification

A natural restriction to make on the size of the final program/grammar is to require that it be of minimal size. In the context of function inference, such a notion was first studied by Freivalds [8]. Definition 9 below describes this criterion and its inferring power.

Definition 9 [8, 13]
(a) $\mathbf{M}$ $\mathbf{Min}_\psi$-*identifies* f (written: $f \in \mathbf{Min}_\psi(\mathbf{M})$) $\iff$ $(\overset{\infty}{\forall}\, n)[\mathbf{M}(f[n]) = \mathbf{minprogram}_\psi(f)]$.
(b) $\mathbf{Min}_\psi = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Min}_\psi(\mathbf{M})]\}$.

Proposition 10 below implies that there exists an acceptable programming system in which a single learning machine can identify the minimal program for some infinite collection of recursive functions.

Proposition 10 $(\exists \psi)(\exists \mathcal{S} \mid \text{card}(\mathcal{S})$ *is infinite* $\wedge$ $\mathcal{S}$ *is an r.e. class of functions*$)[\mathcal{S} \in \mathbf{Min}_\psi]$.

Proof of Proposition 10: Let $\mathcal{S} = \{f \in \mathcal{R} \mid (\forall x > 0)[f(x) = 0]\}$. Clearly $\mathcal{S}$ is an r.e. class and $\text{card}(\mathcal{S})$ is ∞. Consider the acceptable programming system ψ defined below. Note that φ is our standard acceptable programming system.

begin {Definition of ψ_i}
  if i is odd
    then
      $\psi_i = \varphi_{(i-1)/2}$
    else
      $\psi_i(0) = \varphi_{i/2}(0)$;
      $(\forall x > 0)[\psi_i(x) = 0]$
  endif

end {Definition of ψ_i}

Clearly, ψ is an acceptable programming system. A machine **M** $\mathbf{Min}_\psi$-identifies each $f \in \mathcal{S}$ by searching for the minimum even i such that $\psi_i(0) = f(0)$. ∎

With a view to characterize $\mathbf{Min}_\psi$-identification, Freivalds [8] introduced an interesting technical notion called *limit standardizability with a recursive estimate* (abbreviated: **LSR**). This notion, described in Definition 11 below, was later generalized by Chen [6, 7].

Definition 11 [8] $\mathcal{S}$ is *limiting standardizable with a recursive estimate* (written: $\mathcal{S} \in \mathbf{LSR}$) $\iff$ there exist recursive functions g and v such that, for all $f \in \mathcal{S}$ and $i \in N$, if $\varphi_i = f$, then

(a) $\lim_{n\to\infty} g(i,n)$ exists and $\varphi_{\lim_{n\to\infty} g(i,n)} = f$;

(b) for all j, if $\varphi_j = f$, then $\lim_{n\to\infty} g(i,n) = \lim_{n\to\infty} g(j,n)$;

(c) $\mathrm{card}(\{g(i,n) \mid n \in N\}) \leq v(i)$.

The notion of **LSR** above is *implicitly* parametrized by the choice of acceptable programming system; explicit parametrization is not called for because it is easy to verify that the class **LSR** is independent of the choice of acceptable programming system. We write $\mathcal{S} \subseteq \mathbf{LSR}(g,v) \iff$ the recursive functions g and v witness as above that $\mathcal{S} \in \mathbf{LSR}$. Intuitively[4], the role of g in the definition of **LSR** is to provide a limiting recursive solution to a special case of the *program equivalence problem* ($\{\langle x,y\rangle \mid \varphi_x = \varphi_y\}$) where the programs compute functions in $\mathcal{S}$; also, v places some extra constraints on how g reaches its limit. Freivalds [8] used the notion of **LSR** to show the following characterization of $\mathbf{Min}_\psi$-identification.

Theorem 12 [8] $(\forall \mathcal{S})[(\exists \psi)[\mathcal{S} \in \mathbf{Min}_\psi]] \iff \mathcal{S} \in (\mathbf{Ex} \cap \mathbf{LSR})]$.

We study a language learning criterion analogous to $\mathbf{Min}_\psi$-identification. Definition 13 below precisely defines the criterion of minimal grammar identification and its inferring power.

Definition 13

(a) **M** $\mathbf{TxtMin}_\psi$-*identifies* L (written: $L \in \mathbf{TxtMin}_\psi(\mathbf{M})$) $\iff$ ($\forall$ texts T for L) ($\overset{\infty}{\forall}$ n)$[\mathbf{M}(T[n]) = \mathbf{mingrammar}_\psi(L)]$.

(b) $\mathbf{TxtMin}_\psi = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtMin}_\psi(\mathbf{M})]\}$.

Proposition 14 below shows that there exists an acceptable programming system in which a single learning machine can identify the minimal grammar for some infinite collection of r.e. languages.

[4]This interpretation was brought to our attention by John Case.

Proposition 14 $(\exists \psi)(\exists \mathcal{L} \mid \mathrm{card}(\{L \in \mathcal{L} \mid \mathrm{card}(L) \text{ is } \infty\}) \text{ is infinite } \wedge \mathcal{L} \text{ is an r.e. class of languages})[\mathcal{L} \in \mathbf{TxtMin}_{\psi}]$.

Proof of Proposition 14: Let $\mathcal{L} = \{L \text{ is single valued total} \mid (\forall x > 0)[\langle x, 0\rangle \in L]\}$. Consider the acceptable programming system ψ defined below. Note that φ is our standard acceptable programming system.

begin {Definition of W_i^{ψ}}
  if i is odd
    then
      let $W_i^{\psi} = W_{(i-1)/2}^{\varphi}$
    else
      let $W_i^{\psi} = [\{\langle 0, y\rangle \mid y \in N\} \cap W_{i/2}^{\varphi}] \cup [\{\langle x, 0\rangle \mid x > 0\}]$
  endif
end {Definition of W_i^{ψ}}

Clearly, ψ is an acceptable programming system. A machine **M TxtMin**$_{\psi}$-identifies each $L \in \mathcal{L}$ by searching for the minimum even i such that $\{\langle 0, y\rangle \mid y \in N\} \cap L = \{\langle 0, y\rangle \mid y \in N\} \cap W_i^{\psi}$. ∎

With an intention of extending the work of Freivalds to the context of language learning, Case and Chi [4] introduced **TxtLSR** (Definition 15), a notion analogous to **LSR**.

Definition 15 [4] $\mathcal{L}$ is *text limiting standardizable with recursive estimate* (written: $\mathcal{L} \in$ **TxtLSR**) $\iff$ there exist recursive functions g and v such that, for all $L \in \mathcal{L}$ and $i \in N$, if $W_i = L$ then

(a) $\lim_{n\to\infty} g(i, n)$ exists and $W_{\lim_{n\to\infty} g(i,n)} = L$;

(b) for all j, if $W_j = L$, then $\lim_{n\to\infty} g(i, n) = \lim_{n\to\infty} g(j, n)$;

(c) $\mathrm{card}(\{g(i, n) \mid n \in N\}) \leq v(i)$.

We write $\mathcal{L} \subseteq$ **TxtLSR**$(g, v) \iff$ the recursive functions g and v witness as above that $\mathcal{L} \in$ **TxtLSR**. It is easy to verify that the class **TxtLSR** is acceptable programming system independent. Intuitively, the role of g in the definition of **TxtLSR** is to indirectly provide a limiting recursive solution to a special case of the *grammar equivalence problem* $(\{\langle x, y\rangle \mid W_x = W_y\})$ where the grammars generate languages in $\mathcal{L}$; also, v places some extra constraints on how g reaches its limit.

Analogous to Freivalds' characterization of **Min**$_{\psi}$ described in Theorem 12, we characterize **TxtMin**$_{\psi}$ in terms of **TxtEx** and **TxtLSR**. A proof similar to Freivalds' proof of Theorem 12 is sufficient if we only wish to characterize minimal grammar identification of infinite languages. However, a modification of Freivalds' proof technique, which modification handles finite lan-

guages, yields a complete characterization of $\mathbf{TxtMin}_\psi$-identification as stated in Theorem 16 just below.

Theorem 16 $(\forall \mathcal{L})\ [(\exists \psi)[\mathcal{L} \in \mathbf{TxtMin}_\psi] \iff \mathcal{L} \in (\mathbf{TxtEx} \cap \mathbf{TxtLSR})]$.

Proof of Theorem 16: Let $\mathcal{L}$ be given. We first show that $(\exists \psi)[\mathcal{L} \in \mathbf{TxtMin}_\psi] \Rightarrow [\mathcal{L} \in \mathbf{TxtEx} \ \wedge \ \mathcal{L} \in \mathbf{TxtLSR}]$.

Let ψ be an acceptable programming system such that $\mathcal{L} \in \mathbf{TxtMin}_\psi$. Clearly, $\mathcal{L} \in \mathbf{TxtEx}$. We exhibit recursive functions g and v that witness $\mathcal{L} \in \mathbf{TxtLSR}$.

Let r_1 be a recursive function reducing ψ-indices to φ-indices. Let r_2 be a recursive function reducing φ-indices to ψ-indices. Let learning machine $\mathbf{M}$ $\mathbf{TxtMin}_\psi$-identify $\mathcal{L}$. Let T^i denote a text uniformly formed from W_i^φ. Define $v(i) = r_2(i) + 1$ and

$$g(i,n) = \begin{cases} r_1(\mathbf{M}(T^i[n])) & \text{if } \mathbf{M}(T^i[n]) \leq r_2(i); \\ r_1(0) & \text{otherwise.} \end{cases}$$

Clearly, for all i, $\mathrm{card}(\{g(i,n) \mid n \in N\}) \leq r_2(i) + 1 = v(i)$. Now, if $L \in \mathcal{L}$ and $W_i^\varphi = L$, then $\lim_{n\to\infty} \mathbf{M}(T^i[n]) = \mathbf{mingrammar}_\psi(L) \leq r_2(i)$. Hence, $\lim_{n\to\infty} g(i,n) = r_1(\mathbf{mingrammar}_\psi(L))$, which is a φ-grammar for L. This proves one direction of the theorem.

We now show that $[\mathcal{L} \in \mathbf{TxtEx} \ \wedge \ \mathcal{L} \in \mathbf{TxtLSR}] \Rightarrow (\exists \psi)[\mathcal{L} \in \mathbf{TxtMin}_\psi]$. Let learning machine $\mathbf{M}$ $\mathbf{TxtEx}$-identify $\mathcal{L}$ and let g and v witness that $\mathcal{L} \in \mathbf{TxtLSR}$ (without loss of generality assume that for all x, $v(x) \geq 1$). We describe an acceptable programming system ψ such that $\mathcal{L} \in \mathbf{TxtMin}_\psi$. Without loss of generality, let $(\forall i)[\mathrm{card}(\{g(i,n) \mid n \in N\}) \leq v(i)]$. To facilitate the description of ψ, we define a series of functions η, p, ξ, q_1, and q_2 below.

Let η, a partial recursive function, be defined as follows:

$$\eta(i,j) = \begin{cases} g(i,k) & \text{where } k \text{ is the least integer such that} \\ & \mathrm{card}(\{g(i,n) \mid n \leq k\}) = j+1; \\ \uparrow & \text{otherwise.} \end{cases}$$

Intuitively, $\eta(i,j)$ is the $(j+1)^{th}$ grammar output by g on i. Clearly, $\eta(i,j)$ is undefined for $j \geq v(i)$. By the *s-m-n theorem* [24], there exists a recursive function p such that $W^\varphi_{p(m)}$ is as follows:

begin {Definition of $W^\varphi_{p(m)}$}
  go to stage 0;
  begin {Stage s}
    if $\mathrm{card}(\{n \mid g(m,n) = m\}) \geq s$
      then

enumerate $W_m^{\varphi,s}$;
go to stage $s+1$
endif
end {Stage s}
end {Definition of $W_{p(m)}^{\varphi}$}

Hence, for any language $L \in \mathcal{L}$, for any φ-grammar i for L, $W_{p(i)}^{\varphi} = L$ if i is the limiting value of g on any φ-grammar for L; otherwise $W_{p(i)}^{\varphi}$ is a finite set.

Let $\xi(i,j) = p(\eta(i,j))$.

For any $L \in \mathcal{L}$, for any φ-grammar i for L, $W_{\xi(i,j)}^{\varphi} = L$ if the limiting value of g on i is the $(j+1)^{th}$ grammar output by g.

We now define the acceptable programming system ψ. For ease in describing ψ, we first define two increasing recursive functions q_1 and q_2.

$q_1(i) = \sum_{0 \leq j < i}(v(j)+1)$;
$q_2(i) = q_1(i+1) - 1$.

Intuitively, N is divided into segments of length $v(0)+1$, $v(1)+1$, ..., etc. $q_1(i)$ and $q_2(i)$ denote the endpoints of the i-th segment.

begin {Definition of W_i^{ψ}}
if $i = q_2(k)$ for some k
then
let $W_i^{\psi} = W_k^{\varphi}$
{This makes ψ acceptable.}
else
let $j = \min(\{k \mid q_2(k) \geq i\})$;
let $l = i - q_1(j)$;
if $\xi(j,l)\downarrow$
then
let $W_i^{\psi} = W_{\xi(j,l)}^{\varphi}$
endif
endif
end {Definition of W_i^{ψ}}

Clearly, ψ is an acceptable programming system since φ-indices could be reduced to ψ-indices by the recursive function q_2. We now define a two argument recursive function ConvProg which for any j such that $W_j^{\varphi} \in \mathcal{L}$, converges to a ψ-grammar for W_j^{φ}, i.e., $\lim_{n\to\infty} \mathrm{ConvProg}(j,n)$ exists and is equal to a ψ-grammar for W_j^{φ}. Moreover, if W_j^{φ} is infinite, then $\lim_{n\to\infty} \mathrm{ConvProg}(j,n)$ is equal to the minimal ψ-grammar for W_j^{φ}.

begin {Definition of ConvProg(j,n)}

{We first define two values m_1 and m_2.}

{Recall from Section 2 that $\mu n.[Q(n)]$ is 0 if no minimum n exists such that $Q(n)$ is true.}

$m_1 = \mu i.[(i \leq g(j,n)) \ \wedge \ (\exists k < v(i))[\eta(i,k)\downarrow = g(j,n) \text{ in} \leq n \text{ steps }]]$;

$m_2 = \mu k.[(k < v(m_1)) \ \wedge \ (\eta(m_1,k)\downarrow = g(j,n) \text{ in} \leq n \text{ steps })]$;

ConvProg$(j,n) = q_1(m_1) + m_2$

end {Definition of ConvProg(j,n)}

Claim 17 *If $W_j^{\varphi} \in \mathcal{L}$, then* $\lim_{n\to\infty}$ ConvProg(j,n) *exists. Moreover,* $\lim_{n\to\infty}$ ConvProg(j,n) *is a ψ-grammar for W_j^{φ}.*

Proof: Since $\lim_{n\to\infty} g(j,n)$ exists, let $\lim_{n\to\infty} g(j,n) = l$. Let

$$m_1' = \mu i.[(i \leq l) \ \wedge \ (\exists k < v(i))[\eta(i,k)\downarrow = l]]$$

and

$$m_2' = \mu k.[(k < v(m_1')) \ \wedge \ (\eta(m_1',k)\downarrow = l)].$$

Clearly, $\lim_{n\to\infty}$ ConvProg$(j,n) = q_1(m_1') + m_2'$. Also, $W^{\psi}_{q_1(m_1')+m_2'} = W^{\varphi}_{\xi(m_1',m_2')} = W^{\varphi}_{p(\eta(m_1',m_2'))} = W^{\varphi}_{p(l)} = W_l^{\varphi} = W_j^{\varphi}$ (By definition of p,η,ψ,ξ). The claim follows. ∎

Claim 18 $[W_j^{\varphi} \in \mathcal{L} \ \wedge \ \text{card}(W_j^{\varphi}) = \infty] \Rightarrow [\lim_{n\to\infty} \text{ConvProg}(j,n) = \textbf{mingrammar}_{\psi}(W_j^{\phi})]$.

Proof: Let g on any grammar for $L \in \mathcal{L}$ converge to g_L. Let j be a φ grammar for an infinite $L \in \mathcal{L}$. Thus, for large enough $n, g(j,n) = g_L$. Clearly, $W^{\varphi}_{p(g_L)} = W^{\varphi}_{g_L} = L$ (by definition of p). Now, we have the following:

(1) $(\forall k,l \mid q_1(k) \leq l < q_2(k))[[W_l^{\psi}$ is finite $] \ \vee \ [[W_l^{\psi} = W^{\varphi}_{\xi(k,l-q_1(k))} = W^{\varphi}_{p(\eta(k,l-q_1(k)))} = W^{\varphi}_{\eta(k,l-q_1(k))}] \ \wedge \ [\text{card}(\{n \mid g(\eta(k,l-q_1(k)),n) = \eta(k,l-q_1(k))\})$ is infinite$]]]$. (By definition of ψ, p, ξ, η).

(2) Let $m_1' = \mu i.[(i \leq g_L) \ \wedge \ (\exists k < v(i))[\eta(i,k)\downarrow = g_L]]$ and $m_2' = \mu i.[(i < v(m_1')) \ \wedge (\eta(m_1',i)\downarrow = g_L)]$. Clearly, for all $i < m_1'$, $W_i^{\varphi} = W^{\psi}_{q_2(i)} \neq L$ (otherwise by definition of g and η there would be a $k < v(i)$ such that $\eta(i,k) = g_L$).

From (1) and (2), we have that $q_1(m_1') + m_2'$ is the minimal ψ-grammar for $W_j^{\varphi} = L$. Hence, ConvProg$(g(j,n),n)$ converges to the minimal ψ-grammar for W_j^{φ}. This proves the claim. ∎

We now describe a machine $\mathbf{M}'$ that $\mathbf{TxtMin}_{\psi}$-identifies $\mathcal{L}$.

begin {Definition of $\mathbf{M}'(T[n])$}

if $(\exists i \leq \text{ConvProg}(\mathbf{M}(T[n]),n))[W_i^{\psi,n} = \text{content}(T[n])]$

then

output $\mu i.[(i \leq \text{ConvProg}(\mathbf{M}(T[n]), n)) \ \wedge \ W_i^{\psi,n} = \text{content}(T[n])]$

else

output $\text{ConvProg}(\mathbf{M}(T[n]), n)$

endif

end {Definition of $\mathbf{M}'(T[n])$}

Given any text T for $L \in \mathcal{L}$, let n_0 be so large that $(\forall n \geq n_0)[\mathbf{M}(T[n]) = \mathbf{M}(T[n_0])]$. Let $n_1 > n_0$ be so large that $(\forall n \geq n_1)[\text{ConvProg}(\mathbf{M}(T), n) = \text{ConvProg}(\mathbf{M}(T), n_1)]$. Clearly, such n_0, n_1 exist. We now have following two cases:
Case 1: $\text{card}(L) = \infty$.

In this case, by Claim 18, $\text{ConvProg}(\mathbf{M}(T), n)$ converges to the minimal ψ grammar for L. Let $n_2 > n_1$ be so large that $(\forall n \geq n_2)(\forall i < \text{ConvProg}(\mathbf{M}(T), n_1))[W_i^{\psi,n} \not\subseteq \text{content}(T[n]) \ \vee \ \text{content}(T[n]) \not\subseteq W_i^{\psi,n}]$. Clearly, such an n_2 exists. Thus, for all $n > n_2$, $\mathbf{M}'$ outputs $\text{ConvProg}((\mathbf{M}(T), n_1))$ — the minimal ψ-grammar for L.
Case 2: $\text{card}(L)$ is finite.

In this case, by Claim 17, $\text{ConvProg}(\mathbf{M}(T), n)$ converges to a grammar for L in ψ programming system. Let $j \leq \text{ConvProg}(\mathbf{M}(T), n_1)$ be the minimal ψ-grammar for L. By construction of $\mathbf{M}'$, for large enough n, $\mathbf{M}'$ outputs j. This proves the theorem. ∎

5 Nearly Minimal Identification

Freivalds [8] considered **Ex**-identification of programs which are of minimal size modulo a recursive (fudge) factor, i. e., the programs inferred are nearly minimal size. Definition 19 just below describes this notion.

Definition 19 [8, 7]
(a) Let $h \in \mathcal{R}$. $\mathbf{M}$ h-$\mathbf{Mex}_\psi$-*identifies* $\mathcal{S}$ (written: $\mathcal{S} \subseteq h\text{-}\mathbf{Mex}_\psi(\mathbf{M})$) $\iff$ $(\forall f \in \mathcal{S})[\mathbf{M}(f)\downarrow \ \wedge \ \psi_{\mathbf{M}(f)} = f \ \wedge \ \mathbf{M}(f) \leq h(\mathbf{minprogram}_\psi(f))]$.
(b) Let $h \in \mathcal{R}$. $h\text{-}\mathbf{Mex}_\psi = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq h\text{-}\mathbf{Mex}_\psi(\mathbf{M})]\}$.
(c) $\mathbf{Mex}_\psi = \{\mathcal{S} \mid (\exists h \in \mathcal{R})[\mathcal{S} \in h\text{-}\mathbf{Mex}_\psi]\}$.

Intuitively, a learning machine $\mathbf{M}$ h-$\mathbf{Mex}_\psi$-identifies $\mathcal{S}$ iff for each $f \in \mathcal{S}$, $\mathbf{M}$ **Ex**-identifies f in the acceptable programming system ψ, and the final stabilized output of $\mathbf{M}$ on f is no bigger than $h(\mathbf{minprogram}_\psi(f))$. It is easy to verify that for any ψ and ψ', $\mathbf{Mex}_\psi = \mathbf{Mex}_{\psi'}$; hence, we will refer to $\mathbf{Mex}_\psi$ as just **Mex**.

Freivalds showed an interesting result about h-$\mathbf{Mex}_\psi$-identification described in Theorem 20 below. Theorem 20 and Proposition 10 implies Corollary 21 which says that the inferring power of $\mathbf{Min}_\psi$-identification is dependent on the choice of acceptable programming system ψ.

Theorem 20 [8] $(\forall h \in \mathcal{R} \mid (\forall x)[h(x) \geq x])(\exists \psi)(\forall \mathcal{S})\ [\mathcal{S} \in h\text{-}\mathbf{Mex}_{\psi} \iff \text{card}(\mathcal{S}) < \infty]$.

Corollary 21 [8] $(\exists \psi', \psi'')[\mathbf{Min}_{\psi'} \neq \mathbf{Min}_{\psi''}]$.

Case and Chi [4] considered an analog of nearly minimal identification in the context of language learning; Definition 22 below introduces this notion.

Definition 22 [4]
(a) Let $h \in \mathcal{R}$. $\mathbf{M}$ h-$\mathbf{TxtMex}_{\psi}$-*identifies* $\mathcal{L}$ (written: $\mathcal{L} \subseteq h\text{-}\mathbf{TxtMex}_{\psi}(\mathbf{M})$) $\iff$ $(\forall L \in \mathcal{L})$ ($\forall$ texts T for L) $[\mathbf{M}(T)\downarrow\ \wedge\ W^{\psi}_{\mathbf{M}(T)} = L\ \wedge\ \mathbf{M}(T) \leq h(\mathbf{mingrammar}_{\psi}(L))]]$.
(b) Let $h \in \mathcal{R}$. $h\text{-}\mathbf{TxtMex}_{\psi} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq h\text{-}\mathbf{TxtMex}_{\psi}(\mathbf{M})]\}$.
(c) $\mathbf{TxtMex}_{\psi} = \{\mathcal{L} \mid (\exists h \in \mathcal{R})[\mathcal{L} \in h\text{-}\mathbf{TxtMex}_{\psi}]\}$.

Intuitively, a learning machine $\mathbf{M}$ h-$\mathbf{TxtMex}_{\psi}$-identifies $\mathcal{L}$ iff for each L in $\mathcal{L}$, $\mathbf{M}$ $\mathbf{TxtEx}$-identifies L in the acceptable programming system ψ, and for each text T for L the final stabilized output of $\mathbf{M}$ on T is no bigger than $h(\mathbf{mingrammar}_{\psi}(L))$. It is easy to verify that for any ψ and ψ', $\mathbf{TxtMex}_{\psi} = \mathbf{TxtMex}_{\psi'}$; hence, we will refer to $\mathbf{TxtMex}_{\psi}$ as just $\mathbf{TxtMex}$.

We are able to show in Theorem 23 below a language learning analog of Freivalds' result (Theorem 20). It should be noted that Theorem 23 below *does not* hold if '$\Rightarrow$' in the statement of theorem is replaced by '$\iff$'.

Theorem 23 $(\forall h \in \mathcal{R})(\exists \psi)(\forall \mathcal{L})\ [\mathcal{L} \in h\text{-}\mathbf{TxtMex}_{\psi} \Rightarrow \text{card}(\{L \mid L \in \mathcal{L}\ \wedge\ \text{card}(L)\ \textit{is}\ \infty\}) < \infty]$.

Proof of Theorem 23: Let h be as given in the hypothesis of the theorem. Without loss of generality, we can assume that h is increasing. We now define an acceptable programming system ψ such that if $\mathcal{L} \in h\text{-}\mathbf{TxtMex}_{\psi}$, then $\mathcal{L}$ has only a finite number of infinite languages. To facilitate the description of ψ, we first define two recursive functions g_1 and g_2. But, first we have the following notation:

Let $H^0(i) = i$, $H(i) = h(i) + 1$. For $j \geq 2$, let $H^j(i) = H^{j-1}(H(i))$.

$g_1(0) = 0$;

$g_2(i) = H^{i+2}(g_1(i) + 1)$;

$g_1(i+1) = g_2(i)$.

For any $j, s \in N$, let σ^s_j denote a finite sequence uniformly formed from j and s such that the following two conditions hold.

(i) $\text{content}(\sigma^s_j) = W^{\varphi,s}_j$;

(ii) $\sigma^s_j \subset \sigma^{s+1}_j$.

begin {Definition of W^{ψ}_i}

1. **if** $i = g_1(k+1)$ for some $k \geq 0$

then

let $W_i^{\psi} = W_k^{\varphi}$;

exit

{This makes ψ acceptable.}

endif;

2. let $j = \min(\{k \mid g_1(k+1) \geq i\})$;

let $S = \{H^k(g_1(j)+1) \mid 0 \leq k \leq (j+1)\}$;

{Note that S contains $j+2$ elements each H apart.}

if $i \notin S$

then

let $W_i^{\psi} = \emptyset$

else

go to stage 0;

{W_i^{ψ} will either be finite or equal to W_j^{φ}}

begin {stage s}

if there exists $k \leq j$ such that $\mathbf{M}_k(\sigma_j^s) = i$

then

go to stage $s+1$

else

enumerate $W_j^{\varphi,s}$;

go to stage $s+1$

endif

end {stage s}

endif

end {Definition of W_i^{ψ}}

Claim 24 *For any i, W_i^{ψ} is either finite or equal to W_j^{φ}, where $j = \min(\{k \mid g_1(k+1) \geq i\})$.*

Proof: W_i^{ψ} can be infinite either due to step 1 or the enumeration by infinitely many stages in step 2. In both cases, $W_i^{\psi} = W_j^{\varphi}$, where $j = \min(\{k \mid g_1(k+1) \geq i\})$. ∎

Claim 25 *Let $L \in \mathcal{L} \wedge \text{ card}(L) = \infty$. Then, $g_1(\mathbf{mingrammar}_{\varphi}(L)) < \mathbf{mingrammar}_{\psi}(L) \leq g_1(\mathbf{mingrammar}_{\varphi}(L)+1)$.*

Proof: Clear from Claim 24 and step 1 in the construction of W_i^{ψ} above. ∎

Now suppose by way of contradiction that some machine $\mathbf{M}_k$ h-$\mathbf{TxtMex}_{\psi}$-identifies $\mathcal{L}$ which contains infinitely many infinite languages. Consider an infinite language $L \in \mathcal{L}$ such that $\mathbf{mingrammar}_{\varphi}(L) = j > k$. By Claim 25, we have $g_1(j) < \mathbf{mingrammar}_{\psi}(L) \leq g_1(j+1)$.

Clearly, $T = \bigcup_{s \in N} \sigma_j^s$ is a text for L. Assume that $\mathbf{M}_k(T) > g_1(j)$ (otherwise $\mathbf{M}_k$ does not **TxtEx**-identify L). We now consider the following three cases.

Case 1: $\mathbf{M}_k$ on T converges to a grammar $i \in S = \{H^k(g_1(j)+1) \mid 0 \leq k \leq (j+1)\}$.

In this case, by construction, W_i^{ψ} is finite, and thus, $\mathbf{M}_k$ does not h-**TxtMex**$_\psi$-identify L.

Case 2: $\mathbf{M}_k$ on T converges to a grammar i such that $g_1(j) < i < g_1(j+1)$ and $i \notin S = \{H^k(g_1(j)+1) \mid 0 \leq k \leq (j+1)\}$.

In this case, $W_i^{\psi} = \emptyset$, and thus, $\mathbf{M}_k$ does not h-**TxtMex**$_\psi$-identify L.

Case 3: Not Case 1 and not Case 2.

Consider $S = \{H^k(g_1(j)+1) \mid 0 \leq k \leq (j+1)\}$. Since cardinality of S is $j+2$, and there are only $j+1$ machines $\mathbf{M}_{j'}, j' \leq j$, at least for one $i \in S$ we have $W_i^{\psi} = W_j^{\varphi} = L$. Also, for each $i \in S$, $h(i) < g_1(j+1)$. Thus, $\mathbf{M}_k$ does not converge to within h of the minimal ψ-grammar for L.

The above cases prove the theorem. ∎

Using techniques developed in Theorem 16, it is easy to verify that if $\mathcal{L} \in$ **TxtEx** and $\mathrm{card}(\{L \in \mathcal{L} \mid \mathrm{card}(L) = \infty\}) < \infty$, then for all ψ, $\mathcal{L} \in$ **TxtMin**$_\psi$.

Corollary 26 below follows from Theorem 23 and Proposition 14, and it says that **TxtMin**$_\psi$-identification is dependent on the choice of acceptable programming system ψ — a seemingly discouraging result as it precludes the study of **TxtMin**$_\psi$-identification as an interesting criterion of language learning.

Corollary 26 $(\exists \psi', \psi'')[\mathbf{TxtMin}_{\psi'} \neq \mathbf{TxtMin}_{\psi''}]$.

Before we move on to other kinds of program size restrictions, we investigate a few more characteristics of h-**Mex**$_\psi$-identification and h-**TxtMex**$_\psi$-identification criteria. Proposition 27 and Proposition 28 below are simple observations about these criteria.

Proposition 27 *Suppose $h \in \mathcal{R}$ such that $(\forall x)[h(x) > x]$. Then, $(\forall \psi)[\lambda x.[h(x)-1]$-**Mex**$_\psi \subseteq h$-**Mex**$_\psi]$.*

Proposition 28 *Suppose $h \in \mathcal{R}$ such that $(\forall x)[h(x) > x]$. Then, $(\forall \psi)[\lambda x.[h(x)-1]$-**TxtMex**$_\psi \subseteq h$-**TxtMex**$_\psi]$.*

An interesting question to ask is whether there are acceptable programming systems for which the containment in Propositions 27 and 28 is proper. Theorems 29 and 35 below imply that such acceptable programming systems exist. However, Theorems 41 and 42 imply that there exist acceptable programming systems for which this is not the case.

Theorem 29 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that $(\forall x)[h_i(x) > x]$. Then, $(\exists \psi)$ $(\forall i)[\lambda x.[h_i(x)-1]$-**Mex**$_\psi \subset h_i$-**Mex**$_\psi]$.*

Proof of Theorem 29: For this proof, without loss of generality, we assume that the standard acceptable programming system φ is such that there exists an infinite r.e. class of functions which can be $\mathbf{Min}_{\varphi}$-identified; Proposition 10 allows us to make such an assumption. Let $f_0, f_1, \ldots$ be an infinite r. e. sequence of distinct total recursive functions identifiable in the limit by minimal program in φ-system, i.e., $\{f_i \mid i \in N\} \in \mathbf{Min}_{\varphi}$.

Let $\mathcal{S}_i = \{f_{\langle i,j \rangle} \mid j \in N\}$. We will use $\mathcal{S}_i$ to show that h_i-$\mathbf{Mex}_{\psi} \neq (\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_{\psi}$. The function $f_{\langle i,j \rangle}$ will be used to diagonalize against the learning machine $\mathbf{M}_j$.

To facilitate the description of acceptable programming system ψ, we first define recursive functions g_1, g_2 below.

$g_1(0) = 0$;

$g_2(k) = \max(\{h_0(g_1(k)), h_1(g_1(k)), ..., h_k(g_1(k))\})$;

$g_1(k+1) = g_2(k) + 2$.

We set up the programming system ψ in such a way that for all i, for all but finitely many j, ψ-program $h_i(g_1(\mathbf{minprogram}_{\varphi}(f_{\langle i,j \rangle})))$ computes $f_{\langle i,j \rangle}$. We use the numbers from $g_1(\mathbf{minprogram}_{\varphi}(f_{\langle i,j \rangle}))$ to $h_i(g_1(\mathbf{minprogram}_{\varphi}(f_{\langle i,j \rangle}))) - 1$ for diagonalization. We now give a description of ψ.

begin {Definition of $\psi_k(x)$}

execute the following steps until $\psi_k(x)$ is defined:

1. **if** $k = g_2(i) + 1$ for some i

then

let $\psi_k(x) = \varphi_i(x)$;

exit

{Note that this makes ψ an acceptable programming system.}

endif;

2. let lastfun $= \max(\{i \mid g_1(i) \leq k\})$;

if $(\forall x' \leq x)[\varphi_{\text{lastfun}}(x')\downarrow] \ \wedge \ (\forall x' < x)[\psi_k(x')\downarrow]$

then

proceed to step 3

else

diverge

{ Thus, here $\psi_k(x)$ is undefined.}

endif;

3. **if** $(\forall i \leq x)(\exists x' \leq x)[\varphi_{\text{lastfun}}(x') \neq f_i(x')]$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$;

exit

endif;

4. let $j = \min(\{i \mid (\forall x' \leq x)[f_i(x') = \varphi_{\text{lastfun}}(x')]\})$;

Let j_1, j_2 be such that $j = \langle j_1, j_2 \rangle$;

if $k = h_{j_1}(g_1(\text{lastfun}))$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$;

exit

endif;

{Note that this would ensure $\psi_{h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})))} = f_{\langle j_1,j_2\rangle}$.}

5. **if** $[(\exists x' \geq x)[\varphi_{\text{lastfun}}(x')\downarrow \neq f_j(x')]] \vee$

$[(\exists x' \geq x)[[(\forall x'' \leq x')\varphi_{\text{lastfun}}(x'')\downarrow] \wedge [\mathbf{M}_{j_2}(f_j[x']) \neq k]]]$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$

else

diverge

{ Thus, here $\psi_k(x)$ is undefined. }

endif

end {Definition of $\psi_k(x)$}

Claim 30 *ψ is an acceptable programming system.*

Proof: Clearly ψ is a programming system, and $\lambda i.[g_2(i) + 1]$ reduces φ-indices to ψ-indices. Hence, ψ is an acceptable programming system. ∎

Claim 31 $(\forall j)(\forall k)[[g_1(j) \leq k < g_1(j+1)] \Rightarrow (\forall x)[\psi_k(x) = \varphi_j(x) \vee \psi_k(x)\uparrow]]$.

Proof: Whenever $\psi_k(x)$ is defined in the above procedure, $\psi_k(x) = \varphi_{\text{lastfun}}(x)$, where lastfun $= \max(\{i \mid g_1(i) \leq k\})$. The claim follows. ∎

Claim 32 $(\forall j_1, j_2)[[\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}) > j_1] \Rightarrow [[h_{j_1}(\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle})) \geq h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})))] \wedge [\psi_{h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})))} = f_{\langle j_1,j_2\rangle}]]]$.

Proof: Let $\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}) > j_1$. It follows from Claim 31 that

$$\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle}) \geq g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})),$$

which implies

$$h_{j_1}(\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle})) \geq h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))).$$

Now, let $k = h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})))$. We need to show that $\psi_k = f_{\langle j_1,j_2\rangle}$. Suppose by way of contradiction that $(\exists x)[\psi_k(x)\uparrow]$ (if no such x exists then by Claim 31 we will have $\psi_k = f_{\langle j_1,j_2\rangle}$). Let x_0 be the least such x. Now, if j calculated in step 4 is equal to $\langle j_1, j_2\rangle$, then by the **if** statement at step 4, we have $\psi_k(x_0)\downarrow$. Therefore, let $j \neq \langle j_1, j_2\rangle$. But, then in step 5, **if** condition will succeed, and thus, $\psi_k(x_0)\downarrow$. A contradiction. Hence, $(\forall x)[\psi_k(x)\downarrow]$. This proves the claim. ∎

Claim 33 $(\forall i)[\mathcal{S}_i \in h_i\text{-}\mathbf{Mex}_\psi]$.

Proof: Let machine $\mathbf{M}$ $\mathbf{Min}_\varphi$-identify $\{f_i \mid i \in N\}$. We define a machine $\mathbf{M}'$ as follows: $\mathbf{M}'(f[n]) = h_i(g_1(\mathbf{M}(f[n])))$. By Claim 32, it follows that for all $f_{\langle i,j\rangle}$ such that $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) > i$, $\mathbf{M}'$ h_i-$\mathbf{Mex}_\psi$-identifies $f_{\langle i,j\rangle}$. But, there are only finitely many j such that $f_{\langle i,j\rangle} \in \mathcal{S}_i$ and $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) \leq i$. Thus, $\mathbf{M}'$ can easily be modified to h_i-$\mathbf{Mex}_\psi$-identify $\mathcal{S}_i$. ∎

Claim 34 $(\forall i)[\mathcal{S}_i \notin (\lambda x.[h_i(x) - 1])\text{-}\mathbf{Mex}_\psi]$.

Proof:

Suppose by way of contradiction, machine $\mathbf{M}_j$ $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identifies $\mathcal{S}_i$. Without loss of generality, assume that $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) > i$ (since, if $\mathbf{M}_j$ $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identifies $\mathcal{S}_i$, then there are infinitely many machines that $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identify $\mathcal{S}_i$). Consider the function $f = f_{\langle i,j\rangle} \in \mathcal{S}_i$. We show that $\mathbf{M}_j$ does not $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identify f. This would prove the claim.

Assume that $\mathbf{M}_j(f)\downarrow \geq g_1(\mathbf{minprogram}_\varphi(f))$ (otherwise, by Claim 31, $\mathbf{M}_j$ does not **Ex**-identify f). We consider the following two cases:

Case 1: $\mathbf{M}_j$ on f converges to k, where $g_1(\mathbf{minprogram}_\varphi(f)) \leq k < h_i(g_1(\mathbf{minprogram}_\varphi(f)))$.

Let x' be so large that

a) $x' > \langle i, j\rangle$;

b) $(\forall k' < \langle i, j\rangle)(\exists x < x')[f_{k'}(x) \neq f_{\langle i,j\rangle}(x)]$;

c) $\neg(\exists n \geq x')[\mathbf{M}(f[n]) \neq k]$.

Clearly such an x' exists. Now, for $x = x'$, **if** condition at steps $1, 3, 4$, and 5 in the construction of ψ are not satisfied. Thus, $\psi_k(x')\uparrow$. Hence, in this case, $\mathbf{M}_j$ does not $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identify f.

Case 2: Not case 1.

In this case, $\mathbf{M}_j$ on f does not converge to $k_0 = g_1(\mathbf{minprogram}_\varphi(f))$. We claim that $\psi_{k_0} = f$. This would imply that $\mathbf{M}_j$ on f does not converge to a small enough program, and hence, $\mathbf{M}_j$ fails to $(\lambda x.[h_i(x) - 1])$-$\mathbf{Mex}_\psi$-identify f.

Suppose by way of contradiction, $(\exists x)\psi_{k_0}(x)\uparrow$ (otherwise by Claim 31 we would have $\psi_{k_0} = f$). Let x_0 be the least such x. Since, **if** condition at step 2 succeeds, either $\psi_{k_0}(x_0)$ is defined before step 5, or it would be defined at step 5 (since **if** condition at step 5 would succeed). Thus, $\psi_{k_0}(x_0)\downarrow$. A contradiction.

From the above two cases, it follows that $\mathbf{M}_j$ does not $(\lambda x.[h_i(x)-1])$-$\mathbf{Mex}_\psi$-identify $f \in \mathcal{S}_i$. Hence, $\mathcal{S}_i \notin (\lambda x.[h_i(x)-1])$-$\mathbf{Mex}_\psi$. ∎

The theorem follows from Claims 33 and 34. ∎

The above proof of Theorem 29 can be adapted for single valued total languages to show Theorem 35 below—a language learning analog of Theorem 29. We give the details of such an adaptation.

Theorem 35 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that $(\forall x)[h_i(x) > x]$. Then, $(\exists \psi)$ $(\forall i)[\lambda x.[h_i(x)-1]$-$\mathbf{TxtMex}_\psi \subset h_i$-$\mathbf{TxtMex}_\psi]$.*

Proof of Theorem 35: For this proof, without loss of generality, we assume that the standard acceptable programming system φ is such that there exists an infinite r.e. class of single valued total languages which can be $\mathbf{TxtMin}_\varphi$-identified; proof of Proposition 14 allows us to make such an assumption.

Let $L_0, L_1, \ldots$ be an infinite r.e. sequence of *distinct* **svt** languages identifiable in the limit by minimal grammar in the φ-system, i.e., $\{L_i \mid i \in N\} \in \mathbf{TxtMin}_\varphi$.

Let $\mathcal{L}_i = \{L_{\langle i,j\rangle} \mid j \in N\}$. We will use $\mathcal{L}_i$ to show that h_i-$\mathbf{TxtMex}_\psi \neq (\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_\psi$. The language $L_{\langle i,j\rangle}$ will be used to diagonalize against the learning machine $\mathbf{M}_j$.

To facilitate the description of acceptable programming system ψ, we first define recursive functions g_1, g_2 below.

$g_1(0) = 0$;

$g_2(k) = \max(\{h_0(g_1(k)), h_1(g_1(k)), ..., h_k(g_1(k))\})$;

$g_1(k+1) = g_2(k) + 2$.

Clearly, g_1 is an increasing recursive function. We set up the programming system ψ in such a way that ψ-grammar $h_i(g_1(\mathbf{mingrammar}_\varphi(L_{\langle i,j\rangle})))$ enumerates $L_{\langle i,j\rangle}$. We use the numbers from $g_1(\mathbf{mingrammar}_\varphi(L_{\langle i,j\rangle}))$ to $h_i(g_1(\mathbf{mingrammar}_\varphi(L_{\langle i,j\rangle}))) - 1$ for diagonalization. We now give a description of ψ. Let T^j be a text for L_j, such that $T^j[n]$ can be found effectively from j, n.

begin {Definition of W_k^ψ}

1. **if** $k = g_2(i) + 1$ for some i

   then

   let $W_k^\psi = W_i^\varphi$;

   exit

{Note that this makes ψ an acceptable programming system.}

else

proceed to stage 0

endif;

begin {Stage x}

2. let lastlan $= \max(\{i \mid g_1(i) \leq k\})$;

3. **if** $(\exists w, y, z)[y \neq z \ \wedge \ \{\langle w, y\rangle, \langle w, z\rangle\} \subseteq W^{\varphi,x}_{\text{lastlan}}]$

then

go to step 7

{Note that nothing more gets enumerated in W^{ψ}_{k}. }

endif;

{Note that if $W^{\varphi}_{\text{lastlan}}$ is not single valued, then W^{ψ}_{k} is finite.}

4. **if** $(\exists y)[\langle x, y\rangle \in W^{\varphi}_{\text{lastlan}}]$

then

proceed to step 5

else

nothing more gets enumerated in W^{ψ}_{k}

endif;

{Note that steps 3 and 4 ensure that if $W^{\varphi}_{\text{lastlan}}$ is not **svt**, then W^{ψ}_{k} is finite.}

5. let $j = \min(\{i \mid W^{\varphi,x}_{\text{lastlan}} \subseteq L_i\})$;

let j_1, j_2 be such that $j = \langle j_1, j_2\rangle$;

if $k = h_{j_1}(g_1(\text{lastlan}))$

then

enumerate $W^{\varphi,x}_{\text{lastlan}}$ in W^{ψ}_{k};

go to stage $x + 1$

endif;

{note that this would ensure $W^{\psi}_{h_{j_1}(g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1, j_2\rangle})))} = L_{\langle j_1, j_2\rangle}$}

6. **if** $[(\exists x')[x' \in [W^{\varphi}_{\text{lastlan}} - L_j]] \ \vee \ [(\exists x' > x)[\mathbf{M}_{j_2}(T^j[x']) \neq k]]]$

then

enumerate $W^{\varphi,x}_{\text{lastlan}}$ in W^{ψ}_{k};

go to stage $x + 1$

else

nothing more gets enumerated in W^{ψ}_{k}.

endif

end {Stage x}

7. do not enumerate anything more in W^{ψ}_{k}

end {Definition of W^{ψ}_{k}}

Claim 36 *ψ is an acceptable programming system.*

Proof: Clearly, ψ is a programming system, and $\lambda i.[g_2(i)+1]$ reduces φ-indices to ψ-indices. Hence, ψ is an acceptable programming system. $\blacksquare$

Claim 37 $(\forall j)(\forall k)[[g_1(j) \leq k < g_1(j+1)] \Rightarrow [W_k^{\psi} \subseteq W_j^{\varphi}]]$.

Proof: The procedure for W_k^{ψ} described above enumerates only subsets of $W_{\text{lastlan}}^{\varphi}$, where $\text{lastlan} = \max(\{i \mid g_1(i) \leq k\})$. The claim follows. $\blacksquare$

Claim 38 $(\forall j_1, j_2)[[\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle}) > j_1] \Rightarrow [[\ h_{j_1}(\mathbf{mingrammar}_{\psi}(L_{\langle j_1,j_2\rangle})) \geq h_{j_1}(g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})))] \wedge\ [W^{\psi}_{h_{j_1}(g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})))} = L_{\langle j_1,j_2\rangle}]]]$.

Proof: Suppose the hypothesis, i.e., $\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle}) > j_1$. It follows from Claim 37 that $\mathbf{mingrammar}_{\psi}(L_{\langle j_1,j_2\rangle}) \geq g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle}))$, which implies $h_{j_1}(\mathbf{mingrammar}_{\psi}(L_{\langle j_1,j_2\rangle})) \geq h_{j_1}(g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})))$.

Now, let $k = h_{j_1}(g_1(\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})))$. We need to show that $W_k^{\psi} = L_{\langle j_1,j_2\rangle}$. Suppose by way of contradiction that $(\exists x)[W^{\varphi,x}_{\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})} \not\subseteq W_k^{\psi}]$ (if no such x exists then by Claim 37 we will have $W_k^{\psi} = L_{\langle j_1,j_2\rangle}$). Let x_0 be the least such x. Now, if j calculated in step 5 of stage x_0 is equal to $\langle j_1, j_2\rangle$, then by the **if** statement at step 5 we have $W^{\varphi,x_0}_{\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})} \subseteq W_k^{\psi}$. Therefore, let $j \neq \langle j_1, j_2\rangle$. But, then in step 6, the **if** condition will succeed and thus $W^{\varphi,x_0}_{\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})} \subseteq W_k^{\psi}$. This is a contradiction. Hence, $(\forall x)[W^{\varphi,x}_{\mathbf{mingrammar}_{\varphi}(L_{\langle j_1,j_2\rangle})} \subseteq W_k^{\psi}]$. $\blacksquare$

Claim 39 $(\forall i)[\mathcal{L}_i \in h_i\text{-}\mathbf{TxtMex}_{\psi}]$.

Proof: Let machine $\mathbf{M}$ $\mathbf{TxtMin}_{\varphi}$-identify $\{L_i \mid i \in N\}$. We define a machine $\mathbf{M}'$ as follows:

$\mathbf{M}'(T[n]) = h_i(g_1(\mathbf{M}(T[n])))$.

By Claim 38, it follows that for all $L_{\langle i,j\rangle}$ such that $\mathbf{mingrammar}_{\varphi}(L_{\langle i,j\rangle}) > i$, $\mathbf{M}'$ h_i-$\mathbf{TxtMex}_{\psi}$-identifies $L_{\langle i,j\rangle}$. But, there are only finitely many j such that $L_{\langle i,j\rangle} \in \mathcal{L}_i$ and $\mathbf{mingrammar}_{\varphi}(L_{\langle i,j\rangle}) \leq i$. Thus, $\mathbf{M}'$ can easily be modified to h_i-$\mathbf{TxtMex}_{\psi}$-identify $\mathcal{L}_i$. $\blacksquare$

Claim 40 $(\forall i)[\mathcal{L}_i \notin (\lambda x.[h_i(x)-1])\text{-}\mathbf{TxtMex}_{\psi}]$.

Proof: Suppose by way of contradiction, machine $\mathbf{M}_j$ $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_{\psi}$-identifies $\mathcal{L}_i$. Without loss of generality, assume that $\mathbf{mingrammar}_{\varphi}(L_{\langle i,j\rangle}) > i$ (since, if $\mathbf{M}_j$ $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_{\psi}$-identifies $\mathcal{L}_i$, then there are infinitely many machines that $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_{\psi}$-identify $\mathcal{L}_i$). Consider the language $L = L_{\langle i,j\rangle} \in \mathcal{L}_i$. We show that $\mathbf{M}_j$ does not $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_{\psi}$-identify L. This would prove the claim.

Assume that $\mathbf{M}_j(T^{\langle i,j\rangle})\downarrow \geq g_1(\mathbf{mingrammar}_\varphi(L))$ (otherwise $\mathbf{M}_j$ does not **TxtEx**-identify L). We consider the following two cases:

Case 1: $\mathbf{M}_j$ on $T^{\langle i,j\rangle}$ converges to k, where $g_1(\mathbf{mingrammar}_\varphi(L)) \leq k < h_i(g_1(\mathbf{mingrammar}_\varphi(L)))$.

Let x' be so large that

(a) $x' > \langle i,j\rangle$;

(b) $(\forall k' < \langle i,j\rangle)(\exists x)[x \in W^{\varphi,x}_{\mathbf{mingrammar}_\varphi(L)} - L_{k'}]$;

(c) $\neg(\exists n \geq x')[\mathbf{M}(T^{\langle i,j\rangle}[n]) \neq k]$.

Clearly, such an x' exists. Now, in the definition of W^ψ_k, the **if** condition in step 1 is not satisfied and for $x = x'$, the **if** condition in steps 5 and 6 are not satisfied. Thus, W^ψ_k is finite. Hence, in this case $\mathbf{M}_j$ does not $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_\psi$-identify L.

Case 2: Not case 1.

In this case, $\mathbf{M}_j$ on $T^{\langle i,j\rangle}$ does not converge to $k_0 = g_1(\mathbf{mingrammar}_\varphi(L))$. We claim that $W^\psi_{k_0} = L$. This would imply that $\mathbf{M}_j$ on $T^{\langle i,j\rangle}$ does not converge to a small enough grammar, and hence, $\mathbf{M}_j$ fails to $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_\psi$-identify L.

Suppose by way of contradiction, $(\exists x)[W^{\varphi,x}_{\mathbf{mingrammar}_\varphi(L)} \not\subseteq W^\psi_{k_0}]$ (otherwise by Claim 37 we would have $W^\psi_{k_0} = L$). Let x_0 be the least such x. Now at stage x_0, **if** condition at step 3 fails and **if** condition at step 4 succeeds; hence, either $W^{\varphi,x_0}_{\mathbf{mingrammar}_\varphi(L)}$ is enumerated in $W^\psi_{k_0}$ before step 6, or it would be enumerated at step 6 (since the **if** condition at step 6 would succeed). A contradiction. Thus, $W^\psi_{k_0} = L$.

From the above two cases, it follows that $\mathbf{M}_j$ does not $(\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_\psi$-identify $L \in \mathcal{L}_i$. Hence, $\mathcal{L}_i \notin (\lambda x.[h_i(x)-1])$-$\mathbf{TxtMex}_\psi$. ∎

The theorem follows from Claims 39 and 40. ∎

Theorem 41 and Theorem 42 below contrast with Theorem 29 and Theorem 35 above, respectively. Theorem 41 is about function inference and Theorem 42 is about language learning. We give a proof of Theorem 42; a similar proof can be worked out for Theorem 41.

Theorem 41 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that $(\forall x)[h_i(x) \geq x]$. Then, $(\exists \psi)$ $(\forall i)[h_i$-$\mathbf{Mex}_\psi = \lambda x.[x]$-$\mathbf{Mex}_\psi]$.*

Theorem 42 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that $(\forall x)[h_i(x) \geq x]$. Then, $(\exists \psi)$ $(\forall i)[h_i$-$\mathbf{TxtMex}_\psi = \lambda x.[x]$-$\mathbf{TxtMex}_\psi]$.*

Proof of Theorem 42: Let $\{h_i \mid i \in N\}$ be as given in the hypothesis of the theorem. Clearly, for any acceptable programming system ψ and any i, $(\lambda x.[x])$-$\mathbf{TxtMex}_\psi \subseteq h_i$-$\mathbf{TxtMex}_\psi$. We need to construct an acceptable programming system ψ, such that for all h_i, h_i-$\mathbf{TxtMex}_\psi \subseteq (\lambda x.[x])$-$\mathbf{TxtMex}_\psi$. To facilitate the description of such a ψ, we define below two recursive functions g_1 and g_2.

$g_1(0) = 0$;
$g_2(i) = \max(\{h_0(g_1(i)), h_1(g_1(i)), \ldots, h_i(g_1(i))\})$;
$g_1(i+1) = g_1(i) + g_2(i) + 1$.

We let $\psi_k = \varphi_j$, where $j = \max(\{i \mid g_1(i) \leq k\})$. Clearly, ψ is an acceptable programming system.

Consider any h_i in the hypothesis of the theorem. Let $\mathbf{M}$ h_i-$\mathbf{TxtMex}_\psi$-identify $\mathcal{L}$. We construct a machine $\mathbf{M}'$ that $(\lambda x.[x])$-$\mathbf{TxtMex}_\psi$-identifies $\mathcal{L}$.

Let $S_i = \{j \mid [\mathbf{mingrammar}_\varphi(W_j^\psi) \leq i] \ \wedge \ [j \leq h_i(\mathbf{mingrammar}_\psi(W_j^\psi)]\}$. It should be noted that the cardinality of S_i is finite (since, there are only finitely many languages L such that $\mathbf{mingrammar}_\varphi(L) \leq i$ and for each such L there are only finitely many j such that $j \leq h_i(\mathbf{mingrammar}_\psi(L))$).

For each $j \in S_i$, the minimal grammar of W_j^ψ can be stored in a finite table. Formally, let $(\forall j \in S_i)[\text{Table}(j) = \mathbf{mingrammar}_\psi(W_j^\psi)]$.

We now define machine $\mathbf{M}'$ as follows:

$\mathbf{M}'(\sigma) = k$, where

$$k = \begin{cases} \text{Table}(\mathbf{M}(\sigma)) & \text{if } \mathbf{M}(\sigma) \in S_i; \\ g_1(\max(\{i \mid g_1(i) \leq \mathbf{M}(\sigma)\})) & \text{otherwise.} \end{cases}$$

Consider any $L \in \mathcal{L}$. If $\mathbf{mingrammar}_\varphi(L) \leq i$, then by the construction of S_i, $\mathbf{M}'$ $(\lambda x.[x])$-$\mathbf{TxtMex}_\psi$-identifies L. If $\mathbf{mingrammar}_\varphi(L) > i$, then for any text T for L, $g_1(\mathbf{mingrammar}_\varphi(L)) \leq \mathbf{M}(T) < g_1(1 + \mathbf{mingrammar}_\varphi(L))$. Thus, for any text T for L, $\mathbf{M}'(T) = g_1(\mathbf{mingrammar}_\varphi(L)) = \mathbf{mingrammar}_\psi(L)$. Hence, $\mathbf{M}'$ $(\lambda x.[x])$-$\mathbf{TxtMex}_\psi$-identifies L. ∎

6 A Variant of Minimal Identification

Kinber [12], in the context of function inference, considered a variation on the theme of $\mathbf{Min}_\psi$-identification. He considered learning criteria in which, for some positive integer i, a learning machine, fed a graph of a recursive function f, is required to converge to the i^{th} minimal program for f in the acceptable programming system ψ. We study a more general notion than Kinber's, both in the context of function inference and language learning. In the next paragraph, we informally describe our criterion for language learning.

Let $h \in \mathcal{R}^+$ (i.e., h takes only non-zero values). Let L and ψ, respectively, be the language to be learned and the choice acceptable programming system. We say that a learning machine $\mathbf{M}$ *h-$\mathbf{TxtMin}_\psi$-identifies* L iff $\mathbf{M}$, fed any text for L, converges in the limit to the $h(\mathbf{mingrammar}_\psi(L))^{th}$ grammar for L in the acceptable programming system ψ. Analogous learning criteria in the context of function inference are called *h-$\mathbf{Min}_\psi$-identification*; the spe-

cial case of h-$\mathbf{Min}_\psi$-identification, where $h = \lambda x.[i]$ for some $i \in N^+$, was introduced by Kinber [12].

We investigate relationships between these criteria and underlying acceptable programming systems, both in the context of function inference and language learning.

Definition 43 below precisely states what we mean by the i^{th} program for a recursive function in an acceptable programming system ψ. The function inference criteria and its inferring power, based on the variant of minimal identification just described, are introduced in Definition 44.

Definition 43 Suppose $f \in \mathcal{R}$ and $i \in N^+$. We say that k is the i^{th} ψ-program for f (written: $k = i$-$\mathbf{minprogram}_\psi(f)$) $\iff$ $[[\psi_k = f] \ \wedge \ [\mathrm{card}(\{j \mid (j < k) \ \wedge \ (\psi_j = f)\}) = i - 1]]$.

Definition 44 Suppose $h \in \mathcal{R}^+$.
(a) $\mathbf{M}$ h-$\mathbf{Min}_\psi$-*identifies* f (written: $f \in h$-$\mathbf{Min}_\psi(\mathbf{M})$) $\iff$ $(\overset{\infty}{\forall} \ n)[\mathbf{M}(f[n]) = h(\mathbf{minprogram}_\psi(f))$-$\mathbf{minprogram}_\psi(f)]$.
(b) h-$\mathbf{Min}_\psi = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq h\text{-}\mathbf{Min}_\psi(\mathbf{M})]\}$.

Clearly, $\lambda x.[1]$-$\mathbf{Min}_\psi$-identification is the same as $\mathbf{Min}_\psi$-identification. For $i \in N^+$, $\lambda x.[i]$-$\mathbf{Min}_\psi$-identification was introduced by Kinber [12].

In Definition 45 below, we precisely state what we mean by the i^{th} grammar for an r. e. language in acceptable programming system ψ, and introduce the language learning criteria analogous to h-$\mathbf{Min}_\psi$-identification in Definition 46.

Definition 45 Let $i \in N^+$. We say that k is the i^{th} ψ-grammar for L (written: $k = i$-$\mathbf{mingrammar}_\psi(L)$) $\iff$ $[[W_k^\psi = L] \ \wedge \ [\mathrm{card}(\{j \mid (j < k) \ \wedge \ (W_j^\psi = L)\}) = i - 1]]$.

Definition 46 Suppose $h \in \mathcal{R}^+$.
(a) $\mathbf{M}$ h-$\mathbf{TxtMin}_\psi$-*identifies* L (written: $L \in h$-$\mathbf{TxtMin}_\psi(\mathbf{M})$) $\iff$ ($\forall$ texts T for L)$(\overset{\infty}{\forall} \ n)$ $[\mathbf{M}(T[n]) = h(\mathbf{mingrammar}_\psi(L))$-$\mathbf{mingrammar}_\psi(L)]$.
(b) h-$\mathbf{TxtMin}_\psi = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq h\text{-}\mathbf{TxtMin}_\psi(\mathbf{M})]\}$.

Clearly, $\lambda x.[1]$-$\mathbf{TxtMin}_\psi$-identification is the same as $\mathbf{TxtMin}_\psi$-identification.

Consider two non-decreasing recursive functions h_1 and h_2, both members of $\mathcal{R}^+$. Furthermore, let $(\forall x)[h_1(x) \geq h_2(x)]$. Then, for a given acceptable programming system ψ, we would like to compare the inferring powers of h_1-$\mathbf{Min}_\psi$-identification with h_2-$\mathbf{Min}_\psi$-identification and h_1-$\mathbf{TxtMin}_\psi$-identification with h_2-$\mathbf{TxtMin}_\psi$-identification. Theorem 47 below tells us that for any acceptable programming system ψ, h_2-$\mathbf{Min}_\psi$ is at least as big as h_1-$\mathbf{Min}_\psi$. Theorem 48 shows an analogous result for language learning. We give a proof for Theorem 48, an easier version of which proof adapted for function inference is sufficient to show Theorem 47.

Theorem 47 $(\forall \psi)(\forall$ *non-decreasing* $h_1, h_2 \in \mathcal{R}^+ \mid (\forall x)[h_1(x) \geq h_2(x)])[h_1$-$\mathbf{Min}_\psi \subseteq h_2$-$\mathbf{Min}_\psi]$.

Theorem 48 $(\forall\psi)(\forall$ *non-decreasing* $h_1, h_2 \in \mathcal{R}^+ \mid (\forall x)[h_1(x) \geq h_2(x)])[h_1\text{-}\mathbf{TxtMin}_\psi \subseteq h_2\text{-}\mathbf{TxtMin}_\psi]$.

Proof of Theorem 48: Let $\mathbf{M}$ h_1-$\mathbf{TxtMin}_\psi$-identify $\mathcal{L}$. We construct a learning machine $\mathbf{M}'$ that h_2-$\mathbf{TxtMin}_\psi$-identifies $\mathcal{L}$.

begin {Definition of $\mathbf{M}'(T[n])$}
execute all stages n' for $n' \leq n$;
output the result of the largest stage which halts in $\leq n$ steps;
output 0 if none of the stages halt in $\leq n$ steps;
begin {stage n'}
let $j = \mathbf{M}(T[n])$;
search for a set $S \subseteq \{x \mid x \leq j\}$ such that
(1) $[i \in S] \Rightarrow [[W_i^{\psi,n'} \subseteq \text{content}(T[n])] \ \wedge \ [\text{content}(T[n']) \subseteq W_i^\psi]]$;
(2) $[i = \min(S)] \Rightarrow [\text{card}(S) = h_1(i)]$;
if such an S is found, the result of stage n' is
$k \in S$ such that $\text{card}(\{r \mid r \leq k \ \wedge \ r \in S\}) = h_2(\min(S))$
end {stage n'}
end {Definition of $\mathbf{M}'(T[n])$}

Let $S_{n,n'}$ denote the set found in stage n' on input $T[n]$. We show that if $L \in \mathcal{L}$ then $\mathbf{M}'$ h_2-$\mathbf{TxtMin}_\psi$-identifies L. Let T be any text for L. Let n_1 be so large that $(\forall n \geq n_1)\mathbf{M}(T[n]) = j_0$, where $j_0 = h_1(\mathbf{mingrammar}_\psi(L))^{th}$ grammar for L. Let $n_2 \geq n_1$ be so large that $(\forall i \leq j_0)[[W_i^\psi \neq L] \Rightarrow [[W_i^{\psi,n_2} \not\subseteq L] \ \vee \ [\text{content}(T[n_2]) \not\subseteq W_i^\psi]]]$. Clearly, $(\forall n \geq n' \geq n_2)$, if stage n' (of machine $\mathbf{M}'$) on input $T[n]$ halts then $S_{n,n'} = \{i \mid (i \leq j_0) \ \wedge \ (W_i^\psi = L)\}$. Let $n_3 \geq n_2$ be so large that stage $n_2 + 1$ halts on input $T[n]$ for all $n \geq n_3$. Clearly, such an n_3 exists. Thus, for all $n \geq n_3$, machine $\mathbf{M}'$ will output $h_2(\mathbf{mingrammar}_\psi(L))^{th}$ grammar for L. ∎

Continuing our discussion about h_1 and h_2, if we further place the restriction that $(\forall x)[h_1(x) > h_2(x)]$, then we would like to know if h_2-$\mathbf{Min}_\psi$ properly contains h_1-$\mathbf{Min}_\psi$ and h_2-$\mathbf{TxtMin}_\psi$ properly contains h_1-$\mathbf{TxtMin}_\psi$. Theorem 49 below implies that there exist acceptable programming systems for which h_2-$\mathbf{Min}_\psi$ properly contains h_1-$\mathbf{Min}_\psi$ and Theorem 56 implies that there exist acceptable programming systems for which h_2-$\mathbf{TxtMin}_\psi$ properly contains h_1-$\mathbf{TxtMin}_\psi$. However, as will be clear later from Theorem 58 for function inference and Theorem 60 for language learning, this scenario is not true for every acceptable programming system.

Theorem 49 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists\psi)(\forall i)[\lambda x.[h_i(x) + 1]\text{-}\mathbf{Min}_\psi \subset h_i\text{-}\mathbf{Min}_\psi]$.

Corollary 50 [12] $(\exists\psi)(\forall i \in N^+)$ $[\lambda x.[i+1]\text{-}\mathbf{Min}_\psi \subset \lambda x.[i]\text{-}\mathbf{Min}_\psi]$.

Proof of Theorem 49: Let $f_0, f_1, \ldots$ be an infinite r. e. sequence of distinct total recursive functions whose minimal φ-programs are identifiable in the limit, i.e., $\{f_i \mid i \in N\} \in \mathbf{Min}_\varphi$; Proposition 10 allows us to make such an assumption.

Let $\mathcal{S}_i = \{f_{\langle i,j\rangle} \mid j \in N\}$. We will use $\mathcal{S}_i$ to show that $h_i\text{-}\mathbf{Min}_\psi \neq (\lambda x.[h_i(x)+1])\text{-}\mathbf{Min}_\psi$. The function $f_{\langle i,j\rangle}$ will be used to diagonalize against the learning machine $\mathbf{M}_j$. To facilitate the description of ψ, we first define two recursive functions g_1 and g_2 as follows:

$g_1(0) = 0$;

$g_2(k) = \max(\{h_0(g_1(k)), h_1(g_1(k)), ..., h_k(g_1(k))\})$;

$g_1(k+1) = g_1(k) + g_2(k) + 1$.

Clearly, g_1 is an increasing recursive function. We now give a description of acceptable programming system ψ.

begin {Definition of $\psi_k(x)$}

execute the following steps until $\psi_k(x)$ is defined:

1. **if** $k = g_1(i)$ for some i

then

let $\psi_k(x) = \varphi_i(x)$;

exit

{Note that this makes ψ an acceptable programming system.}

endif;

2. let lastfun $= \max(\{i \mid g_1(i) \leq k\})$;

if $(\forall x' \leq x)[\varphi_{\text{lastfun}}(x')\downarrow] \ \wedge \ (\forall x' < x)[\psi_k(x')\downarrow]$

then

proceed to step 3

else

diverge

{ Thus, here $\psi_k(x)$ is undefined.}

endif;

3. **if** $(\forall i \leq x)(\exists x' \leq x)[\varphi_{\text{lastfun}}(x') \neq f_i(x')]$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$;

exit

endif;

4. let $j = \min(\{i \mid (\forall x' \leq x)[f_i(x') = \varphi_{\text{lastfun}}(x')]\})$;

let j_1, j_2 be such that $j = \langle j_1, j_2\rangle$;

if $k < g_1(\text{lastfun}) + h_{j_1}(g_1(\text{lastfun}))$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$;

exit

endif;

5. **if** $[(\exists x' \geq x)[\varphi_{\text{lastfun}}(x')\downarrow \neq f_j(x')]] \vee$
$[(\exists x' \geq x)[[(\forall x'' \leq x')[\varphi_{\text{lastfun}}(x'')\downarrow]] \wedge [\mathbf{M}_{j_2}(f_j[x']) \neq k]]]$

then

let $\psi_k(x) = \varphi_{\text{lastfun}}(x)$;

exit

else

diverge

{ Thus, here $\psi_k(x)$ is undefined. }

endif

end {Definition of $\psi_k(x)$}

Claim 51 *ψ is an acceptable programming system.*

Proof: Clearly, ψ is a programming system, and g_1 reduces φ-indices to ψ-indices. Hence, ψ is an acceptable programming system. ∎

Claim 52 $(\forall j)(\forall k)[[g_1(j) \leq k < g_1(j+1)] \Rightarrow (\forall x)[\psi_k(x) = \varphi_j(x) \ \vee \ \psi_k(x)\uparrow]]$.

Proof: Whenever $\psi_k(x)$ is defined in the above procedure, $\psi_k(x) = \varphi_{\text{lastfun}}(x)$, where lastfun $= \max(\{i \mid g_1(i) \leq k\})$. The claim follows. ∎

Claim 53 $(\forall j_1, j_2)$ $[\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle}) = g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))]$. *Moreover,* $(\forall j_1, j_2)[[\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}) > j_1] \Rightarrow [(\forall k)(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})) \leq k < g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})) + h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))))[\psi_k = f_{\langle j_1,j_2\rangle}]]]$.

Proof: By Claim 52 it follows that $\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle}) \geq g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))$. By step 1 in the construction of ψ, it follows that $\psi_{g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))} = f_{\langle j_1,j_2\rangle}$. Hence, $\mathbf{minprogram}_\psi(f_{\langle j_1,j_2\rangle}) = g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))$. Assume $[\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}) > j_1]$. Hence, $g_2(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})) \geq h_{j_1}(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))$.

Let $g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})) \leq k < (g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle})) + h_{j_1}(g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))))$. Suppose by way of contradiction that $(\exists x)[\psi_k(x)\uparrow]$. (If $(\forall x)$ $\psi_k(x)\downarrow$, then by Claim 52, $\psi_k = \psi_{g_1(\mathbf{minprogram}_\varphi(f_{\langle j_1,j_2\rangle}))} = f_{\langle j_1,j_2\rangle}$.) Also, let x' be the least number such that $\psi_k(x')\uparrow$. If j found in step 4 of the construction is $\langle j_1, j_2\rangle$, then clearly, $\psi_k(x')\downarrow$. Therefore, let j found in step 4 be different from $\langle j_1, j_2\rangle$. But, then the **if** condition at step 5 will succeed, since there is an x' such that $f_{\langle j_1,j_2\rangle}(x') \neq f_j(x')$. Hence, $(\forall x')\psi_k(x')\downarrow$. This proves the claim. ∎

Claim 54 $(\forall i)[\mathcal{S}_i \in h_i\text{-}\mathbf{Min}_\psi]$.

Proof: Let machine $\mathbf{M}$ $\mathbf{Min}_\varphi$-identify $\{f_i \mid i \in N\}$. We define a machine $\mathbf{M}'_i$ as follows:

$\mathbf{M}'_i(f[n]) = g_1(\mathbf{M}(f[n])) + h_i(g_1(\mathbf{M}(f[n]))) - 1$.

Clearly, by Claim 53, for all $f_{\langle i,j\rangle}$) such that $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) > i$, we have that $\mathbf{M}'_i$ h_i-$\mathbf{Min}_\psi$-identifies $f_{\langle i,j\rangle}$. But there are only finitely many j such that $f_{\langle i,j\rangle} \in \mathcal{S}_i$ and $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) \leq i$. Thus, $\mathbf{M}'_i$ can easily be modified to h_i-$\mathbf{Min}_\psi$-identify $\mathcal{S}_i$. ∎

Claim 55 $(\forall i)[\mathcal{S}_i \notin (\lambda x.[h_i(x)+1])\text{-}\mathbf{Min}_\psi]$.

Proof: Suppose by way of contradiction, machine $\mathbf{M}_j$ $(\lambda x.[h_i(x) + 1])$-$\mathbf{Min}_\psi$-identifies $\mathcal{S}_i$. Without loss of generality, assume that $\mathbf{minprogram}_\varphi(f_{\langle i,j\rangle}) > i$ (since if $\mathbf{M}_j$ $(\lambda x.[h_i(x) + 1])$-$\mathbf{Min}_\psi$-identifies $\mathcal{S}_i$, then there are infinitely many machines that $(\lambda x.[h_i(x) + 1])$-$\mathbf{Min}_\psi$-identify $\mathcal{S}_i$). Consider the function $f = f_{\langle i,j\rangle} \in \mathcal{S}_i$. Clearly, by claim 53, $g_1(\mathbf{minprogram}_\varphi(f)), g_1(\mathbf{minprogram}_\varphi(f)) + 1, \ldots, g_1(\mathbf{minprogram}_\varphi(f)) + h_i(g_1(\mathbf{minprogram}_\varphi(f))) - 1$ are the minimal $h_i(\mathbf{minprogram}_\psi(f))$ programs for f in ψ. We show that $g_1(\mathbf{minprogram}_\varphi(f)) + h_i(g_1(\mathbf{minprogram}_\varphi(f)))$ is a program for f iff $\mathbf{M}_j$ on f does not converge to $g_1(\mathbf{minprogram}_\varphi(f)) + h_i(g_1(\mathbf{minprogram}_\varphi(f)))$. This would prove the claim.

Let $k = g_1(\mathbf{minprogram}_\varphi(f)) + h_i(g_1(\mathbf{minprogram}_\varphi(f)))$. We have the following cases:

Case 1: $\mathbf{M}_j$ on f converges to k.

Let x' be so large that

a) $x' > \langle i, j\rangle$;

b) $(\forall k' < \langle i, j\rangle)(\exists x < x')[f_{k'}(x) \neq f_{\langle i,j\rangle}(x)]$;

c) $\neg(\exists n \geq x')[\mathbf{M}(f[n]) \neq k]$.

Clearly, such an x' exists. Now, for $x = x'$, the **if** condition at steps $1, 3, 4$, and 5 in the construction of ψ are not satisfied. Thus $\psi_k(x')\uparrow$.

Case 2: $\mathbf{M}_j$ on f does not converge to k.

In this case, we claim that $\psi_k = f$. Suppose by way of contradiction $(\exists x)\psi_k(x)\uparrow$ (otherwise by Claim 52 we would have $\psi_k = f$). Let x_0 be the least such x. Since, the **if** condition at step 2 in the construction of ψ succeeds, either $\psi_k(x_0)$ is defined before step 5, or it would be defined at step 5 (since the **if** condition at step 5 would succeed), and thus, $\psi_k(x_0)\downarrow$. This is a contradiction.

From the above two cases, it follows that $\mathbf{M}_j$ does not $(\lambda x.[h_i(x)+1])$-$\mathbf{Min}_\psi$-identify $f \in \mathcal{S}_i$. Hence, $\mathcal{S}_i \notin (\lambda x.[h_i(x) + 1])$-$\mathbf{Min}_\psi$. ∎

The theorem follows from Claims 54 and 55.

A proof of Theorem 56 below can be obtained by adapting the above proof of Theorem 49 for single valued total languages. This adaptation is similar to the one in our proof of Theorem 35; we omit the details this time.

Theorem 56 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists \psi)(\forall i)[\lambda x.[h_i(x)+1]\text{-}\mathbf{TxtMin}_\psi \subset h_i\text{-}\mathbf{TxtMin}_\psi]$.

Corollary 57 $(\exists \psi)(\forall i \in N^+)\ [\lambda x.[i+1]\text{-}\mathbf{TxtMin}_\psi \subset \lambda x.[i]\text{-}\mathbf{TxtMin}_\psi]$.

Using Theorem 58 below we can show that for any two functions h_1 and $h_2 \in \mathcal{R}^+$, there exists an acceptable programming system ψ such that $h_1\text{-}\mathbf{Min}_\psi = h_2\text{-}\mathbf{Min}_\psi$. Theorem 60 is the language learning analog of Theorem 58. We give a proof of Theorem 60; a similar proof can be worked out for Theorem 58.

Theorem 58 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists \psi)(\forall i, j)\ [h_i\text{-}\mathbf{Min}_\psi = h_j\text{-}\mathbf{Min}_\psi]$.

The following special case of Theorem 58 is due to Kinber [12].

Corollary 59 [12] $(\exists \psi)(\forall i_1, i_2 \in N^+)[(\lambda x.[i_1])\text{-}\mathbf{Min}_\psi = (\lambda x.[i_2])\text{-}\mathbf{Min}_\psi]$.

Theorem 60 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists \psi)(\forall i, j)\ [h_i\text{-}\mathbf{TxtMin}_\psi = h_j\text{-}\mathbf{TxtMin}_\psi]$.

Proof of Theorem 60: This proof uses a construction similar to the one in the proof of Theorem 42. Let $\{h_i \mid i \in N\}$ be as given in the hypothesis of the theorem. We construct an acceptable programming system ψ, such that for any h_i and h_j, $h_i\text{-}\mathbf{TxtMin}_\psi = h_j\text{-}\mathbf{TxtMin}_\psi$. To facilitate the description of such a ψ, we define below recursive functions g_1 and g_2.

$g_1(0) = 0$;

$g_2(i) = \max(\{h_0(g_1(i)), h_1(g_1(i)), \ldots, h_i(g_1(i))\})$;

$g_1(i+1) = g_1(i) + g_2(i) + 1$.

We let $\psi_k = \varphi_j$, where $j = \max(\{i \mid g_1(i) \leq k\})$. Clearly, ψ is an acceptable programming system.

Consider any h_i, h_j as given in the hypothesis of the theorem. Let $\mathbf{M}^i$ $h_i\text{-}\mathbf{TxtMin}_\psi$-identify $\mathcal{L}$. We construct a machine $\mathbf{M}^j$ that $h_j\text{-}\mathbf{TxtMin}_\psi$-identifies $\mathcal{L}$.

Let $\mathcal{L}_0 = \{L \mid \mathbf{mingrammar}_\varphi(L) \leq \max(\{i, j\})\}$. Let $S_i = \{h_i(\mathbf{mingrammar}_\psi(L))\text{-}\mathbf{mingrammar}_\psi(L) \mid L \in \mathcal{L}_0\}$. Clearly, S_i is a finite set since $\mathcal{L}_0$ is a finite set. For each $k \in S_i$, let $h_j(\mathbf{mingrammar}_\psi(W_k^\psi))\text{-}\mathbf{mingrammar}_\psi(W_k^\psi)$ be stored in a finite table. Formally, let $(\forall k \in S_i)[\text{Table}(k) = h_j(\mathbf{mingrammar}_\psi(W_k^\psi))\text{-}\mathbf{mingrammar}_\psi(W_k^\psi)]$.

We now define machine $\mathbf{M}^j$ as follows:

$$\mathbf{M}^j(\sigma) = \begin{cases} \text{Table}(\mathbf{M}^i(\sigma)) & \text{if } \mathbf{M}^i(\sigma) \in S_i; \\ g_1(\max(\{i \mid g_1(i) \leq \mathbf{M}^i(\sigma)\}))+ & \\ \quad h_j(g_1(\max(\{i \mid g_1(i) \leq \mathbf{M}^i(\sigma)\}))) - 1 & \text{otherwise.} \end{cases}$$

Consider any $L \in \mathcal{L}$. We have the following two cases:

Case 1: $\mathbf{mingrammar}_{\varphi}(L) \leq \max(\{i, j\})$.

In this case, by the definition of S_i and Table, $\mathbf{M}^j$ h_j-$\mathbf{TxtMin}_{\psi}$-identifies L.

Case 2: $\mathbf{mingrammar}_{\varphi}(L) > \max(\{i, j\})$.

In this case, the following four statements are true.

(1) $g_1(\mathbf{mingrammar}_{\varphi}(L)) = \mathbf{mingrammar}_{\psi}(L)$;

(2) $g_1(\mathbf{mingrammar}_{\varphi}(L)) \leq h_i(\mathbf{mingrammar}_{\psi}(L))$-$\mathbf{mingrammar}_{\psi}(L)$
$< g_1(1 + \mathbf{mingrammar}_{\varphi}(L))$;

(3) $g_1(\mathbf{mingrammar}_{\varphi}(L)) \leq h_j(\mathbf{mingrammar}_{\psi}(L))$-$\mathbf{mingrammar}_{\psi}(L)$
$< g_1(1 + \mathbf{mingrammar}_{\varphi}(L))$;

(4) $(\forall k \mid g_1(\mathbf{mingrammar}_{\varphi}(L)) \leq k < g_1(1 + \mathbf{mingrammar}_{\varphi}(L)))[W_k^{\psi} = L]$.

Let T be any text for L. Then, $\mathbf{M}^i(T) = h_i(\mathbf{mingrammar}_{\psi}(L))$-$\mathbf{mingrammar}_{\psi}(L)$. It is clear from the above statements 1, 2, 3, 4, and the definition of machine $\mathbf{M}^j$ that $\mathbf{M}^j(T) = h_j(\mathbf{mingrammar}_{\psi}(L))$-$\mathbf{mingrammar}_{\psi}(L)$. Hence, $\mathbf{M}^j$ h_j-$\mathbf{TxtMin}_{\psi}$-identifies L. ∎

An interesting open question concerns given acceptable programming system ψ, the relationship between $\bigcup_{h \in \mathcal{R}}(h\text{-}\mathbf{Min}_{\psi})$ and $\mathbf{Mex}$ and also between $\bigcup_{h \in \mathcal{R}}(h\text{-}\mathbf{TxtMin}_{\psi})$ and $\mathbf{TxtMex}$.

7 Relaxing the Variant of Minimal Identification

We consider a relaxation of h-$\mathbf{Min}_{\psi}$-identification and h-$\mathbf{TxtMin}_{\psi}$-identification criteria introduced in the previous section. We illustrate this new identification criteria in the context of language learning.

Let $h \in \mathcal{R}^+$ (i.e., h takes only non-zero values). Let L and ψ, respectively, be the language to be learned and the choice acceptable programming system. We say that a learning machine $\mathbf{M}$ h-$\mathbf{TxtLemin}_{\psi}$-*identifies* L iff $\mathbf{M}$, fed any text for L, converges in the limit to j^{th} ψ-grammar for L, where $j \leq h(\mathbf{mingrammar}_{\psi}(L))$. An analogous criterion in the context of function inference is called h-$\mathbf{Lemin}_{\psi}$-*identification*. A special case of h-$\mathbf{Lemin}_{\psi}$-identification was briefly considered by Kinber [12]. As in the previous section, we study the relationships between these new identification criteria and acceptable programming systems.

Definition 61 below describes our new function inference criteria and its inferring power.

Definition 61 Suppose $h \in \mathcal{R}^+$.

(a) $\mathbf{M}$ h-$\mathbf{Lemin}_{\psi}$-*identifies* f (written: $f \in h\text{-}\mathbf{Lemin}_{\psi}(\mathbf{M})$) $\iff$ $(\exists j \leq h(\mathbf{minprogram}_{\psi}(f)))(\overset{\infty}{\forall}\, n)$ $[\mathbf{M}(f[n]) = j\text{-}\mathbf{minprogram}_{\psi}(f)]$.

(b) $h\text{-}\mathbf{Lemin}_{\psi} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq h\text{-}\mathbf{Lemin}_{\psi}(\mathbf{M})]\}$.

For $i \in N^+$, $\lambda x.[i]$-**Lemin**$_\psi$-identification was introduced by Kinber [12]. Definition 62 below is the language learning analog of Definition 61.

Definition 62 Suppose $h \in \mathcal{R}^+$.
(a) **M** h-**TxtLemin**$_\psi$-*identifies* L (written: $L \in h$-**TxtLemin**$_\psi(\mathbf{M})$) $\iff$ ($\forall$ texts T for L) ($\exists j \leq h(\mathbf{mingrammar}_\psi(L))$) ($\overset{\infty}{\forall}\, n$) [$\mathbf{M}(T[n]) = j$-**mingrammar**$_\psi(L)$].
(b) h-**TxtLemin**$_\psi = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq h\text{-}\mathbf{TxtLemin}_\psi(\mathbf{M})]\}$.

Recall that φ is our standard acceptable programming system. Kinber [12] showed that for all $i \in N^+$, $[\mathcal{S} \in \lambda x.[i]\text{-}\mathbf{Lemin}_\varphi] \Rightarrow [(\exists \psi)\ [\mathcal{S} \in \mathbf{Min}_\psi]]$. If we restrict ourselves to identification of *infinite* languages only, then a proof similar to Kinber's shows an analogous result for **TxtLemin**-identification. However, a modification of Kinber's proof to take care of finite languages yields a complete analog of Kinber's result for language learning. Theorem 63 below describes this result.

Theorem 63 ($\forall$ *non-decreasing* $h \in \mathcal{R}^+$) $[[\mathcal{L} \in h\text{-}\mathbf{TxtLemin}_\varphi] \Rightarrow [(\exists \psi)\ [\mathcal{L} \in \mathbf{TxtMin}_\psi]]]$.

Proposition 64 below is an easy observation about h-**Lemin**$_\psi$-identification. Proposition 65 is the language learning analog of Proposition 64.

Proposition 64 $(\forall h \in \mathcal{R} \mid (\forall x)[h(x) \geq 2])(\forall \psi)[\lambda x.[h(x)-1]\text{-}\mathbf{Lemin}_\psi \subseteq h\text{-}\mathbf{Lemin}_\psi]$.

Proposition 65 $(\forall h \in \mathcal{R} \mid (\forall x)[h(x) \geq 2])(\forall \psi)[\lambda x.[h(x)-1]\text{-}\mathbf{TxtLemin}_\psi \subseteq h\text{-}\mathbf{TxtLemin}_\psi]$.

Using Theorems 66 and 67 below we can show that there exist acceptable programming systems for which the containment in Propositions 64 and 65 are proper. Proofs of Theorem 66 and 67 can be worked out using techniques illustrated in the earlier parts of this paper.

Theorem 66 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that* $(\forall x)[h_i(x) \geq 2]$. $(\exists \psi)(\forall i)[\lambda x.[h_i(x)-1]\text{-}\mathbf{Lemin}_\psi \subset h_i\text{-}\mathbf{Lemin}_\psi]$.

Theorem 67 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct non-decreasing recursive functions such that* $(\forall x)[h_i(x) \geq 2]$. $(\exists \psi)(\forall i)[\lambda x.[h_i(x)-1]\text{-}\mathbf{TxtLemin}_\psi \subset h_i\text{-}\mathbf{TxtLemin}_\psi]$.

A proof similar to that used to prove Theorem 60 can be used to show the following two theorems.

Theorem 68 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists \psi)(\forall i, j)[h_i\text{-}\mathbf{Lemin}_\psi = h_j\text{-}\mathbf{Lemin}_\psi]$.

Theorem 69 *Let $h_0, h_1, h_2, \ldots$ be an infinite r.e. sequence of distinct recursive functions $\in \mathcal{R}^+$.* $(\exists \psi)(\forall i, j)\ [h_i\text{-}\mathbf{TxtLemin}_\psi = h_j\text{-}\mathbf{TxtLemin}_\psi]$.

Proposition 70 below shows that h-**TxtLemin**$_\psi$-identification and h-**TxtMex**$_\psi$-identification are, in some sense, similar. Proposition 71 is the language learning analog of Proposition 70. We give a proof of Proposition 71; Proposition 70 has a similar proof.

Proposition 70 $(\forall\psi)[\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Lemin}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Mex}_\psi)]$.

Proposition 71 $(\forall\psi)[\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtLemin}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtMex}_\psi)]$.

Proof of Proposition 71: Let $h \in \mathcal{R}$ be given. A simple result from recursive function theory tells us that, for any acceptable programming system ψ, there exists a $pad \in \mathcal{R}^2$, monotonically increasing in the second argument, such that $(\forall x)(\forall i)[W^\psi_{pad(x,i)} = W^\psi_x]$. Then, it is easy to see that h-**TxtLemin**$_\psi \subseteq$ **TxtMex**$(\lambda x.[pad(x, h(x))], \psi)$. Also, **TxtMex**$(h, \psi) \subseteq (\lambda x.[h(x) + 1])$-**TxtLemin**$_\psi$. This is because, if j is a grammar for L such that $j \leq h(\mathbf{mingrammar}_\psi(L))$, then $\mathrm{card}(\{i \leq j \mid W^\psi_i = L\}) \leq h(\mathbf{mingrammar}_\psi(L)) + 1$. ∎

Since, for any two acceptable programming systems ψ and ψ', $\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Mex}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Mex}_{\psi'})$ and $\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtMex}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtMex}_{\psi'})$, we have following Corollaries 72 and 73 to Propositions 70 and 71 above, respectively.

Corollary 72 $(\forall\psi, \psi')[\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Lemin}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{Lemin}_{\psi'})]$.

Corollary 73 $(\forall\psi, \psi')[\bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtLemin}_\psi) = \bigcup_{h\in\mathcal{R}}(h\text{-}\mathbf{TxtLemin}_{\psi'})]$.

8 Conclusion

On first observation, results presented above seem to say that learning criteria requiring final programs to conform to seemingly "natural" notions of succinctness are uninteresting (or, mathematically dirty), as they are dependent on the programming system. However, we would like to note that Freivalds [9] has shown, in the context of function inference, that some of these programming system dependence results still hold if attention is restricted to a very 'nice' subclass of acceptable programming systems called Kolmogorov numberings (by definition, every acceptable programming system can be reduced to a Kolmogorov numbering via a recursive function with no more rapid than linear growth). Analogs of Freivalds' results can be shown to hold for language learning also.

All this seems to suggest that complexity restrictions on final hypothesis in general models of learning will most likely result in learning criteria which are dependent on the choice of the underlying acceptable programming system. This dependence may turn out to be a fundamental fact about learning rather than a mere mathematical inconvenience. Thus, for the task of learning succinctly, it may be desirable to investigate different programming systems for different learning situations. Adapting the ideas presented in the present paper to practical programming systems is a very interesting open direction.

9 Acknowledgements

We would like to express our gratitude to John Case, Mark Fulk, and Dan Osherson for encouragement. Motivations for the present study came out of discussions with John Case. Our results build on the techniques of Freivalds, Kinber, and Chen.

Sanjay Jain was supported in part by NSF grant CCR 832-0136 at the University of Rochester. Arun Sharma was supported in part by NSF Grant CCR 871-3846 at SUNY at Buffalo and University of Delaware.

We would also like to thank Prof. S.N. Maheshwari of IIT Delhi for making the facilities of his department available to us during the preparation of this manuscript.

Finally, we thank an anonymous referee for several helpful suggestions.

References

[1] D. Angluin and C. Smith. A survey of inductive inference: Theory and methods. *Computing Surveys*, 15:237–289, 1983.

[2] M. Blum. A machine independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

[3] J. Case. Learning machines. In W. Demopoulos and A. Marras, editors, *Language Learning and Concept Acquisition*. Ablex Publishing Company, 1986.

[4] J. Case and Chi. Machine learning of nearly minimal size grammars. Unpublished Manuscript, 1986.

[5] J. Case and C. Lynes. Machine inductive inference and language identification. *Lecture Notes in Computer Science*, 140:107–115, 1982.

[6] K. Chen. *Tradeoffs in Machine Inductive Inference*. PhD thesis, SUNY at Buffalo, 1981.

[7] K. Chen. Tradeoffs in inductive inference of nearly minimal sized programs. *Information and Control*, 52:68–86, 1982.

[8] R. Freivalds. Minimal Gödel numbers and their identification in the limit. *Lecture Notes in Computer Science*, 32:219–225, 1975.

[9] R. Freivalds. Inductive inference of minimal programs. In M. Fulk and J. Case, editors, *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 3–20. Morgan Kaufmann Publishers, Inc., August 1990.

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

[11] J. Hopcroft and J. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[12] E. B. Kinber. A note on limit identification of c-minimal indices. *Electronische Informationverarbeitung und Kybernetik*, 19:459–463, 1983.

[13] E.B. Kinber. On a theory of inductive inference. *Lecture Notes in Computer Science*, 56:435–440, 1977.

[14] R. Klette and R. Wiehagen. Research in the theory of inductive inference by GDR mathematicians – A survey. *Information Sciences*, 22:149–169, 1980.

[15] D. T. Langendoen and P. M. Postal. *The Vastness of Natural Languages*. Basil Blackwell Publisher Limited, Oxford, England, 1984.

[16] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms*. North Holland, New York, 1978.

[17] Ernst E. Moody. *The Logic of William of Ockham*. New York, Sheed and Ward Inc., 1935.

[18] D. Osherson, M. Stob, and S. Weinstein. *Systems that Learn, An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, Cambridge, Mass., 1986.

[19] S. Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.

[20] H. Putnam. Reductionism and the nature of psychology. *Cognition*, 2:131–146, 1973.

[21] G. Riccardi. *The Independence of Control Structures in Abstract Programming Systems*. PhD thesis, SUNY/ Buffalo, 1980.

[22] G. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22:107–143, 1981.

[23] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

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

[25] J. Royer. *A Connotational Theory of Program Structure*. Lecture Notes in Computer Science 273. Springer Verlag, 1987.

[26] W. M. Thorburn. Occam's Razor. *Mind*, pages 287–288, 1915.

[27] W. M. Thorburn. The myth of Occam's Razor. *Mind*, pages 345–353, 1918.

[28] K. Wexler. On extensional learnability. *Cognition*, 11:89–95, 1982.

[29] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition*. MIT Press, Cambridge, Mass, 1980.