Learning with the Knowledge of an Upper Bound on Program Size

Sanjay Jain *
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware 19716

Arun Sharma †
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
E10-043, Cambridge, MA 02139

March 11, 2007

*Supported by NSF grant CCR 832-0136 at the University of Rochester.
†Supported by NSF grant CCR 871-3846 at the State University of New York at Buffalo and the University of Delaware.

Abstract

Two learning situations are considered: machine identification of programs from graphs of recursive functions (modeling inductive hypothesis formation) and machine identification of grammars from texts of recursively enumerable languages (modeling first language acquisition). Both these learning models are extended to account for situations in which a learning machine is provided additional information in the form of knowledge about an upper-bound on the minimal size program (grammar) for the function (language) being identified. For a number of such extensions, it is shown that larger classes of functions (languages) can be algorithmically identified in the presence of upper-bound information.

Numerous interesting relationships are shown between different models of learning, number of anomalies allowed in the inferred program (grammar), and number of anomalies allowed in the upper-bound information.

1 Introduction

Consider the following description of a typical learning situation involving a subject acquiring a concept. At any given time, a finite piece of data about the concept is made available to the subject. Based upon this finite information, the subject comes to have a certain belief about the concept. The belief of the subject may become fixed over time as it sees more data about the concept. The subject *learns* or *explains* the concept just in case the fixed belief, eventually held by it, is a correct explanation of the concept. *Computational learning theory* provides a framework for studying problems of this nature when the subject is a machine.

In the present paper, we formalize learning models in which the learner has some knowledge about the "size" of an explanation for the concept being learned. In particular, we investigate numerous learning models in which the learner is given, in addition to data about the concept, an "upper-bound" on the "minimal size explanation" of the concept. Our treatment is recursion theoretic.

We investigate two learning situations: *inductive hypothesis formation* modeled as machine identification of programs for computable functions (described in Section 1.1) and *first language acquisition* modeled as machine identification of grammars for recursively enumerable languages (described in Section 1.2). In Section 1.3, we informally describe some of our notions for language identification with additional information. Section 1.4 briefly mentions related work on learning with additional information.

1.1 Ex-identification: A Model of Inductive Hypothesis Formation

Picture a scientist performing all possible experiments (in arbitrary order) associated with a phenomenon, noting the result of each experiment, while simultaneously conjecturing a succession of candidate explanations for predicting the results of any experiment about the phenomenon. A criterion of success is for the scientist to eventually conjecture an explanation which he or she never gives up and which explanation correctly predicts the results of every experiment about the phenomenon. The set of all pairs of the form ⟨experiment, corresponding result⟩ associated with the phenomenon can be coded by a function from N to N, where N is the set of natural numbers.

Assuming a suitable mechanistic viewpoint, such a function associated with a phenomenon is computable. Also, a *predictive* explanation of the phenomenon can naturally be modeled as a program for the associated function. Thus, replacing the ever experimenting scientist in the above scenario by a machine yields a plausible model for, at least, a part of the practice of science—algorithmic identification in the limit of programs for computable functions from their graphs. We formalize this notion in Definition 1 just below.

Definition 1 [Gol67, BB75, CS83] A learning machine **M** **Ex**-*identifies* a function $f \Leftrightarrow \mathbf{M}$, fed graph of f, converges in the limit to a program for f.

Definition 1 above introduces a criterion for a learning machine to successfully identify a function. To facilitate comparison between different criteria of learning, we define the *inferring power* of a learning criterion which is a set theoretic summary of the capability of various learning machines to learn according to that criteria. Definition 2, just below, describes the class **Ex**, the inferring power of **Ex**-identification.

Definition 2 [Gol67, BB75, CS83] **Ex** denotes the class of all sets $\mathcal{S}$ of computable functions such that some learning machine **Ex**-identifies each function in $\mathcal{S}$.

The additional information to a function identifying machine is an upper-bound on the minimal size program for the function. This approach to modeling additional information for function identification is due to Freivalds and Wiehagen [FW79].

1.2 TxtEx-identification: A Model of Language Acquisition

Motivated by psycholinguistic studies which conclude that children are rarely, if ever, informed of grammatical errors, Gold [Gol67] introduced the seminal notion of *identification* as a model for first language acquisition. According to this paradigm, a child (modeled as a machine) receives (in arbitrary order) all the well-defined sentences, a *text*, of a language, and simultaneously, conjectures a succession of grammars. A criterion of success is for the child to eventually conjecture a correct grammar for the language being received and never to change its conjecture 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 [HU79]) 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 in the class of recursively enumerable languages. 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. Definition 3, just below, formalizes this notion.

Definition 3 [Gol67, CL82, OW82a] A learning machine **M** **TxtEx**-*identifies* a language $L \Leftrightarrow \mathbf{M}$, fed any text for L, converges in the limit to a grammar for L.

The inferring power of **TxtEx**-identification criteria, introduced above, is described in Definition 4 below.

Definition 4 [Gol67, CL82, OW82a] **TxtEx** denotes the class of all sets $\mathcal{L}$ of recursively enumerable languages such that some learning machine **TxtEx**-identifies each language in $\mathcal{L}$.

The additional information to a language identifying machine is an upper-bound on the minimal size grammar for the language being identified.

1.3 Language Identification with Additional Information

Gold's model of **TxtEx**-identification assumes that a language learner is only exposed to strings in the language. In practice, a learning machine may have more information about the language. One such additional information could be an *upper bound on the minimal size grammar* for the language. Seen in the perspective of child learning, this is, in some sense, plausible additional information, as an upper bound on the size of a child's head is also an upper bound on the minimal size grammar for any language the child has an ability to learn. Modeling this additional information yields an extension of Gold's paradigm. We introduce this extension as a new criteria of language learning. A learning machine **M** is said to **TxtBex**-*identify* a language L iff (by definition) **M**, fed an upper-bound on the minimal grammar for L in addition to a text for L, converges to a correct grammar for L. **TxtBex**, the inferring power of **TxtBex**-identification, is defined to be the class of sets $\mathcal{L}$ of recursively enumerable languages such that some learning machine **TxtBex**-identifies each language in $\mathcal{L}$. As indicated by the following Result 1, **TxtBex**-identification is strictly more powerful than **TxtEx**-identification in the sense that larger sets of languages can be learned according to **TxtBex**-identification.

$$\mathbf{TxtEx} \subset \mathbf{TxtBex} \tag{1}$$

An interesting restriction on **TxtBex**-identification is a further requirement that the learning machine infer the same grammar for any upper-bound. We express this restriction as a new language identification criterion. A learning machine **M** **TxtResBex**-*identifies* a language L iff (by definition) there exists a grammar G_L for L such that **M**, fed any upper-bound on the minimal size grammar for L in addition to a text for L, converges to G_L. **TxtResBex**, the inferring power of **TxtResBex**-identification, can be defined in the usual manner. As shown by Result 2, even the restricted additional information criterion, **TxtResBex**-identification is strictly more powerful than Gold's **TxtEx**-identification. Result 3 shows that **TxtResBex**-identification is indeed a restriction on **TxtBex**-identification.

$$\mathbf{TxtEx} \subset \mathbf{TxtResBex} \tag{2}$$

$$\mathbf{TxtResBex} \subset \mathbf{TxtBex} \tag{3}$$

Result 3 could be read as saying "if a child can learn a language in different ways, we are going to restrict its learning power by requiring it to learn in some particular way." In a metaphoric sense, Result 3 seems to "corroborate" an observation about language learning in humans. Children tend to learn a language effortlessly compared to adults in a class room setting. A possible explanation may be that the learning criteria used by children allows them to converge to any grammar for the language whereas adults are constrained to learn some unique grammar described by the teacher; thus, it may be impossible to learn large sets of languages in a class-room setting because class-room learning employs a more restricted language learning criteria!

We extend both **TxtResBex**-identification and **TxtBex**-identification in the following ways:

- We allow the inferred grammars to have anomalies (in the style of Case and Lynes [CL82] and Osherson and Weinstein [OW82a]). After all, it is all right not to learn a language perfectly. We show that larger sets of languages can be learned if we allow anomalies in the language generated by the inferred grammar.
- Instead of giving additional information about the minimal grammar of the language being learned, we give an upper-bound on the minimal grammar for a finite variant of the language. After all, information from our teachers could always be somewhat inaccurate. We show that there is a decrease in the inferring power if we allow anomalous additional information.

We investigate similar extensions to other models of language learning: **TxtFex**-identification [Cas88] and **TxtBc**-identification [CL82]. Analogous results in the context of function identification are also presented. It should be noted that analogs of Results 1, 2, and 3 for function identification were first observed by Freivalds and Wiehagen [FW79].

1.4 Related Research

In the present work, we are concerned with extending **TxtEx**-identification and **Ex**-identification by providing additional information to the learning machine. We briefly note other attempts to extending these fundamental learning paradigms. L. Blum and M. Blum [BB75] and Case and Smith [CS83], in the context of function inference, consider the case where the program inferred by the learning machine is allowed to make a finite number of mistakes. For language learning, Case and Lynes [CL82] and Osherson and Weinstein [OW82a] consider learning criteria in which the grammar inferred is allowed to be a grammar for a finite variant of the language being learned. Smith [Smi82] considers the function inference criteria in which the learning machine is replaced by a "team" of learning machines and successful learning takes place if any one member of the team succeeds in learning the function. Osherson, Stob, and Weinstein [OSW86a] consider a generalized notion of team learning. Pitt [Pit84] has shown that the power of probabilistic machines can be neatly characterized in terms of teams [Smi82] of deterministic machines. Jain and Sharma [JS90a] consider team inference in the context of language learning. Royer [Roy86] and Smith and Velauthapillai [SV86] consider the case where the inferred program may have infinitely many anomalies, but the "density" of these anomalies is bounded. Case [Cas88], based on Osherson and Weinstein [OW82a], considers language learning criteria in which the learning agent is allowed to converge in the limit to a finite set of grammars instead of one. Case, Jain and Sharma [CJS89] consider grammar size restrictions in Case's vacillating language learning criteria [Cas88]. Fulk [Ful85, Ful90a] and Jain and Sharma [JS90b] consider other forms of additional information to learning machines.

2 Preliminaries

2.1 Notation

Recursion-theoretic concepts not explained below are treated in [Rog67]. 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 let $(\forall n \in N)[n < * < \infty]$. Generally, e and lower case letters near the beginning, middle, and end of the alphabet, with or without decorations, $a, b, c, \ldots, i, j, k, l, m, n, \ldots, x, y, z$, range over N; it will be explicitly specified if any of these variables locally range over N^+, $N \cup \{*\}$, or $N^+ \cup \{*\}$.

$[m, n]$ denotes the set $\{x \mid m \leq x \leq n\}$. We let P, S, with or without decorations, range over subsets of N and we let D, with or without decorations, range over finite subsets of N. $\|P\|$ denotes the cardinality of P. So then, '$\|P\| \leq *$' means that $\|P\|$ is finite. $\min(P)$ and $\max(P)$ respectively denote the minimum and maximum element in P. We take $\min(\emptyset)$ to be ∞ and $\max(\emptyset)$ to be 0.

Let $\lambda x, y.\langle x, y\rangle$ denote a fixed pairing function (a recursive, bijective mapping: $N \times N \rightarrow N$) [Rog67]. $\lambda x, y.\langle x, y\rangle$ and its inverses are useful to simulate the effect of having multiple argument functions. π_1 and π_2 are corresponding projection functions, i.e., $(\forall x, y)[\pi_1(\langle x, y\rangle) = x \wedge \pi_2(\langle x, y\rangle) = y]$.

L, with or without decorations, ranges over *recursively enumerable* (r.e.) subsets of N, which subsets are usually construed as codings of formal languages. $\mathcal{E}$ denotes the class of all recursively enumerable languages $\subseteq N$. We let $\mathcal{L}$, with or without decorations, range over subsets of $\mathcal{E}$. $L_1 \triangle L_2$ 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 $\|L_1 \triangle L_2\| \leq a$.

η and ξ range over partial functions. For $a \in (N \cup \{*\})$, $\eta_1 =^a \eta_2$ means that $\|\{x \mid \eta_1(x) \neq \eta_2(x)\}\| \leq a$. $\text{domain}(\eta)$ and $\text{range}(\eta)$ respectively denote the domain and range of partial function η.

$S \subseteq N$ 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* just in case the function it represents is total.

$\mathcal{R}$ denotes the class of all *recursive* functions, i.e., total computable functions with arguments and values from N. f, g, h, and p, with or without decorations, range over $\mathcal{R}$. $\mathcal{S}$ ranges over subsets of $\mathcal{R}$.

We fix φ to be an *acceptable programming system* [Rog58, Rog67, MY78] for the partial recursive functions: $N \rightarrow N$. φ_i denotes the partial recursive function computed by φ-program i. W_i denotes $\text{domain}(\varphi_i)$. W_i is, then, the r.e. set/language ($\subseteq N$) accepted (or equivalently, generated) by the φ-program i. We let Φ be an arbitrary Blum complexity measure [Blu67] associated with acceptable programming system φ; such measures exist for any acceptable programming system [Blu67]. Then, $W_{i,s}$ denotes the set $\{x \mid x < s \wedge \Phi_i(x) \leq s\}$. For a given total computable function f and an r.e. language L, we define $\text{minprog}(f)$ to denote $\min(\{i \mid \varphi_i = f\})$ and $\text{mingram}(L)$ to denote $\min(\{i \mid W_i = L\})$.

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 undefined otherwise. For any set A, 2^A denotes the power set of A. The quantifiers '$\forall^\infty$' and '$\exists^\infty$' mean 'for all but finitely many' and 'there exist infinitely many,' respectively. The quantifier '$\exists!$' means 'there exists a unique.'

We concern ourselves with formally investigating learning 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.

2.2 Learning Machines

In Definition 5 below, we formally introduce what we mean by a machine that learns a function, and in Definition 7, 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 segment. SEG denotes the set of all finite initial segments. Note that $f[n] \cup \{(n, x)\}$ is a new finite initial segment of length $n + 1$ formed by extending $f[n]$ suitably.

Definition 5 [Gol67] A *function learning machine* is an algorithmic device which computes a mapping from SEG into N.

We now consider language learning machines. Definition 6 below introduces a notion that facilitates discussion about elements of a language being fed to a learning machine.

Definition 6 A *sequence* σ is a mapping from an initial segment of N into $(N \cup \{\#\})$. The *content* of a sequence σ, denoted content(σ), is the set of natural numbers in the range of σ. The *length* of σ, denoted by $|\sigma|$, is the number of elements in σ.

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

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

Definition 7 A *language-learning machine* is an algorithmic device which computes a mapping from SEQ into N.

The set of all finite initial segments, SEG, can be coded onto N. Also, the set of all finite sequences of natural numbers and #'s, SEQ, can be coded onto N. Thus, in both Definitions 5 and 7, we can view these machines as taking natural numbers as input and emitting natural numbers as output. Henceforth, we will refer to both function-learning machines and language-learning machines as just learning machines. We let $\mathbf{M}$, with or without decorations, range over learning machines.

2.3 Fundamental Function Identification Paradigms

In Definition 8 below we spell out what it means for a learning machine on a function to converge in the limit.

Definition 8 Suppose $\mathbf{M}$ is a learning machine and f is a computable function. $\mathbf{M}(f)\downarrow$ (read: $\mathbf{M}(f)$ *converges*) $\iff$ $(\exists i)(\forall^\infty n)$ $[\mathbf{M}(f[n]) = i]$. If $\mathbf{M}(f)\downarrow$, then $\mathbf{M}(f)$ is defined = the unique i such that $(\forall^\infty n)[\mathbf{M}(f[n]) = i]$, otherwise we say that $\mathbf{M}(f)$ diverges (written: $\mathbf{M}(f)\uparrow$).

We now introduce two different criteria for a learning machine to successfully infer a function.

2.3.1 Explanatory Function Identification

Definition 9 [Gol67, BB75, CS83] Let $a \in N \cup \{*\}$.
(i) **M** $\mathbf{Ex}^a$-*identifies* f (written: $f \in \mathbf{Ex}^a(\mathbf{M})$) $\iff$ $(\exists i \mid \varphi_i =^a f)[\mathbf{M}(f)\downarrow = i]$.
(ii) $\mathbf{Ex}^a = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Ex}^a(\mathbf{M})]\}$.

Case and Smith [CS83] motivate anomalies (or, mistakes) in the final programs in Definition 9 from the fact that physicists sometimes do employ explanations with anomalies. The $a = *$ case was introduced by L. Blum and M. Blum [BB75] and the other $a > 0$ cases were first considered by Case and Smith [CS83].

2.3.2 Behaviorally Correct Function Identification

Case and Smith [CS83] introduced another infinite hierarchy of identification criteria which we describe below. "**Bc**" stands for *behaviorally correct*. Barzdin [Bar74] independently introduced a similar notion.

Definition 10 [CS83] Let $a \in N \cup \{*\}$.
(i) **M** $\mathbf{Bc}^a$-*identifies* f (written: $f \in \mathbf{Bc}^a(\mathbf{M})$) $\iff$ $(\forall^\infty n)[\varphi_{\mathbf{M}(f[n])} =^a f]$.
(ii) $\mathbf{Bc}^a = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bc}^a(\mathbf{M})]\}$.

We usually write **Ex** for $\mathbf{Ex}^0$ and **Bc** for $\mathbf{Bc}^0$. Theorem 1 below describes some of the basic results about the two kinds of function identification criteria described above.

Theorem 1 *For all $a \in N$,*
(a) $\mathbf{Ex}^a \subset \mathbf{Ex}^{a+1}$.
(b) $\bigcup_{a \in N} \mathbf{Ex}^a \subset \mathbf{Ex}^*$.
(c) $\mathbf{Ex}^* \subset \mathbf{Bc}$.
(d) $\mathbf{Bc}^a \subset \mathbf{Bc}^{a+1}$.
(e) $\bigcup_{a \in N} \mathbf{Bc}^a \subset \mathbf{Bc}^*$.
(f) $\mathcal{R} \in \mathbf{Bc}^*$.

Parts (a), (b), (d), and (e) are due to Case and Smith [CS83]. Part (f) is due to Harrington [CS83]. Blum and Blum [BB75] first showed that $\mathbf{Ex} \subset \mathbf{Ex}^*$. Barzdin [Bar74] independently showed $\mathbf{Ex} \subset \mathbf{Bc}$.

2.4 Fundamental Language Identification Paradigms

Definition 11 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 initial sequence of T with length n. Hence, domain($T[n]$) $= \{x \mid x < n\}$.

2.4.1 Explanatory Language Identification

In Definition 12 below we spell out what it means for a learning machine on a text to converge in the limit.

Definition 12 Suppose $\mathbf{M}$ is a learning machine and T is a text. $\mathbf{M}(T)\downarrow$ (read: $\mathbf{M}(T)$ *converges*) $\iff$ $(\exists i)(\forall^{\infty} n)$ $[\mathbf{M}(T[n]) = i]$. If $\mathbf{M}(T)\downarrow$, then $\mathbf{M}(T)$ is defined = the unique i such that $(\forall^{\infty} n)[\mathbf{M}(T[n]) = i]$, otherwise we say that $\mathbf{M}(T)$ diverges (written: $\mathbf{M}(T)\uparrow$).

We now introduce criteria for a learning machine to be successful on a language. $\mathbf{TxtEx}^0$ from Definition 13 just below is the same as **TxtEx**.

Definition 13 [Gol67, CL82, OW82a] Let $a \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtEx}^a$-*identifies* L (written: $L \in \mathbf{TxtEx}^a(\mathbf{M})$) $\iff$ ($\forall$ texts T for $L)(\exists i \mid W_i =^a L)[\mathbf{M}(T)\downarrow = i]$.
(ii) $\mathbf{TxtEx}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtEx}^a(\mathbf{M})]\}$.

The generalization of Gold's paradigm to the $a > 0$ case above was motivated by the observation that humans rarely learn a language perfectly. The $a > 0$ case in Definition 13 is due to Case and Lynes [CL82]. Osherson and Weinstein [OW82a], independently of Case and Lynes, introduced the $a = *$ case. The influence of Gold's paradigm to human language learning is discussed by Pinker [Pin79], Wexler and Culicover [WC80], Wexler [Wex82], and Osherson, Stob, and Weinstein [OSW82, OSW84, OSW86b].

Definition 14 [Ful85, Ful90b] σ is a **TxtEx**-*stabilizing sequence for* $\mathbf{M}$ *on* L $\iff$ content$(\sigma) \subseteq L$ and $(\forall \sigma' \mid \text{content}(\sigma') \subseteq L \ \wedge \ \sigma \subseteq \sigma')[\mathbf{M}(\sigma') = \mathbf{M}(\sigma)]$.

Definition 15 [BB75, OW82b] Let $a \in N \cup \{*\}$. σ is a $\mathbf{TxtEx}^a$-*locking sequence for* $\mathbf{M}$ *on* L $\iff$ σ is a **TxtEx**-stabilizing sequence for $\mathbf{M}$ on L and $W_{\mathbf{M}(\sigma)} =^a L$.

We now present a very important lemma in learning theory due to L. Blum and M. Blum [BB75].

Lemma 1 [BB75, OW82b] *If* $\mathbf{M}$ $\mathbf{TxtEx}^a$*-identifies* L*, then there is a* $\mathbf{TxtEx}^a$*-locking sequence for* $\mathbf{M}$ *on* L*.*

2.4.2 Vacillatory Language Identification

Case [Cas88], as a refinement of a result by Osherson and Weinstein [OW82a], considered the question whether humans converge to more than one distinct, but equivalent, correct grammars. He captured this notion through a new criterion of language learning, viz., **TxtFex**-*identification*—a more general criteria than Gold's **TxtEx**-identification. We also study the effect of additional information on this criteria.

Before we describe **TxtFex**-identification, we first consider in Definition 16 just below what it means for a learning machine to converge on a text to a finite set of grammars.

Definition 16 [Cas88] Suppose $\mathbf{M}$ is a learning machine and T is a text. Then, $\mathbf{M}(T)$ *finitely-converges* (written: $\mathbf{M}(T)\Downarrow$) $\iff$ $\{\mathbf{M}(\sigma) \mid \sigma \subset T\}$ is finite. If $\mathbf{M}(T)\Downarrow$, then we say that $\mathbf{M}(T)\Downarrow = D \iff D = \{i \mid (\exists^{\infty}\sigma \subset T)[\mathbf{M}(\sigma) = i]\}$, otherwise we say that $\mathbf{M}(T)$ *finitely-diverges* (written: $\mathbf{M}(T)\Uparrow$).

Definition 17 [Cas88] Let $a \in N \cup \{*\}$ and $b \in N^{+} \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtFex}_b^a$*-identifies* L (written: $L \in \mathbf{TxtFex}_b^a(\mathbf{M})$) $\iff$ ($\forall$ texts T for L)($\exists D \mid \|D\| \leq b \wedge (\forall i \in D)[W_i =^a L]$)$[\mathbf{M}(T)\Downarrow = D]$.
(ii) $\mathbf{TxtFex}_b^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtFex}_b^a(\mathbf{M})]\}$.

In $\mathbf{TxtFex}_b^a$-identification, the b is a "bound" on the number of final grammars and the a is a "bound" on the number of anomalies allowed in these final grammars. A "bound" of $*$ just means unbounded, but finite. We sometimes refer to $\mathbf{TxtFex}_b^0$ as $\mathbf{TxtFex}_b$. $\mathbf{TxtFex}_*$-identification was first studied by Osherson and Weinstein [OW82a].

2.4.3 Behaviorally Correct Language Identification

Definition 18 [CL82] Let $a \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtBc}^a$-identifies L (written: $L \in \mathbf{TxtBc}^a(\mathbf{M})$) $\iff$ ($\forall$ texts T for L)($\forall^{\infty} n$)$[W_{\mathbf{M}(T[n])} =^a L]$.
(ii) $\mathbf{TxtBc}^a = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtBc}^a(\mathbf{M})]\}$.

We usually write $\mathbf{TxtBc}$ for $\mathbf{TxtBc}^0$.

The following definition is an analog of Definition 15 for $\mathbf{TxtBc}$-identification.

Definition 19 (Based on [BB75, CL82]) Let $a \in N \cup \{*\}$. σ is a $\mathbf{TxtBc}^a$*-locking sequence for* $\mathbf{M}$ *on* $L \iff$ content$(\sigma) \subseteq L$ and $(\forall \sigma' \mid [\sigma \subseteq \sigma'] \wedge [\text{content}(\sigma') \subseteq L])[W_{\mathbf{M}(\sigma')} =^a L]$.

There is an analog of Lemma 1 for $\mathbf{TxtBc}$-identification [CL82].

Lemma 2 (Based on [BB75, CL82]) *If* $\mathbf{M}$ $\mathbf{TxtBc}^a$*-identifies* L*, then there is a* $\mathbf{TxtBc}^a$*-locking sequence for* $\mathbf{M}$ *on* L.

Theorem 2 below states some of the basic results about the three kinds of language identification criteria just described.

Theorem 2 *For all* $a, b \in N$,
(a) $\mathbf{TxtEx}^{a+1} - \mathbf{TxtFex}_*^a \neq \emptyset$.
(b) $\mathbf{TxtEx}^{2a+1} - \mathbf{TxtBc}^a \neq \emptyset$.
(c) $\mathbf{TxtEx}^{2a} \subset \mathbf{TxtBc}^a$.
(d) $\mathbf{TxtFex}_{b+1}^0 - \mathbf{TxtFex}_b^* \neq \emptyset$.
(e) $\bigcup_{a \in N}\ \mathbf{TxtFex}_b^a \subset \mathbf{TxtFex}_b^*$.
(f) $\bigcup_{a \in N}\ \mathbf{TxtBc}^a \subset \mathbf{TxtBc}^*$.

Parts (a), (d) and (e) are due to Case [Cas88]. Parts (b) and (c) are due to Case and Lynes [CL82]. Part (f) follows from part (e) in Theorem 1. Osherson and Weinstein [OW82a] have independently shown that $\mathbf{TxtEx} \subset \mathbf{TxtFex}_*$.

3 Function Identification with Additional Information

3.1 Definitions

In the approach we take, a machine identifying a program for a recursive function from its graph, is provided with an upper-bound on the minimal program for the function as additional information. Hence, to simplify our exposition, learning machines with additional information can be viewed as taking two arguments: upper-bound on the minimal program and initial segment of graph of a function. In other words, learning machines identifying functions with additional information are algorithmic devices that compute a mapping from $N \times \text{SEG}$ into N. For $c \in N \cup \{*\}$, $\text{minprog}^c(f)$ denotes the minimal program in φ-programming system that computes f with at most c errors, i.e., $\text{minprog}^c(f) = \mu i . \varphi_i =^c f$.

In Definition 20 just below, we describe what it means for a learning machine to converge on additional information and graph of a function.

Definition 20 $\mathbf{M}(x, f)\downarrow$ (read: $\mathbf{M}$ on f with additional information x converges) $\iff (\exists i)(\forall^\infty n)$ $[\mathbf{M}(x, f[n]) = i]$. If $\mathbf{M}(x, f)\downarrow$, then $\mathbf{M}(x, f)$ is defined = the unique i such that $(\forall^\infty n)[\mathbf{M}(x, f[n]) = i]$; otherwise $\mathbf{M}(x, f)$ is said to be undefined.

Definition 21 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{Bex}^{a,c}$-*identifies* f (written: $f \in \mathbf{Bex}^{a,c}(\mathbf{M})$) $\iff$ $(\forall x \geq \text{minprog}^c(f))$ $(\exists i \mid \varphi_i =^a f)$ $[\mathbf{M}(x, f)\downarrow = i]$.
(ii) $\mathbf{Bex}^{a,c} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bex}^{a,c}(\mathbf{M})]\}$.

Intuitively, $\mathbf{M}$ $\mathbf{Bex}^{a,c}$-identifies f iff $\mathbf{M}$, fed x, at least as large as the minimal program that computes f with at most c errors, and graph of f, converges to a program that computes f with at most a errors. The notion $\mathbf{Bex}^{0,0}$-identification was first studied by Freivalds and Wiehagen [FW79].
Remark: We may modify the above definition to allow the additional information to converge to a bound instead of giving the bound itself as an additional information. However, this does not change the class of functions identifiable in the limit since the machine can assume the last additional information received as the correct additional information. These remarks hold for all the additional information learning criteria introduced in the present paper.

Proposition 1 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{Bex}^{a_1,c} \subseteq \mathbf{Bex}^{a_2,c}$.
(ii) $\mathbf{Bex}^{a,c_1} \supseteq \mathbf{Bex}^{a,c_2}$.

If, in Definition 21, we further require that the final program conjectured be the same for any upper-bound, we get a new identification criterion described in Definition 22 below.

Definition 22 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{ResBex}^{a,c}$-*identifies* f (written: $f \in \mathbf{ResBex}^{a,c}(\mathbf{M})$) $\iff$ $(\exists i \mid \varphi_i =^a f)$ $(\forall x \geq$ $\text{minprog}^c(f))$ $[\mathbf{M}(x, f)\downarrow = i]$.
(ii) $\mathbf{ResBex}^{a,c} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{ResBex}^{a,c}(\mathbf{M})]\}$.

$\mathbf{ResBex}^{0,0}$-identification criterion was first introduced by Freivalds and Wiehagen.

Proposition 2 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{ResBex}^{a_1,c} \subseteq \mathbf{ResBex}^{a_2,c}$.
(ii) $\mathbf{ResBex}^{a,c_1} \supseteq \mathbf{ResBex}^{a,c_2}$.

Proposition 3 *Let $a, c \in N \cup \{*\}$. Then,*
$\mathbf{ResBex}^{a,c} \subseteq \mathbf{Bex}^{a,c}$.

A plausible way to apply additional information for **Bc**-identification is introduced in Definition 23 below.

Definition 23 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{Bbc}^{a,c}$-*identifies* f (written: $f \in \mathbf{Bbc}^{a,c}(\mathbf{M})$) $\iff$ $(\forall x \geq \text{minprog}^c(f))\,(\forall^\infty n)[\varphi_{\mathbf{M}(x,f[n])} =^a f]$.
(ii) $\mathbf{Bbc}^{a,c} = \{\mathcal{S} \mid (\exists \mathbf{M})[\mathcal{S} \subseteq \mathbf{Bbc}^{a,c}(\mathbf{M})]\}$.

Proposition 4 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{Bbc}^{a_1,c} \subseteq \mathbf{Bbc}^{a_2,c}$.
(ii) $\mathbf{Bbc}^{a,c_1} \supseteq \mathbf{Bbc}^{a,c_2}$.

3.2 Results

In this section we compare the relative inferring powers of various learning criteria defined in the previous section. In the discussion to follow, the information about the upper-bound on program size given to a learning machine in **ResBex**-identification will be referred to as *restricted additional information*. The term *general additional information* will be used to refer to the upper bound information about the program size in **Bex**, **Bbc**.

Theorem 3 $(\forall a \in N)[\mathbf{Ex}^{a+1} - \mathbf{ResBex}^{a,0} \neq \emptyset]$.

Theorem 3 says that there are classes of recursive functions that can be algorithmically learned in the **Ex** sense by allowing up to $a+1$ anomalies in the final inferred program but *cannot* be learned by allowing up to a anomalies in the final program even if the learning machine is given the best possible restricted additional information. In other words, extra anomalies allowed in the inferred program *cannot*, in general, be compensated by any restricted additional information.

Lemma 3 (Operator Recursion Theorem, Case [Cas74]). *Suppose Θ is an effective operator* [Rog67]. *Then one can effectively find a recursive one-to-one function p such that $(\forall i, x)[\varphi_{p(i)}(x) = \Theta(p)(\langle i, x\rangle)]$.*

We use Lemma 3 extensively to prove some of the theorems in this paper. Intuitively, Lemma 3 allows us to construct a repetition free r.e. sequence of programs $p(0), p(1), \ldots$ such that each program $p(i)$ on input x can construct copies of any subcollection (effectively computed from i and x) of the entire collection of programs $p(0), p(1), \ldots, p(i), p(i+1), \ldots$; $p(i)$ can then use this subcollection together with i and x as data in any further preassigned effective computation.

Proof of Theorem 3: Consider the class of functions $\mathcal{S} = \{f \mid \varphi_{f(0)} =^{a+1} f\}$. Clearly, $\mathcal{S} \in \mathbf{Ex}^{a+1}$. Suppose learning machine $\mathbf{M}$ is given. It remains to exhibit a function $f \in (\mathcal{S} - \mathbf{ResBex}^{a,0}(\mathbf{M}))$. By implicit use of the operator recursion theorem (Lemma 3), we obtain a repetition free r.e. sequence of programs $p(0), p(1), p(2), \ldots$ such that either $\varphi_{p(0)}$ or for some $s > 0$, $\varphi_{p(s)}$ computes such an f. We proceed to give an informal effective construction of the $\varphi_{p(i)}$'s in successive stages $s > 0$. Let n_s denote the least x such that $\varphi_{p(0)}(x)$ has not been defined before stage s. Let $\varphi_{p(0)}(0) = p(0)$. Go to Stage 1.

Stage s

let $\varphi_{p(0)}(n_s) = 1$;
for each $x \leq n_s$, let $\varphi_{p(s)}(x) = \varphi_{p(0)}(x)$;
let $j = \mathbf{M}(p(0), \varphi_{p(0)}[n_s + 1])$;
if $\mathbf{M}(p(s), \varphi_{p(s)}[n_s + 1]) = j$, then go to Step 1, else go to Step 2;

Step 1: let n'_s be the least x such that $\varphi_{p(0)}(x)$ has not been defined till now;
for $0 \leq l \leq a$, let $\varphi_{p(s)}(n'_s + l) = 0$;
let $y = n'_s + a + 1$;
repeat following steps

let $\varphi_{p(s)}(y) = \varphi_{p(0)}(y) = 0$;
let $y = y + 1$
until either Condition $1A$ or Condition $1B$ is true:

Condition $1A$: $\mathbf{M}(p(s), \varphi_{p(s)}[y]) \neq j$;
Condition $1B$: $(\exists l)[0 \leq l \leq a$ and $\Phi_j(n'_s + l) \leq y]$;

Step $1a$: if Condition $1A$ holds then go to Step 2;
Step $1b$: if Condition $1B$ holds then

let l be the convergence point discovered in Condition $1B$.
let $\varphi_{p(0)}(n'_s + l) = \varphi_j(n'_s + l) + 1$;
for each $0 \leq r \leq a$ and $r \neq l$, let $\varphi_{p(0)}(n'_s + r) = 0$;
go to Stage $s + 1$.

Step 2: let n''_s be the least x such that $\varphi_{p(s)}(x)$ has not been defined till now;
for all x such that $x < n''_s$ and $\varphi_{p(0)}(x)$ has not been defined, let $\varphi_{p(0)}(x) = \varphi_{p(s)}(x)$;
repeat following steps
let $\varphi_{p(0)}(n''_s) = \varphi_{p(s)}(n''_s) = 0$;
let $n''_s = n''_s + 1$

until either Condition $2A$ or Condition $2B$ is true:

Condition $2A$: $\mathbf{M}(p(0), \varphi_{p(0)}[n''_s]) \neq j$;
Condition $2B$: $\mathbf{M}(p(s), \varphi_{p(s)}[n''_s]) = j$;

Step 2*a*: if Condition 2A holds then go to Stage $s+1$;
Step 2*b*: if Condition 2B holds then go to Step 1
End of *Stage* s.

Now consider the following cases:
Case 1: There are infinitely many stages. In this case, let additional information be $p(0)$ and $f = \varphi_{p(0)}$. Clearly, $\varphi_{p(0)}$ is total and is in $\mathcal{S}$.
Case 1a: $\mathbf{M}(p(0), f)$ diverges. In this case, $\mathbf{M}$ does not $\mathbf{ResBex}^{a,0}$-identify $f \in \mathcal{S}$.
Case 1b: $\mathbf{M}(p(0), f)$ converges to j at Stage s. In this case, the only way in which infinitely many stages can exist is by the execution of Step 1b infinitely-often. But, then $\varphi_{\mathbf{M}(p(0),f)}$ is infinitely different from f.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt.
Case 2a: Steps 1 and 2 are both executed infinitely-often. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in \mathcal{S}$. Now, $\mathbf{M}$ on f with additional information $p(s)$ does not converge (the only way in which Steps 1 and 2 can be executed infinitely-often without leaving Stage s is by repeated execution of Step 1a and Step 2b, and thus $\mathbf{M}$ infinitely-often outputs j without converging to j).
Case 2b: In Stage s, the algorithm never leaves Step 1. In this case, let $f = \varphi_{p(s)} \in \mathcal{S}$ (since $\varphi_{p(0)} =^{a+1} \varphi_{p(s)}$). Now, $\mathbf{M}$ on f with additional information $p(s)$ converges to j (otherwise Condition 1A would succeed) and φ_j does not converge on at least $a+1$ values (otherwise Condition 1B would succeed). Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{a,0}$-identify f.
Case 2c: In Stage s, the algorithm never leaves Step 2. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in \mathcal{S}$. Now, $\mathbf{M}$ on f with additional information $p(s)$ does not converge to j (otherwise Condition 2B would succeed) and $\mathbf{M}$ on f with additional information $p(0)$ converges to j (otherwise Condition 2A would succeed). Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{a,0}$-identify f.

From the above cases, we have that $\mathbf{M}$ does not $\mathbf{ResBex}^{a,0}$-identify $\mathcal{S}$. □

Theorem 3 yields an anomaly hierarchy for **ResBex**-identification criteria. If the number of anomalies allowed in the additional information is fixed, then allowing extra anomalies in the program inferred by the learning machine results in increased inferring power with respect to **ResBex**-identification criteria.

Corollary 1 $(\forall c \in N \cup \{*\})$
$\mathbf{ResBex}^{0,c} \subset \mathbf{ResBex}^{1,c} \subset \mathbf{ResBex}^{2,c} \cdots \subset \mathbf{ResBex}^{i,c} \subset \mathbf{ResBex}^{i+1,c} \subset \cdots \subset \mathbf{ResBex}^{*,c}$.

We now turn our attention to the effects of general additional information to a function learning machine. However, Proposition 5 below limits us to a weaker form of Theorem 3 for **Bex**-identification.

Proposition 5 $(\forall n \in N)[\mathcal{R} \in \mathbf{Bex}^{n,n}]$.

Proof of Proposition 5: We need to show the existence of a learning machine that $\mathbf{Bex}^{n,n}$-identifies any $f \in \mathcal{R}$. We describe the behavior of such a machine on x, the given additional information, and finite initial segment $f[s]$. Let $S = \{j \mid j \leq x \wedge \|\{y \mid y \leq s \wedge \Phi_j(y) \leq s \wedge \varphi_j(y) \neq f(y)\}\| \leq n\}$. Intuitively, S is a set of programs which appear correct for the given additional information x and the finite initial segment $f[s]$. For each $j \in S$, let errors$(j) = \{y \mid y \leq s \wedge \Phi_j(y) \leq s \wedge \varphi_j(y) \neq f(y)\}$. For each $j \in S$, let patch(j) be a program obtained from program j

by patching the anomalies in errors(j). Thus, if $j \leq x$ is a program which computes f with atmost n errors, then, for large enough s, program patch(j) will not make any convergent errors. **M** then outputs a program i, which on input y outputs the value output by the first converging program in the set $\{\text{patch}(j) \mid j \in S\}$. It is easy to verify that **M** $\mathbf{Bex}^{n,n}$-identifies f (since, the only errors program i makes are divergent errors whose total count is bounded by n). □

Proposition 5 says that if the number of anomalies tolerated in the additional information and the number of anomalies tolerated in the final program are finite and equal, then there exists a single learning machine that learns every recursive function with respect to the **Bex**-identification criteria. However, if we allow an unbounded finite number of anomalies both in the additional information and the final program inferred, there is a reduction in inferring power of **Bex**-identification criteria as shown by Proposition 6 below.

Proposition 6 $\mathcal{R} \notin \mathbf{Bex}^{*,*}$.

Proof of Proposition 6 : Consider the following classes of functions:

$\mathcal{S}_1 = \{f \mid \varphi_{f(0)} = f\}$;

$\mathcal{S}_2 = \{f \mid \|\{x \mid f(x) \neq 0\}\| \text{ is finite }\}$.

Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. We show that $\mathcal{S} \notin \mathbf{Bex}^{*,*}$. Suppose by way of contradiction, there exists a learning machine **M** such that $\mathcal{S} \subseteq \mathbf{Bex}^{*,*}$. Let i_0 be a program for $\lambda x.0$, the everywhere 0 function. We describe a machine $\mathbf{M}'$ which works as follows. $\mathbf{M}'$, upon being fed graph of f, feeds $\max(f(0), i_0)$ along with the graph of f to machine **M**, and emits the output of **M**. Clearly, $\max(f(0), i_0)$ is an upper-bound on the minimal program for a finite variant of any function $f \in \mathcal{S}$. Hence, $\mathbf{M}'$ $\mathbf{Ex}^*$-identifies any $f \in \mathcal{S}$. A contradiction, since it has been shown by Case and Smith [CS83] that $\mathcal{S} \notin \mathbf{Ex}^*$. Therefore, $\mathcal{S} \notin \mathbf{Bex}^{*,*}$. □

Because of Proposition 5, we can only show the following weak form of Theorem 3 for **Bex**-identification.

Theorem 4 $(\forall a \in N)[\mathbf{Ex}^{a+1} - \mathbf{Bex}^{a,a+1} \neq \emptyset]$.

Proof of Theorem 4: Consider the class of recursive functions $\mathcal{S} = \{f \mid \varphi_{f(0)} =^{a+1} f\}$. Clearly, $\mathcal{S} \in \mathbf{Ex}^{a+1}$. Suppose by way of contradiction, there exists a learning machine **M** such that $\mathcal{S} \subseteq \mathbf{Bex}^{a,a+1}(\mathbf{M})$. Then, consider a machine $\mathbf{M}'$ which works as follows. $\mathbf{M}'$, upon being fed graph of $f \in \mathcal{R}$, feeds $f(0)$ along with the graph of f to machine **M**, and emits the program output by **M**. Since, $f(0)$ is an upper-bound on the minimal program computing f with at most $a+1$ errors, $\mathbf{M}'$ converges in the limit to a program that computes f with at most a errors. This implies that $\mathbf{M}'$ $\mathbf{Ex}^a$-identifies $\mathcal{S}$. A contradiction, since it has been shown by Case and Smith [CS83] that $\mathcal{S} \notin \mathbf{Ex}^a$. □

Theorem 4 and Propositions 5 and 6 give us Corollaries 2 and 3 below. According to Corollary 2, if we fix the number of anomalies allowed in the additional information, then allowing extra anomalies in the inferred program results in increased inferring power with respect to **Bex**-identification criteria. However, when the number of anomalies allowed in the inferred program equals the number of anomalies allowed in the additional information, we don't gain any extra inferring power by allowing more anomalies in the inferred programs. This is because, when the number of anomalies allowed in the additional information and the inferred program are equal, we can identify every recursive function with respect to the **Bex**-identification criteria.

Corollary 2 $(\forall c \in N)$
$\mathbf{Bex}^{0,c} \subset \mathbf{Bex}^{1,c} \subset \cdots \subset \mathbf{Bex}^{c,c} = \mathbf{Bex}^{c+1,c} = \cdots = 2^{\mathcal{R}}$.

However, allowing a finite but unbounded number of anomalies in the additional information decreases the learning power with respect to **Bex**-identification criteria, as we cannot identify all the recursive functions even if we allow an unbounded finite number of anomalies in the inferred program.

Corollary 3 $\mathbf{Bex}^{0,*} \subset \mathbf{Bex}^{1,*} \subset \mathbf{Bex}^{2,*} \subset \cdots \subset \mathbf{Bex}^{n,*} \subset \mathbf{Bex}^{n+1,*} \subset \cdots \subset \mathbf{Bex}^{*,*} \subset 2^{\mathcal{R}}$.

Now we consider the effects of extra additional information on learning capability of machines. The following theorems show that extra additional information can indeed help in certain cases. They also give most of the hierarchy results except the ones noted.

Theorem 5 $(\forall c \in N)[\mathbf{ResBex}^{0,c} - \mathbf{ResBex}^{*,c+1} \neq \emptyset]$.

Proof of Theorem 5: Consider the following classes of functions:
$\mathcal{S}_1 = \{f \mid \varphi_{f(0)} = f \ \wedge \ \|\{x \mid f(x) \neq 0\}\| \text{ is infinity }\}$;
$\mathcal{S}_2 = \{f \mid \|\{x \mid f(x) \neq 0\}\| \leq \mu k.[\varphi_k =^c (f)]\}$.
Let $\mathcal{S} = \mathcal{S}_1 \bigcup \mathcal{S}_2$. We claim that $\mathcal{S} \in \mathbf{ResBex}^{0,c}$. To see this, consider the behavior of learning machine $\mathbf{M}_0$ on additional information x and finite initial segment $f[n]$ described below.

Let F be a function with arguments from the set of all finite sets and values from N. For any finite set D, $F(D)$ is defined to be an index of a program for the function $\lambda y.[\min(\{z \mid \langle y, z\rangle \in D\})$ if $(\exists z)[\langle y, z\rangle \in D]$; 0 otherwise].

Definition of $\mathbf{M}_0(x, f[n])$
if $\|\{y \mid f(y) \neq 0 \wedge y < n\}\| > x$
then output $f(0)$
else output $F(\{\langle y, z\rangle \mid y < n \wedge z \neq 0 \wedge f(y) = z\})$

End of Definition of $\mathbf{M}_0(x, f[n])$

Consider any $f \in \mathcal{S}_1$. Clearly, the "if condition" in the definition of $\mathbf{M}_0$ will succeed for any upper-bound x and sufficiently large n. Thus, $\mathbf{M}_0$ in the limit will output $f(0)$ which by definition of $\mathcal{S}_1$ is a program for f. For any $f \in \mathcal{S}_2$ and for any $x \geq \mu k.[\varphi_k =^c f]$, the "if condition" of $\mathbf{M}_0$ will always fail. Thus, for sufficiently large n, $\mathbf{M}_0(x, f[n])$ will stabilize to a program for f. Hence, $\mathcal{S} \in \mathbf{ResBex}^{0,c}$.

Now, we show that $\mathcal{S} \notin \mathbf{ResBex}^{*,c+1}$. Suppose a learning machine $\mathbf{M}$ is given. It remains to exhibit a function $f \in (\mathcal{S} - \mathbf{ResBex}^{*,i+1}(\mathbf{M}))$. By implicit use of the operator recursion theorem (Lemma 3), we obtain a repetition free r.e. sequence of programs $p(0), p(1), p(2), \ldots$ such that either $\varphi_{p(0)}$ or for some $s > 0$, a function derived from $\varphi_{p(s)}$ computes such an f. We proceed to give an informal effective construction of the $\varphi_{p(i)}$'s in successive stages $s > 0$. Let n_s denote the least x such that $\varphi_{p(0)}(x)$ has not been defined before Stage s. Let $\varphi_{p(0)}(0) = p(0)$. Go to stage 1.

Stage s
let $\varphi_{p(0)}(n_s) = 1$;
for each $x \leq n_s$, let $\varphi_{p(s)}(x) = \varphi_{p(0)}(x)$;
let $j = \mathbf{M}(p(0), \varphi_{p(0)}[n_s + 1])$;
if $\mathbf{M}(p(s), \varphi_{p(s)}[n_s + 1]) = j$, then go to Step 1, else go to Step 2;

Step 1: let n'_s be the least x such that $\varphi_{p(0)}(x)$ has not been defined till now;
for each $k \in N$, let f_k denote the following recursive function:

$$f_k(x) = \begin{cases} \varphi_{p(s)}(x) & \text{if } x < n'_s; \\ k & \text{if } n'_s \leq x \leq n'_s + c; \\ 0 & \text{otherwise.} \end{cases}$$

let $y = n'_s + c + 1$;
repeat following steps

let $\varphi_{p(s)}(y) = 0$; let $y = y + 1$
until either Condition $1A$ or Condition $1B$ is true:

Condition $1A$: $(\exists k < y)[\mathbf{M}(p(s), f_k[y]) \neq j]$;
Condition $1B$: $(\exists x)[(n'_s \leq x)$ and $\Phi_j(x) \leq y]$;

Step $1a$: if Condition $1A$ holds then

for each x such that $n'_s \leq x \leq n'_s + c$, let $\varphi_{p(s)}(x) = k$, where k is as found in Condition $1A$;
go to Step 2;

Step $1b$: if Condition $1B$ holds then
let x be the convergence point discovered in Condition $1B$;
let $\varphi_{p(0)}(x) = \varphi_j(x) + 1$;
for all z such that $n'_s \leq z < x$, let $\varphi_{p(0)}(z) = 0$; go to Stage $s + 1$;

Step 2: let n''_s be the least x such that $\varphi_{p(s)}(x)$ has not been defined till now;
for all x such that $x < n''_s$ and $\varphi_{p(0)}(x)$ has not been defined, let $\varphi_{p(0)}(x) = \varphi_{p(s)}(x)$;
repeat following steps
let $\varphi_{p(0)}(n''_s) = \varphi_{p(s)}(n''_s) = 1$;
let $n''_s = n''_s + 1$

until either Condition $2A$ or Condition $2B$ is true:

Condition $2A$: $\mathbf{M}(p(0), \varphi_{p(0)}[n''_s]) \neq j$;
Condition $2B$: $\mathbf{M}(p(s), \varphi_{p(s)}[n''_s]) = j$;

Step $2a$: if Condition $2A$ holds then go to Stage $s + 1$;
Step $2b$: if Condition $2B$ holds then go to Step 1
End of *Stage* s.

Now consider the following cases:
Case 1: There are infinitely many stages. In this case, let additional information be $p(0)$ and $f = \varphi_{p(0)}$. Clearly, $\varphi_{p(0)}$ is total and is in $\mathcal{S}$.
Case 1a: $\mathbf{M}(p(0), f)$ diverges. In this case, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,c+1}$-identify $f \in \mathcal{S}$.

Case 1b: $\mathbf{M}(p(0), f)$ converges to j at Stage s.

In this case, the only way in which infinitely many stages can exist is by the execution of Step 1b infinitely-often. But, then $\varphi_{\mathbf{M}(p(0),f)}$ is infinitely different from f.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt.
Case 2a: Steps 1 and 2 are both executed infinitely often. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in S$. Now, $\mathbf{M}$ on f with additional information $p(s)$ does not converge (the only way in which Steps 1 and 2 can be executed infinitely-often without leaving Stage s is by repeated execution of Step 1a and Step 2b, and thus $\mathbf{M}$ infinitely often outputs j without converging to j).
Case 2b: In stage s, the algorithm never leaves Step 1.

In this case, for all $m \in N$, let f_m be as follows:

$$f_m(y) = \begin{cases} \varphi_{p(s)}(y) & \text{if } y \in \text{domain}(\varphi_{p(s)}); \\ m & \text{otherwise.} \end{cases}$$

Now, $\varphi_{p(s)}(y)$ is defined for all y except $n'_s \leq x \leq n'_s + c$. Also, $\varphi_{p(s)}$ is a $c+1$ variant of each f_m, and for all but finitely many m, $f_m \in \mathcal{S}_2$. But, for all m, $\varphi_{\mathbf{M}(p(s),f_m)}$ does not converge on infinitely many points. Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,c+1}$-identify $\mathcal{S}$.
Case 2c: In Stage s, the algorithm never leaves Step 2. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in \mathcal{S}$. Now, $\mathbf{M}$ on f with additional information $p(s)$ does not converge to j (otherwise Condition 2b would succeed) and $\mathbf{M}$ on f with additional information $p(0)$ converges to j (otherwise Condition 2a would succeed). Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,c+1}$-identify f.

From the above cases, we have that $\mathbf{M}$ does not $\mathbf{ResBex}^{*,c+1}$-identify $\mathcal{S}$. □

The following proposition follows from the techniques of [CS83].

Proposition 7 $(\forall c \in N \cup \{*\})[\mathbf{Bex}^{*,c} \subseteq \mathbf{Bbc}^{0,c}]$.

Proof of Proposition 7: Let $\mathbf{M}$ $\mathbf{Bex}^{*,c}$-identify $\mathcal{S}$. It can easily be verified that learning machine $\mathbf{M}'$, whose behavior is described below, $\mathbf{Bbc}^{0,c}$-identifies $\mathcal{S}$. $\mathbf{M}'$, on additional information x and finite initial segment $f[s]$, computes $j = \mathbf{M}(x, f[s])$ and outputs a program $\text{patch}(j, f[s])$, where $\varphi_{\text{patch}(j,f[s])}(y) = f(y)$ if $y < s$, otherwise $\varphi_{\text{patch}(j,f[s])}(y) = \varphi_j(y)$. □

Theorem 6 $(\forall a \in N)[\mathbf{ResBex}^{0,*} - \mathbf{Bc}^a \neq \emptyset]$.

Proof of Theorem 6: Consider the following class of functions:

$$\begin{aligned} \mathcal{S} = \{f \in \mathcal{R} \mid & \\ & [\varphi_{f(0)} = f \wedge \|\{y \mid f(y+1) \neq f(y)\}\| \text{ is } \infty] \vee \\ & [\|\{y \mid f(y+1) \neq f(y)\}\| \leq \mu k.[\varphi_k =^* f]]\} \end{aligned}$$

It is easy to verify that $\mathcal{S} \in \mathbf{ResBex}^{0,*}$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{Bc}^a$-identifies $\mathcal{S}$. Then, by the implicit use of Kleene's recursion theorem [Rog67], we construct a program e informally described below.

We construct φ_e in stages. Let n_s denote the least x such that $\varphi_e(x)$ has not been defined prior to the execution of Stage s. Let $\varphi_e(0) = e$. Go to stage 0. Clearly, $n_0 = 1$.

Stage s

Step 1: search for k, n, and $a+1$ numbers $y_1, y_2, \ldots, y_{a+1}$ such that

$k > \varphi_e(n_s - 1) \wedge$
$n_s \leq n < y_1 < y_2 < \cdots < y_{a+1} \wedge$
$(\forall l \mid 1 \leq l \leq a+1)[\varphi_{\mathbf{M}(\varphi_e[n_s] \cup \{(n_s,k)\} \cup \{(n_s+1,k)\} \cup \cdots \cup \{(n,k)\})}(y_l)\downarrow]$;

Step 2: if k, n, and $a+1$ numbers $y_1, y_2, \ldots, y_{a+1}$ are found in Step 1, then

for each l, such that $1 \leq l \leq a+1$, let $\varphi_e(y_l) = \varphi_{\mathbf{M}(\varphi_e[n_s] \cup \{(n_s,k)\} \cup \{(n_s+1,k)\} \cup \cdots \cup \{(n,k)\})}(y_l) + 1$;
for each y such that $n_s \leq y \leq y_{a+1} \wedge y \notin \{y_1, y_2, \ldots, y_{a+1}\}$, let $\varphi_e(y) = k$;
go to *Stage* $s+1$

End *Stage* s

Now, consider the following cases:
Case 1: All stages halt. In this case, let $f = \varphi_e$. Clearly, $f \in \mathcal{S}$. But, by construction of program e, for infinitely many n, $\varphi_{\mathbf{M}(f[n])} \neq^a f$. Thus, $\mathbf{M}$ does not $\mathbf{Bc}^a$-identify $f \in \mathcal{S}$.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt. In this case, for all $m \in N$, let f_m be as follows:

$$f_m(y) = \begin{cases} \varphi_e(y) & \text{if } x < n_s; \\ m & \text{otherwise.} \end{cases}$$

Now, for all but finitely many m, $f_m \in \mathcal{S}$. However, for all but finitely many m, for all $n > n_s$, $\varphi_{\mathbf{M}(f_m[n])}$ is defined on only finitely many inputs. Thus, $\mathbf{M}$ does not $\mathbf{Bc}^a$-identify f_m for some $f_m \in \mathcal{S}$.

From the above two cases, we have that $\mathbf{M}$ does not $\mathbf{Bc}^a$-identify $\mathcal{S}$. □

The following corollary to Theorem 5 and Theorem 6, says that even the worst quality additional information (allowing an unbounded finite number of anomalies) increases the inferring power of **Ex**-identification criteria. Also, if we fix the number of anomalies allowed in the inferred program, we get a reduction in inferring power by allowing more anomalies in additional information.

Corollary 4 *For all* $a \in N \cup \{*\}$, $\mathbf{ResBex}^{a,0} \supset \mathbf{ResBex}^{a,1} \supset \cdots \supset \mathbf{ResBex}^{a,i} \supset \mathbf{ResBex}^{a,i+1} \supset \cdots \supset \mathbf{ResBex}^{a,*} \supset \mathbf{Ex}^a$.

Theorem 7 $(\forall c \in N)[\mathbf{ResBex}^{0,c} - \mathbf{Bex}^{c,c+1} \neq \emptyset]$.

The following Corollary 5 to Theorem 7 above says that with respect to **Bex**- identification criteria, we can identify all the recursive functions if the number of anomalies allowed in the additional information and the inferred program are equal. However, if we increase the number of anomalies allowed in the additional information, we get a reduction in learning power, the greatest reduction being for the case where we allow an unbounded finite number of anomalies in the inferred program.

Corollary 5 *For all* $c \in N$, $2^{\mathcal{R}} = \mathbf{Bex}^{c,c} \supset \mathbf{Bex}^{c,c+1} \supset \cdots \supset \mathbf{Bex}^{c,c+j} \supset \mathbf{Bex}^{c,c+j+1} \supset \cdots \supset \mathbf{Bex}^{c,*}$.

Proof of Theorem 7: Consider the following classes of functions:

$\mathcal{S}_1 = \{f \mid \varphi_{f(0)} = f \wedge \|\{y \mid f(y) \neq 0\}\| \text{ is } \infty\}$;

$\mathcal{S}_2 = \{f \mid \|\{y \mid f(y) \neq 0\}\| \leq \mu k.\varphi_k =^c (f)\}$.

Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$.

It is easy to verify that $\mathcal{S} \in \mathbf{ResBex}^{0,c}$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{Bex}^{c,c+1}$-identifies $\mathcal{S}$. Then, by the implicit use of Kleene's recursion theorem [Rog67], we construct a program e informally described below.

We define φ_e in stages. Let n_s denote the least x such that $\varphi_e(x)$ has not been defined prior to the execution of Stage s. Let $\varphi_e(0) = e$. Go to stage 0. Clearly, $n_0 = 1$.

Stage s

let $\varphi_e(n_s) = 1$;

for each $m \in N$, let f_m denote the following recursive function:

$$f_m(y) = \begin{cases} \varphi_e(y) & y \leq n_s; \\ m & n_s < y \leq n_s + c + 1; \\ 0 & \text{otherwise.} \end{cases}$$

let $y = n_s + c + 2$;

repeat following steps

let $\varphi_e(y) = 0$;

let $y = y + 1$

until either Condition A or Condition B is true:

Condition A: $(\exists k \leq y)[\mathbf{M}(e, f_k[y]) \neq \mathbf{M}(e, \varphi_e[n_s])]$;

Condition B: $(\exists l \mid n_s + 1 \leq l \leq n_s + c + 1)[\Phi_{\mathbf{M}(e,\varphi_e[n_s])}(l) \leq y]$;

if Condition A holds then

for $n_s < z \leq n_s + c + 1$, let $\varphi_e(z) = k$, where k is as found in Condition A;

go to Stage $s + 1$;

if Condition B holds then

let l be as found in Condition B;

let $\varphi_e(l) = \varphi_{\mathbf{M}(e,\varphi_e[n_s])}(l) + 1$;

for $n_s < z \leq n_s + c + 1 \wedge z \neq l$, let $\varphi_e(z) = 0$;

go to Stage $s + 1$;

End of *Stage* s

Now consider the following cases:

Case 1: There are infinitely many stages. In this case, let $f = \varphi_e$. Clearly, $f \in \mathcal{S}_1$.
Case 1a: $\mathbf{M}$ on f with additional information e diverges. In this case, $\mathbf{M}$ does not $\mathbf{Bex}^{c,c+1}$-identify f.
Case 1b: $\mathbf{M}$ on f with additional information e converges to program i at Stage s. In this case, by construction of program e, φ_i is infinitely different from f. Therefore, $\mathbf{M}$ does not $\mathbf{Bex}^{c,c+1}$-identify f.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt. In this case, for all m, e is an upper-bound on $c+1$ variant of f_m. Also, for all but finitely many m, $f_m \in \mathcal{S}_2$. But, for all m, $\mathbf{M}$ on f_m and additional information e converges to a program that does not halt on at least $c+1$ inputs. Hence, $\mathbf{M}$ does not $\mathbf{Bex}^{c,c+1}$-identify $\mathcal{S}_2$.

From the above cases, it is clear that $\mathbf{M}$ does not $\mathbf{Bex}^{c,c+1}$-identify $\mathcal{S}$. □.

Theorem 8 $(\forall c \in N)[\mathbf{ResBex}^{0,c} - \mathbf{Bex}^{*,*} \neq \emptyset]$.

Proof of Theorem 8: Consider the following classes of functions:
$\mathcal{S}_1 = \{f \mid \varphi_{f(0)} = f \ \wedge \ \|\{y \mid f(y) \neq 0\}\| \text{ is } \infty\}$;
$\mathcal{S}_2 = \{f \mid \|\{y \mid f(y) \neq 0\}\| \leq \mu k.[\varphi_k =^c (f)]\}$.
Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$.

It is easy to verify that $\mathcal{S} \in \mathbf{ResBex}^{0,c}$. However, since $\max(f(0), i_0)$, where i_0 is a program for $\lambda y.0$, the everywhere 0 function, is a valid additional information for $\mathbf{Bex}^{*,*}$-identification for any $f \in \mathcal{S}$, $\mathcal{S} \in \mathbf{Bex}^{*,*} \Leftrightarrow \mathcal{S} \in \mathbf{Ex}^*$. But, $\mathcal{S} \notin \mathbf{Ex}^*$, as evidenced by Theorem 5. Thus, $\mathcal{S} \notin \mathbf{Bex}^{*,*}$. □

Theorem 9 $\mathbf{Bc} - \mathbf{Bex}^{*,*} \neq \emptyset$.

Proof of Theorem 9: Consider the following class of functions:

$\mathcal{S} = \{f \mid$ 1, 2, and 3 are satisfied
1. $\varphi_{f(0)} = f \vee \|\{y \mid f(y) \neq 0\}\|$ is finite
2. $(\forall^\infty y)[f(y) \neq 0 \Rightarrow \varphi_{f(y)} = f]$
3. $\|\{y \mid f(y) \neq 0\}\|$ is finite $\Rightarrow \varphi_{f(\max(\{y \mid f(y) \neq 0\})} = f$ $\}$

It is easy to verify that $\mathcal{S} \in \mathbf{Bc}$. Clearly, $\max(f(0), i_0)$, where i_0 is a program for $\lambda y.0$, the everywhere 0 function, is an upper-bound on the minimal program for a finite variant for any $f \in \mathcal{S}$. Thus, $\mathcal{S} \in \mathbf{Bex}^{*,*} \Leftrightarrow \mathcal{S} \in \mathbf{Ex}^*$. We prove below that $\mathcal{S} \notin \mathbf{Ex}^*$. The rest of the proof given below is similar to that used by Case and Smith [CS83] to prove that $\mathbf{Bc} - \mathbf{Ex}^* \neq \emptyset$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{Ex}^*$-identifies $\mathcal{S}$. We need to exhibit a function $f \in (\mathcal{S} - \mathbf{Ex}^*(\mathbf{M}))$. By implicit use of the operator recursion theorem (Lemma 3), we obtain a repetition free r.e. sequence of programs $p(0), p(1), p(2), \ldots$ such that for all i, $p(i) > 0$ and either $\varphi_{p(0)}$ or for some $s > 0$, $\varphi_{p(s)}$ computes such an f. We proceed to give an informal effective construction of the $\varphi_{p(i)}$'s in successive stages $s > 0$. Let n_s denote the least x such that $\varphi_{p(0)}(x)$ has not been defined before Stage s. Let $\varphi_{p(0)}(0) = p(0)$. Go to Stage 1.

Stage s

Step 1: for $y < n_s$, let $\varphi_{p(s)}(y) = \varphi_{p(0)}(y)$;
let $\varphi_{p(s)}(n_s) = p(s)$;
let $u = n_s + 1$;
execute Steps 1a, 1b, and 1c in parallel until the search in either Step 1b or Step 1c is successful:

Step 1a:
repeat
let $\varphi_{p(s)}(u) = 0$; let $u = u + 1$;
forever

Step 1b: search for $n \geq n_s$ such that $\mathbf{M}(\varphi_{p(0)}[n_s]) \neq \mathbf{M}(\varphi_{p(s)}[n_s + 1] \cup \{(n_s + 1, 0)\} \cup \cdots \cup \{(n, 0)\})$;

Step 1c: search for $l \geq n_s$ such that $\varphi_{\mathbf{M}(\varphi_{p(0)}[n_s])}(l)\downarrow$;

Step 2: if search in Step 1b succeeds then

let u be the least x such that $\varphi_{p(s)}(x)$ has not been defined till now.
for $n_s \leq y \leq \max(u - 1, n)$, let $\varphi_{p(0)}(y) = \varphi_{p(s)}(y)$;
let $\varphi_{p(s)}$ be same as $\varphi_{p(0)}$ from now on;
go to Stage $s + 1$;

Step 3: if search in Step 1c succeeds then

for $n_s \leq y \leq l$, let $\varphi_{p(0)}(y) = 0$ if $\varphi_{\mathbf{M}(\varphi_{p(0)}[n_s])}(l) = p(0)$; otherwise let $\varphi_{p(0)}(y) = p(0)$;
go to Stage $s + 1$

End of *Stage* s

Now consider the following cases:
Case 1: There are infinitely many stages. In this case, let $f = \varphi_{p(0)} \in \mathcal{S}$.
Case 1a: $\mathbf{M}$ on f diverges. Then, $\mathbf{M}$ does not $\mathbf{Ex}^*$-identify f.
Case 1b: $\mathbf{M}$ on f converges to program i at Stage s. In this case, the only way in which infinitely many stages can occur is if the search in Step 1c succeeds infinitely-often. But, then by the above construction, φ_i is infinitely different from f.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt. In this case, let $f = \varphi_{p(s)} \in \mathcal{S}$. Also, $\mathbf{M}$ on f converges to $\mathbf{M}(\varphi_{p(s)}[n_s])$. But, $\varphi_{\mathbf{M}(\varphi_{p(s)}[n_s])}$ converges on only finitely many inputs. Hence, $\mathbf{M}$ does not $\mathbf{Ex}^*$-identify f.
From the above cases, it follows that $\mathbf{M}$ does not $\mathbf{Ex}^*$-identify $\mathcal{S}$. □.

Theorem 10 $(\forall a \in N)[\mathbf{Bc}^{a+1} - \mathbf{Bbc}^{a,*} \neq \emptyset]$.

Proof of Theorem 10: Consider the following class of functions:

$\mathcal{S} = \{f \mid$ 1, 2, and 3 are satisfied

1. $\varphi_{f(0)} = f \vee \|\{y \mid f(y) \neq 0\}\|$ is finite
2. $(\forall^{\infty} y)[f(y) \neq 0 \Rightarrow \varphi_{f(y)} =^{a+1} f]$
3. $\|\{y \mid f(y) \neq 0\}\|$ is finite $\Rightarrow \varphi_{f(\max(\{y|f(y)\neq 0\})} =^{a+1} f$ $\}$

It is easy to verify that $\mathcal{S} \in \mathbf{Bc}^{a+1}$. Clearly, $\max(f(0), i_0)$, where i_0 is a program for $\lambda y.0$, the everywhere 0 function, is an upper-bound on the minimal program for a finite variant for any $f \in \mathcal{S}$. Thus, $\mathcal{S} \in \mathbf{Bbc}^{a,*} \Leftrightarrow \mathcal{S} \in \mathbf{Bc}^{a}$. The rest of the proof is similar to that used by Case and Smith [CS83] to prove that $\mathbf{Bc}^{a+1} - \mathbf{Bc}^{a} \neq \emptyset$. We omit the details.

Theorem 11 $(\forall a, c \in N)[\mathbf{ResBex}^{0,c} - \mathbf{Bbc}^{a,*} \neq \emptyset]$.

Proof of Theorem 11: Consider the following classes of functions:
$\mathcal{S}_1 = \{f \mid \varphi_{f(0)} = f \ \wedge \ \|\{y \mid f(y) \neq 0\}\|$ is $\infty\}$;
$\mathcal{S}_2 = \{f \mid \|\{y \mid f(y) \neq 0\}\| \leq \mu k.\varphi_k =^{c} (f)]\}$.
Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$.

It is easy to verify that $\mathcal{S} \in \mathbf{ResBex}^{0,c}$. Clearly, $\max(f(0), i_0)$, where i_0 is a program for $\lambda y.0$, the everywhere 0 function, is a bound on the minimal program for a finite variant for any $f \in \mathcal{S}$. Thus, $\mathcal{S} \in \mathbf{Bbc}^{a,*} \Leftrightarrow \mathcal{S} \in \mathbf{Bc}^{a}$. We prove below that $\mathcal{S} \notin \mathbf{Bc}^{a}$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{Bc}^{a}$-identifies $\mathcal{S}$. Then, by the use of Kleene's recursion theorem [Rog67], there exists a program e described informally below in stages. Let n_s denote the least x such that $\varphi_e(x)$ has not been defined before Stage s. Let $\varphi_e(0) = e$. Clearly, $n_0 = 1$. Go to stage 0.

Stage s

Step 1: search for k, n, and $a+1$ numbers $x_1, x_2, x_3, \ldots, x_{a+1}$ such that
$n_s + c < n < x_1 < x_2 < \ldots < x_{a+1}$ and
$(\forall l \mid 1 \leq l \leq a+1)[\varphi_{\mathbf{M}(\varphi_e[n_s] \cup \{(n_s,k)\} \cup \cdots \cup \{(n_s+c,k)\} \cup \{(n_s+c+1,0)\} \cup \cdots \cup \{(n,0)\})}(x_l)\downarrow]$;

Step 2: if the search for $k, n, x_1, x_2, \ldots, x_{a+1}$ is successful in Step 1 then

for each l such that $1 \leq l \leq a+1$, let
$\varphi_e(x_l) = \varphi_{\mathbf{M}(\varphi_e[n_s] \cup \{(n_s,k)\} \cup \cdots \cup \{(n_s+c,k)\} \cup \{(n_s+c+1,0)\} \cup \cdots \cup \{(n,0)\})}(x_l) + 1$;
for $n_s \leq x \leq n_s + c$, let $\varphi_e(x) = k$;
for $n_s + c < x < x_{a+1}$ and $x \notin \{x_1, x_2, \ldots, x_{a+1}\}$, let $\varphi_e(x) = 0$;
go to Stage $s+1$

End of *Stage s*

Now consider the following cases:
Case 1: All stages halt. In this case, let $f = \varphi_e$. Clearly, $f \in \mathcal{S}$. But, by the above construction, for infinitely many n, $\varphi_{\mathbf{M}(f[n])} \neq^{a} f$.

Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt. For all k, define f_k as follows:

$$f_k(y) = \begin{cases} \varphi_e(y) & \text{if } y < n_s; \\ k & \text{if } n_s \leq x \leq n_s + c; \\ 0 & \text{otherwise.} \end{cases}$$

Now, for all but finitely many k, $f_k \in \mathcal{S}$. But, for all k and for all $n > n_s + c$, $\varphi_{\mathbf{M}(f_k[n])}$ converges on only finitely many inputs. Thus, for some k, $\mathbf{M}$ does not $\mathbf{Bc}^a$-identify $f_k \in \mathcal{S}$.

From the above two cases, we have that $\mathbf{M}$ does not $\mathbf{Bc}^a$-identify $\mathcal{S}$. □

Theorem 12 $\mathbf{Bc}^1 - \mathbf{ResBex}^{*,0} \neq \emptyset$.

With respect to the **ResBex**-identification criteria, we can achieve increased learning power by either increasing the number of anomalies in the inferred program, or by decreasing the number of anomalies in the additional information. This means that allowing an unbounded finite number of anomalies in the inferred program and allowing no anomalies in additional information (best possible additional information) would result in maximum inferring power for **ResBex**-identification. Unfortunately, as the following corollary shows, even this most powerful **ResBex**-identification criteria is not good enough to identify the entire class of recursive functions.

Corollary 6 $\mathcal{R} \notin \mathbf{ResBex}^{*,0}$.

Proof of Theorem 12: Consider the following class of functions:

$\mathcal{S} = \{f \mid (\forall^\infty n)[\varphi_{f(n)} =^1 f]\}$.

Clearly, $\mathcal{S} \in \mathbf{Bc}^1$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{ResBex}^{*,0}$-identifies $\mathcal{S}$. We need to exhibit a function $f \in (\mathcal{S} - \mathbf{ResBex}^{*,0}(\mathbf{M}))$. By implicit use of the operator recursion theorem (Lemma 3), we obtain a repetition free r.e. sequence of programs $p(0), p(1), p(2), \ldots$ such that either $\varphi_{p(0)}$ or for some $s > 0$, $\varphi_{p(s)}$ computes such an f. We proceed to give an informal effective construction of the $\varphi_{p(i)}$'s in successive stages $s > 0$. Let n_s denote the least x such that $\varphi_{p(0)}(x)$ has not been defined before stage s. Let $n_1 = 0$. Go to Stage 1.

Stage s

let $\varphi_{p(0)}(n_s) = p(0)$;
for each $x \leq n_s$, let $\varphi_{p(s)}(x) = \varphi_{p(0)}(x)$;
let $j = \mathbf{M}(p(0), \varphi_{p(0)}[n_s + 1])$;
if $\mathbf{M}(p(s), \varphi_{p(s)}[n_s + 1]) = j$, then go to Step 1, else go to Step 2;

Step 1: let n'_s be the least x such that $\varphi_{p(0)}(x)$ has not been defined till now;
let $\varphi_{p(s)}(n'_s) = p(s)$;
let $y = n'_s + 1$;
repeat following steps

let $\varphi_{p(s)}(y) = p(s)$;
let $y = y + 1$;

until either Condition $1A$ or Condition $1B$ is true:

Condition $1A$: $\mathbf{M}(p(s), \varphi_{p(s)}[y]) \neq j$;
Condition $1B$: for some $x \geq n'_s$, $\Phi_j(x) \leq y$;

Step $1a$: if Condition $1A$ holds then go to Step 2;
Step $1b$: if Condition $1B$ holds then

let x be the convergence point discovered in Condition $1B$;
if $\varphi_j(x) = p(s)$, then let $\varphi_{p(0)}(x) = p(0)$, else let $\varphi_{p(0)}(x) = p(s)$;
for each z such that $n'_s \leq z < y$ and $z \neq x$, let $\varphi_{p(0)}(z) = p(s)$;
let $\varphi_{p(s)}$ be the same as $\varphi_{p(0)}$ from this point onwards;
go to Stage $s + 1$;

{Note that if $p(s)$ is in range of $\varphi_{p(0)}$ and $\varphi_{p(0)}$ is total, then $\varphi_{p(0)} =^1 \varphi_{p(s)}$}

Step 2: let n''_s be the least x such that $\varphi_{p(s)}(x)$ has not been defined till now;
for all x such that $x < n''_s$ and $\varphi_{p(0)}(x)$ has not been defined, let $\varphi_{p(0)}(x) = \varphi_{p(s)}(x)$;
repeat following steps
let $\varphi_{p(0)}(n''_s) = \varphi_{p(s)}(n''_s) = p(0)$;
let $n''_s = n''_s + 1$

until either Condition $2A$ or Condition $2B$ is true:

Condition $2A$: $\mathbf{M}(p(0), \varphi_{p(0)}[n''_s]) \neq j$;
Condition $2B$: $\mathbf{M}(p(s), \varphi_{p(s)}[n''_s]) = j$;

Step $2a$: if Condition $2A$ holds then go to Stage $s + 1$;
Step $2b$: if Condition $2B$ holds then go to Step 1
End of *Stage* s.

Now consider the following cases:
Case 1: There are infinitely many stages. In this case, let additional information be $p(0)$ and $f = \varphi_{p(0)} \in \mathcal{S}$.
Case 1a: $\mathbf{M}(p(0), f)$ diverges. In this case, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,0}$-identify $f \in \mathcal{S}$.
Case 1b: $\mathbf{M}(p(0), f)$ converges to j at Stage s. In this case, the only way in which infinitely many stages can exist is by the execution of Step $1b$ infinitely-often. But, then $\varphi_{\mathbf{M}(p(0),f)}$ is infinitely different from f.
Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt.
Case 2a: Steps 1 and 2 are both executed infinitely-often. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in \mathcal{S}$. Now, $\mathbf{M}$ on f with additional information $p(s)$ does not converge (the only way in which Steps 1 and 2 can be executed infinitely-often without leaving Stage s is by repeated execution of Step $1a$ and Step $2b$, and thus $\mathbf{M}$ infinitely-often outputs j without converging to j).
Case 2b: In stage s, the algorithm never leaves Step 1. In this case, let $f = \varphi_{p(s)} \in \mathcal{S}$. Now, $\mathbf{M}$ on f with additional information $p(s)$ converges to j (otherwise Condition $1A$ would succeed) and φ_j converges only on finitely many inputs (otherwise Condition $1B$ would succeed). Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,0}$-identify f.

Case 2c: In stage s, the algorithm never leaves Step 2. In this case, let $f = \varphi_{p(0)} = \varphi_{p(s)} \in \mathcal{S}$. Now $\mathbf{M}$ on f with additional information $p(s)$ does not converge to j (otherwise Condition $2B$ would succeed) and $\mathbf{M}$ on f with additional information $p(0)$ converges to j (otherwise Condition $2A$ would succeed). Thus, $\mathbf{M}$ does not $\mathbf{ResBex}^{*,0}$-identify f.

From the above cases, we have that $\mathbf{M}$ does not $\mathbf{ResBex}^{*,0}$-identify $\mathcal{S}$. □

A similar proof can be given for the following Theorem 13 below.

Theorem 13 $\mathbf{Bc} - \mathbf{ResBex}^{*,1} \neq \emptyset$.

It is open at present if the above theorems can be extended to $\mathbf{Bc}^0 - \mathbf{ResBex}^{*,0} \neq \emptyset$.

The following theorem shows the advantage of allowing the learning machine to converge to different programs on different bounds.

Theorem 14 $(\forall c \in N)[\mathbf{Bex}^{0,c} - \mathbf{ResBex}^{*,0} \neq \emptyset]$.

Corollary 7 shows that if we fix the number of anomalies allowed in the additional information and the inferred program then allowing a machine to converge to any program for the function, as opposed to some unique program for any upper-bound, results in increased inferring power.

Corollary 7 $(\forall a \in N \cup \{*\})(\forall c \in N)[\mathbf{ResBex}^{a,c} \subset \mathbf{Bex}^{a,c}]$.

Proof of Theorem 14: Consider the following class of functions:

$\mathcal{S} = \{f \mid (\forall y)(\forall z)[f(\langle y, 0\rangle) = f(\langle y, z\rangle)]\}$.

Suppose x is an upper bound on the minimal program index for a function $f \in \mathcal{S}$. Since $\mathcal{R} \in \mathbf{Bex}^{c,c}$, a program p, which computes f with at most c errors can be found in the limit. For all y let w_y be such that $\|\{z \mid z < 2c+1 \wedge \varphi_p(\langle y, z\rangle) = w_y\}\| \geq c+1$. Note that there exists a unique such w_y by the definition of $\mathcal{S}$ and the fact that p computes f with atmost c errors. Let modify(p) be a program obtained from p which behaves as follows: For all y, z, $\varphi_{\text{modify}(p)}(\langle y, z\rangle) = w_y$, where w_y is as defined above. It follows immediately that modify(p) is a program for f. Thus $\mathcal{S} \in \mathbf{Bex}^{0,c}$.

If $\mathbf{M}$ $\mathbf{ResBex}^{*,0}$-identifies $\mathcal{S}$ then it is easy to modify $\mathbf{M}$ to $\mathbf{ResBex}^{*,0}$-identify $\mathcal{R}$. But this is not true (Corollary 6). □

It is open at present if the Theorem 14 above can be extended to $\mathbf{Bex}^{0,*} - \mathbf{ResBex}^{*,0} \neq \emptyset$.

4 Language Identification with Additional Information

4.1 Definition

Analogous to the approach taken in Section 3, a learning machine identifying a language in the presence of additional information can be viewed as taking two arguments: upper-bound on minimal grammar and initial sequence of a text of a language. In other words, learning machines identifying languages with additional information are algorithmic devices that compute a mapping from $N \times$ SEQ to N. For $c \in N \cup \{*\}$, mingram$^c(L)$ denotes the minimal grammar in the φ-system that generates L with at most c errors, i.e., mingram$^c(L) = \mu i.W_i =^c L$.

In Definition 24 just below, we describe what it means for a learning machine to converge on additional information and a text for a language.

Definition 24 $\mathbf{M}(x,T)\downarrow$ (read: $\mathbf{M}$ on text T with additional information x converges) $\iff$ $(\exists i)(\forall^{\infty} n)$ $[\mathbf{M}(x,T[n]) = i]$. If $\mathbf{M}(x,T)\downarrow$, then $\mathbf{M}(x,T)$ is defined = the unique i such that $(\forall^{\infty} n)[\mathbf{M}(x,T[n]) = i]$; otherwise $\mathbf{M}(x,T)$ is said to be undefined.

Definition 25 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtBex}^{a,c}$-*identifies* L (written: $L \in \mathbf{TxtBex}^{a,c}(\mathbf{M})$) $\iff$ $(\forall x \geq \text{mingram}^{c}(L))$ $(\exists i \mid W_i =^{a} L)$ ($\forall$ texts T for L) $(\forall^{\infty} n)[\mathbf{M}(x,T[n]) = i]$.
(ii) $\mathbf{TxtBex}^{a,c} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtBex}^{a,c}(\mathbf{M})]\}$.

Proposition 8 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{TxtBex}^{a_1,c} \subseteq \mathbf{TxtBex}^{a_2,c}$.
(ii) $\mathbf{TxtBex}^{a,c_1} \supseteq \mathbf{TxtBex}^{a,c_2}$.

Definition 26 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtResBex}^{a,c}$-*identifies* L (written: $L \in \mathbf{TxtResBex}^{a,c}(\mathbf{M})$) $\iff$ $(\exists i \mid W_i =^{a} L)$ $(\forall x \geq \text{mingram}^{c}(L))$ ($\forall$ texts T for L) $(\forall^{\infty} n)[\mathbf{M}(x,T[n]) = i]$.
(ii) $\mathbf{TxtResBex}^{a,c} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtResBex}^{a,c}(\mathbf{M})]\}$.

Proposition 9 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{TxtResBex}^{a_1,c} \subseteq \mathbf{TxtResBex}^{a_2,c}$.
(ii) $\mathbf{TxtResBex}^{a,c_1} \supseteq \mathbf{TxtResBex}^{a,c_2}$.

Proposition 10 *Let $a, c \in N \cup \{*\}$. Then,*
$\mathbf{TxtResBex}^{a,c} \subseteq \mathbf{TxtBex}^{a,c}$.

Definition 27 Suppose $\mathbf{M}$ is a learning machine, x is some upper-bound information, and T is a text. Then, $\mathbf{M}(x,T)$ *finitely-converges* (written: $\mathbf{M}(x,T)\Downarrow$) $\iff$ $\{\mathbf{M}(x,\sigma) \mid \sigma \subset T\}$ is finite. If $\mathbf{M}(x,T)\Downarrow$, then we say that $\mathbf{M}(x,T)\Downarrow = D$ $\iff$ $D = \{i \mid (\exists^{\infty} \sigma \subset T)[\mathbf{M}(x,\sigma) = i]\}$, otherwise we say that $\mathbf{M}(x,T)$ *finitely-diverges* (written: $\mathbf{M}(x,T)\Uparrow$).

Definition 28 Let $a, c \in N \cup \{*\}$ and $b \in N^{+} \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtBfex}_{b}^{a,c}$-*identifies* L (written: $L \in \mathbf{TxtBfex}_{b}^{a,c}(\mathbf{M})$) $\iff$ $(\forall x \geq \text{mingram}^{c}(L))$ $(\exists D \mid \|D\| \leq b \wedge (\forall i \in D)[W_i =^{a} L])$ ($\forall$ texts T for L)$[\mathbf{M}(x,T)\Downarrow = D]$.
(ii) $\mathbf{TxtBfex}_{b}^{a,c} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtBfex}_{b}^{a,c}(\mathbf{M})]\}$.

Proposition 11 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Let $b, b_1, b_2 \in N^{+} \cup \{*\}$ such that $b_1 \leq b_2$. Then,*
(i) $\mathbf{TxtBfex}_{b}^{a_1,c} \subseteq \mathbf{TxtBfex}_{b}^{a_2,c}$.
(ii) $\mathbf{TxtBfex}_{b}^{a,c_1} \supseteq \mathbf{TxtBfex}_{b}^{a,c_2}$.
(iii) $\mathbf{TxtBfex}_{b_1}^{a,c} \subseteq \mathbf{TxtBfex}_{b_2}^{a,c}$.

Definition 29 Let $a, c \in N \cup \{*\}$ and $b \in N^{+} \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtResBfex}_{b}^{a,c}$-*identifies* L (written: $L \in \mathbf{TxtResBfex}_{b}^{a,c}(\mathbf{M})$) $\iff$ $(\exists D \mid \|D\| \leq b \wedge (\forall i \in D)[W_i =^{a} L])$ $(\forall x \geq \text{mingram}^{c}(L))$ ($\forall$ texts T for L)$[\mathbf{M}(x,T)\Downarrow = D]$.
(ii) $\mathbf{TxtResBfex}_{b}^{a,c} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtResBfex}_{b}^{a,c}(\mathbf{M})]\}$.

Proposition 12 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Let $b, b_1, b_2 \in N^+ \cup \{*\}$ such that $b_1 \leq b_2$. Then,*
(i) $\mathbf{TxtResBfex}_b^{a_1,c} \subseteq \mathbf{TxtResBfex}_b^{a_2,c}$.
(ii) $\mathbf{TxtResBfex}_b^{a,c_1} \supseteq \mathbf{TxtResBfex}_b^{a,c_2}$.
(iii) $\mathbf{TxtResBfex}_{b_1}^{a,c} \subseteq \mathbf{TxtResBfex}_{b_2}^{a,c}$.

Proposition 13 *Let $a, c \in N \cup \{*\}$ and $b \in N^+ \cup \{*\}$. Then,*
$\mathbf{TxtResBfex}_b^{a,c} \subseteq \mathbf{TxtBfex}_b^{a,c}$.

Definition 30 Let $a, c \in N \cup \{*\}$.
(i) $\mathbf{M}$ $\mathbf{TxtBbc}^{a,c}$-*identifies* L (written: $L \in \mathbf{TxtBbc}^{a,c}(\mathbf{M})$) $\iff$ $(\forall x \geq \text{mingram}^c(L))$ ($\forall$ texts T for L) $(\forall^\infty n)[W_{\mathbf{M}(x,T[n])} =^a L]$.
(ii) $\mathbf{TxtBbc}^{a,c} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathcal{L} \subseteq \mathbf{TxtBbc}^{a,c}(\mathbf{M})]\}$.

Proposition 14 *Let $a, a_1, a_2 \in N \cup \{*\}$ such that $a_1 \leq a_2$. Let $c, c_1, c_2 \in N \cup \{*\}$ such that $c_1 \leq c_2$. Then,*
(i) $\mathbf{TxtBbc}^{a_1,c} \subseteq \mathbf{TxtBbc}^{a_2,c}$.
(ii) $\mathbf{TxtBbc}^{a,c_1} \supseteq \mathbf{TxtBbc}^{a,c_2}$.

4.2 Results

The diagonalization results about function identification have a natural counterpart for language identification. We now proceed to give results which are unique to language identification.

Theorem 15 $(\forall c \in N)[\mathbf{TxtResBex}^{0,c} \ - \ \mathbf{TxtBbc}^{*,*} \neq \emptyset]$.

Proof of Theorem 15: Consider the following class of languages:
$\mathcal{L} = \{L \mid L = N \ \vee \ \|L\| < \mu k.W_k =^c L\}$.
Clearly, $\mathcal{L} \in \mathbf{TxtResBex}^{0,c}$. Let G_N be a grammar for N and $G_\emptyset$ be a grammar for $\emptyset$. Since $\max(G_N, G_\emptyset)$ is a upper-bound on the minimal grammar for a $*$-variant of any language in $\mathcal{L}$, $\mathcal{L} \in \mathbf{TxtBbc}^{*,*} \Leftrightarrow \mathcal{L} \in \mathbf{TxtBc}^*$. Now, suppose by way of contradiction, there exists a machine $\mathbf{M}$ that $\mathbf{TxtBc}^*$-identifies $\mathcal{L}$. Let σ be a $\mathbf{TxtBc}^*$ locking sequence for $\mathbf{M}$ on N. Thus, $\mathbf{M}$ does not $\mathbf{TxtBc}^*$ identify any finite language which is a superset of content(σ). For all j let $L_j = \text{content}(\sigma) \cup [j, j + c]$. Now, for all but finitely many j, $\mu k.[W_k =^c L_j]$ is greater than $\|L_j\|$ and $\|L_j - \text{content}(\sigma)\| = c + 1$. Let L' be one such language (clearly, there exists one). Thus, $\mathbf{M}$ does not $\mathbf{TxtBc}^*$-identify $L' \in \mathcal{L}$. $\square$.

The following proposition follows easily from the proof techniques used in [CL82].

Proposition 15 $(\forall a \in N)(\forall c \in N \cup \{*\})[\mathbf{TxtBfex}_*^{2a,c} \subseteq \mathbf{TxtBbc}^{a,c}]$.

Proof of Proposition 15 : Our proof uses a technique by Case and Lynes [CL82]. Let $\mathbf{M}$ $\mathbf{TxtBfex}^{2a,c}$-identify $\mathcal{L}$. $\mathbf{M}'$ can $\mathbf{TxtBbc}^{a,c}$-identify $\mathcal{L}$ as follows. On input x, σ, $\mathbf{M}'$ behaves as follows: Let $s = |\sigma|$. Let $\mathbf{M}(x, \sigma) = j$. Let $D = \{x \mid x \in W_{j,s} - \text{content}(\sigma)\}$. Let S be the set of a least elements of D (if $\|D\| < a$, then let $S = D$). Output $p(j)$ where $W_{p(j)} = (W_j \cup \text{content}(\sigma)) - S$. It is easy to see that $\mathbf{M}'$ $\mathbf{TxtBbc}^{a,c}$-identifies $\mathcal{L}$. $\square$

Proposition 16 $(\forall a \in N)[\mathcal{E} \in \mathbf{TxtBfex}_*^{a,a}]$.

Corollary 8 $(\forall a \in N)[\mathcal{E} \in \mathbf{TxtBbc}^{a,2a}]$.

Proof of Proposition 16 : Let $\mathrm{cons}(\sigma, j) = \max(\{r < n \mid \|\{x \leq r \mid [x \in (\mathrm{content}(\sigma) \triangle W_{j,n})]\} \leq a\|\})$, where n is the length of σ.

A machine $\mathbf{M}$ can $\mathbf{TxtBfex}^{a,a}_*$-identify $\mathcal{E}$ by emitting, on input x and σ, the minimum $k \leq x$ such that $(\forall j \leq x)[\mathrm{cons}(\sigma, j) \leq \mathrm{cons}(\sigma, k)]$.

□

Theorem 16 $(\forall b \in N^+)$ $[\mathbf{TxtFex}^0_{b+1} - \mathbf{TxtBfex}^{*,0}_b \neq \emptyset]$.

Proof of Theorem 16: It was shown in [Cas88] that $\mathbf{TxtFex}^0_{b+1} - \mathbf{TxtFex}^*_b \neq \emptyset$. Let $\mathcal{L}$ be a class of languages in $\mathbf{TxtFex}^0_{b+1} - \mathbf{TxtFex}^*_b$. We claim that $\mathcal{L} \notin \mathbf{TxtBfex}^{*,0}_b$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{TxtBfex}^{*,0}_b$-identifies $\mathcal{L}$. Let $\mathbf{M}'$ be a machine which $\mathbf{TxtFex}^0_{b+1}$-identifies $\mathcal{L}$. We construct a machine $\mathbf{M}''$ which $\mathbf{TxtFex}^*_b$-identifies any $L \in \mathcal{L}$. $\mathbf{M}''$ on input σ gives machine $\mathbf{M}$, σ with additional information $x = \max(\{\mathbf{M}'(\tau) \mid \tau \subseteq \sigma\})$. $\mathbf{M}''$ then outputs the output of $\mathbf{M}$. Clearly, for large enough initial segment, σ, of any text for any $L \in \mathcal{L}$, $\max(\{\mathbf{M}'(\tau) \mid \tau \subseteq \sigma\})$ converges to an upper bound on the minimal grammar for L (since $\mathbf{M}'$ $\mathbf{TxtFex}^0_{b+1}$-identifies $\mathcal{L}$). Since $\mathbf{M}$ $\mathbf{TxtBfex}^{*,0}_b$-identifies L, $\mathbf{M}''$ $\mathbf{TxtFex}^*_b$-identifies L. Since this is not possible no such machine $\mathbf{M}$ can exist. □.

As a contrast to Proposition 5, in the context of language identification it turns out that the class of all r.e. languages, $\mathcal{E}$, does not belong to $\mathbf{TxtBex}^{*,0}$.

Corollary 9 $\mathcal{E} \notin \mathbf{TxtBex}^{*,0}$.

The following Corollary 10 to the Theorem 16 above says that if we fix the number of anomalies allowed in the additional information and the converged grammars, inferring power of machines is increased by allowing the machine to converge to more number of grammars.

Corollary 10 $(\forall a, b \in N \cup \{*\})$
$\mathbf{TxtBfex}^{a,b}_1 \subset \mathbf{TxtBfex}^{a,b}_2 \subset \ldots \subset \mathbf{TxtBfex}^{a,b}_*$.

Theorem 17 $(\forall a \in N)(\forall b \in N^+)[\mathbf{TxtFex}^{a+1}_1 - \mathbf{TxtBfex}^{a,0}_b \neq \emptyset]$.

Proof of Theorem 17: Consider the following class of languages:

$\mathcal{L} = \{L \mid$
$[L$ is infinite $] \wedge$
$[[(\exists r)[\langle 0, r\rangle \in L]] \wedge [(\forall r)[\langle 0, r\rangle \in L \Rightarrow W_r =^{a+1} L]]] \wedge$
$[(\forall^\infty \langle x+1, y\rangle \in L)[W_x = L]] \wedge$
$[\ \|\{x \mid \langle x+1, y\rangle \in L\} \bigcup \{x \mid \langle 0, x\rangle \in L\}\| < \infty]\ \}$

Clearly, $\mathcal{L} \in \mathbf{TxtFex}^{a+1}_1$. Also, $\mathcal{L} \in \mathbf{TxtBfex}^{a,0}_b \Leftrightarrow \mathcal{L} \in \mathbf{TxtFex}^a_b$. Suppose by way of contradiction, there exists a machine $\mathbf{M}$ that $\mathbf{TxtFex}^a_b$-identifies $\mathcal{L}$. Then, by the implicit use of $(a+1) * b + 2$-ary recursion theorem (based on Smullyan's double recursion theorem [Smu61]), there exist $e_0 < e_1 < ... < e_{(a+1)*b+1}$ such that $W_{e_0}, W_{e_1}, \ldots, W_{e_{(a+1)*b+1}}$ are defined as follows. Enumerate $\langle 0, e_0\rangle$ in W_{e_0}. Let $W^s_{e_i}$ denote W_{e_i} enumerated before Stage s. Let $\sigma_0 = (\langle 0, e_0\rangle)$. Go to Stage 0.

Stage s

enumerate $W^s_{e_0} \cup W^s_{e_1} \cup \cdots \cup W^s_{e_{(a+1)*b+1}}$ in $W_{e_0}, W_{e_1}, \ldots, W_{e_{(a+1)*b+1}}$;
let D be the set of last b grammars output by $\mathbf{M}$ on σ_s;
let $\tau_1 = \sigma_s$;
for each $l \in D$, let $d_l = 0$;
go to Substage 1;

Substage k

enumerate W_{e_0} enumerated until now in W_{e_k};
let $m = \max(\{l \mid \langle e_k + 1, l\rangle \in \text{content}(\tau_k)\})$;
enumerate $\langle e_k + 1, m + l\rangle, 1 \leq l \leq a + 1$ in W_{e_k};
let $n = m + a + 1$;
repeat following steps

let $n = n + 1$;
enumerate $\langle e_k + 1, n\rangle$ in W_{e_0}, W_{e_k};

until either Condition A or Condition B is true;

Condition A: $(\exists \sigma \supseteq \tau_k)[\text{content}(\sigma) \subseteq W_{e_k} \bigwedge \mathbf{M}(\sigma) \notin D]$;
Condition B: $(\exists l \in D)[[W_l \cap \{\langle e_k+1, m+1\rangle, \ldots, \langle e_k+1, m+a+1\rangle\} \neq \emptyset] \wedge [d_l \leq a]]$;

if Condition A holds then
enumerate content(σ) in W_{e_0};
let σ_{s+1} be an extension of σ such that content(σ_{s+1}) = W_{e_0} enumerated till now;
go to Stage $s + 1$;

if Condition B holds then
let $d_l = d_l + 1$;
let $\tau_{k+1} \supseteq \tau_k$ be such that content(τ_k) = W_{e_0} enumerated until now;
go to Substage $k + 1$;

End of *Substage k*
End of *Stage s*

Now consider the following cases:

Case 1: There are infinitely many stages. In this case, let $L = W_{e_0}$. Clearly, $L \in \mathcal{L}$. But, $\mathbf{M}$ on a text for L does not converge to a set of $\leq b$ grammars.

Case 2: Only finitely many stages halt. Let Stage s be the least stage that does not halt. Clearly, at any stage at most $(a + 1) * b + 1$ substages can be executed (since in each substage, at least one of d_l, $(l \in D)$ increases and d_l is bounded by $a + 1$). Let k be the substage which never terminates. Let $L = W_{e_k}$. Clearly, $L \in \mathcal{L}$. Also, the last b grammars emitted by $\mathbf{M}$ on a text for W_{e_k} differ from L by at least $a + 1$ elements (since either W_l does not enumerate any of $\langle e_k + 1, m + 1\rangle, \ldots, \langle e_k + 1, m + a + 1\rangle$ or $d_l = a + 1$ (in which case $\|W_l - W_{e_k}\| \geq d_l = a + 1$)).

From the above cases, it follows that $\mathbf{M}$ does not $\mathbf{TxtFex}^a_b$-identify $\mathcal{L}$. □

The following corollary follows from Theorem 3.

Corollary 11 $(\forall a \in N)[\mathbf{TxtEx}^{a+1} - \mathbf{TxtResBfex}_{*}^{a,0} \neq \emptyset]$.

Proof of Corollary 11: Consider the class of languages $\mathcal{L} = \{L \mid L$ is single valued total $\wedge[\langle 0, x\rangle \in L \Rightarrow \varphi_x =^{a+1} [\lambda y.[z \mid \langle y, z\rangle \in L]]]\}$. Clearly, $\mathcal{L} \in \mathbf{TxtEx}^{a+1}$. Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ such that $\mathcal{L} \subseteq \mathbf{TxtResBfex}_{*}^{a,0}(\mathbf{M})$. It is easy to see that a grammar for an a-variant of a single valued total r.e. language can be effectively transformed into a program for an a-variant of the corresponding recursive function. Hence, $\mathbf{M}$ can easily be modified to $\mathbf{ResBex}^{a,0}$-identify $\{f \mid \varphi_{f(0)} =^{a+1} f\}$ (Since $\mathbf{Fex}^a = \mathbf{Ex}^a$ [CS83]). This is a contradiction because according to Theorem 3 $\{f \mid \varphi_{f(0)} =^{a+1} f\} \notin \mathbf{ResBex}^{a,0}$. Hence, $\mathcal{L} \notin \mathbf{TxtResBfex}_{*}^{a,0}$. □

Corollary 12 *For all* $a, b \in N \cup \{*\}$,
$\mathbf{TxtResBfex}_{b}^{0,a} \subset \cdots \subset \mathbf{TxtResBfex}_{b}^{i,a} \subset \cdots \subset \mathbf{TxtResBfex}_{b}^{*,a}$.

Corollary 13 *Suppose* $c \in N \cup \{*\}$. *Then,*
$\mathbf{TxtResBex}^{0,c} \subset \cdots \subset \mathbf{TxtResBex}^{i,c} \subset \mathbf{TxtResBex}^{i+1,c} \subset \cdots \subset \mathbf{TxtResBex}^{*,c}$.

Corollary 14 *For* $a \in N \cup \{*\}, j \in N$, $\mathbf{TxtBfex}_{j}^{0,a} \subset \cdots \subset \mathbf{TxtBfex}_{j}^{i,a} \subset \cdots \subset \mathbf{TxtBfex}_{j}^{*,a}$.

Corollary 15 *For* $a \in N \cup \{*\}$, $\mathbf{TxtBex}^{0,a} \subset \cdots \subset \mathbf{TxtBex}^{i,a} \subset \cdots \subset \mathbf{TxtBex}^{*,a}$.

The above corollaries establish anomaly hierarchies for both **TxtResBex**-identification and **TxtBex**-identification of languages. For both the language learning criteria, if we fix the number of anomalies allowed in the language about which additional information is provided, we get increased learning power by allowing more anomalies in the language generated by the inferred grammar. The greatest increase in the learning power is achieved by allowing an unbounded finite number of anomalies in the language generated by the inferred grammar.

The following corollary follows from theorem 4.

Corollary 16 $(\forall a \in N)[\mathbf{TxtEx}^{a+1} - \mathbf{TxtBfex}_{*}^{a,a+1} \neq \emptyset]$.

As a corollary to Proposition 16 and Corollary 16 we have

Corollary 17 *For all* $c \in N$, $\mathbf{TxtBfex}_{*}^{0,c} \subset \mathbf{TxtBfex}_{*}^{1,c} \subset \cdots \subset \mathbf{TxtBfex}_{*}^{c,c} = \mathbf{TxtBfex}_{*}^{c+1,c} = \cdots = 2^{\mathcal{E}}$.

Theorem 18 $(\forall a \in N)[\mathbf{TxtEx}^{2a+1} - \mathbf{TxtBbc}^{a,2a+1} \neq \emptyset]$.

Corollary 18 $(\forall a \in N)$ $[\mathbf{TxtBc}^{a+1} - \mathbf{TxtBbc}^{a,2a+1} \neq \emptyset]$.

Proof of Theorem 18: Consider the class of languages:
$\mathcal{L} = \{L \mid L =^{2a+1} N\}$.
Clearly, $\mathcal{L} \in \mathbf{TxtEx}^{2a+1}$. Also, $\mathcal{L} \in \mathbf{TxtBbc}^{a,2a+1} \Leftrightarrow \mathcal{L} \in \mathbf{TxtBc}^{a}$. In [CL82] it was shown that $\mathcal{L} \notin \mathbf{TxtBc}^{a}$. This proves the theorem. □

Theorem 19 $(\forall a \in N)$
1. $\mathbf{TxtResBex}^{0,2a} - \mathbf{TxtBbc}^{a,2a+1} \neq \emptyset$.
2. $\mathbf{TxtResBex}^{0,2a+1} - \mathbf{TxtBbc}^{a,2a+2} \neq \emptyset$.

Corollary 19 $(\forall a \in N)$
1. $\mathbf{TxtBbc}^{0,2a} - \mathbf{TxtBbc}^{a,2a+1} \neq \emptyset$.
2. $\mathbf{TxtBbc}^{0,2a+1} - \mathbf{TxtBbc}^{a,2a+2} \neq \emptyset$.

Proof of Theorem 19: We prove the first part of the theorem. Second part can be proved similarly.

Consider the class of languages:

$$\mathcal{L} = \{L \mid L =^{2a+1} N \bigwedge \max(\{x \mid x \notin L\}) \leq 1 + 2a + (4a+2) * (1 + \mu k.[W_k =^{2a} L]) \}$$

Clearly, $\mathcal{L} \in \mathbf{TxtResBex}^{0,2a}$. Clearly, $\mathcal{L} \in \mathbf{TxtBbc}^{a,2a+1} \Leftrightarrow \mathcal{L} \in \mathbf{TxtBc}^{a}$. Suppose by way of contradiction that $\mathbf{M}$ $\mathbf{TxtBc}^{a}$-identifies $\mathcal{L}$. Let σ be a $\mathbf{TxtBc}^{*}$ locking sequence for $\mathbf{M}$ on N. Clearly, for any $L \supseteq \text{content}(\sigma)$ such that $\|N - L\| = 2a + 1$, $\mathbf{M}$ does not $\mathbf{TxtBc}^{a}$-identify L. We will give a language $L \in \mathcal{L}$ such that $\|N - L\| = 2a + 1, L \supseteq \text{content}(\sigma)$. This would prove the theorem.

Let $m = \max(\text{content}(\sigma))$. Consider the set $S = \{i \mid i \leq m \bigwedge \|N - W_i\| \leq 4a + 1\}$. Let $X = \{x \mid x \notin W_i \text{ for some } i \in S\}$. Clearly $\|X\| \leq (m + 1) * (4a + 1)$. Thus there exists a set D of cardinality $2a+1$ such that $\min(D) > m \bigwedge \max(D) \leq m+1+2a+1+(m+1)*(4a+1) \bigwedge D \bigcap X = \emptyset$. Thus for $L = N - D$, $\mu k.[W_k =^{2a} L] > m$. Thus $L \in \mathcal{L}$. This proves the theorem. □

Theorem 20 $(\forall b \in N^{+})(\forall c \in N)$ $[\mathbf{TxtResBex}^{0,c} - \mathbf{TxtBfex}_{b}^{*,c+1} \neq \emptyset]$.

Corollaries 20 and 21 establish **TxtResBex** and **TxtBex** hierarchies respectively for the number of anomalies allowed in the language about which additional information is provided. For both the language learning criteria, if we fix the number of anomalies allowed in the language whose grammar is inferred, we get a decrease in learning power by allowing more anomalies in the language about which additional information is provided to the language learning device. The greatest decrease is seen when an unbounded finite number of anomalies is allowed in the language about which additional information is provided.

Corollary 20 *For all* $a \in N \cup \{*\}$,
$\mathbf{TxtResBex}^{a,0} \supset \cdots \supset \mathbf{TxtResBex}^{a,i} \supset \mathbf{TxtResBex}^{a,i+1} \supset \cdots \supset \mathbf{TxtResBex}^{a,*}$.

Corollary 21 *For all* $a \in N \cup \{*\}$,
$\mathbf{TxtBex}^{a,0} \supset \cdots \supset \mathbf{TxtBex}^{a,i} \supset \mathbf{TxtBex}^{a,i+1} \supset \cdots \supset \mathbf{TxtBex}^{a,*}$.

Proof of Theorem 20 : Consider the class of languages $\mathcal{L}_c$ defined below.

$$\mathcal{L}_c = \{L \mid [\|L\| \text{ is } \infty \ \wedge \ L = W_{\min(L)}] \vee \|L\| < \min(\{j \mid W_j =^{c} L\}) \}$$

It is easy to verify that $\mathcal{L}_c \in \mathbf{TxtResBex}^{0,c}$. We give a proof to show that $\mathcal{L}_0 \notin \mathbf{TxtBex}^{*,1}$ $(= \mathbf{TxtBfex}_1^{*,1})$. This proof can easily be extended to show that $\mathcal{L}_c \notin \mathbf{TxtBfex}_b^{*,c+1}$.

Suppose by way of contradiction, there exists a learning machine $\mathbf{M}$ that $\mathbf{TxtBex}^{*,1}$-identifies $\mathcal{L}_0$. Then, by the implicit use of double recursion theorem [Smu61], there exists $e_0 < e_1$ such that W_{e_0}, W_{e_1} are defined as follows.

Enumerate e_0 in both W_{e_0} and W_{e_1}. Let $\sigma_0 = (e_0)$.Go to stage 0.

Stage s

enumerate $s + e_0$ in both W_{e_0} and W_{e_1};
let $\sigma'' = \sigma' = \sigma_s \diamond s + e_0$;
dovetail Step 1 and Step 2 below until a mind change is found in either:

Step 1: let $m = s + e_0$;
repeat the following steps
let $m = m + 1$;
enumerate m in W_{e_0};
let $\sigma' = \sigma' \diamond m$;

until $\mathbf{M}(e_1, \sigma') \neq \mathbf{M}(e_1, \sigma_s)$;

Step 2: search for $k, n > e_0$ such that

$$\mathbf{M}(e_1, \sigma'' \diamond \underbrace{k \diamond k \diamond \cdots \diamond k}_{n \text{ times}}) \neq \mathbf{M}(e_1, \sigma_s);$$

let σ denote the sequence for which a mind change was found in either Step 1 or Step 2;
enumerate content(σ) in W_{e_0};
enumerate W_{e_0} enumerated till now in W_{e_1};
let σ_{s+1} be an extension of σ such that content$(\sigma_{s+1}) = W_{e_0}$ enumerated till now;
go to Stage $s + 1$;

End of *Stage s*

Now, consider the following cases:
Case 1: There are infinitely many stages. In this case, let $L = W_{e_0} = W_{e_1}$. Clearly, $L \in \mathcal{L}_0$. But, $\mathbf{M}$, on a text for L with e_1 as the additional information, does not converge.
Case 2: Only finitely many stages halt. Let Stage s be the least such stage that does not halt. Clearly, $W_{e_0} \in \mathcal{L}_0$. For all k, let $L_k = \text{content}(\sigma_s) \cup \{k, e_0 + s\}$. Now for all but finitely many k, $L_k \in \mathcal{L}_0$. Let k_1 be one such k. Note that e_1 is a valid upper-bound for $\mathbf{TxtBex}^{*,1}$-identification of W_{e_0} because $e_1 > e_0$. Also, note that e_1 is a valid upper-bound for $\mathbf{TxtBex}^{*,1}$-identification of L_{k_1} because $W_{e_1} =^1 L_{k_1}$. Now, $\mathbf{M}$ on some text for W_{e_0} and additional information e_1, and on some text for L_{k_1} with additional information e_1 converges to $\mathbf{M}(e_1, \sigma_s)$. But, W_{e_0} is an infinite language and L_{k_1} is a finite language. Hence, $W_{\mathbf{M}(e_1,\sigma_s)}$ cannot be a finite variant of both W_{e_0} and L_{k_1}. Thus, $\mathbf{M}$ does not $\mathbf{TxtBex}^{*,1}$-identify $\mathcal{L}_0$. □

The following follows as corollary to theorem 7.

Corollary 22 $(\forall c \in N)$ $[\mathbf{TxtResBex}^{0,c} - \mathbf{TxtBfex}_*^{c,c+1} \neq \emptyset]$.

As a corollary to Theorem 14 we have,

Corollary 23 $(\forall c \in N)[\mathbf{TxtBex}^{0,c} - \mathbf{TxtResBfex}^{*,0}_{*} \neq \emptyset]$.

Corollary 24 is the language learning counterpart of corollary 7 . If we fix the number of anomalies allowed, both in the additional information and in inferred grammar, then allowing the learning machine to converge to any grammar for the language is better than restricting it to converge to some unique grammar.

Corollary 24 $(\forall a \in N \cup \{*\})\ (\forall b \in N^{+} \cup \{*\})\ (\forall c \in N)\ [\mathbf{TxtResBfex}^{a,c}_{b} \subset \mathbf{TxtBfex}^{a,c}_{b}]$.

5 Conclusion

We present a pictorial summary of results obtained in the following six figures. Figure 1 describes results pertaining to function inference. Figures 2-6 present results about language learning. While interpreting the figure, it should be noted that classes of functions (languages) learnable by an identification criteria A is contained in the classes of functions (languages) learnable by another identification criteria B *if and only if* the containment follows from some inclusion path (except for the open problem mentioned through broken arrows and the ones mentioned below). Solution to the following open problems would complete the picture for the entire corpus of learning criteria introduced in this paper. It is open if theorem 12 and 13 can be extended to $\mathbf{Bc}^{0} - \mathbf{ResBex}^{*,0} \neq \emptyset$. It is also open if theorem 14 can be extended to $\mathbf{Bex}^{0,*} - \mathbf{ResBex}^{*,0} \neq \emptyset$. Corresponding problems in the context of language learning are also open.

6 Acknowledgements

We would like to express our sincere gratitude to John Case and Mark Fulk for helpful advice and encouragement. An anonymous referee made helpful suggestions. Zuzana Dobes, Lata Narayanan, Stuart Kurtz, and Rajeev Raman provided helpful discussions.

Sanjay Jain was supported by NSF grant CCR 832-0136 at the University of Rochester and Arun Sharma was supported by NSF grant CCR 871-3846 to John Case at the State University of New York at Buffalo and the University of Delaware.

References

[Bar74] J. A. Barzdin. Two theorems on the limiting synthesis of functions. *In Theory of Algorithms and Programs, Latvian State University, Riga, U.S.S.R*, 210:82–88, 1974.

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

[Blu67] M. Blum. A machine independent theory of the complexity of recursive functions. *Journal of the ACM*, 14:322–336, 1967.

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

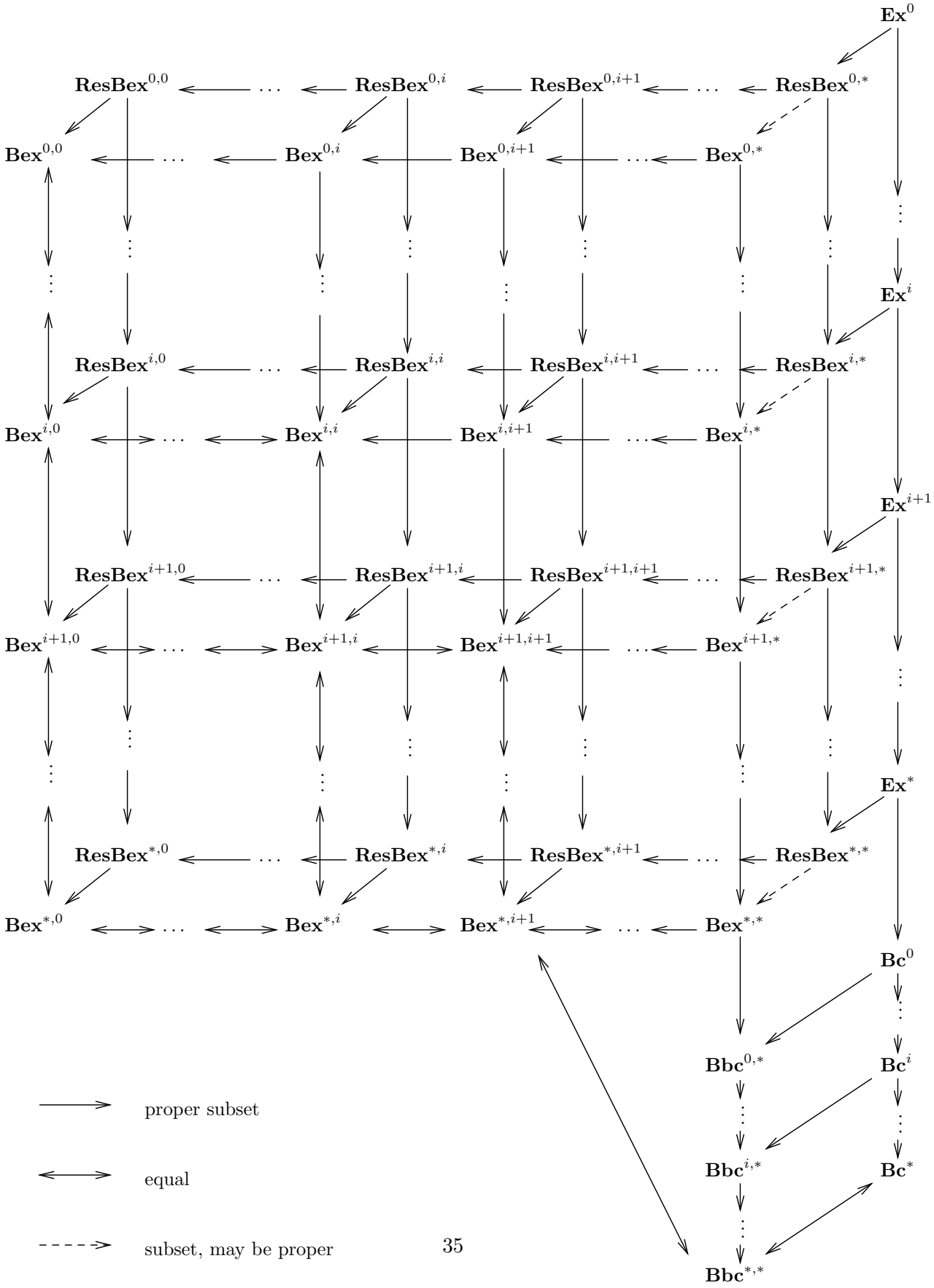


Figure 1: Function Learning

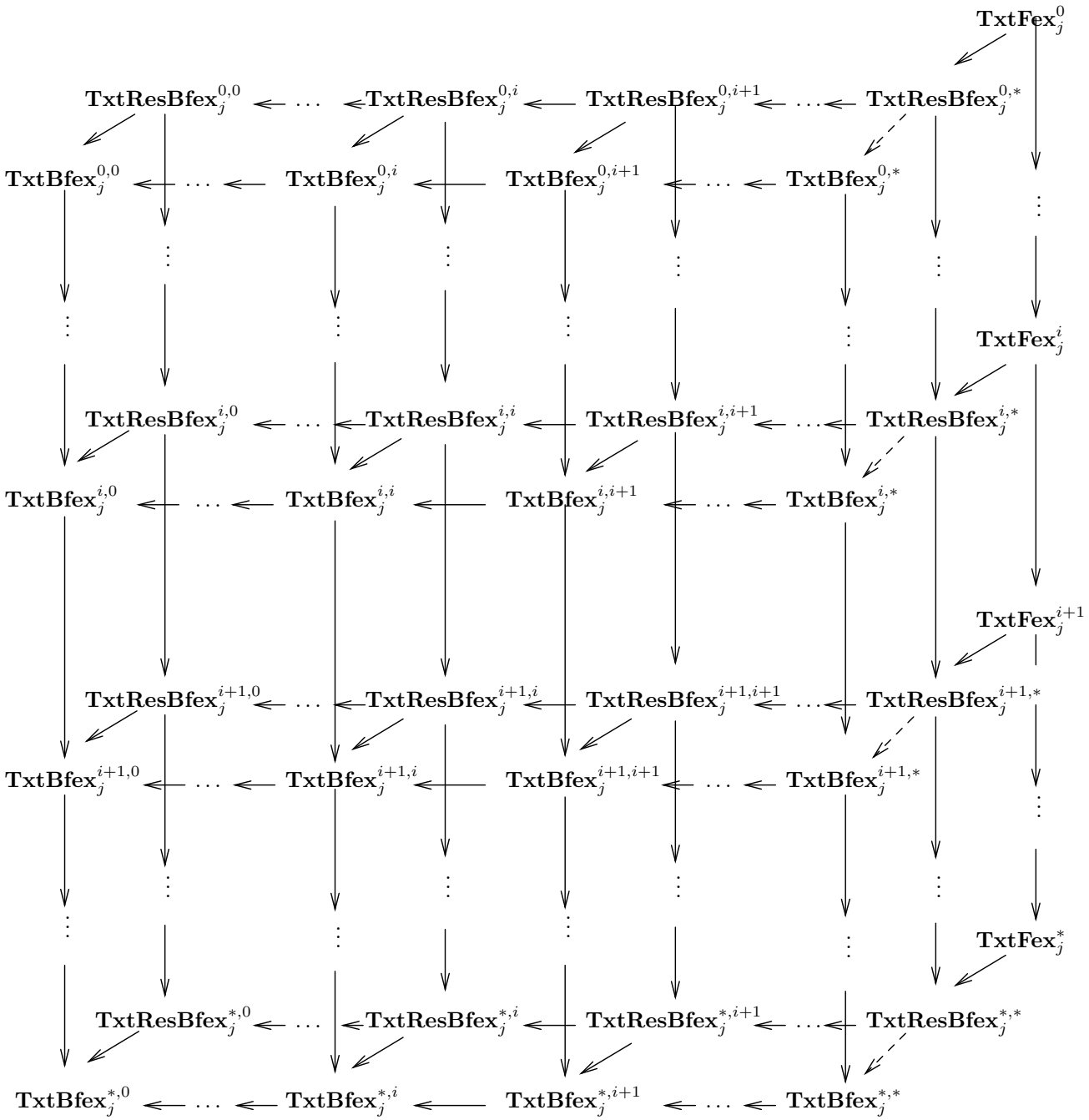


Figure 2: Language Learning - $\mathbf{TxtFex}_j$

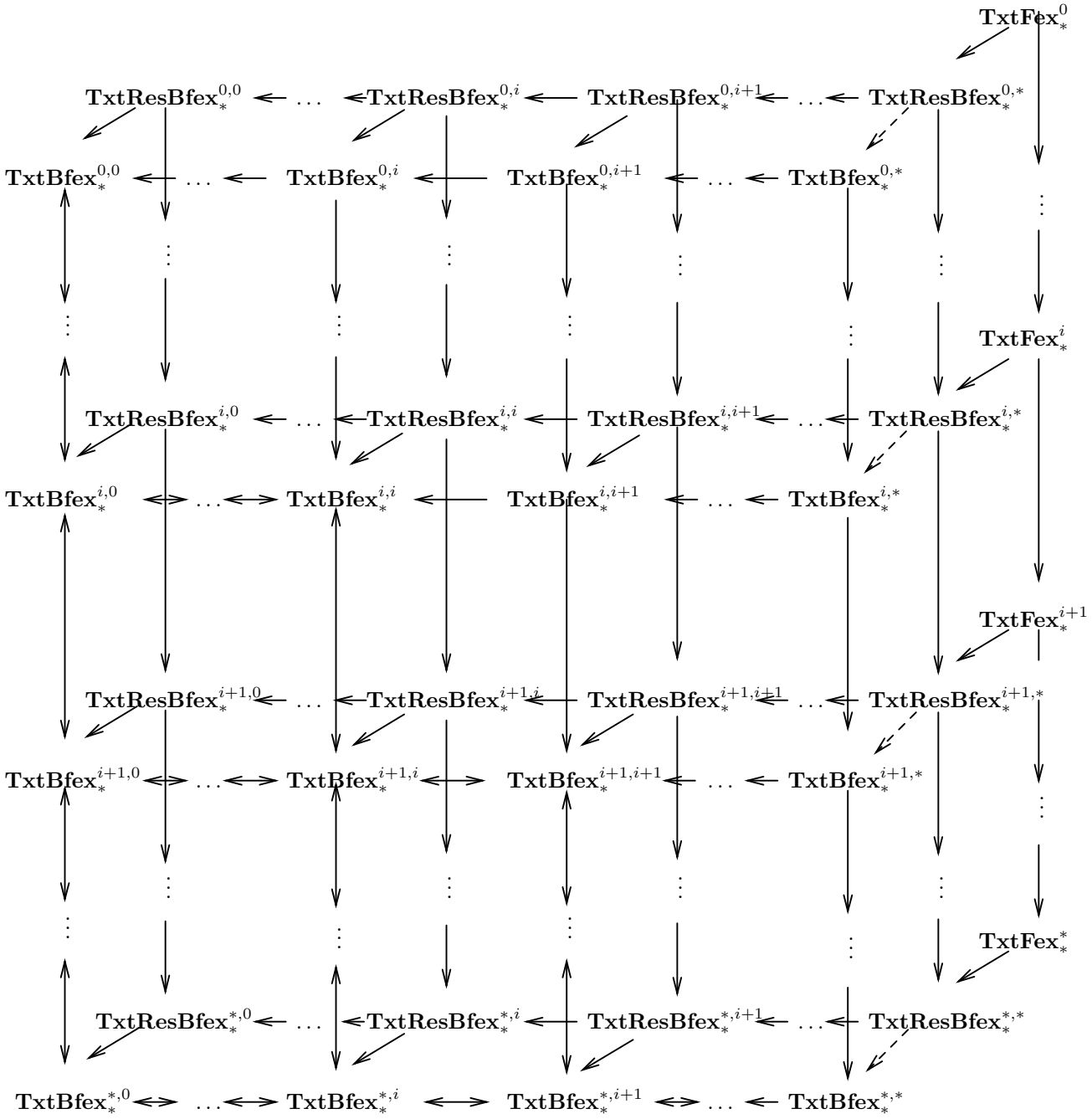
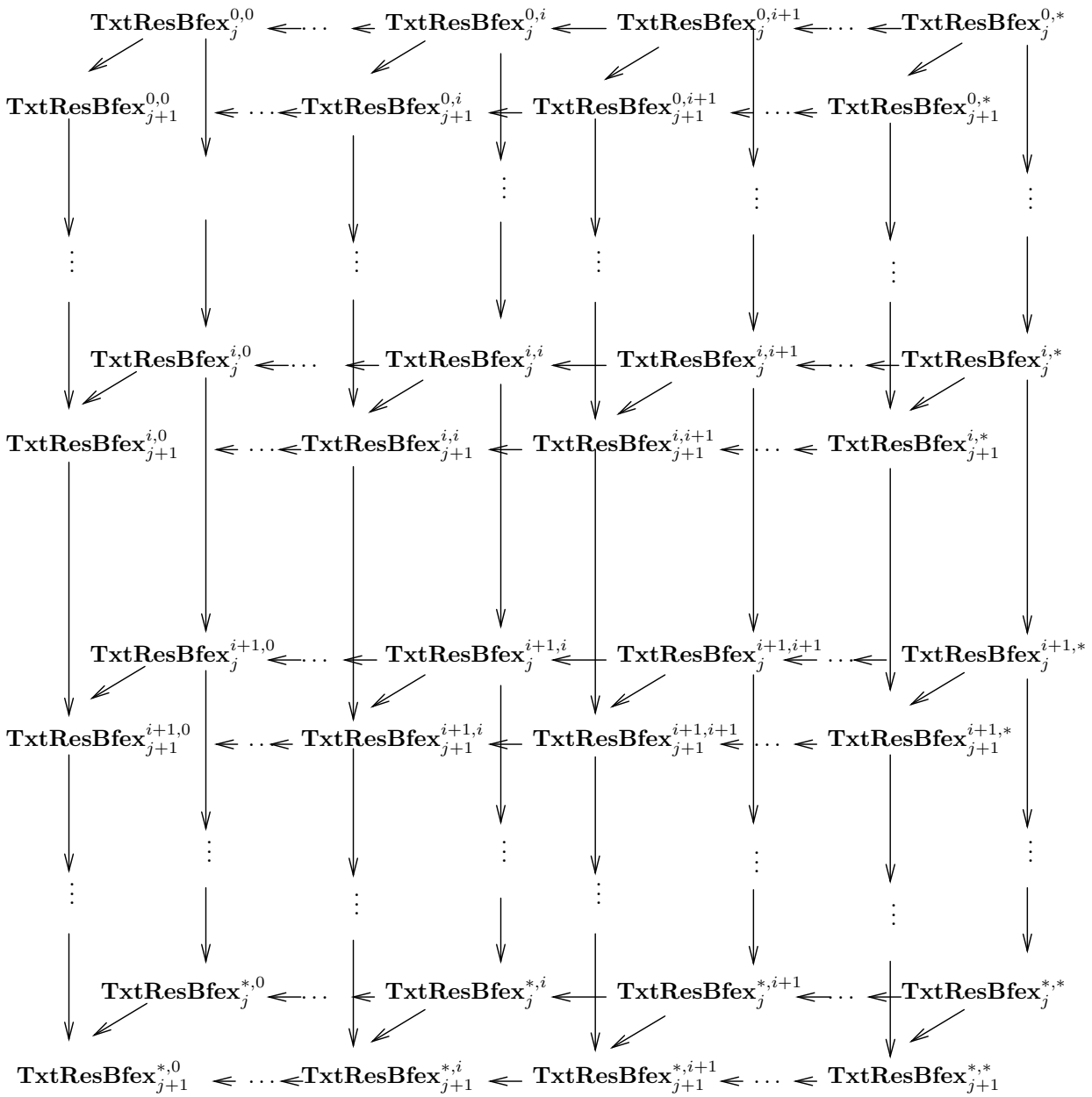


Figure 3: Language Learning - $\mathbf{TxtFex}_*$

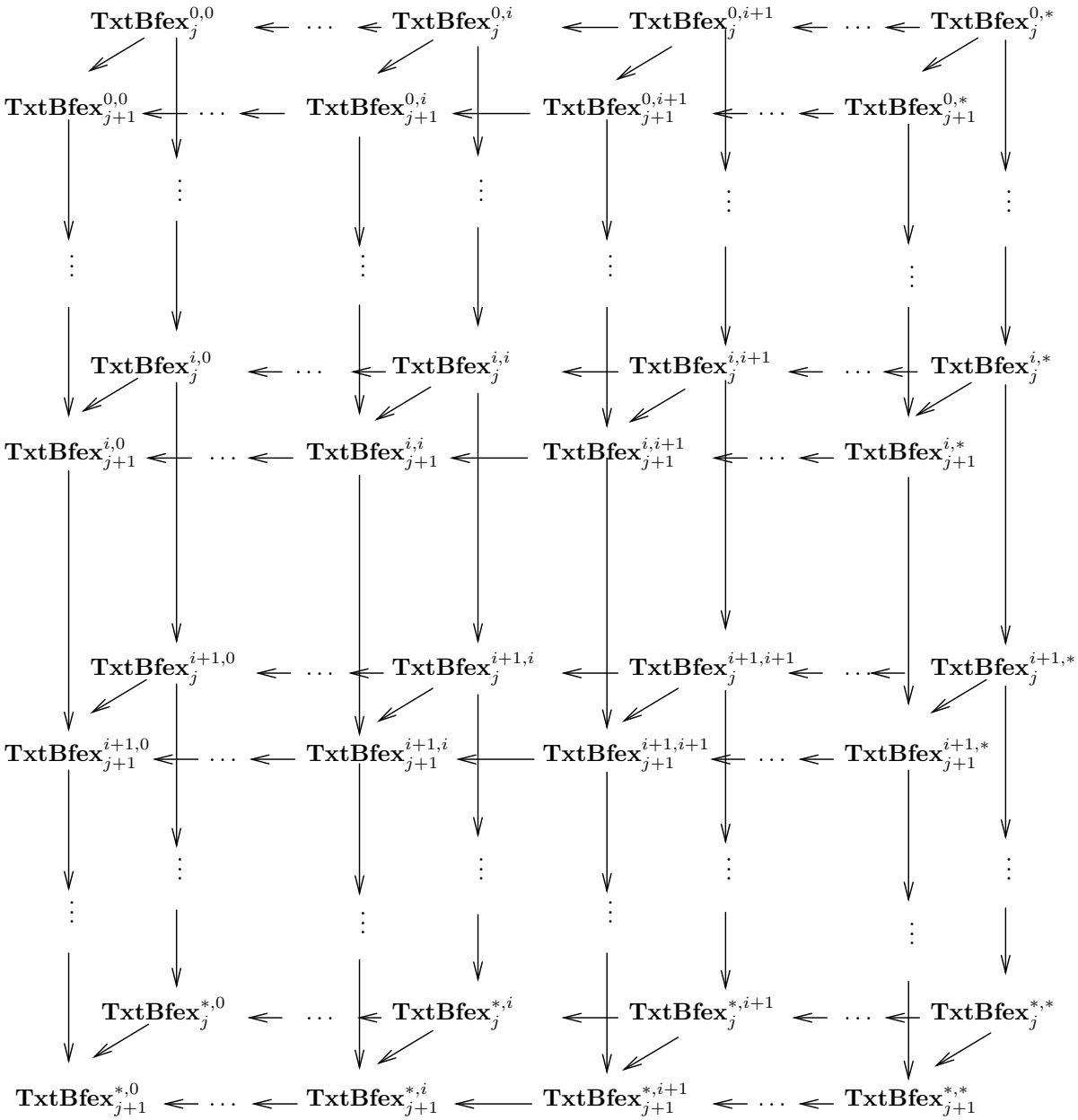


proper subset

equal

subset, may be proper

Figure 4: Language Learning - **TxtResBfex**

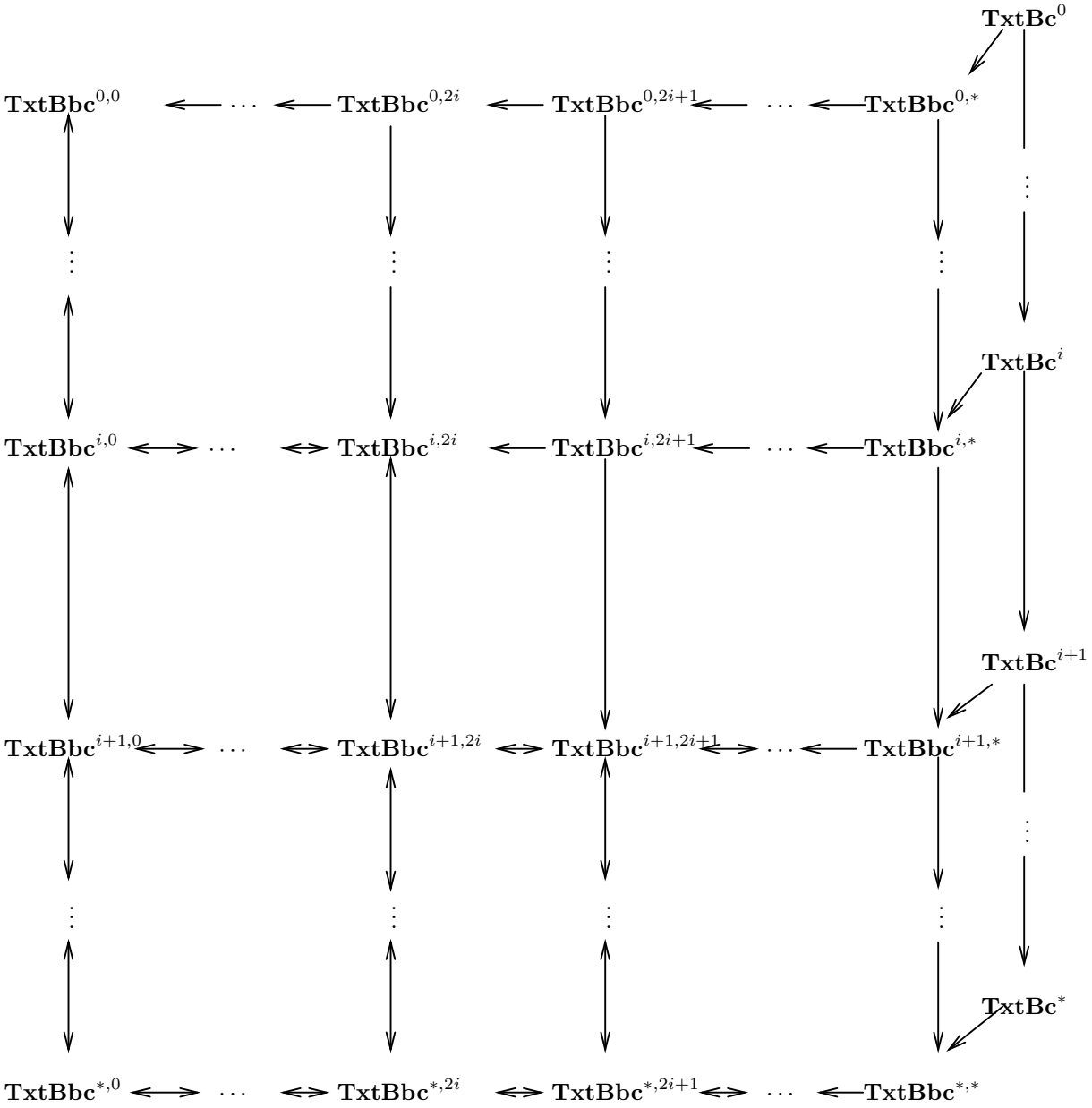


proper subset

equal

subset, may be proper

Figure 5: Language Learning - **TxtBfex**



$\mathbf{TxtBfex}^{2b,a} \dashrightarrow \mathbf{TxtBbc}^{b,a}$

$\longrightarrow$ proper subset

$\dashrightarrow$ subset, proper if $a = *$ or $2b < a$.

$\longleftrightarrow$ equal

$\dashrightarrow$ subset, may be proper

Figure 6: Language Learning - **TxtBbc**

[Cas88] J. Case. The power of vacillation. In D. Haussler and L. Pitt, editors, *Proceedings of the Workshop on Computational Learning Theory*, pages 133–142. Morgan Kaufmann Publishers, Inc., 1988.

[CJS89] J. Case, S. Jain, and A. Sharma. Convergence to nearly minimal size grammars by vacillating learning machines. In R. Rivest, D. Haussler, and M.K. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 189–199. Morgan Kaufmann Publishers, Inc., 1989.

[CL82] J. Case and C. Lynes. Machine inductive inference and language identification. *Lecture Notes in Computer Science*, 140:107–115, 1982.

[CS83] J. Case and C. Smith. Comparision of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[Ful85] M. Fulk. *A Study of Inductive Inference machines*. PhD thesis, SUNY at Buffalo, 1985.

[Ful90a] M. Fulk. Inductive inference with additional information. *Journal of Computer and System Sciences*, 1990. To appear.

[Ful90b] M. Fulk. Prudence and other conditions on formal language learning. *Information and Computation*, 85:1–11, 1990.

[FW79] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Electronische Informationverarbeitung und Kybernetik*, 15:179–195, 1979.

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

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[JS90a] S. Jain and A. Sharma. Language learning by a team. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 153–166, 1990.

[JS90b] S. Jain and A. Sharma. Learning in the presence of partial explanations. *Information and Computation*, 1990. To Appear.

[LP84] D. T. Langendoen and P. M. Postal. *The Vastness of Natural Languages*. Basil Blackwell Publisher Limited, Oxford, England, 1984.

[MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms*. North Holland, New York, 1978.

[OSW82] D. Osherson, M. Stob, and S. Weinstein. Ideal learning machines. *Cognitive Science*, 6:277–290, 1982.

[OSW84] D. Osherson, M. Stob, and S. Weinstein. Learning theory and natural language. *Cognition*, 17:1–28, 1984.

[OSW86a] D. Osherson, M. Stob, and S. Weinstein. Aggregating inductive expertise. *Information and Control*, 70:69–95, 1986.

[OSW86b] 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.

[OW82a] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.

[OW82b] D. Osherson and S. Weinstein. A note on formal learning theory. *Cognition*, 11:77–88, 1982.

[Pin79] S. Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.

[Pit84] L. Pitt. *A characterization of probabilistic inference*. PhD thesis, Yale University, 1984.

[Rog58] H. Rogers. Gödel numberings of partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.

[Roy86] J. Royer. Inductive inference of approximations. *Information and Control*, 70:156–178, 1986.

[Smi82] C. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.

[Smu61] R. M. Smullyan. Theory of formal systems. In *Annals of Mathematics Studies, no. 47*. Princeton, N.J., 1961.

[SV86] C. Smith and M. Velauthapillai. On the inference of programs approximately computing the desired function. *Lecture Notes in Computer Science*, 265:164–176, 1986.

[WC80] K. Wexler and P. Culicover. *Formal Principles of Language Acquisition*. MIT Press, Cambridge, Mass, 1980.

[Wex82] K. Wexler. On extensional learnability. *Cognition*, 11:89–95, 1982.