Generality's Price:
Inescapable Deficiencies in Machine-Learned Programs

John Case[1], K-J. Chen[2], Sanjay Jain[3], Wolfgang Merkle[4], and James S. Royer[5]

[1] Dept. of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586 USA. case@cis.udel.edu
[2] Institute of Information Science, Academia Sinica, Nankang 115, Taipei, Taiwan, ROC. kchen@iis.sinica.edu.tw
[3] School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543, Republic of Singapore. sanjay@comp.nus.edu.sg
[4] Universität Heidelberg, Mathematisches Institut, Im Neuenheimer Feld 294, D-69120 Heidelberg, Germany. merkle@math.uni-heidelberg.de
[5] Dept. of Elec. Eng. and Computer Science, Syracuse University, Syracuse, NY 13244 USA. royer@ecs.syr.edu

Abstract. This paper investigates some delicate tradeoffs between the *generality* of an algorithmic learning device and the *quality* of the programs it learns successfully. There are results to the effect that, thanks to small increases in generality of a learning device, the computational complexity of some successfully learned programs is provably unalterably *sub*optimal. There are also results in which the complexity of successfully learned programs *is* asymptotically optimal and the learning device is general, but, still thanks to the generality, some of those optimal, learned programs are provably unalterably *information deficient* — in some cases, deficient as to safe, algorithmic extractability/provability of the fact that they are even approximately optimal.

1 Introduction

We abbreviate *class of characteristic functions of languages* by CCFL. Suppose $\mathcal{C}_0 \subseteq \mathcal{C}_1$ is a pair of complexity CCFLs which *do* (perhaps barely) separate. For example, let α be a *very slow growing*, linear time computable inverse of Ackermann's function as from [CLRS01, §21.4]; let $\mathcal{C}_1$ be $\mathrm{DTIME}(n \cdot (\log n) \cdot \alpha(n))$; and let $\mathcal{C}_0$ be $\mathrm{DTIME}(n)$.[6] These classes have long been known to separate [HS65,HU79]. Furthermore, it is straightforward that *some* learning device (synonymously, inductive inference machine or IIM) $\mathbf{M}_0$, fed the values of any element f of this $\mathcal{C}_0$, outputs nothing but linear-time programs and eventually converges to a fixed linear-time program which correctly computes f. This kind

[6] $\mathrm{DTIME}(t(n))$ denotes the set of languages decidable by a deterministic, multi-tape Turing machine within $O(t(n))$ time, where n is the length of the machine's input. $\mathrm{DTimeF}(t(n))$ denotes the set of *functions* over strings computable by this same class of machines within $O(t(n))$ time.

of syntactically converging learning in the limit is called EX-*learning* (or EX-*identification*) [Gol67,BB75,CS83,JORS99]. Let $\mathcal{Z}^*$ be the CCFL for (all and only) the *finite* languages. Clearly, $\mathcal{Z}^*$ is an especially simple, proper subclass of our example $\mathcal{C}_0$. Two of our main theorems (Theorems 19 and 20 in §6 below) each implies that, nonetheless, if $\mathbf{M}_1$ is *any* learning device which is slightly more general than $\mathbf{M}_0$ in that it EX-learns every function in our example $\mathcal{C}_1$, then, for some especially "easy" function f, more particularly for an $f \in \mathcal{Z}^*$, $\mathbf{M}_1$ on f syntactically converges to a correct program p for f, *but* this p runs in worse than any linear-time bound *on all but finitely many inputs*. This *inherent* run-time deficiency of p is the *inescapable* price for employing a more general learning device to learn $\mathcal{C}_1$ instead of learning only $\mathcal{C}_0$. Theorems 19 and 20 on which this example is based are proved by delayed diagonalization (or slowed simulation) [Lad75,RC94] with cancellation [Blu67] (or zero injury), complexity-bounded self-reference [RC94], and careful subrecursive programming [RC94].

Fix $k \geq 1$. Let $\mathcal{C}_1 = \text{DTIME}(n^k \cdot (\log n) \cdot \alpha(n))$ and $\mathcal{C}_0 = \text{DTIME}(n^k)$. These classes separate [HS65,HU79], and it is straightforward that some learning device EX-learns this $\mathcal{C}_0$ outputting only conjectures that run in k-degree polytime. However, again from Theorems 19 and 20, for any slightly more general learning device $\mathbf{M}$ which EX-learns this $\mathcal{C}_1$, there will be an easy f, an $f \in \mathcal{Z}^*$, so that, on f, $\mathbf{M}$'s final program p will run worse than any k-degree polytime bound on all but finitely many inputs.

One way to circumvent the complexity-deficiency-in-learned-programs price of generality in the above examples is to consider a most general learning criterion called BC*-*learning* [CS83,CCJ01]. In this type of learning, in contrast to EX-learning, one foregoes syntactic convergence in favor of semantic convergence and one foregoes requiring the final programs to be perfectly correct at computing the input function: convergence is to an infinite sequence of programs all but finitely many of which are each correct on all but finitely many inputs. Harrington [CS83] showed that there is a learning device that BC*-learns *every* computable function. (On the other hand, fairly simple classes of computable functions cannot be EX-learned [CS83].) One of our main positive results (Theorem 23 in §8 below) says that there is a learning device $\mathbf{M}_*$ that BC*-learns the CCFL for the polytime decidable languages in such a way that: (i) all of $\mathbf{M}_*$'s output conjectures run in polytime, (ii) for each $k \geq 1$, on each $f \in \text{DTIME}(n^k)$, all but finitely many of $\mathbf{M}_*$'s outputs run in k-degree polytime; *and* (iii) $\mathbf{M}_*$ EX-learns all the linear-time computable functions.

There is, though, another kind of deficiency-in-learned-programs price for generality of learning, and this affects BC*-learning, EX-learning, and the learning criteria of strength in between them and which are discussed beginning in §2 below. Let $\mathcal{PF}^k = \text{DTimeF}(n^k)$, and let $\mathcal{QF}^k_\alpha = \text{DTimeF}(n^k \cdot (\log n) \cdot \alpha(n))$. Let φ_q be the (partial) function computed by multi-tape Turing machine q. Suppose $\mathbf{M}$ is any device BC*-learning $\mathcal{QF}^k_\alpha$.[7] Corollary 6 in §4 below says, then, that there is an easy f, an $f \in \mathcal{Z}^*$, so that, if $\mathbf{M}$ is fed the values of f (which it at least BC*-learns), then for all but finitely many of $\mathbf{M}$'s corresponding output

[7] One of many *special cases of this hypothesis* is that $\mathbf{M}$ actually EX-learns $\mathcal{QF}^k_\alpha$.

conjectures p, Peano Arithmetic [Men97] (PA) *fails* to prove that some finite variant of φ_p is k-degree polytime computable. Of course, for such p's, some finite variant of φ_p, e.g., f, *is* trivially *linear-time* computable. Hence, these p's are *information-deficient.* If, for example, $\mathbf{M}_*$, the learning device of Theorem 23 (discussed in the previous paragraph), is used for $\mathbf{M}$, then, on the corresponding f, this $\mathbf{M}$ outputs a *perfectly correct* final program p which runs in linear-time, but Peano Arithmetic can*not* prove the weaker result about this p that some finite variant of φ_p is k-degree polytime computable. Hence, for the device of Theorem 23, its final output on f is information-deficient, but *not* complexity-deficient. Corollary 6 discussed in this paragraph (as well as Corollaries 7 and 8 in the same section and other corollaries that will be included in an expanded version of this paper) is a consequence of one of our main sufficient condition results, Theorem 5 (also in the same section).

Here is another example. Let $(\mathcal{C}_0, \mathcal{C}_1) = (\mathcal{REG}, \mathcal{CF})$, where $\mathcal{REG}$ and $\mathcal{CF}$ are the CCFLs of regular and context free languages, respectively. Of course, for this example, the separation is not particularly tight. However, importantly, for this example, direct, aggressive diagonalization methods such those as mentioned above are not available. Let $\mathcal{U}^*$ be the CCFL for the *co-finite* languages, i.e., the languages whose complements are finite. Clearly, $\mathcal{U}^*$ is an especially simple, proper subclass of $\mathcal{REG}$. First note that some learning device outputs only deterministic finite automata and EX-learns $\mathcal{REG}$ [Gol67], where deterministic finite automata should be thought of as a degenerate case of Turing machines which use *no* tape squares for workspace [HU79]. EX*-learning is the variant of EX-learning in which the final program need be correct only on all but finitely many inputs. By contrast, still in §4 below, as a corollary of our other main sufficient condition result, Theorem 9, we have Corollary 12 as follows. Suppose $\mathbf{M}$ EX*-learns $\mathcal{CF}$ and $k, n \geq 1$.[8] Then there is an easy f, an $f \in \mathcal{U}^*$, such that, if p is $\mathbf{M}$'s final program on f, for *some distinct* $x_0, \ldots, x_{n-1}$, program p uses more than k workspace squares on each of the inputs $x_0, \ldots, x_{n-1}$. This is a complexity-deficiency result for $(\mathcal{REG}, \mathcal{CF})$. Theorem 9 has other complexity-deficiency corollaries, e.g., Corollary 13, an interesting one for $(\mathrm{P}, \mathrm{NP})$ — *assuming* they separate. Corollaries 7 and 8 of Theorem 5 provide information-deficiency results for $(\mathcal{REG}, \mathcal{CF})$ and $(\mathrm{P}, \mathrm{NP})$, respectively.

Those corollaries, discussed in the previous paragraph, of our sufficient condition results, Theorems 5 and 9, involve classes $(\mathcal{C}_0, \mathcal{C}_1)$ for which direct, aggressive diagonalization is (apparently) not available. These sufficient condition results are proved herein with the aid of some refined *inseparability* results from [RC94]. §3 below provides the details. In [RC94] the inseparabilities were used to characterize relative program succinctness between (possibly barely) separated subrecursive programming systems. Herein they are used to obtain higher-type inseparabilities providing our sufficient conditions (not characterizations) for deficiencies in machine-learned programs. We also use Theorem 5 to obtain all our *information-deficiency* results, including the one for $(\mathcal{PF}^k, \mathcal{QF}^k_\alpha)$

[8] One of many special cases of the hypothesis that $\mathbf{M}$ EX*-learns $\mathcal{CF}$ is that $\mathbf{M}$ actually EX-learns $\mathcal{CF}$.

described above. Actually, Theorem 9 can be used to prove a weak special case of our strong complexity-deficiency result (Theorem 20) for $(\mathcal{PF}^k, \mathcal{QF}^k_\alpha)$. This is Corollary 11. In this corollary the quantifier *order* between the $f \in \mathcal{Z}^*$ and the k-degree polynomial time bounds is weakened and the for-all-but-finitely-many-inputs-x quantifier is weakened to exists-n-distinct-inputs.

Some of our results herein whose proofs employ tricks from [RC94] can likely be proved employing instead related methods from [Sch85,Sch82,Reg88]. We may pursue this in an expanded version of the present paper. We also hope to provide, in a future expansion, results in the style of the present paper for P versus various quantum polynomial time classes — assuming separation.

The order of presentation in this introduction differs from that of the remaining sections. The latter order was dictated, to some extent, by the need to introduce required technology in a particular order. Due to space limitations, the proofs of many results have been omitted in this version of the paper.

2 Conventions and notation

Strings and numbers. Each $x \in \mathbb{N}$, the set of non-negative integers, is identified with its dyadic representation over $\{\mathbf{0}, \mathbf{1}\}$. Thus, $0 \equiv \epsilon$, $1 \equiv \mathbf{0}$, $2 \equiv \mathbf{1}$, $3 \equiv \mathbf{00}$, etc.[9] For each $x \in \mathbb{N}$, $|x|$ denotes the length of its dyadic representation. By convention, for $\boldsymbol{x} \in \mathbb{N}^n$, $|\boldsymbol{x}|$ denotes $|x_0| + \cdots + |x_{n-1}|$. (By convention, $\boldsymbol{x}, \boldsymbol{y}, \ldots$ range over finite sequences of numbers, the length of which will always be clear from context.) We will freely pun between $x \in \mathbb{N}$ as a number and a $\mathbf{0}$-$\mathbf{1}$-string.

Let $\langle \cdot, \cdot \rangle$ be a standard, poly-time computable pairing function, e.g., the one from Rogers [Rog67]. By convention, for each $n \geq 2$ and each $x_1, \ldots, x_{n+1} \in \mathbb{N}$, we recursively define $\langle x_1, \ldots, x_{n+1} \rangle =_{\text{def}} \langle x_1, \langle x_2, \ldots, x_{n+1} \rangle \rangle$.

Encoding lists. For each $x, y \in \mathbb{N}$, let $x \diamond y$ denote the concatenation of (the dyadic representations of) x and y. For each $x \in \mathbb{N}$, let $E(x) = \mathbf{1}^{|x|}\mathbf{0} \diamond x$. (Note that the image of E in $\{\mathbf{0}, \mathbf{1}\}^*$ is a collection of prefix codes [LV97, §1.4].) We use E to provide a low complexity encoding of lists of numbers as follows. Let $[\,] = 0$, and for each $x_0, \ldots, x_k \in \mathbb{N}$, let $[x_0, \ldots, x_k] = E(x_0) \diamond \cdots \diamond E(x_k)$. By convention, any element of $\mathbb{N}$ that is not of the form $[x_0, \ldots, x_j]$ is considered as coding the empty list. It is clear from our definition of $[\cdot]$ that concatenations, projections, and so on, involving coded lists are all linear-time computable.

Functions. $A \to B$ denotes the set of all total functions from A to B and $A \rightharpoonup B$ denotes the set of (possibly) partial functions from A to B. Let $\mathcal{F} = (\mathbb{N} \to \mathbb{N})$. For each $\mathcal{C} \subseteq \mathcal{F}$, let $\mathcal{C}_{0\text{-}1}$ denote the 0–1 valued elements of $\mathcal{C}$. $\mathcal{PR}$ denotes the set of partial recursive functions and $\mathcal{R}$ denotes the total recursive functions.

For each $f \colon \mathbb{N} \to \mathbb{N}$, we define $O(f) = \{\, g \colon \mathbb{N} \to \mathbb{N} \mid (\exists a)(\forall x)[\, g(x) \leq a \cdot (f(x) + 1)\,]\,\}$. By convention $O(f(n))$ is short for $O(\lambda x \centerdot f(|x|))$.

Suppose $f, g \colon \mathbb{N} \rightharpoonup \mathbb{N}$. We say that f and g are *finite variants* if and only if $\{\, x \mid f(x) \neq g(x)\,\}$ is finite. Let $\mathcal{FV}(f)$ = the set of finite variants of f. For

[9] We employ dyadic representation instead of the more standard binary number representation to avoid problems with the degeneracy of lead zeros in the latter [RC94].

$\mathcal{S} \subseteq (\mathbb{N} \rightharpoonup \mathbb{N})$, let $\mathcal{FV}(\mathcal{S}) = \bigcup_{f \in \mathcal{S}} \mathcal{FV}(f)$. For each $n \in \mathbb{N}$, $f =^n g$ means that $\{ x \mid f(x) \neq g(x) \}$ is of size n or less; and $f =^* g$ means $g \in \mathcal{FV}(f)$.

Programming systems. We say that a partial recursive $\psi\colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ is a *programming system* for a class of functions $\mathcal{S} \subseteq \mathcal{PR}$ if and only if $\mathcal{S} = \{ \lambda x . \psi(i,x) : i \in \mathbb{N}\}$. We typically write $\psi_i(x)$ for $\psi(i,x)$. We say that ψ is an *acceptable* programming system of $\mathcal{S}$ [Rog67,RC94] if and only if it is the case that for any other programming system for $\mathcal{S}$, say θ, there is a recursive function t such that, for all i, $\psi_{t(i)} = \theta_i$, that is, there is an effective way of translating (or reducing) θ-programs into equivalent ψ-programs. Let φ be an acceptable programming system of $\mathcal{PR}$ based on deterministic multi-tape Turing machines [HU79,Jon97]. By convention, for each i and $x_0, \ldots, x_k$, $\varphi_i(x_0, \ldots, x_k) = \varphi_i(\langle x_0, \ldots, x_k\rangle)$. Also, for each i and x, let $\Phi_i(x)$ denote the run time of the computation of the TM encoded by i on input x and let $\Phi_i^{\mathrm{WS}}(x) =$ the work space used by the TM encoded by i on input x, provided $\varphi_i(x){\downarrow}$; ∞, otherwise.

Arithmetic sets. The Σ_0- and Π_0-predicates over $\mathbb{N}^k$ are just the recursive predicates over $\mathbb{N}^k$. P is a Σ_{n+1} predicate over $\mathbb{N}^k$ if and only if for some m, there is a Π_n predicate Q over $\mathbb{N}^{k+m}$ such that $P(\boldsymbol{x}) \equiv (\exists y_1) \ldots (\exists y_m) Q(\boldsymbol{x}, \boldsymbol{y})$ for each $x \in \mathbb{N}^k$. P is a Π_{n+1} predicate over $\mathbb{N}^k$ if and only if for some m, there is a Σ_n predicate Q over $\mathbb{N}^{k+m}$ such that $P(\boldsymbol{x}) \equiv (\forall y_1) \ldots (\forall y_m) Q(\boldsymbol{x}, \boldsymbol{y})$ for each $\boldsymbol{x}$. A is a Σ_n (respectively, Π_n) set if and only if for some Σ_n (respectively, Π_n) predicate P we have $A = \{ x \mid P(x) \}$. Let $\langle W_i^n \rangle_{i \in \mathbb{N}}$ be an acceptable indexing of the Σ_n sets [Rog67].

Subrecursive classes of functions. For each recursive $t\colon \mathbb{N} \to \mathbb{N}$, let $\mathrm{DTimeF}(t) = \{ \varphi_i \mid (\exists a)(\forall x)[\, \Phi_i(x) \leq a \cdot (t(x)+1) \,] \}$ and $\mathrm{DTIME}(t) = (\mathrm{DTimeF}(f))_{0\text{-}1}$. For each $k > 0$, we define $\mathcal{PF}^k = \mathrm{DTimeF}(\lambda x . |x|^k) =$ the class of functions computable in $O(n^k)$ time on a deterministic multi-tape TM, and $\mathcal{PF} = \cup_{k>0} \mathcal{PF}^k =$ the class of poly-time computable functions. Let $\mathcal{LS}low = \{ f \in \mathcal{PF}^1 \mid f$ is nondecreasing and unbounded$\}$. By standard results [CM81,LLR81], for each recursive, increasing, unbounded f, there is an $s \in \mathcal{LS}low$ that grows slower than the inverse of f in the sense that, for all x, $s(f(x)) \leq x$. $\mathcal{QF}_s^k = \mathrm{DTimeF}(\lambda x . |x|^k \cdot (\log |x|) \cdot s(|x|))$, for $k > 0$ and $s \in \mathcal{LS}low$. By standard results from [HS65,HU79], for all $k > 0$ and $s \in \mathcal{LS}low$, $(\mathcal{QF}_s^k - \mathcal{PF}^k)_{0\text{-}1} \neq \emptyset$.

Let $\mathcal{NP}$, $\mathcal{P}$, $\mathcal{CF}$, and $\mathcal{REG}$ respectively denote the classes of characteristic functions of NP, poly-time decidable, context free, and regular languages over $\{ \mathbf{0}, \mathbf{1} \}^*$. (Recall that we are punning between $\mathbb{N}$ as the natural numbers and $\mathbf{0}$-$\mathbf{1}$-strings.) $\mathcal{Z}^*$ (respectively, $\mathcal{U}^*$) denotes the class of 0–1 valued functions that are 0 (respectively, 1) almost everywhere.

$\mathcal{S} \subseteq \mathcal{R}$ is an *r.e. subrecursive class* when there is a programming system for $\mathcal{S}$. By standard results, $\mathcal{P}$, $\mathcal{P}^k$, $\mathcal{QF}_s^k$, $\mathcal{NP}$, $\ldots$ are each r.e. subrecursive classes.

Finite initial segments. For $f \in \mathcal{F}$ and $n \in \mathbb{N}$, $f|_n$ denotes the sequence $f(0), f(1), \ldots, f(n-1)$, the finite initial segment of f of length n. Clearly, $f|_0$ denotes the empty segment. Let SEG denote the set of all such finite initial segments; σ, with or without decorations, will range over SEG. If $\sigma = a_0, a_1, \ldots, a_{n-1}$ and $m \leq n$, then $\sigma|_m = a_0, a_1, \ldots, a_{m-1}$.

Inductive inference machines. An *inductive inference machine* [Gol67] is an algorithmic device that computes a possibly partial function from SEG to $\mathbb{N}$. $\mathbf{M}$, with or without decorations, ranges over such machines. Since SEG can be coded into $\mathbb{N}$, we can view an $\mathbf{M}$ as computing a partial function over $\mathbb{N}$. We say that $\mathbf{M}(f)$ *converges* to i (written: $\mathbf{M}(f)\downarrow = i$) if and only if, for all but finitely many n, $\mathbf{M}(f|_n) = i$; $\mathbf{M}(f)$ is undefined if no such i exists. The *point of convergence* of $\mathbf{M}$ on f is, if it exists, the smallest $m \in \mathbb{N}$ such that $\mathbf{M}(f|_m)$ is defined and, for all $n \geq m$, $\mathbf{M}(f|_n) = \mathbf{M}(f|_m)$.

The EX *and* EX* *identification criteria.* Suppose $f \in \mathcal{R}$ and $\mathcal{S} \subseteq \mathcal{R}$. We say that $\mathbf{M}$ EX-identifies f if and only if, for some i, $\mathbf{M}(f)\downarrow = i$ and i is a program for f (i.e., $\varphi_i = f$). We say that $\mathbf{M}$ EX-identifies $\mathcal{S}$ if and only if $\mathcal{S} \subseteq \mathrm{EX}(\mathbf{M}) =_{\mathrm{def}} \{\, f \in \mathcal{R} \mid \mathbf{M}$ EX-identifies $f\,\}$. EX denotes $\{\, \mathcal{S} \mid$ some $\mathbf{M}$ EX-identifies $\mathcal{S}\,\}$. The notion of EX-identification originated with Gold [Gol67]. Gold [Gol67] showed that every r.e. subrecursive class is in EX. We say that $\mathbf{M}$ EX*-identifies f if and only if, for some i, $\mathbf{M}(f)\downarrow = i$ and $\varphi_i =^* f$. We define $\mathrm{EX}^*(\mathbf{M})$ and EX* analogously to our definitions of $\mathrm{EX}(\mathbf{M})$ and EX. EX*-identification is due to Blum and Blum [BB75] who showed that $\mathrm{EX} \subsetneq \mathrm{EX}^*$.

The BC, BC^n, *and* BC* *identification criteria.* Suppose $f \in \mathcal{R}$ and $\mathcal{S} \subseteq \mathcal{R}$. For each $k \in \mathbb{N}$, we say that $\mathbf{M}$ BC^k-identifies f if and only if, for all but finitely many n, $\varphi_{\mathbf{M}(f|_n)} =^k f$. We say that $\mathbf{M}$ BC^k-identifies $\mathcal{S}$ if and only if $\mathcal{S} \subseteq \mathrm{BC}^k(\mathbf{M}) =_{\mathrm{def}} \{\, f \in \mathcal{R} \mid \mathbf{M}$ BC^k-identifies $f\,\}$. BC^k denotes $\{\, \mathcal{S} \mid$ some $\mathbf{M}$ BC^k-identifies $\mathcal{S}\,\}$. We usually write BC^0 as simply BC. We define the BC* criterion in the obvious fashion. BC-identification was first formalized by Bārzdiņš [Bar74]. Independently, Case and Smith [CS83] defined BC^m- and BC*-identification. Steel [CS83] showed $\mathrm{EX}^* \subseteq \mathrm{BC}$, Harrington and Case showed this inclusion to be proper [CS83], and Case and Smith [CS83] showed $\mathrm{BC}^0 \subsetneq \mathrm{BC}^1 \subsetneq \mathrm{BC}^2 \subsetneq \cdots \subsetneq \mathrm{BC}^*$. Moreover, as noted in §1, Harrington [CS83] showed that $\mathcal{R} \in \mathrm{BC}^*$.

3 Inseparability notions

Suppose A, B, and S are sets. We say that S *separates* B from A if and only if $B \subseteq S \subseteq \overline{A}$. (See Figure 1.)

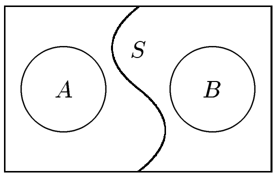

Fig. 1. S separates B from A.

Definition 1. Suppose A and $B \subseteq \mathbb{N}$.

(a) B is *Σ_n-inseparable* from A if and only if A and B are nonempty and disjoint, but no Σ_n set separates B from A.

(b) B is *effectively Σ_n-inseparable* from A if and only if A and B are nonempty and disjoint and there is a recursive f such that, for each i, $f(i) \in (W_i^n \cap A) \cup (\overline{W_i^n} \cap B)$, i.e., $f(i)$ witnesses that $B \subseteq W_i^n \subseteq \overline{A}$ fails. ◇

Definition 2. Suppose $R \subseteq \mathcal{F}^k \times \mathbb{N}^\ell$.

(a) R is *recursive* if and only if the characteristic function of R is a total recursive functional of type $\mathcal{F}^k \times \mathbb{N}^\ell \to \{\,0,1\,\}$.

(b) R is *arithmetical* if and only if either R is recursive or

$$R = \{\, (\boldsymbol{f}, \boldsymbol{x}) \mid (Q_1\, y_1) \ldots (Q_m\, y_m)[\, S(\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{y})\,]\,\} \tag{1}$$

where each Q_i is either $\exists$ or $\forall$ and where $S \subseteq \mathcal{F}^k \times \mathbb{N}^{\ell+m}$ is recursive. (**N.B.** All the quantifiers in (1) are numeric.)

(c) R is in $\Sigma_n^{(\mathrm{fn})}$ if and only if R is recursive or R is expressible as in (1) with the quantifiers in Σ_n form.

(d) R is in $\Pi_n^{(\mathrm{fn})}$ if and only if R's complement is in $\Sigma_n^{(\mathrm{fn})}$. $\diamond$

Indexings. For each k, ℓ, and n, let $\langle W_i^{(\mathrm{fn}),k,\ell,n} \rangle_{i \in \mathbb{N}}$ be an acceptable indexing of the class of all $R \subseteq \mathcal{F}^k \times \mathbb{N}^\ell$ in $\Sigma_n^{(\mathrm{fn})}$. For each i, let $\mathcal{W}_i^n = W_i^{(\mathrm{fn}),1,0,n}$.

Next we introduce the higher-type inseparabilities needed for our results.

Definition 3. Suppose $\mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{F}$.

(a) $\mathcal{B}$ is $\Sigma_n^{(\mathrm{fn})}$-*inseparable* from $\mathcal{A}$ if and only if $\mathcal{A}$ and $\mathcal{B}$ are nonempty and disjoint, but no $\Sigma_n^{(\mathrm{fn})}$ set separates $\mathcal{B}$ from $\mathcal{A}$.

(b) $\mathcal{B}$ is *effectively* $\Sigma_n^{(\mathrm{fn})}$-*inseparable* from $\mathcal{A}$ if and only if $\mathcal{A}$ and $\mathcal{B}$ are nonempty and disjoint and there is a recursive f such that for each i, $\varphi_{f(i)} \in (\mathcal{W}_i^n \cap \mathcal{A}) \cup (\overline{\mathcal{W}_i^n} \cap \mathcal{B})$. $\diamond$

In the journal version of this paper we establish several sufficient conditions for $\Sigma_2^{(\mathrm{fn})}$-inseparability. From these sufficient conditions we can deduce:

Corollary 4. *Suppose that $k > 0$ and $s \in \mathcal{LS}low$.*

(a) $(\mathcal{QF}_s^k - \mathcal{PF}^k)_{0\text{-}1}$ is effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable from $\mathcal{Z}^*$.

(b) $(\mathcal{CF} - \mathcal{REG})$ is effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable from $\mathcal{U}^*$.

(c) If $\mathrm{P} \neq \mathrm{NP}$, then $(\mathcal{NP} - \mathcal{P})$ is effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable from $\mathcal{Z}^*$.

4 Sufficient conditions theorems

In the following, think of $\mathcal{A}$ as some set of very modest functions (e.g., $\mathcal{Z}^*$ above), $\mathcal{B}$ as some set of immodest functions, and G as some set of "good" programs such that no finite variant of a member of $\mathcal{B}$ has a program in G.[10]

Theorem 5, our first sufficient condition theorem, provides us with our information deficiency corollaries (Corollaries 6 through 8).

Theorem 5. *Suppose that: (i) $\mathcal{B}$ is $\Sigma_2^{(\mathrm{fn})}$-inseparable from $\mathcal{A}$. (ii) G is a Σ_1-set such that $\mathcal{FV}(\mathcal{B}) \cap \{\, \varphi_i \mid i \in G \,\} = \emptyset$. (iii) $\mathbf{M}$ is an IIM such that $\mathcal{B} \subseteq \mathrm{BC}^*(\mathbf{M})$. Then there is an $f \in \mathcal{A}$ such that for all but finitely many n, $\mathbf{M}(f|_n) \notin G$.*

[10] For example, the members of G may run efficiently and/or be easy to prove things about.

Proof. Since G is a Σ_1 set, we know that there is a recursive R_G such that $G = \{\, x \mid (\exists m) R_G(x,m) \,\}$. Consider $\mathcal{S} = \{\, f \in \mathcal{F} \mid (\overset{\infty}{\forall} n)[\, \mathbf{M}(f|_n) \notin G \,] \,\} = \{\, f \in \mathcal{F} \mid (\exists n_0)(\forall n > n_0)(\forall m)[\, \neg R_G\,(\mathbf{M}(f|_n), m) \,] \,\}$. Thus, $\mathcal{S} \in \Sigma_2^{(\mathrm{fn})}$. Also, by (ii) and (iii) it follows that $\mathcal{B} \subseteq \mathcal{S}$. Now suppose by way of contradiction that, for all $f \in \mathcal{A}$, we have: $(\overset{\infty}{\exists} n)[\, \mathbf{M}(f|_n) \in G \,]$. Clearly, $\mathcal{A} \cap \mathcal{S} = \emptyset$. Therefore, $\mathcal{S}$ is a $\Sigma_2^{(\mathrm{fn})}$-set that separates $\mathcal{B}$ from $\mathcal{A}$, contradiction. □

The next three corollaries involve provability and PA, Peano Arithmetic [Men97]. We write $\vdash$ for the provability relation and $\nvdash$ for 'does *not* prove.' The following predicates are expressible in Peano Arithmetic (and herein we do not distinguish between expressions in Peano Arithmetic and expressions in the metalanguage).

$$P_k(i) \;\equiv_{\mathrm{def}}\; (\exists c)(\forall x)[\, \Phi_i(x) \leq c \cdot (|x|+1)^k \,]. \quad S_k(i) \;\equiv_{\mathrm{def}}\; (\forall x)[\, \Phi_i^{\mathrm{WS}}(x) \leq k \,].$$
$$P_k^*(i) \;\equiv_{\mathrm{def}}\; (\exists j \mid \varphi_j =^* \varphi_i)[\, P_k(j) \,]. \quad P^*(i) \;\equiv_{\mathrm{def}}\; (\exists j \mid \varphi_j =^* \varphi_i)(\exists k)[\, P_k(j) \,].$$
$$REG^*(i) \;\equiv_{\mathrm{def}}\; (\exists k)(\exists j \mid \varphi_j =^* \varphi_i)[\, S_k(j) \,].$$

N.B. Each of Corollaries 6, 7, and 8 remains true if PA is replaced by any true and computably axiomatized theory [Men97] extending the language of PA. Such theories, including PA itself, should be thought of as safe, algorithmic extractors of information: the safety is they prove only true sentences; and, since they are computably axiomatized, there is an associated automatic theorem prover, i.e., the set of theorems is r.e. [Men97].

Corollary 6. *Suppose that* $k > 0$, $s \in \mathcal{LS}low$, *and* $\mathrm{BC}^*(\mathbf{M}) \supseteq \mathcal{QF}_s^k$. *Then there is an* $f \in \mathcal{Z}^*$ *such that, for all but finitely many* n, $\mathrm{PA} \nvdash P_k^*(\mathbf{M}(f|_n))$.

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{QF}_s^k - \mathcal{PF}^k)$, and $G = \{\, i \mid \mathrm{PA} \vdash P_k^*(i) \,\}$. Now, applying Corollary 4(a) and Theorem 5, we are done. □

Interpretation. Let $\mathbf{M}$ and f be as in Corollary 6.[11] Then it must be the case that, for all but finitely many n, the program $\mathbf{M}(f|_n)$ computes a (finite variant of a) function f that is almost everywhere zero. Of course *some* program computes f in *linear time.* Yet, even so, for sufficiently large n, the programs $\mathbf{M}(f|_n)$ are so information deficient that PA fails to prove of them that they compute a finite variant of something (like f) that has *some* program running in k-degree polynomial time. Analogous interpretive remarks apply to the next two corollaries (which, by the way, have proofs similar to that just above for Corollary 6).

Corollary 7. *Suppose* $\mathrm{BC}^*(\mathbf{M}) \supseteq \mathcal{CF}$. *Then there is an* $f \in \mathcal{U}^*$ *such that, for all but finitely many* n, $\mathrm{PA} \nvdash REG^*(\mathbf{M}(f|_n))$.

Corollary 8. *Suppose* $\mathrm{BC}^*(\mathbf{M}) \supseteq \mathcal{NP}$ *and that* $\mathrm{P} \neq \mathrm{NP}$. *Then there is an* $f \in \mathcal{Z}^*$ *such that, for all but finitely many* n, $\mathrm{PA} \nvdash P^*(\mathbf{M}(f|_n))$.

[11] As noted in §1, an allowed special case is that $\mathbf{M}$ actually EX-learns $\mathcal{QF}_\alpha^k$.

Theorem 9, our second sufficient condition theorem, provides us with complexity deficiency corollaries (Corollaries 11 through 13).

Theorem 9. *Suppose that: (i)* $\mathcal{B}$ *is* $\Sigma_2^{(\mathrm{fn})}$*-inseparable from* $\mathcal{A}$*. (ii)* G *is a* Π_2*-set such that* $\mathcal{FV}(\mathcal{B}) \cap \{\, \varphi_i \mid i \in G \,\} = \emptyset$*. (iii)* $\mathbf{M}$ *is an IIM such that (a)* $\mathcal{A} \subseteq \mathrm{EX}^*(\mathbf{M})$ *and (b)* $\mathcal{B} \subseteq \mathrm{EX}^*(\mathbf{M})$*. Then there is an* $f \in \mathcal{A}$ *such that* $\mathbf{M}(f) \notin G$*.*

Scholium 10. The fact that $G \in \Pi_2$ in Theorem 9 does not provide as much generality as one might hope. Here is why. It is a well-worn observation that if $\mathcal{C}$ is closed under finite variants and P is a Σ_2 set such that $\mathcal{C} \subseteq \{\, \varphi_i \mid i \in P \,\} \subseteq \mathcal{F}$, then there is an r.e. set P' such that $\mathcal{C} \subseteq \{\, \varphi_i \mid i \in P \,\} \subseteq \{\, \varphi_i \mid i \in P' \,\} \subseteq \mathcal{F}$. It is a minor variation on this observation that if hypotheses (ii) and (iii) of Theorem 9 hold, then there is a Π_1-set G' such that $\mathcal{A} \subseteq \{\varphi_i \mid i \in G\} \subseteq \{\, \varphi_i \mid i \in G' \,\} \subseteq (\mathcal{PR} - \mathcal{B})$. Hence, G in Theorem 9 might as well be Π_1—which is what it is in our applications of this theorem. ◇

As was mentioned in §1, the following corollary of Theorem 9 provides a weak special case of our strong complexity-deficiency result (Theorem 20) for $(\mathcal{PF}^k, \mathcal{QF}^k_\alpha)$: the quantifier *order* between the $f \in \mathcal{Z}^*$ and the k-degree polynomial time bounds is weakened and the for-all-but-finitely-many-inputs-x quantifier is weakened to exists-n-distinct-inputs.

Corollary 11. *Suppose* $a, k, n > 0$*,* $s \in \mathcal{LS}low$*, and* $\mathrm{EX}^*(\mathbf{M}) \supseteq \mathcal{QF}^k_s$*. Then there is an* $f \in \mathcal{Z}^*$ *such that, for some* i*,* $\mathbf{M}(f){\downarrow} = i$*, but there are distinct* $x_0, \ldots, x_{n-1}$ *such that for each* $j < n$*,* $\Phi_i(x_i) > a \cdot (|x_j| + 1)^k$*.*

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{QF}^k_s - \mathcal{PF}^k)$, and $G = \{\, i \mid (\forall x_0)(\forall x_1 > x_0) \ldots (\forall x_{n-1} > x_{n-2})(\exists j < n)[\Phi_i(x_j) \leq a \cdot (|x_j| + 1)^k \,] \,\}$. Now, applying Corollary 4(a) and Theorem 9, we are done. □

As mentioned in §1, the next two corollaries seem difficult to establish by aggressive diagonalization techniques. It is open for each as to whether the quantifier on the inputs to the programs i can be strengthened.

Corollary 12. *Suppose* $\mathrm{EX}^*(\mathbf{M}) \supseteq \mathcal{CF}$ *and* $k, n > 0$*. Then there is an* $f \in \mathcal{U}^*$ *such that, for some* i*,* $\mathbf{M}(f){\downarrow} = i$*, but, there are distinct* $x_0, \ldots, x_{n-1}$ *such that for each* $j < n$*,* $\Phi_i^{\mathrm{WS}}(x_j) > k$*.*

Proof. Let $\mathcal{A} = \mathcal{U}^*$, $\mathcal{B} = (\mathcal{CF} - \mathcal{REG})$, and $G = \{\, i \mid (\forall x_0)(\forall x_1 > x_0) \ldots (\forall x_{n-1} > x_{n-2})(\exists j < n)[\Phi_i^{\mathrm{WS}}(x_j) \leq k \,] \,\}$. Now, applying Corollary 4(b) and Theorem 9, we are done. □

Interpretation. Suppose $\mathbf{M}$ EX^*-identifies $\mathcal{CF}$.[12] Then Corollary 12 implies that there are members of $\mathcal{U}^*$ for which $\mathbf{M}$ infers programs that use arbitrarily large (but finite) amounts of workspace on arbitrarily large (but finite) sets of inputs. Thus $\mathbf{M}$ is quite far from inferring space efficient programs for easy

[12] As noted in §1, an allowed special case is that $\mathbf{M}$ actually EX-learns $\mathcal{CF}$.

members of $\mathcal{REG}$, and members of $\mathcal{REG}$ have programs that use no workspace at all.

The next corollary has a similar proof to that for Corollary 12.

Let φ^{ND} be based on a natural programming system of non-deterministic, multi-tape Turing machines for accepting sets.

Corollary 13. *Suppose* P $\neq$ NP. *Suppose* **M** EX**-identifies* $\mathcal{NP}$ *using polynomial-time (deterministic and nondeterministic)* φ^{ND}*-programs*[13], q *is a polynomial, and* $n > 0$. *Then there is an* $f \in \mathcal{Z}^*$ *such that, for some* i, $\mathbf{M}(f){\downarrow} = i$, *but, there are distinct* $x_0, \ldots, x_{n-1}$ *such that for each* $j < n$, φ^{ND}*-program* i *on input* x_j *runs* non-*deterministically and, in fact, has a computation tree of more than* $q(|x_j|)$ *paths.*

Interpretation. Suppose **M** EX*-identifies $\mathcal{NP}$ using polynomial-time (deterministic and nondeterministic) φ^{ND}-programs. Then the above corollary implies that there are members of $\mathcal{Z}^*$ for which **M** infers programs that employ arbitrarily polynomially much unpleasant non-determinism on arbitrarily large (but finite) sets of inputs. Thus **M** is far from inferring efficient programs for easy members of P.

5 A few more diagonalization and structural tools

Here we state a few more tools for the proofs of the results in the next three sections. These depend on a few special features of our programming system φ and its associated complexity measure Φ. The details of these features are mostly straightforward and are omitted here, but can be found in Chapter 3 of [RC94].

Proposition 14 (The cost of simulations, [RC94] Theorem 3.6). *Suppose* $S, T \colon \mathbb{N}^3 \to \mathbb{N}$ *are given by:*

$$S(i,x,t) \;=\; \begin{cases} \varphi_i(x), & \text{if } \Phi_i(x) \le |t|; \\ 0, & \text{otherwise.} \end{cases} \qquad T(i,x,t) \;=\; \begin{cases} 1, & \text{if } \Phi_i(x) \le |t|; \\ 0, & \text{otherwise.} \end{cases}$$

Then S *and* T *are computable in time* $O(|x| + (|i|+1)\cdot(|t|\cdot\log|t|+1))$.

Proposition 15. *Suppose that* $m, n \ge 1$. *In the following* i, j, *and* k *range over* $\mathbb{N}$, *and* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *over* $\mathbb{N}^m$ *and* $\mathbb{N}^n$, *respectively.*

(a) COMPLEXITY-BOUNDED CONDITIONALS, [RC94] LEMMA 3.14. *There is a poly-time computable* if_m *and an* $a_m \in \mathbb{N}$ *such that, for all* i, j, k, *and* $\boldsymbol{x}$*:*

$$\varphi_{\mathrm{if}_m(i,j,k)}(\boldsymbol{x}) = \begin{cases} \varphi_j(\boldsymbol{x}), & \text{if } \varphi_i(\boldsymbol{x}) > 0; \\ \varphi_k(\boldsymbol{x}), & \text{if } \varphi_i(\boldsymbol{x}) = 0; \\ \uparrow, & \text{if } \varphi_i(\boldsymbol{x}){\uparrow}. \end{cases}$$

$$\Phi_{\mathrm{if}_m(i,j,k)}(\boldsymbol{x}) \le \begin{cases} \Phi_i(\boldsymbol{x}) + \Phi_j(\boldsymbol{x}) + a_m\cdot(|\boldsymbol{x}|+1), & \text{if } \varphi_i(\boldsymbol{x}) > 0; \\ \Phi_i(\boldsymbol{x}) + \Phi_k(\boldsymbol{x}) + a_m\cdot(|\boldsymbol{x}|+1), & \text{if } \varphi_i(\boldsymbol{x}) = 0; \\ \infty, & \text{if } \varphi_i(\boldsymbol{x}){\uparrow}. \end{cases}$$

[13] *Note:* NP is trivially in EX as witnessed by some $\mathbf{M}'$ also outputting φ^{ND}-programs.

(b) COMPLEXITY-BOUNDED SELF-REFERENCE, [RC94] THEOREM 4.6. *There is a poly-time computable $r_{m,n}$ and an $a_{m,n} \in \mathbb{N}$ such that, for all i, $\boldsymbol{x}$, and $\boldsymbol{y}$:*

$$\varphi^n_{r_{m,n}(i,\boldsymbol{x})}(\boldsymbol{y}) = \varphi_i^{m+n+1}(r_{m,n}(i,\boldsymbol{x}),\boldsymbol{x},\boldsymbol{y}).$$

$$\Phi^n_{r_{m,n}(i,\boldsymbol{x})}(\boldsymbol{y}) \le \Phi_i^{m+n+1}(r_{m,n}(i,\boldsymbol{x}),\boldsymbol{x},\boldsymbol{y}) + a_{m,n}\cdot(|\boldsymbol{x}|+|\boldsymbol{y}|+1).$$

Proposition 16 (Low complexity delayed enumeration, [RC94] Theorem 7.1). *For each $m > 0$ and $s \in \mathcal{LS}low$, there is a linear time computable function $\mathrm{rng}_{m,s}$ such that, for all i with φ_i total and all $\boldsymbol{w} \in \mathbb{N}^m$, there is a strictly increasing sequence of numbers y_0, y_1, $y_2, \ldots$ such that*

(a) for each $y \in \{0,\ldots,y_0-1\}$, $\mathrm{rng}_{m,s}(i,\boldsymbol{w},y)=0$, and

(b) for each x and each $y \in \{y_x \ldots, y_{x+1}-1\}$, $\mathrm{rng}_{m,s}(i,\boldsymbol{w},y) = 1+\varphi_i(\boldsymbol{w},x)$, and moreover, $|\varphi_i(\boldsymbol{w},x)| \le s(|\max(i,\boldsymbol{w},y)|)$.

Convention: For each m, let rng_m denote $\mathrm{rng}_{m,s}$ where $s = \lambda n \boldsymbol{.} \log^{(2)}(n)$.

6 Negative, almost everywhere results for EX^* and BC^0

For simplicity of proof exposition we begin with two theorems essentially announced in [Che81] and based on a suggestion of Sipser for the EX case. In [Che81] it was merely asserted without proof that the constructions could be done in polytime. At that time, the machinery to supply really convincing proofs of these results was not yet available (at least to us). For the present paper we have the needed machinery for this and for proving the main theorems of this section (Theorems 19 and 20) which follow. These main theorems provide considerably tighter complexity bounds *and* stronger quantifier order.

Although $\mathrm{EX}^* \subsetneq \mathrm{BC}^0$, Theorems 17 through 20 handle separately the cases of EX^* and BC^0. This is because, if $\mathbf{M}$ witnesses that a class is in EX^*, the same $\mathbf{M}$ need not witness the class is in BC^0: the latter can require a different machine $\mathbf{M}'$.

Theorem 17. *Suppose that* $\mathrm{BC}^0(\mathbf{M}) \supseteq \mathcal{PF}$. *Then for each polynomial q, there is an $f \in \mathcal{Z}^*$ such that $(\overset{\infty}{\forall} n)(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f|_n)}(x) > q(|x|)]$.*

Theorem 18. *Suppose that* $\mathrm{EX}^*(\mathbf{M}) \supseteq \mathcal{PF}$. *Then, for each polynomial q, there is an $f \in \mathcal{Z}^*$ such that $(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f)}(x) > q(|x|)]$.*

Proof (of Theorem 17). Fix a polynomial q. *Terminology:* We say that p is *available* at w if and only if $\Phi_p(w) \le q(|w|)$. For each σ, define the set

$$\mathrm{Candidates}(\sigma) = \left\{ p \;\middle|\; \begin{array}{l} \text{for some } n \le |\sigma|+1,\ p = \mathbf{M}(\sigma|_n) \text{ and, for each } w \in \\ \mathrm{dom}(\sigma), \text{ if } p \text{ is available at } w, \text{ then } \varphi_p(w) = \sigma(w) \end{array} \right\}.$$

Let d be a φ-program such that, for all e and x,

$$\varphi_d(e,x) = \begin{cases} \uparrow, & \text{if, for some } w < x,\ \ \varphi_e(w)\uparrow; \\ [p_1,\ldots,p_k] & \text{otherwise, where } \{p_1 < \cdots < p_k\} = \\ & \qquad \mathrm{Candidates}(\varphi_e|_x). \end{cases}$$

Intuitively, when $\varphi_d(e,x)\downarrow = [p_1, \ldots, p_k]$, then $p_1, \ldots, p_k$ is a list of conjectures that $\mathbf{M}$ makes on φ_e that are candidates for diagonalization. Now let u be a φ-program such that, for all e and y, $\varphi_u(e,y) =$

$$\begin{cases} 0, & \text{if (i) } \mathrm{rng}_1(d,e,y) = 0 \text{ or } \mathrm{rng}_1(d,e,y) = 1 + [p_1, \\ & \quad \ldots, p_k], \text{ but none of the } p_i\text{'s is available at } y; \\ 1 \dot{-} S(p,y,\mathbf{0}^{q(|y|)}), & \text{(ii) otherwise, where } p \text{ is the least } p_i \text{ available at } y. \end{cases}$$

Terminology: If (ii) holds above for a particular input e and y, we then say that the p is *canceled* for e at y. Since S, rng_1, and the availability predicate are all poly-time computable, it is straightforward that φ_u is poly-time computable. So, without loss of generality, we assume that Φ_u is polynomially bounded. Thus by Proposition 15(b), there is a φ-program e_0 and a polynomial q_0 such that, for all y, $\varphi_{e_0}(y) = \varphi_u(e_0,y)$ and $\Phi_{e_0}(y) \le q_0(|y|)$. Hence, $\varphi_{e_0} \in \mathcal{PF}$. Thus, $\lambda x\,.\,\varphi_d(e_0,x)$ is total.

Claim 1: No p is canceled for e_0 infinitely many times. *Proof:* Suppose p is canceled for e_0 on some number. Then it follows by the definition of φ_d that, for all but finitely many x, p is not on the list output by $\varphi_d(e_0,x)$. Hence, by the definition of rng_1, for all but finitely many y, $p \notin \{p_1^y, \ldots, p_{k_y}^y\}$ where $1 + [p_1^y, \ldots, p_{k_y}^y] = \mathrm{rng}_1(d,e_0,y)$. Thus, by the definition of u, Claim 1 follows.

Claim 2: Suppose $\varphi_{\mathbf{M}(\varphi_{e_0}|_n)} = \varphi_{e_0}$. Then $\mathbf{M}(\varphi_{e_0}|_n)$ is never canceled for e_0 on any y. *Proof:* If p is canceled for e_0 on some y, then $\varphi_p \neq \varphi_{e_0}$. Hence the claim follows.

Claim 3: Suppose $\varphi_{\mathbf{M}(\varphi_{e_0}|_n)} = \varphi_{e_0}$. Then it is the case that, for all but finitely many y, $\Phi_{\mathbf{M}(\varphi_{e_0}|_n)}(y) > q(|y|)$. *Proof:* Suppose by way of contradiction that $\mathbf{M}(\varphi_{e_0}|_n)$ is available for e_0 on infinitely many y. Then it follows by standard arguments that $\mathbf{M}(\varphi_{e_0}|_n)$ is eventually canceled for e_0 on some y, contradicting Claim 2. Hence, the present claim follows.

Since $\mathbf{M}$ BC-identifies φ_{e_0}, it follows from Claim 3 that, for all but finitely many n and all but finitely many y, $\Phi_{\mathbf{M}(\varphi_{e_0}|_n)}(y) > q(|y|)$.

It follows from Claims 1 and 2 and the BC-identification of φ_{e_0} by $\mathbf{M}$ that there are only finitely many y on which any p is canceled for e_0. Thus, by the definition of u, $\varphi_{e_0} \in \mathcal{Z}^*$. Therefore, the theorem follows. □

By a more delicate choice of complexity functions and a correspondingly more careful complexity analysis of the proofs of the previous two theorems, we can obtain the following two improvements which are our main theorems of the present section.

Theorem 19. *Suppose* $\mathbf{M}$ BC^0*-identifies* $\mathcal{QF}_s^k$*, where* $k \ge 1$ *and* $s \in \mathcal{L}Slow$*. Then there is an* $f \in \mathcal{Z}^*$ *such that* $(\forall a)(\overset{\infty}{\forall} n)(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f|_n)}(x) > a \cdot (|x|+1)^k]$.

Theorem 20. *Suppose* $\mathbf{M}$ EX^**-identifies* $\mathcal{QF}_s^k$*, where* $k \ge 1$ *and* $s \in \mathcal{L}Slow$*. Then there is an* $f \in \mathcal{Z}^*$ *such that* $(\forall a)(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f)}(x) > a \cdot (|x|+1)^k]$.

Interpretation. Let $\mathbf{M}$, k, s and f be as in Theorem 19.[14] Then for all most all n, the program $\mathbf{M}(f|_n)$ must compute f, an almost everywhere zero function, yet the run time of this program is almost everywhere worse than any degree-k polynomial in the size of the input. This is a profound failure of $\mathbf{M}$ to infer anything like asymptotically optimal programs for even easy members of $\mathcal{PF}^k$. Similar remarks apply to Theorem 20.

Proof (of Theorem 19). Let $s' \in \mathcal{LS}low$ be such that $\lim_{n\to\infty}(s'(n))^2/s(n) = 0$. The construction is identical to the one given in the proof of Theorem 17 with q replaced by $\lambda n \centerdot s'(n)\cdot(n+1)^k$ and rng_1 replaced by $\mathrm{rng}_{1,s'}$. Without loss of generality we assume that, for all x, $s(x)$ and $s'(x) > 0$.

Let us consider the cost of computing the function computed by φ_u. Recall that p is available at y if and only if $\Phi_p(y) \leq s'(|y|)\cdot(|y|+1)^k$ if and only if $T(p, y, \mathbf{0}^{s'(|y|)\cdot(|y|+1)^k}) = 1$. By standard time-constructibility results [HU79], given y (in dyadic representation), constructing a string of $\mathbf{0}$'s of length $s'(|y|)\cdot(|y|+1)^k$ can be done in time $O(s'(|y|)\cdot(|y|+1)^k)$. Hence by Proposition 14, testing, for a given p and y, whether p is available at y can be done in $O((|p|+1)\cdot(|y|+1)^k\cdot(1+\log|y|)\cdot s'(|y|))$ time.

Recall from Proposition 16 that $\mathrm{rng}_{1,s'}$ is linear time computable *and*, for all d, e, and y, $|\mathrm{rng}_1(d,e,y)| \leq s'(\max(|d|,|e|,|y|))$. Consequently it follows that when $\mathrm{rng}_1(d,e,y) = 1+[p_1,\ldots,p_m]$, we have that each of m, $|p_1|,\ldots,|p_m|$ is less that $s'(\max(|d|,|e|,|m|))$. Hence it follows that searching for the least i such that p_i is available at y can be done in $O((s'(\max(|d|,|e|,|y|)))^2\cdot(|y|+1)^k\cdot(1+\log|y|))$ time. Since, by Proposition 14, computing $S(p,y,\mathbf{0}^{s'(|y|)\cdot(|y|+1)^k})$ has the same complexity as testing whether p is available at y, it follows from Proposition 15(a) that φ_u on input (e,y) is computable in $O((s'(\max(|e|,|y|)))^2\cdot(|y|+1)^k\cdot(1+\log|y|))$ time. (Since d is a constant, we can absorb its contribution into the constant hidden by the O.) Without loss of generality, we can assume that Φ_u has such an upper bound. Therefore, by Proposition 15(b), there is an e_0 such that, for all y, $\varphi_{e_0} = \varphi_u(e_0,y)$ and $\Phi_{e_0}(y)$ has an upper bound which is in $O\left((s'(\max(|e_0|,|y|)))^2\cdot(|y|+1)^k\cdot(1+\log|y|)+(|y|+1)\right)$ which by some algebra is contained in $O(|y|^k(\log|y|)\cdot s(|y|))$. It thus follows that $\mathbf{M}$ BC^0 identifies φ_{e_0}. Now the rest of the proof follows the argument given for Theorem 17. □

7 Infinitely often results for $\mathbf{BC}^m$

In this section we deal with the criteria BC^m, especially for $m \geq 1$. The stronger version of the $m = 0$ case was handled in Theorem 19. It is technically surprising that the $m \geq 1$ cases provably do not permit as strong a quantifier on the inputs x as does the $m = 0$ case.

Theorem 21. *Suppose* $\mathbf{M}$ BC^m*-identifies* $\mathcal{QF}^k_s$*, where* $k \geq 1$ *and* $s \in \mathcal{LS}low$*. Then there is an* $f \in \mathcal{Z}^*$ *such that* $(\forall a)(\overset{\infty}{\forall} n)(\overset{\infty}{\exists} x)[\Phi_{\mathbf{M}(f|_n)}(x) > a\cdot(|x|+1)^k]$.

[14] As noted in §1, an allowed special case is that $\mathbf{M}$ actually EX-learns $\mathcal{QF}^k_s$.

The proof is a straightforward modification of the proof of Theorem 19; however, to prove Theorem 21 we need to diagonalize over $m+1$ points at once. It is *not* possible to replace the $(\stackrel{\infty}{\exists} x)$ in Theorem 21 with an $(\stackrel{\infty}{\forall} x)$ as shown by:

Theorem 22. *There is an* $\mathbf{M}$ *that both: (i)* BC^1*-identifies* $\mathcal{PF}$ *using programs having polynomial-bounded run times, and (ii)* EX*-identifies* $\mathcal{PF}^1$ *and moreover, for each* $f \in \mathcal{PF}^1$*, there is a constant* c_f *such that* $(\stackrel{\infty}{\exists} x)[\,\Phi_{\mathbf{M}(f)}(x) \le c_f \cdot |x|\,]$.

8 Positive, almost everywhere results for BC*

This section contains our strongest positive results. After the theorem, we state informally a generalization.

Theorem 23. *There is an IIM* $\mathbf{M}_*$ *that* BC^**-identifies* $\mathcal{PF}$ *with all outputs running in polynomial time and such that:*

(a) For each $k \ge 1$ *and each* $f \in \mathcal{PF}^k$, $(\stackrel{\infty}{\forall} n)[\,\Phi_{\mathbf{M}_*(f|_n)} \in O(\lambda x \,.\, |x|^k)\,]$.

(b) Moreover, $\mathbf{M}_*$ EX*-identifies* $\mathcal{PF}^1$.

Interpretation. In contrast to Theorems 17 through 21, the above result is quite a surprise. Not only does the $\mathbf{M}_*$ of the theorem BC*-infer programs that have $O(n^k)$ run-time bounds for each member of $\mathcal{PF}^k$ for every k, but for each $f \in \mathcal{PF}^1$, $\mathbf{M}_*$ also *syntactically* converges to a program for this f that has an $O(n)$ run-time bound. However, as noted in §1, Corollary 6 applies to $\mathbf{M}_*$ of the above theorem. Hence, for each $\ell \ge 1$, there is an $f \in \mathcal{Z}^*$ such that $\mathbf{M}_*$ EX-identifies f and the perfectly correct φ-program $\mathbf{M}_*(f)$ has a *linear run-time bound* (by Theorem 23); *however*, by Corollary 6, $\mathbf{M}_*(f)$ is so *information deficient* that PA fails to prove even that it computes a finite variant of something having *some* program running in ℓ-degree polynomial time. Thus part of the price $\mathbf{M}_*$ pays for the asymptotically optimal run times of its output programs is that these programs, even on some easy functions, must necessarily be highly information deficient.

A generalization of Theorem 23 also holds by a similar proof. In the generalization one introduces an arbitrary $j \ge 1$ but requires $k \ge j$; then part (b) becomes $\mathbf{M}_*$ EX-identifies $\mathcal{PF}^j$ with all but finitely many of $\mathbf{M}_*$'s conjectures running in time $O(n^j)$.

Acknowledgments. Special thanks go to Prof. Dr. Klaus Ambos-Spies for some very helpful suggestions and observations. Grant support was received by J. Case from NSF grant CCR-0208616, by S. Jain from NUS grant R252-000-127-112, and by J. Royer from NSF grant CCR-0098198.

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** (1974), 82–88.

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

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

[CCJ01] J. Case, K. Chen, and S. Jain, *Costs of general purpose learning*, Theoretical Computer Science **259** (2001), 455–473.

[Che81] K. Chen, *Tradeoffs in machine inductive inference*, Ph.D. thesis, SUNY at Buffalo, 1981.

[CLRS01] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*, 2 ed., MIT Press, 2001.

[CM81] P. Chew and M. Machtey, *A note on structure and looking back applied to the relative complexity of computable functions*, Journal of Computer and System Sciences **22** (1981), 53–59.

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

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

[HS65] J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Transactions of the American Mathematical Society **117** (1965), 285–306.

[HU79] J. Hopcroft and J. Ullman, *Introduction to automata theory languages and computation*, Addison-Wesley Publishing Company, 1979.

[Jon97] N. Jones, *Computability and complexity from a programming perspective*, MIT Press, 1997.

[JORS99] S. Jain, D. Osherson, J. Royer, and A. Sharma, *Systems that learn: An introduction to learning theory*, second ed., MIT Press, Cambridge, Mass., 1999.

[Lad75] R. Ladner, *On the structure of polynomial time reducibility*, Journal of the ACM **22** (1975), 155–171.

[LLR81] L. Landweber, R. Lipton, and E. Robertson, *On the structure of sets in NP and other complexity classes*, Theoretical Computer Science **15** (1981), 181–200.

[LV97] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications, second edition*, Springer-Verlag, 1997.

[Men97] E. Mendelson, *Introduction to mathematical logic*, fourth ed., Chapman & Hall, London, 1997.

[RC94] J. Royer and J. Case, *Subrecursive programming systems: Complexity & succinctness*, Birkhäuser, 1994.

[Reg88] K. Regan, *The topology of provability in complexity theory*, Journal of Computer and System Sciences **36** (1988), 384–432.

[Rog67] H. Rogers, *Theory of recursive functions and effective computability*, McGraw Hill, New York, 1967, Reprinted. MIT Press. 1987.

[Sch82] U. Schöning, *A uniform approach to obtain diagonal sets in complexity classes*, Theoretical Computer Science **18** (1982), 95–103.

[Sch85] D. Schmidt, *The recursion-theoretic structure of complexity classes*, Theoretical Computer Science **38** (1985), 143–156.