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

John Case

Dept. of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586 USA. Email: case@cis.udel.edu.

Keh-Jiann Chen

Institute of Information Science, Academia Sinica, Nankang 115, Taipei, Taiwan, ROC. Email: kchen@iis.sinica.edu.tw.

Sanjay Jain

School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543, Republic of Singapore. Email: sanjay@comp.nus.edu.sg.

Wolfgang Merkle

Universität Heidelberg, Mathematisches Institut, Im Neuenheimer Feld 294, D-69120 Heidelberg, Germany. Email: merkle@math.uni-heidelberg.de.

James S. Royer

Dept. of Elec. Eng. and Computer Science, Syracuse University, Syracuse, NY 13244 USA. Email: 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. For these results, the safe, algorithmic methods of information extraction will be by proofs in arbitrary, true, computably axiomatizable extensions of Peano Arithmetic.

Key words: Computational learning theory; Applications of computability theory

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, as from [1, §21.4], be a *very slow growing*, linear time computable function $\leq$ an inverse of Ackermann's function; let $\mathcal{C}_1$ be DTIME($n \cdot (\log n) \cdot \alpha(n)$); and let $\mathcal{C}_0$ be DTIME(n). [1] These classes have long been known to separate [2,3]. 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 of syntactically converging learning in the limit is called EX-*learning* (or EX-*identification*) [4–7]. Let $\mathcal{Z}^*$ be the CCFL for precisely the *finite* languages. Clearly, $\mathcal{Z}^*$ is an especially simple, proper subclass of our example $\mathcal{C}_0$. Two of our main theorems (Theorems 27 and 28 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 27 and 28, on which this example is based, are proved by delayed diagonalization (or slowed simulation) [8,9] with cancellation [10] (or zero injury), complexity-bounded self-reference [9], and careful subrecursive programming [9].

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 [2,3], and it is straightforward that some learning device EX-

[1] 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. DTimeF($t(n)$) denotes the set of *functions* over strings computable by this same class of machines within $O(t(n))$ time.

learns this $\mathcal{C}_0$ outputting only conjectures that run in k-degree polytime. However, again from Theorems 27 and 28, for any slightly more general learning device $\mathbf{M}_1$ which EX-learns this $\mathcal{C}_1$, there will be an easy f, an $f \in \mathcal{Z}^*$, so that, on f, $\mathbf{M}_1$'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* [6,11]. 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 [6] 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 [6].) One of our main positive results (Theorem 31 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 intermediate strength discussed beginning in §2 below. Let $\mathcal{PF}^k = \text{DTimeF}(n^k)$ and $\mathcal{QF}^k_\alpha = \text{DTimeF}(n^k \cdot (\log n) \cdot \alpha(n))$. Let φ_q be the (partial) function computed by multi-tape Turing machine (number) q. Suppose $\mathbf{M}$ is any device BC*-learning $\mathcal{QF}^k_\alpha$.[2] Corollary 9 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 conjectures p, Peano Arithmetic [12] (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 31 (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 learning device of Theorem 31, its final output on f is information-deficient, but *not* complexity-deficient. Corollary 9 discussed

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

in this paragraph is one of several corollaries of Theorem 8, our first main sufficient condition result, all given in §4.

Here is another example. Let $\mathcal{C}_0 = \mathcal{REG}$ and $\mathcal{C}_1 = \mathcal{CF}$, 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 as those mentioned above are not available. Let $co\mathcal{Z}^*$ be the CCFL for the *co-finite* languages, i.e., the languages whose complements are finite. Clearly, $co\mathcal{Z}^*$ is an especially simple, proper subclass of $\mathcal{REG}$. First note that some learning device outputs only deterministic finite automata and EX-learns $\mathcal{REG}$ [4], where deterministic finite automata should be thought of as a degenerate case of Turing machines that use *no* tape squares for workspace [3]. 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 14, we have Corollary 17 as follows. Suppose $\mathbf{M}$ EX*-learns $\mathcal{CF}$ and $k, n \geq 1$.[3] Then there is an easy f, an $f \in co\mathcal{Z}^*$, 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 inputs $x_0, \ldots, x_{n-1}$. This is a complexity-deficiency result for $(\mathcal{REG}, \mathcal{CF})$. Theorem 14 has other complexity-deficiency corollaries, e.g., Corollary 18, an interesting one for $(\mathrm{P}, \mathrm{NP})$ — *assuming* they separate. See also Remark 20 for a related interesting corollary involving BQP, a quantum version of polynomial-time [13], instead of NP. In these results the complexity deficient learned programs have *un*necessary non-determinism or quantum parallelism. Corollaries 10 and 11 of Theorem 8 provide information-deficiency results for $(\mathcal{REG}, \mathcal{CF})$ and $(\mathrm{P}, \mathrm{NP})$, respectively. Remark 12 provides information-deficiency corollaries of Theorem 8 for $(\mathrm{P}, \mathrm{BQP})$ as well as other examples.

Those corollaries, discussed in the previous paragraph, of our sufficient condition results, Theorems 8 and 14, 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 [9]. §3 below provides the details. In [9] 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 8 to obtain all our *information-deficiency* results, including the one for $(\mathcal{PF}^k, \mathcal{QF}^k_\alpha)$ de-

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

scribed above. Actually, Theorem 14 can be used to prove a weak special case of our strong complexity-deficiency result (Theorem 28) for $(\mathcal{PF}^k, \mathcal{QF}^k_\alpha)$. This is Corollary 16. 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 whose proofs employ tricks from [9] can also be shown through related methods from [14–16], but we do not pursue this further here.

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.

2 Conventions and notation

Strings and numbers. $\mathbb{N}$ denotes the set of non-negative integers. Each element of $\mathbb{N}$ 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. We will freely pun between $x \in \mathbb{N}$ as a number and a **0**-**1**-string. Let $|x|$ = the length of the dyadic representation of $x \in \mathbb{N}$. By convention, for $\vec{x} \in \mathbb{N}^n$, $|\vec{x}| = |x_0| + \cdots + |x_{n-1}|$.

Encoding tuples. Let $\langle\cdot,\cdot\rangle$ be a linear time pairing function. That is, $\langle\cdot,\cdot\rangle \colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is 1–1 and onto and each of $\lambda x, y \,.\, \langle x, y\rangle$, $\pi_0 = \lambda\langle x, y\rangle \,.\, x$, and $\pi_1 = \lambda\langle x, y\rangle \,.\, y$ is computable on a multi-tape deterministic Turing Machine in time linear in the lengths of its inputs. Examples of such pairing functions can be found in [17,9]. By convention, for each $n \geq 2$ and $x_0, \ldots, x_n \in \mathbb{N}$, we inductively define $\langle x_0, \ldots, x_n\rangle = \langle x_0, \langle x_1, \ldots, x_n\rangle\rangle$.

Functions. $A \to B$ (respectively, $A \rightharpoonup B$) denotes the set of all total (respectively, possibly partial) functions from A to B. For $f \colon A \rightharpoonup B$, $f(x)\!\downarrow$ means f is defined on x and $f(x)\!\uparrow$ means that $f(x)$ is undefined on x; $\uparrow$ by itself denotes *undefined.* Suppose $f, g \colon \mathbb{N} \rightharpoonup \mathbb{N}$. 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 that $\{\, x \mid f(x) \neq g(x)\,\}$ is finite, i.e., that f and g are *finite variants.* For each $f \colon \mathbb{N} \to \mathbb{N}$, let

$$O(f) \stackrel{\text{def}}{=} \{\, g \colon \mathbb{N} \to \mathbb{N} \mid (\exists a)(\forall x)[\, g(x) \leq a \cdot (f(x) + 1)\,]\,\}.$$

By convention $O(f(n))$ is short for $O(\lambda x \,.\, f(|x|))$. For each $\mathcal{C} \subseteq (\mathbb{N} \to \mathbb{N})$, 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. Let $\log 0 = 0$ and, for each positive integer x, $\log x = \lfloor \log_2 x \rfloor$.

Programming systems. A partial recursive $\psi \colon \mathbb{N}^2 \rightharpoonup \mathbb{N}$ is a *programming system* for $\mathcal{S} \subseteq \mathcal{PR}$ if and only if $\mathcal{S} = \{\, \lambda x \mathbin{.} \psi(i,x) \mid i \in \mathbb{N} \,\}$. We typically write $\psi_i(x)$ for $\psi(i,x)$. Let φ be an acceptable programming system of $\mathcal{PR}$ based on deterministic multi-tape Turing machines [3,18]; φ being *acceptable* means that for any other programming system for an $\mathcal{S} \subseteq \mathcal{PR}$, say θ, there is an effective translation of θ-programs into equivalent φ-programs, i.e., a computable t with $\varphi_{t(i)} = \theta_i$ for all i. By convention, for each i and $x, x_0, \ldots, x_k$, $\varphi_i(x_0, \ldots, x_k) = \varphi_i(\langle x_0, \ldots, x_k \rangle)$; $\Phi_i(x) =$ the run time of TM i on input x; and $\Phi_i^{\mathrm{WS}}(x) =$ the work space used by TM i on input x, provided $\varphi_i(x)\!\downarrow$; ∞, otherwise.

Subrecursive classes of functions. For each recursive $t \colon \mathbb{N} \to \mathbb{N}$, let $\mathrm{DTimeF}(t) = \{\, \varphi_i \mid (\exists a)(\forall x)[\, \Phi_i(x) \le a \cdot (t(x)+1) \,] \,\}$ and $\mathrm{DTIME}(t) = (\mathrm{DTimeF}(t))_{\text{0-1}}$. For each $k > 0$, let

$$\mathcal{PF}^k \stackrel{\text{def}}{=} \mathrm{DTimeF}(\lambda x \mathbin{.} |x|^k),$$

the functions computable in $O(n^k)$ time on a deterministic multi-tape TM, and let $\mathcal{PF} = \cup_{k>0} \mathcal{PF}^k$, the polynomial-time computable functions. Let

$$\mathcal{LS}low \stackrel{\text{def}}{=} \Big\{\, f \in \mathcal{PF}^1 \mid f \text{ is nondecreasing and unbounded} \,\Big\}.$$

By standard results [19,20], 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 $s(f(x)) \le x$ for all x. For each $k > 0$ and $s \in \mathcal{LS}low$,

$$\mathcal{QF}^k_s \stackrel{\text{def}}{=} \mathrm{DTimeF}\left(\lambda x \mathbin{.} |x|^k \cdot (\log |x|) \cdot s(|x|)\right).$$

By standard results [2,3], $(\mathcal{QF}^k_s - \mathcal{PF}^k)_{\text{0-1}} \neq \emptyset$ for each $k > 0$ and $s \in \mathcal{LS}low$.

Let $\mathcal{Z}^*$ (respectively, $co\mathcal{Z}^*$) denote the class of 0–1 valued functions that are 0 (respectively, 1) almost everywhere. Let $\mathcal{NP}$, $\mathcal{BPP}$, $\mathcal{P}$, $\mathcal{CF}$, and $\mathcal{REG}$ respectively denote the classes of characteristic functions of NP, BPP [21], polynomial-time decidable, context free, and regular languages over $\{\,\mathbf{0},\mathbf{1}\,\}^*$. (Recall: $\mathbb{N} \equiv \{\,\mathbf{0},\mathbf{1}\,\}^*$.)

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

Finite initial segments. For $f \colon \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, $f|_n$ denotes the sequence $f(0), f(1), \ldots, f(n-1)$, the length-$n$ initial segment of f. So, $f|_0 =$ the empty segment. Let SEG $=$ the set of all such finite initial segments; σ, with or without decorations, ranges 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* [4] is an algorithmic device that computes a SEG $\rightharpoonup \mathbb{N}$ function. $\mathbf{M}$, with or without decorations, ranges over such machines. Since SEG can be coded into $\mathbb{N}$, an $\mathbf{M}$ can be viewed as computing an element of $\mathcal{PR}$. $\mathbf{M}$ on f *converges* to i (written: $\mathbf{M}(f){\downarrow} = i$) when, 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 with $\mathbf{M}(f|_m){\downarrow}$ and $\mathbf{M}(f|_n) = \mathbf{M}(f|_m)$ for each $n > m$.

The EX *and* EX* *identification criteria.* Suppose $f \in \mathcal{R}$ and $\mathcal{S} \subseteq \mathcal{R}$. $\mathbf{M}$ EX-*identifies* f if and only if, for some i, $\mathbf{M}(f){\downarrow} = i$ and i is a program for f (i.e., $\varphi_i = f$). $\mathbf{M}$ EX-*identifies* $\mathcal{S}$ if and only if $\mathcal{S} \subseteq \mathrm{EX}(\mathbf{M}) \overset{\mathrm{def}}{=} \{\, f \in \mathcal{R} \mid \mathbf{M}$ EX-identifies $f \,\}$. EX $\overset{\mathrm{def}}{=} \{\, \mathcal{S} \mid$ some $\mathbf{M}$ EX-identifies $\mathcal{S} \,\}$. EX-identification originated with Gold [4] who showed that every r.e. subrecursive class is in EX. $\mathbf{M}$ EX*-identifies f if and only if, for some i, $\mathbf{M}(f){\downarrow} = i$ and $\varphi_i =^* f$. EX$^*(\mathbf{M})$ and EX* are defined analogously to our definitions of EX($\mathbf{M}$) and EX. EX*-identification is due to Blum and Blum [5] who showed that EX $\subsetneq$ EX*.

The BC*,* BC^n*, and* BC* *identification criteria.* Suppose $f \in \mathcal{R}$, $\mathcal{S} \subseteq \mathcal{R}$, and $k \in \mathbb{N}$. $\mathbf{M}$ BC^k-*identifies* f if and only if, for all but finitely many n, $\varphi_{\mathbf{M}(f|_n)} =^k f$. $\mathbf{M}$ BC^k-*identifies* $\mathcal{S}$ if and only if $\mathcal{S} \subseteq \mathrm{BC}^k(\mathbf{M}) \overset{\mathrm{def}}{=} \{\, f \in \mathcal{R} \mid \mathbf{M}$ BC^k-identifies $f \,\}$. BC$^k \overset{\mathrm{def}}{=} \{\, \mathcal{S} \mid$ some $\mathbf{M}$ BC^k-identifies $\mathcal{S} \,\}$. We usually write BC^0 as simply BC. The BC* criterion is defined in the obvious fashion. BC-identification was first formalized by Bārzdiņš [22]. Independently, Case and Smith [6] defined BC^m- and BC*-identification. Steel [6] showed EX$^* \subseteq$ BC, Harrington and Case showed this inclusion to be proper [6], and Case and Smith [6] showed BC$^0 \subsetneq$ BC$^1 \subsetneq$ BC$^2 \subsetneq \cdots \subsetneq$ BC*. Moreover, as noted in §1, Harrington [6] showed that $\mathcal{R} \in$ BC*.

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 $\mathbb{N}^k$ when, for some m, there is a Π_n predicate Q over $\mathbb{N}^{k+m}$ such that $P(\vec{x}) \equiv (\exists y_1)\ldots(\exists y_m)Q(\vec{x},\vec{y})$. P is a Π_{n+1} predicate over $\mathbb{N}^k$ when, for some m, there is a Σ_n predicate Q over $\mathbb{N}^{k+m}$ such that $P(\vec{x}) \equiv (\forall y_1)\ldots(\forall y_m)Q(\vec{x},\vec{y})$. A is a Σ_n (respectively, Π_n) set if and only if $A = \{\, x \mid P(x) \,\}$ for some Σ_n (respectively, Π_n) predicate P. Let $\langle W_i^n \rangle_{i \in \mathbb{N}}$ be an acceptable indexing of the Σ_n-sets [23].

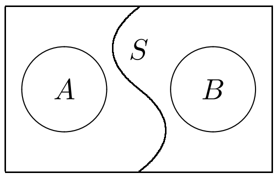

Fig. 1. S separates B from A.

3 Inseparability notions

Suppose A, B, and S are subsets of some fixed set U. We say that S *separates* B from A if and only if $B \subseteq S \subseteq \overline{A}$. (See Figure 1.)

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

Definition 2. Suppose $R \subseteq (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell$. R is *recursive* if and only if the characteristic function of R is a total recursive functional of type $(\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell \to \{0,1\}$. R is *arithmetical* if and only if either R is recursive or

$$R = \left\{ (\vec{f}, \vec{x}) \,\vdots\, (Q_1 y_1) \ldots (Q_m y_m)[\, S(\vec{f}, \vec{x}, \vec{y}) \,] \right\} \tag{1}$$

where each Q_i is either $\exists$ or $\forall$ and where $S \subseteq (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^{\ell+m}$ is recursive. (**N.B.** All the quantifiers in (1) are numeric.) 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. 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 $R \subseteq (\mathbb{N} \to \mathbb{N})^k \times \mathbb{N}^\ell$ in $\Sigma_n^{(\mathrm{fn})}$. (See [23, §15.2]). 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 $\mathcal{B} \subseteq (\mathbb{N} \to \mathbb{N})$. $\mathcal{B}$ 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 $\mathcal{A}$. Also, $\mathcal{B}$ 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, $\varphi_{f(i)} \in (\mathcal{W}_i^n \cap \mathcal{A}) \cup (\overline{\mathcal{W}_i^n} \cap \mathcal{B})$. $\diamond$

The next proposition gives us a way of establishing an $\Sigma_2^{(\mathrm{fn})}$-inseparability through the Σ_2-inseparability of certain sets of programs.

Proposition 4. *Suppose the following:*

(i) $\mathcal{C}$ *is a subrecursive class with programming system* ψ.
(ii) $\mathcal{A}$ *and* $\mathcal{B}$ *are disjoint and both* $\mathcal{A}\cap\mathcal{C}$ *and* $\mathcal{B}\cap\mathcal{C}$ *are nonempty.*
(iii) $\{\, i \mid \psi_i \in \mathcal{B} \,\}$ *is effectively* Σ_2*-inseparable from* $\{\, i \mid \psi_i \in \mathcal{A} \,\}$.

Then $\mathcal{B}$ *is effectively* $\Sigma_2^{(\mathrm{fn})}$*-inseparable from* $\mathcal{A}$.

Proof. Suppose r witnesses the effective Σ_2-inseparability of $\{\, i \mid \psi_i \in \mathcal{B} \,\}$ from $\{\, i \mid \psi_i \in \mathcal{A} \,\}$. By standard results (see [23, §15.2]), there is a recursive $R \subseteq (\mathbb{N} \to \mathbb{N}) \times \mathbb{N}^3$ such that, for all j, $\mathcal{W}^2_j = \{\, f \mid (\exists m)(\forall n) R(f,m,n,j) \,\}$. By a few more standard results, there is a recursive function g such that, for all j, $W^2_{g(j)} = \{\, i \mid (\exists m)(\forall n) R(\psi_i,m,n,j) \,\} = \{\, i \mid \psi_i \in \mathcal{W}^2_j \,\}$. Let t be a recursive function such that $\varphi_{t(i)} = \psi_i$ for all i. It follows that $r' = t \circ r \circ g$ is recursive and witnesses the effective $\Sigma_2^{(\mathrm{fn})}$-inseparability of $\mathcal{B}$ from $\mathcal{A}$. □

The next proposition provides an alternative, often handier, way of showing $\Sigma_2^{(\mathrm{fn})}$-inseparability. Note, however, the proof of this proposition depends on Proposition 4. Recall: $\mathcal{PF}^1$ = the linear-time computable functions and $(\mathcal{C})_{0\text{-}1}$ = the 0–1 valued elements of $\mathcal{C}$.

Proposition 5. *Suppose that* $\mathcal{C}_0, \mathcal{C}_1 \subseteq (\mathbb{N} \to \mathbb{N})$ *are such that:*

(i) Both $\mathcal{C}_0$ *and* $\mathcal{C}_1$ *are closed under 0–1 valued finite variants.*
(ii) $\mathcal{PF}^1 \subseteq \mathcal{C}_0 \cap \mathcal{C}_1$.
(iii) For each $f \in \mathcal{PF}^1$ *and* $g, h \in \mathcal{C}_1$, $f\circ g$, $g\circ f$, *and* $\lambda x \,.\, \langle g(x), h(x)\rangle \in \mathcal{C}_1$.
(iv) There is a programming system for $\mathcal{C}_1$.
(v) $(\mathcal{C}_1 - \mathcal{C}_0)_{0\text{-}1} \neq \emptyset$.

Then, $(\mathcal{C}_1 - \mathcal{C}_0)_{0\text{-}1}$ *is effectively* $\Sigma_2^{(\mathrm{fn})}$*-inseparable from* $\mathcal{Z}^*$.

There are many ways to establish this proposition using results from the structural complexity literature, for example [15,14,16,9]. The proof given below uses a tool from [9], restated as the following lemma. The φ^* of condition f is an acceptable programming system for the partial recursive functions relative to an oracle for the halting problem [23,9]. *Notation:* For each $f, g, h \colon \mathbb{N} \rightharpoonup \mathbb{N}$, define *Cond*: $(\mathbb{N} \rightharpoonup \mathbb{N})^3 \to (\mathbb{N} \rightharpoonup \mathbb{N})$ by:

$$\mathit{Cond}(f,g,h)(x) \;=\; \begin{cases} g(x), & \text{if } f(x) > 0; \\ h(x), & \text{if } f(x) = 0;; \\ \uparrow, & \text{if } f(x)\uparrow. \end{cases} \tag{2}$$

Lemma 6 ([9, Theorem 9.11]). *Suppose that* $\mathcal{C}_0$*, and* $\mathcal{C}_1$ *are subrecursive* *classes, $\mathcal{A} \subseteq (\mathcal{C}_0 \cap \mathcal{C}_1)_{0\text{-}1}$, $\mathcal{D} \subseteq \mathcal{C}_1$, and there is a programming system ψ for $\mathcal{C}_1$. Moreover, suppose that the following conditions hold.*

a. There is a $g_1 \in (\mathcal{C}_1)_{0\text{-}1}$ with $\{ f\colon \mathbb{N} \to \{0,1\} \mid f =^* g_1 \} \subseteq (\mathcal{C}_1 - \mathcal{C}_0)$.
b. There is a $g_0 \in \mathcal{A}$ with $\{ f\colon \mathbb{N} \to \{0,1\} \mid f =^* g_0 \} \subseteq \mathcal{A}$.
c. For each $f \in \mathcal{D}$ and $g, h \in \mathcal{C}_1$, $Cond(f,g,h) \in \mathcal{C}_1$.
d. For each $f \in \mathcal{C}_1$, $f \circ \pi_0$, $f \circ \pi_1 \in \mathcal{C}_1$.
e. For each m and $n > 0$, there is a computable function s such that, for all $i, x_1, \ldots, x_m, y_1, \ldots, y_n$, $\psi_{s(i,\vec{x})}(\langle \vec{y} \rangle) = \psi_i(\langle \vec{x}, \vec{y} \rangle)$.
f. There is an $L \in \mathcal{D}$ such that, for all i and x, $\lim_{t\to\infty} L(\langle i,x,t\rangle) = \varphi_i^*(x)$, where $\varphi_i^*(x)\downarrow = y \iff (\overset{\infty}{\forall} t)[L(\langle i,x,t\rangle) = y]$.

Then $\{ i \mid \psi_i \in (\mathcal{C}_1 - \mathcal{C}_0)_{0\text{-}1} \}$ is effectively Σ_2-inseparable from $\{ i \mid \psi_i \in \mathcal{A} \}$.

Proof (of Proposition 5). Let $\mathcal{A} = \mathcal{Z}^*$ and $\mathcal{D} = \mathcal{PF}^1$. By (ii), (iv), and general results on programming systems in [9, §4.2], we may assume that ψ is a programming system for $\mathcal{C}_1$ that satisfies condition e. We note that $\mathcal{C}_0$, $\mathcal{C}_1$, $\mathcal{A}$, $\mathcal{D}$, and ψ together satisfy the hypotheses of the lemma. Specifically: conditions a and b follow from (i) and (v), conditions c and d follow from (ii) and (iii), condition e follows from our choice of ψ, and condition f follows from (ii) and Theorem 7.4 of [9]. Hence by the lemma, $\{ i \mid \psi_i \in (\mathcal{C}_1 - \mathcal{C}_0)_{0\text{-}1} \}$ is effectively Σ_2-inseparable from $\{ i \mid \psi_i \in \mathcal{A} \}$. Therefore, by Proposition 4 we have that $(\mathcal{C}_1 - \mathcal{C}_0)_{0\text{-}1}$ is effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable from $\mathcal{Z}^*$. □

Now, using Propositions 4 and 5 and few other results from the literature, we can establish some sample $\Sigma_2^{(\mathrm{fn})}$-inseparability results for some of the subrecursive classes introduced in §2.

Corollary 7. *Suppose $k > 0$ 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 co$\mathcal{Z}^*$.

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

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

Proof. *Part (a).* As previously noted, $(\mathcal{QF}_s^k)_{0\text{-}1}$ and $(\mathcal{PF}^k)_{0\text{-}1}$ separate by classic results [2,3]. It is then straightforward that hypotheses (i) through (v) of Proposition 5 are satisfied. Thus, part (a) follows from the proposition.

Part (b). By the proof of Corollary 11.17 of [9], there is a programming system

ψ for $\mathcal{CF}$ such that $\{\, i \mid \psi_i \in (\mathcal{CF} - \mathcal{REG}) \,\}$ is effectively Σ_2-inseparable from $\{\, i \mid \psi_i \in co\mathcal{Z}^* \,\}$. Thus, part (b) follows by Proposition 4.

Part (c). Suppose $\mathcal{C}_1 = \mathcal{NP}$ and $\mathcal{C}_0 = \mathcal{P}$. Then it is straightforward that (i) through (iv) of Proposition 5 are satisfied. The P $\neq$ NP hypothesis implies (v). Thus, part (c) follows from the proposition.

Part (d). This follows from an argument similar to the one for part (c). $\square$

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[4] such that no finite variant of a member of $\mathcal{B}$ has a program in G.

Theorem 8, our first sufficient condition theorem, provides us with our information deficiency corollaries (Corollaries 9 through 11). *Notation:* $\mathcal{FV}(\mathcal{B}) = \{\, f \colon \mathbb{N} \to \mathbb{N} \mid f \text{ is a finite variant of some element of } \mathcal{B} \,\}$.

Theorem 8. *Suppose that:*

(i) $\mathcal{B}$ *is* $\Sigma_2^{(\mathrm{fn})}$*-inseparable from* $\mathcal{A}$.
(ii) G *is a* Σ_1*-set with* $\mathcal{FV}(\mathcal{B}) \cap \{\, \varphi_i \mid i \in G \,\} = \emptyset$.
(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, $\mathbf{M}(f|_n) \notin G$.

Proof. Since G is a Σ_1-set, there is a recursive predicate R_G such that $G = \{\, x \mid (\exists m) R_G(x,m) \,\}$. Consider $\mathcal{S} = \{\, f \mid (\overset{\infty}{\forall} n)[\mathbf{M}(f|_n) \notin G] \} =$

$$\{\, f \mid (\exists n_0)(\forall n > n_0)(\forall m)[\, \neg R_G\,(\mathbf{M}(f|_n), m)\,] \,\}\,.$$

Thus, $\mathcal{S} \in \Sigma_2^{(\mathrm{fn})}$. Also, by (ii) and (iii) it follows that $\mathcal{B} \subseteq \mathcal{S}$. Now suppose the negation of the conclusion: that for all $f \in \mathcal{A}$, $(\overset{\infty}{\exists} n)[\, \mathbf{M}(f|_n) \in G\,]$. Clearly, $\mathcal{A} \cap \mathcal{S} = \emptyset$. Therefore, not (i) since $\mathcal{S}$ is a $\Sigma_2^{(\mathrm{fn})}$-set separating $\mathcal{B}$ from $\mathcal{A}$. $\square$

The next three corollaries involve provability and PA, Peano Arithmetic [12]. We write $\vdash$ for the provability relation and $\nvdash$ for 'does *not* prove.' The following predicates are expressible in PA (and herein we do not distinguish between

4 E.g., the members of G may run efficiently and/or be easy to prove things about.

expressions in PA and expressions in the metalanguage).

$$\begin{aligned}
P_k(i) &\equiv_{\text{def}} (\exists c)(\forall x)[\,\Phi_i(x) \le c \cdot (|x|+1)^k\,].\\
P^*_k(i) &\equiv_{\text{def}} (\exists j \mid \varphi_j =^* \varphi_i)[\,P_k(j)\,].\\
P^*(i) &\equiv_{\text{def}} (\exists k)[\,P^*_k(i)\,].\\
S_k(i) &\equiv_{\text{def}} (\forall x)[\,\Phi^{\text{WS}}_i(x) \le k\,].\\
\mathit{REG}^*(i) &\equiv_{\text{def}} (\exists k)(\exists j \mid \varphi_j =^* \varphi_i)[\,S_k(j)\,].
\end{aligned}$$

N.B. Each of Corollaries 9, 10, and 11 remains true if PA is replaced by any true and computably axiomatized theory [12] 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. [12].

Corollary 9. *Suppose that* $k > 0$, $s \in \mathcal{LS}low$, *and* $\text{BC}^*(\mathbf{M}) \supseteq \mathcal{QF}^k_s$. *Then there is an* $f \in \mathcal{Z}^*$ *such that, for all but finitely many* n, $\text{PA} \nvdash P^*_k(\mathbf{M}(f|_n))$.

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{QF}^k_s - \mathcal{PF}^k)$, and $G = \{\, i \mid \text{PA} \vdash P^*_k(i)\,\}$. Now, applying Corollary 7(a) and Theorem 8, we are done. $\square$

Interpretation. Let $\mathbf{M}$ and f be as in Corollary 9.[5] Then it must be the case that, for all but finitely many n, the program $\mathbf{M}(f|_n)$ computes a finite variant of f, an almost everywhere zero function. 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 remarks apply to the next two corollaries.

Corollary 10. *Suppose* $\text{BC}^*(\mathbf{M}) \supseteq \mathcal{CF}$. *Then there is an* $f \in co\mathcal{Z}^*$ *such that, for all but finitely many* n, $\text{PA} \nvdash \mathit{REG}^*(\mathbf{M}(f|_n))$.

Proof. Let $\mathcal{A} = co\mathcal{Z}^*$, $\mathcal{B} = (\mathcal{CF} - \mathit{REG})$, and $G = \{\, i \mid \text{PA} \vdash \mathit{REG}^*(i)\,\}$. Now, applying Corollary 7(b) and Theorem 8, we are done. $\square$

Corollary 11. *Suppose* $\text{BC}^*(\mathbf{M}) \supseteq \mathcal{NP}$ *and that* $\text{P} \neq \text{NP}$. *Then there is an* $f \in \mathcal{Z}^*$ *such that, for all but finitely many* n, $\text{PA} \nvdash P^*(\mathbf{M}(f|_n))$.

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{NP} - \mathcal{P})$, and $G = \{\, i \mid \text{PA} \vdash P^*(i)\,\}$. Now, applying Corollary 7(c) and Theorem 8, we are done. $\square$

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

Remark 12. Corollaries 9, 10, and 11 provide only a small sample of the wide range of situations to which Theorem 8 applies. For example, one can replace NP in Corollary 11 with essentially any natural complexity class C containing P; then under the assumptions that $\mathrm{BC}^*(\mathbf{M}) \supseteq$ the class of characteristic functions of members of C and C $\neq$ P, one has the same conclusion as Corollary 11. So for C one can have BPP (bounded probabilistic polynomial-time [21]), BQP (a quantum version of polynomial-time [13]), PSPACE, and so on. The only work involved in showing these results is in establishing the analogue of Corollary 7(c) for each of these classes, and this is straightforward using the results and tools of [9]. [6] $\diamond$

Remark 13. We call the f asserted to exist in Theorem 8 a *witness to the deficiency.* If we change "$\Sigma_2^{(\mathrm{fn})}$-inseparable" to "effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable" in Theorem 8, then we can strengthen that theorem's conclusion to: *there is a computable function w such that, for each i, if i is the index of an IIM satisfying hypothesis (iii), then $\varphi_{w(i)} \in \mathcal{A}$ and, for all but finitely many n, $\mathbf{M}(\varphi_{w(i)}|_n) \notin G$.* So, thanks to Corollary 7, each of Corollaries 9, 10, and 11 can have its conclusion correspondingly strengthened. In particular situations we can do much better than this. For example, using the tools of [9] we can improve the conclusion of Corollary 9 to: *There is a linear time computable function w such that, for all i, $\Phi_{w(i)} \in O(n^k \cdot \log n \cdot s(n))$ and, if i is the index of an $\mathbf{M}$ with $\mathrm{BC}^*(\mathbf{M}) \supseteq \mathcal{QF}_s^k$, then $\varphi_{w(i)} \in \mathcal{Z}^*$ and, for all but finitely many n, $\mathrm{PA} \nvdash P_k^*(\mathbf{M}(\varphi_{w(i)}|_n))$.* $\diamond$

Theorem 14, our second sufficient condition theorem, provides us with complexity deficiency corollaries (Corollaries 16 through 19). Recall: $\mathcal{FV}(\mathcal{B}) = \{ f \mid (\exists g \in \mathcal{B})[f =^* g]\}$.

Theorem 14. *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 $\mathcal{A} \cup \mathcal{B} \subseteq \mathrm{EX}^*(\mathbf{M})$.

Then there is an $f \in \mathcal{A}$ such that $\mathbf{M}(f) \notin G$.

[6] BQP is not discussed in [9]. However, as BQP amounts to a quantum version of BPP, all the results needed to show the BQP analogue of Corollary 7(c) can be obtained by a straightforward modification of the BPP results in [9]. Of course, then, [9, Corollary 11.10], a relative program succinctness result for, for example, BPP vs. P, also holds for BQP vs. P (each assuming separation).

Proof. Since G is a Π_2-set, there is a recursive predicate R_G such that $G = \{\, x \mid (\forall m)(\exists n) R_G(x,m,n) \,\}$. Consider $\mathcal{S} = \{\, f \mid \mathbf{M}(f){\downarrow} \notin G] \,\} =$

$$\Big\{\, f \mid (\exists i, m)(\forall n_0, n_1 \geq m) \Big[\mathbf{M}(f|_{n_0}) = i \;\; \& \;\; \neg R_G(i, m, n_1) \Big] \,\Big\}.$$

Thus, $\mathcal{S} \in \Sigma_2^{(\mathrm{fn})}$. Also, by (ii) and (iii) it follows that $\mathcal{B} \subseteq \mathcal{S}$. Now suppose the negation of the conclusion: that for all $f \in \mathcal{A}$, $\mathbf{M}(f){\downarrow} \in G$. Then clearly, $\mathcal{A} \cap \mathcal{S} = \emptyset$. Therefore, not (i) since $\mathcal{S}$ is a $\Sigma_2^{(\mathrm{fn})}$-set separating $\mathcal{B}$ from $\mathcal{A}$. $\square$

Scholium 15. The fact that $G \in \Pi_2$ in Theorem 14 fails to provide as much generality as one might hope. Here is why. It is a well-worn observation that if $\mathcal{C}$ is closed under total finite variants and P is a Σ_2-set such that $\mathcal{C} = \{\, \varphi_i \mid i \in P \,\}$, then there is an r.e. set P' such that $\mathcal{C} = \{\, \varphi_i \mid i \in P' \,\}$. It is a minor variation on this observation that if hypotheses (ii) and (iii) of Theorem 14 hold, then there is a Π_1-set G' such that $\{\, \varphi_i \mid i \in G \,\} \subseteq \{\, \varphi_i \mid i \in G' \,\} \subseteq (\mathcal{PR} - \mathcal{FV}(\mathcal{B}))$. Hence, G in Theorem 14 might as well be Π_1—which is what it is in our applications of this theorem. $\diamond$

As was mentioned in §1, the following corollary of Theorem 14 provides a weak special case of our strong complexity-deficiency result (Theorem 28) 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 "there exist n distinct inputs."

Corollary 16. *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 $j < n$, $\Phi_i(x_j) > a \cdot (|x_j| + 1)^k$.*

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{QF}^k_s - \mathcal{PF}^k)$, and $G = \{\, i \mid (\forall x_0, \ldots, x_{n-1} | x_0 < \cdots < x_{n-1})(\exists j < n)[\Phi_i(x_j) \leq a \cdot (|x_j| + 1)^k \,] \,\}$. Now, applying Corollary 7(a) and Theorem 14, we are done. $\square$

As mentioned in §1, the next three 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 17. *Suppose $\mathrm{EX}^*(\mathbf{M}) \supseteq \mathcal{CF}$ and $k, n > 0$. Then there is an $f \in co\mathcal{Z}^*$ such that, for some i, $\mathbf{M}(f){\downarrow} = i$, but, there are distinct $x_0, \ldots, x_{n-1}$ such that for each $j < n$, $\Phi^{\mathrm{WS}}_i(x_j) > k$.*

Proof. Let $\mathcal{A} = co\mathcal{Z}^*$, $\mathcal{B} = (\mathcal{CF} - \mathcal{REG})$, and $G = \{\, i \mid (\forall x_0, \ldots, x_{n-1} | x_0 < \cdots < x_{n-1})(\exists j < n)[\Phi^{\mathrm{WS}}_i(x_j) \leq k \,] \,\}$. Now, applying Corollary 7(b) and Theorem 14, we are done. $\square$

Interpretation. Suppose $\mathbf{M}$ EX*-identifies $\mathcal{CF}$.[7] Then by Corollary 17, there are members of $co\mathcal{Z}^*$ 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 members of $\mathcal{REG}$, and members of $\mathcal{REG}$ have programs that use no workspace at all.

Let φ^{ND} be based on a natural programming system of nondeterministic, multi-tape Turing machines for accepting sets. Let $\mathrm{Paths}_i(x) =$ the number of paths in the computation tree of φ^{ND}-program i on input x.

Corollary 18. *Suppose* $\mathrm{P} \neq \mathrm{NP}$. *Suppose* $\mathbf{M}$ EX**-identifies* $\mathcal{NP}$ *using polynomial-time (deterministic and nondeterministic)* φ^{ND}*-programs,*[8] q *is a polynomial, and* $n > 0$. *Then there is an* $f \in \mathcal{Z}^*$ *for which there are distinct* $x_0, \ldots, x_{n-1}$ *such that for* $i = \mathbf{M}(f)$ *and for* $x = x_0, \ldots, x_{n-1}$, φ^{ND}*-program* i *on input* x *runs* non-*deterministically and, in fact,* $\mathit{Paths}_i(x) > q(|x|)$.

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{NP} - \mathcal{P})$, and $G = \{\, i \mid (\forall x_0, \ldots, x_{n-1} | x_0 < \cdots < x_{n-1})(\exists j < n)[\mathrm{Paths}_i(x_j) \leq q(|x_j|)\,]\,\}$. G is easily shown to be in Π_1. So, applying Corollary 7(c) and Theorem 14, we are done. $\square$

Interpretation. Suppose $\mathbf{M}$ EX*-identifies $\mathcal{NP}$ using polynomial-time (deterministic and nondeterministic) φ^{ND}-programs.[9] Then by Corollary 18, there are members of $\mathcal{Z}^*$ for which $\mathbf{M}$ infers programs that employ arbitrarily polynomially many unpleasant non-determinist paths on arbitrarily large (but finite) sets of inputs.

Our final corollary of Theorem 14 concerns the probabilistic complexity class BPP. This corollary and its setup are representative of how one obtains complexity-deficiency results for probabilistic [21], counting [24], and quantum [13] complexity classes.

Let φ^{PR} be the modification of φ^{ND} in which all nondeterministic branch points are binary and decided upon by the flip of a fair coin. A φ^{PR}-program's run time on an input is the length of the longest possible computation of the program on that input. For $\delta \in (\frac{1}{2}, 1]$, a φ^{PR}-program is said to δ-*confidently*

[7] As noted in §1, an allowed special case is that $\mathbf{M}$ actually EX-learns $\mathcal{CF}$.
[8] *Note:* $\mathcal{NP} \in$ EX trivially as witnessed by some $\mathbf{M}'$ also outputting φ^{ND}-programs.
[9] As noted in §1, an allowed special case is that $\mathbf{M}$ actually EX-learns $\mathcal{NP}$.

decide A when, for all x,

$$\left.\begin{array}{l} x \in A \implies \text{Prob}[\,\text{the program accepts } x\,] \geq \delta; \\ x \notin A \implies \text{Prob}[\,\text{the program rejects } x\,] \geq \delta. \end{array}\right\} \qquad (3)$$

BPP $\overset{\text{def}}{=} \{\, A \mid (\exists i)(\exists \delta \in (\frac{1}{2}, 1])[\,\varphi^{\text{PR}}$-program i runs in polynomial time and δ-confidently decides $A\,]\,\}$. It turns out [25] that for any fixed $\delta_0 \in (\frac{1}{2}, 1)$, BPP $= \{\, A \mid (\exists i)\ [\,\varphi^{\text{PR}}$-program i runs in polynomial time and δ_0-confidently decides $A\,]\,\}$. Let $\text{Flips}_i(x)$ = the maximum number of coin-flip branch points along any branch of φ^{PR}-program i's computation tree on input x. Note: if i is a polynomial-time φ^{PR}-program that δ-confidently decides A with $\text{Flips}_i(x) \in O(\log |x|)$, then $A \in P$.

For each $A \subseteq \mathbb{N}$, let χ_A = the characteristic function of A. An IIM $\mathbf{M}$ is said to δ-*confidently* EX-*identify* BPP when, for each $A \in$ BPP, $\mathbf{M}(\chi_A)\!\downarrow = i_A$, a polynomial-time φ^{PR}-program that δ-confidently decides A. Similarly, $\mathbf{M}$ is said to δ-*confidently* EX*-*identify* BPP when, for each $A \in$ BPP, $\mathbf{M}(\chi_A)\!\downarrow = i_A$, a polynomial-time φ^{PR}-program such that, for all but finitely many x, (3) holds. It turns out that, for each $\delta \in (\frac{1}{2}, 1)$, there is an IIM $\mathbf{M}_\delta$ that δ-confidently EX-identifies BPP. [10]

Corollary 19. *Suppose* P $\neq$ BPP. *Suppose that* $\mathbf{M}$ *δ-confidently* EX*-*identifies* BPP *where $\delta \in (\frac{1}{2}, 1)$ and that k and n are positive integers. Then there is an $f \in \mathcal{Z}^*$ for which there are distinct $x_0, \ldots, x_{n-1}$ such that for $i = \mathbf{M}(f)$ and for $x = x_0, \ldots, x_{n-1}$, we have $\mathit{Flips}_i(x) > k \cdot \log |x|$.*

Proof. Let $\mathcal{A} = \mathcal{Z}^*$, $\mathcal{B} = (\mathcal{BPP} - \mathcal{P})$, and $G = \{\, i \mid (\forall x_0, \ldots, x_{n-1} | x_0 < \cdots < x_{n-1})(\exists j < n)[\,\text{Flips}_i(x_j) \leq k \cdot \log |x|\,]\,\}$. G is easily shown to be in Π_1. So, applying Corollary 7(d) and Theorem 14, we are done. $\square$

Interpretation. Suppose $\mathbf{M}$ δ-confidently EX*-identifies $\mathcal{BPP}$ as supposed in the above corollary. [11] Then the corollary implies that there are members of $\mathcal{Z}^*$ for which $\mathbf{M}$ infers witnessing programs that employ arbitrarily logarithmically many unpleasant coin flips on arbitrarily large (but finite) sets of inputs.

[10] E.g., let $\mathbf{M}_\delta(\sigma)$ = the least $i \leq |\sigma|$, if any, such that for each $x \in \text{dom}(\sigma)$: (i) φ^{PR}-program i runs in $i \cdot (|x|+1)^i$-time, and (ii) for φ^{PR}-program i, for $A = \{\, x \mid \sigma(x) = 1\,\}$, and for each $x \in \text{dom}(\sigma)$, (3) holds; let $\mathbf{M}_\delta(\sigma) = 0$ if there is no such i. Note that if φ^{PR}-program i runs in polynomial time, but not in time $i \cdot (|x|+1)^i$, then there is a larger, padded version of i, say i', that will run in time $i' \cdot (|x|+1)^{i'}$.
[11] An allowed special case is that $\mathbf{M}$ actually δ-confidently EX-learns $\mathcal{BPP}$.

Remark 20. Corollaries 16 through 19 provide only a small sample of the wide range of situations to which Theorem 14 applies. For example, as in Remark 12, one can replace NP in Corollary 18 with essentially any natural complexity class C containing P; then under the assumptions that $\mathrm{EX}^*(\mathbf{M}) \supseteq$ the class of characteristic functions of members of C and C $\neq$ P, one has the same conclusion as Corollary 18. So for C one can have BQP (a quantum version of polynomial-time [13]), PSPACE, and so on. The main work involved in showing these results is (i) a set up, as for Corollaries 18 and 19, to handle the computational resource in question and (ii) establishing the analogue of Corollary 7(c) for each of these classes, and the results of [9] make this later straightforward. (For BQP, the remarks of footnote 6 again apply.) Then, for *example*, for C = BQP$\neq$ P, the corresponding complexity deficient learned programs exhibit *un*necessary quantum parallelism—just as in Corollary 19, if P $\neq$ BPP, the corresponding complexity deficient learned programs exhibit *un*necessary amounts of randomization. $\diamond$

Remark 21. Applications of Theorem 14 (e.g., Corollary 18 above) typically involve details of specific programming systems and resource measures. Because of this Theorem 14 does not have the same breadth of generality as Theorem 8. We also note that if one changes "$\Sigma_2^{(\mathrm{fn})}$-inseparable" to "effectively $\Sigma_2^{(\mathrm{fn})}$-inseparable" in Theorem 14, then one can strengthen that theorem's conclusion so that witnesses are effectively found. $\diamond$

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 tools depend on a few special features of our programming system φ and its associated complexity measure Φ introduced in §2. The details of these features are mostly straightforward and are omitted here, but can be found in Chapter 3 of [9].

The first of these tools is simply a uniform version of the classic result of Hennie and Stearns [26] on the cost of simulations. (*Note:* "uniform" here means that the cost of interpreting the program is taken into account.)

Proposition 22 (The cost of simulations, [9] 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 $O(|x| + (|i| + 1) \cdot (|t| \cdot \log |t| + 1))$.

Next is a technical proposition about the complexity overhead of applying simple control structures such as, in part (a), conditionals to sub-programs. Part (b) is about the overhead of storing data or programs inside programs, and part (c) is about complexity-bounded self-reference. Machtey, Winklmann, Young [27,28] and Kozen [29] were among the first to establish "polynomial-time overhead" results of these sorts. The proposition below is based on somewhat more refined work in [9]. Recall that *Cond* was defined by (2) in §3.

Proposition 23 (Complexity-bounded control structures). *Suppose that $m, n \geq 1$. In the following i, j, and k range over $\mathbb{N}$, and $\vec{x}$ and $\vec{y}$ over $\mathbb{N}^m$ and $\mathbb{N}^n$, respectively.*

(a) (Conditionals, [9] Lemma 3.14.) *There is a linear-time computable* if_m *and an $a_m \in \mathbb{N}$ such that, for all i, j, k, and $\vec{x}$:*

$$\begin{aligned}
\varphi_{\mathrm{if}_m(i,j,k)}(\vec{x}) &= \mathit{Cond}(\varphi_i, \varphi_j, \varphi_k)(\vec{x}). \\
\Phi_{\mathrm{if}_m(i,j,k)}(\vec{x}) &\leq \begin{cases} \Phi_i(\vec{x}) + \Phi_j(\vec{x}) + a_m \cdot (|\vec{x}| + 1), & \text{if } \varphi_i(\vec{x}) > 0; \\ \Phi_i(\vec{x}) + \Phi_k(\vec{x}) + a_m \cdot (|\vec{x}| + 1), & \text{if } \varphi_i(\vec{x}) = 0; \\ \infty, & \text{if } \varphi_i(\vec{x})\uparrow. \end{cases}
\end{aligned}$$

(b) (S-m-n, [9] Theorem 4.4.) *There is a linear-time computable $s_{m,n}$ and an $a_{m,n} \in \mathbb{N}$ such that, for all i, $\vec{x}$, and $\vec{y}$:*

$$\begin{aligned}
\varphi^n_{s_{m,n}(i,\vec{x})}(\vec{y}) &= \varphi^{m+n}_i(\vec{x}, \vec{y}). \\
\Phi^n_{s_{m,n}(i,\vec{x})}(\vec{y}) &\leq \Phi^{m+n}_i(\vec{x}, \vec{y}) + a_{m,n} \cdot (|\vec{x}| + |\vec{y}| + 1).
\end{aligned}$$

(c) (Self-reference, [9] Theorem 4.6.) *There is a linear-time computable $r_{m,n}$ and an $a_{m,n} \in \mathbb{N}$ such that, for all i, $\vec{x}$, and $\vec{y}$:*

$$\begin{aligned}
\varphi^n_{r_{m,n}(i,\vec{x})}(\vec{y}) &= \varphi^{m+n+1}_i(r_{m,n}(i, \vec{x}), \vec{x}, \vec{y}). \\
\Phi^n_{r_{m,n}(i,\vec{x})}(\vec{y}) &\leq \Phi^{m+n+1}_i(r_{m,n}(i, \vec{x}), \vec{x}, \vec{y}) + a_{m,n} \cdot (|\vec{x}| + |\vec{y}| + 1).
\end{aligned}$$

Kleene [30] showed that any nonempty r.e. set is the range of some primitive recursive function. The next proposition takes the basic idea behind Kleene's construction, lowers the complexity, slows the enumeration, and recasts things in terms of the ranges of partial recursive functions.

Proposition 24 (Delayed enumeration, [9] 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 $\vec{w} \in \mathbb{N}^m$, there is a strictly increasing sequence of numbers y_0, y_1, $y_2, \ldots$ such that*

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

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

Convention: For each m, let $\mathrm{rng}_m = \mathrm{rng}_{m,s}$ where $s = \lambda n \centerdot \max(1, \log^{(2)}(n))$.

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

For simplicity of the technical exposition we begin with two theorems essentially announced in [31] and based on a suggestion of Sipser for the EX case. In [31] 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 not only for the results from [31], but also for the two main results of this section (Theorems 27 and 28 below). These main theorems provide considerably tighter complexity bounds *and* stronger quantifier order than the results from [31].

Although EX$^* \subsetneq$ BC^0, Theorems 25 through 28 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 25. *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 26. *Suppose that $\mathrm{EX}^*(\mathbf{M}) \supseteq \mathcal{PF}$. Then, for each polynomial q, there is an $f \in \mathcal{Z}^*$ such that $(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f)}(x) > q(|x|)]$.*

We start with the proof of Theorem 26 which is a bit simpler than that of Theorem 25.

Proof (of Theorem 26). Fix a polynomial q. *Terminology:* We say that p is *available* at w if and only if $\Phi_p(w) \le q(|w|)$. Since $[\,\Phi_p(w) \le q(|w|)\,]$ is equivalent to $[\,T(p, w, \mathbf{0}^{q(|w|)}) = 1\,]$, by Proposition 22 we have that availability is testable in time polynomial in $|p|$ and $|w|$. Let d be a φ-program such that, for all e and x,

$$\varphi_d(e,x) \;=\; \begin{cases} \uparrow, & \text{if, for some } w < x,\ \varphi_e(w)\uparrow; \\ \mathbf{M}(\varphi_e|_x), & \text{otherwise.} \end{cases}$$

Now let u be a φ-program such that, for all e and y,

$$\varphi_u(e,y) \;=\; \begin{cases} 0, & \text{if (i) } \mathrm{rng}_1(d,e,y) = 0 \text{ or } \mathrm{rng}_1(d,e,y) = \\ & \quad 1+p, \text{ but } p \text{ is not available at } y; \\ 1 \dot{-} S(p,y,\mathbf{0}^{q(|y|)}), & \text{(ii) otherwise, where } \mathrm{rng}_1(d,e,y) = 1+p. \end{cases}$$

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 polynomial-time computable, it is straightforward that φ_u is polynomial-time computable. So, without loss of generality, we assume that Φ_u is polynomially bounded. Thus by Proposition 23(c), 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. Also note that if p is canceled for e_0 at y, then $\varphi_{e_0}(y) = 1 \dot{-} \varphi_p(y) \neq \varphi_p(y)$.

Since $\varphi_{e_0} \in \mathcal{PF}$, by hypothesis there is a p_0 such that $\mathbf{M}(\varphi_{e_0})\downarrow = p_0$ and $\varphi_{p_0} =^* \varphi_{e_0}$. So by the definition of d, we have that for all but finitely many x, $\varphi_d(e_0,x) = p_0$. Hence, by Proposition 24 we have that, for all but finitely many y, $\mathrm{rng}_1(d,e_0,y) = 1+p_0$.

Claim: p_0 is canceled for e_0 only finitely many times. *Proof:* Since $\varphi_{p_0} =^* \varphi_{e_0}$, the claim follows from the definition of cancellation.

Since for all but finitely many y, $\mathrm{rng}_1(d,e_0,y) = 1+p_0$ and since by the claim p_0 is canceled for e_0 only finitely many times, it follows that p_0 is available only finitely many times, i.e., for all but finitely many y, $\Phi_{p_0}(y) > q(|y|)$. It also follows that there are only finitely many y on which any p is canceled for e_0. Hence, by the construction of u, $\varphi_{e_0} \in \mathcal{Z}^*$. □

Notation: For the next proof and for the proofs of Theorems 30 and 31 below, we introduce a low-complexity way to encode lists of numbers. For each $x, y \in \mathbb{N}$ ($\cong \{\mathbf{0},\mathbf{1}\}^*$), let $x \diamond y$ = the concatenation of x and y. For each $x \in \mathbb{N}$, let $E(x) = \mathbf{1}^{|x|}\mathbf{0} \diamond x$. Note that $\{\, E(x) \mid x \in \mathbb{N} \,\}$ is a collection of prefix codes [32, §1.4]. Let $[\,] = 0$, and for each $x_0, \ldots, x_k \in \mathbb{N}$, let $[x_0,\ldots,x_k] = E(x_0) \diamond \cdots \diamond E(x_k)$. Elements of $\mathbb{N}$ not of the form $[x_0,\ldots,x_k]$

are 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.

Proof (of Theorem 25). Fix a polynomial q. *Terminology:* We again say that p is *available* at w when $\Phi_p(w) \leq q(|w|)$. For each σ, define the set

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

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

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

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,

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

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 polynomial-time computable, it is straightforward that φ_u is polynomial-time computable. So without loss of generality, we assume that Φ_u is polynomially bounded. Thus by Proposition 23(c), 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 $\Phi_{e_0}(y) \leq q_0(|y|)$. Hence, $\varphi_{e_0} \in \mathcal{PF}$. Thus, $\lambda x \centerdot \varphi_d(e_0,x)$ 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 $\varphi_d(e_0,x)$. Thus, by the definition of rng_1, for all but finitely many y, $p \notin \{\, p_1^y,\ldots,p_{k_y}^y \,\}$ where $1 + [p_1^y,\ldots,p_{k_y}^y] = \text{rng}_1(d,e_0,y)$. Hence, 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 y, then $\varphi_p(y)\downarrow$ and $\varphi_{e_0}(y)\downarrow = 1 \dot{-} \varphi_p(y) \neq \varphi_p(y)$. 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. $\square$

By a more delicate choice of complexity classes 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 27. *Suppose* $\mathbf{M}$ BC^0*-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* $(\forall a)(\overset{\infty}{\forall} n)(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f|_n)}(x) > a \cdot (|x|+1)^k]$.

Theorem 28. *Suppose* $\mathbf{M}$ EX^**-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* $(\forall a)(\overset{\infty}{\forall} x)[\Phi_{\mathbf{M}(f)}(x) > a \cdot (|x|+1)^k]$.

Interpretation. Let $\mathbf{M}$, k, s and f be as in Theorem 27.[12] 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 28.

Proof (of Theorem 27). Let $s' \in \mathcal{LS}low$ be such that $\lim_{n\to\infty} \frac{(s'(n))^2}{s(n)} = 0$. (Without loss of generality we assume s and s' are everywhere nonzero.) The construction is identical to the one given in the proof of Theorem 25 with q replaced by $\lambda n \centerdot s'(n) \cdot (n+1)^k$ and rng_1 replaced by $\mathrm{rng}_{1,s'}$.

Let us consider the cost of computing the function 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 $T(p, y, \mathbf{0}^{s'(|y|)\cdot(|y|+1)^k}) =$

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

1. By standard time-constructibility results [3], given y (in dyadic representation), constructing a string of $\mathbf{0}$'s of length ℓ can be done in time $O(\ell)$. Hence by Proposition 22, 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 24 that $\mathrm{rng}_{1,s'}$ is linear time computable *and*, for all d, e, and y, $|\mathrm{rng}_1(d,e,y)| \leq s'(\max(|d|,|e|,|y|))$. It thus follows that when $\mathrm{rng}_1(d,e,y) = 1 + [p_1,\dots,p_m]$, each of m, $|p_1|,\dots,|p_m|$ is less than $s'(\max(|d|,|e|,|y|))$. Hence we have 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 22, 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 23(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, its contribution can be absorbed 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 23(c), 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 25. $\square$

The proof of Theorem 28 is left to the reader. We note that in Theorems 27 and 28 we could have replaced $\mathcal{QF}_s^k$ and "$\Phi_{\mathbf{M}(f|_n)}(x) > a\cdot(|x|+1)^k$" with $\mathrm{DTIME}(T_2(n))$ and "$\Phi_{\mathbf{M}(f|_n)}(x) > T_1(|x|)$" where T_2 is a nonzero, fully time-constructible function [3] and $\lim_{n\to\infty}(T_1(n)\log T_1(n)/T_2(n)) = 0$. The cost of this would be somewhat more involved proofs. Analogous remarks hold for Corollaries 9 and 16 above and Theorem 29 below.

7 Infinitely often results for 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 27. 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 29. *Suppose* $\mathbf{M}$ BC^m*-identifies* $\mathcal{QF}_s^k$*, where* $k \geq 1$ *and* $s \in \mathcal{LS}low$*. Then there is an* $f \in \mathcal{Z}^*$ *such that* $(\forall a)(\overset{\infty}{\forall} n)(\overset{\infty}{\exists} x)[\Phi_{\mathbf{M}(f|_n)}(x) > a\cdot(|x|+1)^k]$.

The proof is a straightforward modification of Theorem 27's proof; however, to prove Theorem 29 we need to diagonalize over $m+1$ points at once. It is *not* possible to replace the $(\overset{\infty}{\exists} x)$ in Theorem 29 with an $(\overset{\infty}{\forall} x)$ as shown by:

Theorem 30. *There is an* $\mathbf{M}$ *that both:*

(a) EX*-identifies* $\mathcal{PF}^1$ *and moreover, for each* $f \in \mathcal{PF}^1$*, there is a constant* c_f *such that* $(\overset{\infty}{\exists} x)[\,\Phi_{\mathbf{M}(f)}(x) \leq c_f \cdot |x|\,]$*, and*

(b) BC^1*-identifies* $\mathcal{PF}$ *using programs having polynomial-bounded run times.*

Proof. Define $g\colon \mathbb{N} \to \mathbb{N}$ recursively by $g(0) = \mathbf{0}$ and $g(m+1) = \mathbf{0}^k$, where $k = 2^{2^{|g(m)|}}$. Clearly g is strictly increasing. It is straightforward that range(g) is linear time decidable and, in fact, that:

$$inv_g \;=\; \lambda x \centerdot \begin{cases} 0, & \text{if } x \notin \text{range}(g); \\ 1 + [g(0), \ldots, g(m-1)], & \text{if } m = g^{-1}(x). \end{cases}$$

is $O(|x|)$ time computable and $inv_g \in O(\log_2 |x|)$. (Recall that $[\,\cdot\,]$ is our linear-time encoding of lists.) Our goal is to define an $\mathbf{M}$ such that:

(1) For each $f \in \mathcal{PF}^1$, $\mathbf{M}$ EX-identifies f and there is a constant c_f such that, $\Phi_{\mathbf{M}(f)}(x) \leq c_f \cdot |x|$ for all $x \in$ range(g).
(2) For each $f \in (\mathcal{PF} - \mathcal{PF}^1)$, for sufficiently large n, $\varphi_{\mathbf{M}(f|_n)}(x) = f(x)$ for all x except perhaps for one $x \in$ range(g) (so, $\mathbf{M}$ BC^1 identifies f).

The fact that the elements of range(g) are spaced so far apart will help with the "looking back" part of the construction.

Let u be such that, for all i_0, i_1, and x,

$$u(i_0, i_1, x) \;=\; \begin{cases} 0, & \text{if for some } m,\ g(m) = x \text{ and, for each } w \in \{\, g(0), \ldots, g(m-1) \,\}, \text{ we have that } \Phi_{i_0}(w) \leq \sqrt{|x|},\ \Phi_{i_1}(w) \leq \sqrt{|x|}, \text{ and } \varphi_{i_0}(w) = \varphi_{i_1}(w); \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

It follows from the noted properties of inv_g, Proposition 22, and a little algebra that u is $O(|i_0| + |i_1| + |x|)$ time computable.

Suppose for the moment that i is a φ-program with polynomial run time. Then it follows from the noted properties of inv_g that, for all but finitely many m, we have, for each $w \in \{\, g(0), \ldots, g(m-1) \,\}$, that $\Phi_i(w) \leq \sqrt{|g(m)|}$. From this and (4) we have:

Claim: Suppose i_0 and i_1 are φ-programs with polynomial run times and, for all $x \in \text{range}(g)$, $\varphi_{i_0}(x) = \varphi_{i_1}(x)$. Then, for all but finitely many $x \in \text{range}(g)$, $u(i_0, i_1, x) = 0$.

Now, since u is linear time computable, it follows from Proposition 23 that there is a recursive h and a constant c_0 such that, for all i_0, i_1, and x:

$$\varphi_{h(i_0,i_1)}(x) \;=\; \begin{cases} \varphi_{i_0}(x), & \text{if } u(i_0,i_1,x) = 0; \\ \varphi_{i_1}(x), & \text{otherwise.} \end{cases} \tag{5}$$

$$\Phi_{h(i_0,i_1)}(x) \;\le\; \begin{cases} \Phi_{i_0}(x) + c_0 \cdot (|i_0| + |i_1| + |x|), & \text{if } u(i_0,i_1,x) = 0; \\ \Phi_{i_1}(x) + c_0 \cdot (|i_0| + |i_1| + |x|), & \text{otherwise.} \end{cases} \tag{6}$$

Fix $\mathbf{M}_0$ and $\mathbf{M}_1$ such that (i) $\mathbf{M}_0$ EX-identifies $\mathcal{PF}^1$ and outputs only conjectures that run in linear time, and (ii) $\mathbf{M}_1$ EX-identifies $\mathcal{PF}$ and outputs only conjectures that run in polynomial time. Define $\mathbf{M}$ by:

$$\mathbf{M}(\sigma) \;=\; h(\mathbf{M}_0(\sigma), \mathbf{M}_1(\sigma)). \tag{7}$$

Since both $\mathbf{M}_0$ and $\mathbf{M}_1$ output only conjectures with polynomial run times, it follows from (6) and (7) that $\mathbf{M}$ also outputs only conjectures with polynomial run times.

Suppose $f \in \mathcal{PF}^1$. Then both $\mathbf{M}_0$ and $\mathbf{M}_1$ EX-identify f. Let n be the maximum of the points of convergence of $\mathbf{M}_0$ and $\mathbf{M}_1$ on f. Then by (7), $\mathbf{M}(f|_n) = \mathbf{M}(f)$. Since $\varphi_{\mathbf{M}_0(f)} = \varphi_{\mathbf{M}_1(f)}$, it follows from (5) and (7) that $\varphi_{\mathbf{M}(f)} = f$ and it follows from the claim and (6) that there is a constant c_f such that, for all $x \in \text{range}(g)$, $\Phi_{\mathbf{M}(f)} \le c_f \cdot |x|$. Therefore, part (a) follows.

Suppose $f \in (\mathcal{PF} - \mathcal{PF}^1)$. Then $\mathbf{M}_1$ EX-identifies f. Let n be greater than or equal to the point of convergence of $\mathbf{M}_1$ on f and set $i_0 = \mathbf{M}_0(f|_n)$, $i_1 = \mathbf{M}_1(f|_n)$, and $p = \mathbf{M}(f|_n)$. Note that $\varphi_{i_1} = f$. We claim that $\varphi_p =^1 f$. If $\varphi_p = f$, we are done. So suppose that for some x, $\varphi_p(x) \neq f(x)$. By (4), (5), and (7) it follows that $x \in \text{range}(g)$ and that $u(i_0, i_1, x) = 0$. Let x_0 be the least such x. But then, for each $x \in \text{range}(g)$ with $x = g(m) > x_0$, we have that $x_0 \in \{\, g(0), \dots, g(m-1) \,\}$ and $\varphi_{i_0}(x_0) \neq \varphi_{i_1}(x_0)$. Thus, $u(i_0, i_1, x)$ cannot be 0. Therefore, $u(i_0, i_1, x) = 1$ for all $x > x_0$. Hence, by (5) and (7), $\varphi_p =^1 \varphi_{i_1} = f$ as required. Therefore, part (b) follows. $\square$

8 Positive, almost everywhere results for BC*

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

Theorem 31. *There is an IIM* $\mathbf{M}_*$ *that* BC**-identifies* $\mathcal{PF}$ *with all outputs running in polynomial time and such that:*

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

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

Interpretation. In contrast to Theorems 25 through 29, 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 9 applies to $\mathbf{M}_*$ of the above theorem. Hence, for each $\ell \geq 1$, 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 31); *however*, by Corollary 9, $\mathbf{M}_*(f)$ 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.

Proof. For each n, let $\mathcal{P}_n = \{\, \langle k, a, p\rangle \;\mid\; k, a, p \leq n \,\}$, *triples*$(n) = [\langle k_1, a_1, p_1\rangle, \ldots, \langle k_m, a_m, p_m\rangle]$, where the list enumerates $\mathcal{P}_n$ in lexicographical order, i.e., $\langle 0,0,0\rangle, \langle 0,0,1\rangle, \ldots, \langle 0,0,n\rangle, \langle 0,1,0\rangle, \ldots, \langle n,n,n\rangle$, and $\ell(n)$ = the length of *triples*(n). Let d be a φ-program such that, for all j_0, j_1, n, and x,

$$\varphi_d(j_0, j_1, n, x) \;= \tag{8}$$
$$\begin{cases} \uparrow, & \text{if (i): for some } w \leq x,\ \ \varphi_{j_0}(w)\uparrow \text{ or } \varphi_{j_1}(w)\uparrow; \\ 0, & \text{if (ii): for all } w \leq x,\ \ \varphi_{j_0}(w)\downarrow = \varphi_{j_1}(w)\downarrow; \\ i, & \text{if (iii): not [(i) or (ii)] and } i \text{ is the least number,} \\ & \quad \text{if any, such that } \langle k_i, a_i, p_i\rangle \in \textit{triples}(n) \text{ and,} \\ & \quad \text{for each } w \leq x,\ \ \Phi_{p_i}(x) \leq a_i \cdot (|w|+1)^{k_i} \text{ and} \\ & \quad \varphi_{p_i}(w) = \varphi_{j_1}(w); \\ \ell(n)+1, & \text{otherwise.} \end{cases}$$

Since rng_3 is linear time computable, it follows from parts (a) and (c) of

Proposition 23 that there is a recursive function g and, for each j_0, j_1, and n, there is a constant $c_{j_0,j_1,n}$ such that, for all y:

$$\varphi_{g(j_0,j_1,n)}(y) \;=\; \begin{cases} \varphi_{j_0}(y), & \text{if } \mathrm{rng}_3(d,j_0,j_1,n,y)=0; \\ \varphi_{p_1}(y), & \text{if } \mathrm{rng}_3(d,j_0,j_1,n,y)=1 \text{ and } \Phi_{p_1}(y) \le a_1\cdot(|y|+1)^{k_1}; \\ \vdots & \quad\vdots \\ \varphi_{p_{\ell(n)}}(y), & \text{if } \mathrm{rng}_3(d,j_0,j_1,n,y)=\ell(n) \text{ and } \Phi_{p_{\ell(n)}}(y) \le \\ & \quad a_{\ell(n)}\cdot(|y|+1)^{k_{\ell(n)}}; \\ \varphi_{j_1}(y), & \text{otherwise;} \end{cases} \tag{9}$$

$$\Phi_{g(j_0,j_1,n)}(y) \;=\; \begin{cases} \Phi_{j_0}(y)+c_{j_0,j_1,n}\cdot(|y|+1), & \text{if } \mathrm{rng}_3(d,j_0,j_1,n,y)=0; \\ c_{j_0,j_1,n}\cdot a_i\cdot(|y|+1)^{k_i}, & \text{if } 0<\mathrm{rng}_3(d,j_0,j_1,n,y)=i\le\ell(n) \\ & \quad \text{and } \Phi_{p_i}(y)\le a_i(|y|+1)^{k_i}; \\ \Phi_{j_1}(y)+c_{j_0,j_1,n}\cdot(|y|+1), & \text{otherwise;} \end{cases} \tag{10}$$

where $[\langle k_1,a_1,p_1\rangle,\ldots,\langle k_{\ell(n)},a_{\ell(n)},p_{\ell(n)}\rangle] = \mathit{triples}(n)$.

Now let $\mathbf{M}_0$ be an IIM that EX-identifies all of $\mathcal{PF}^1$ and that outputs only conjectures that run in linear time, and let $\mathbf{M}_1$ be an IIM that EX-identifies all of $\mathcal{PF}$ and that outputs only conjectures that run in polynomial time. Moreover, we assume without loss of generality that, for each $f\in(\mathcal{PF}-\mathcal{PF}^1)$, $\mathbf{M}_0$ on f has infinitely many mind changes. For each σ, define

$$\mathbf{M}_*(\sigma) \;=\; g(\mathbf{M}_0(\sigma),\mathbf{M}_1(\sigma),m_\sigma), \text{ where}$$

$$m_\sigma \;=\; \max\left\{ m \;\middle|\; \begin{array}{c} 0 \;<\; m \;\le\; |\sigma| \text{ and either} \\ \mathbf{M}_0(\sigma|_{m-1}) \;\ne\; \mathbf{M}_0(\sigma|_m) \text{ or} \\ \mathbf{M}_1(\sigma|_{m-1}) \ne \mathbf{M}_1(\sigma|_m) \end{array} \right\}.$$

(Recall that $\max(\emptyset)=0$.)

The argument for part (b). Suppose $f\in\mathcal{PF}^1$. Let m be the maximum of the points of convergence of $\mathbf{M}_0$ and $\mathbf{M}_1$ on f. Thus, for all $n\ge m$, $m_{f|_n}=m$. Let $j_0=\mathbf{M}_0(f|_m)=\mathbf{M}_0(f)$ and $j_1=\mathbf{M}_1(f|_m)=\mathbf{M}_1(f)$. By the definition of $\mathbf{M}_*$, we have that, for all $n\ge m$, $\mathbf{M}_*(f|_n)=g(j_0,j_1,m)$. Since $\varphi_{j_0}=\varphi_{j_1}$, by (8) we have, for all n and x, $\varphi_d(j_0,j_1,n,x)=0$. Hence by (9) and (10), $\varphi_{g(j_0,j_1,m)}=\varphi_{j_0}$ and, for all y, $\Phi_{g(j_0,j_1,m)}(y)\le\Phi_{j_0}(y)+c_{j_0,j_1,m}\cdot(|y|+1)$. By our hypotheses on $\mathbf{M}_0$, $\varphi_{j_0}=f$ and Φ_{j_0} is linearly bounded. Therefore, part (b) follows.

The argument for part (a). Suppose $f \in (\mathcal{PF} - \mathcal{PF}^1)$. Let m be the point of convergence of $\mathbf{M}_1$ on f and $j_1 = \mathbf{M}_1(f|_m) = \mathbf{M}_1(f)$. Let k be the least number such that $f \in \mathcal{PF}^k$and let a be the least number such that

$$\varphi_p = f \text{ and, for all } x,\ \Phi_p(x) \leq a \cdot (|x| + 1)^k \tag{11}$$

for some p. Let p be the least number such that (11) holds. Finally, let σ be an initial segment of f with the property that $m_\sigma \geq \max(m, k, a, p)$. Since $f \in (\mathcal{PF} - \mathcal{PF}^1)$, by hypothesis $\mathbf{M}_0$ on f makes infinitely many mind changes, hence, all but finitely many initial segments of f have this property. Let $j_0 = \mathbf{M}_0(\sigma)$. By our definition of $\mathbf{M}_*$, $\mathbf{M}_*(\sigma) = g(j_0, j_1, m_\sigma)$. Part (a) will thus follow if we show that $\varphi_{g(j_0,j_1,m_\sigma)} =^* f$ and that $\Phi_{g(j_0,j_1,m_\sigma)}$ has an $O(n^k)$ bound.

By our hypotheses on $\mathbf{M}_0$ and $\mathbf{M}_1$ and our choices of m and m_σ, it follows that φ_{j_0} is total and $\neq f$ and that $\varphi_{j_1} = f$. Hence, by (8) and our choices of k, a, p, and m_σ, it follows that, for all but finitely many x, we have $\varphi_d(j_0, j_1, m_\sigma, x) = i$, where $\langle k_i, a_i, p_i \rangle$ is the element of *triples*(m_σ) with $k_i = k$, $a_i = a$, and $p_i = p$. It thus follows from (9), (10), and (11) that, for all but finitely many y:

$$\begin{aligned}
\varphi_{g(j_0,j_1,m_\sigma)}(y) &= \varphi_{p_i}(y) = \varphi_p(y) = f(y). \\
\Phi_{g(j_0,j_1,m_\sigma)}(y) &\leq \Phi_p(y) + c_{j_0,j_1,m_\sigma} \cdot (|y| + 1) \\
&\leq a \cdot (|y| + 1)^k + c_{j_0,j_1,m_\sigma} \cdot (|y| + 1).
\end{aligned}$$

Therefore $g(j_0, j_1, m_\sigma)$ is as required and part (a) follows. □

A generalization of Theorem 31 also holds by a similar proof. In the generalization one introduces an arbitrary $j \geq 1$ but requires $k \geq j$ in part (a); 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. Thanks to the anonymous referee for several suggestions that helped tighten and improve the paper. 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

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, 2nd Edition, MIT Press, 2001.

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

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

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

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

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

[7] S. Jain, D. Osherson, J. Royer, A. Sharma, Systems that Learn: An Introduction to Learning Theory, 2nd Edition, MIT Press, Cambridge, Mass., 1999.

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

[9] J. Royer, J. Case, Subrecursive Programming Systems: Complexity *&* Succinctness, Birkhäuser, 1994.

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

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

[12] E. Mendelson, Introduction to Mathematical Logic, 4th Edition, Chapman & Hall, London, 1997.

[13] E. Bernstein, U. Vazirani, Quantum complexity theory, SIAM Journal of Computing 26 (1997) 1411–1473.

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

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

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

[17] K. Regan, Minimum-complexity pairing functions, Journal of Computer and System Sciences 45 (1992) 285–295.

[18] N. Jones, Computability and Complexity From a Programming Perspective, MIT Press, 1997.

[19] P. Chew, 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.

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

[21] J. Gill, Computational complexity of probabilistic complexity classes, SIAM Journal of Computing 6 (1977) 675695.

[22] J. A. Bārzdiņš, 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.

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

[24] L. Fortnow, Counting complexity, in: A. Selman, L. Hemaspaandra (Eds.), Complexity Theory Retrospective II, Springer Verlag, 1997, pp. 81–107.

[25] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[26] F. Hennie, R. Stearns, Two-tape simulation of multitape Turing machines, Journal of the ACM 13 (1966) 433–446.

[27] M. Machtey, K. Winklmann, P. Young, Simple Gödel numberings, SIAM Journal of Computing 7 (1978) 39–60.

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

[29] D. Kozen, Indexings of subrecursive classes, Theoretical Computer Science 11 (1980) 277–301.

[30] S. Kleene, General recursive functions of natural numbers, Math. Ann. 112 (1936) 727–742.

[31] K. Chen, Tradeoffs in machine inductive inference, Ph.D. thesis, SUNY at Buffalo (1981).

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