BANISHING ROBUST TURING COMPLETENESS *

LANE A. HEMASPAANDRA †
Department of Computer Science
University of Rochester
Rochester, New York 14627, USA

SANJAY JAIN ‡
Institute of Systems Science
National University of Singapore
Singapore 0511
Republic of Singapore

NIKOLAI K. VERESHCHAGIN
The Institute of New Technologies
Kirovogradskaja 11, Moscow, 113587 Russia

Received
Revised
Communicated by T. Ito

ABSTRACT

This paper proves that "promise classes" are so fragilely structured that they do not robustly (i.e. with respect to all oracles) possess Turing-hard sets even in classes far larger than themselves. In particular, this paper shows that FewP does not robustly possess Turing hard sets for UP ∩ coUP and IP ∩ coIP does not robustly possess Turing hard sets for ZPP. It follows that ZPP, R, coR, UP∩coUP, UP, FewP∩coFewP, FewP, and IP ∩ coIP do not robustly possess Turing complete sets. This both resolves open questions of whether promise classes lacking robust downward closure under Turing reductions (e.g., R, UP, FewP) might robustly have Turing complete sets, and extends the range of classes known not to robustly contain many-one complete sets.

Keywords: Structural complexity theory; Polynomial-time reductions; Complete languages.

---

*Some of these results were reported at Logic at TVER '92—Symposium on Logical Foundations of Computer Science.

†Supported in part by a Hewlett-Packard Corporation equipment grant and by the National Science Foundation under grants CCR-8809174/CCR-8996198, CCR-8957604, and NSF-INT-9116781/JSPS-ENG-207.

‡Part of the work done while the second author was at the University of Rochester and supported in part by the National Science Foundation grant CCR-8320136 to the University of Rochester.

1 Introduction

Complete languages have long been a useful tool in complexity theory. Much of our knowledge about NP comes from studying the NP-complete set SAT (see, e.g., [28]). Most common complexity classes—NP, coNP, PSPACE, etc.—have many-one complete sets[a] that help us study them. Sipser noted, however, that some classes may lack complete sets [35]. His paper sparked much research into which classes robustly [33,18] (i.e., with respect to all oracles [4]) possess complete languages, and what strengths of completeness results (e.g., many-one or Turing) can be obtained. The crucial property of classes such as NP and PSPACE that causes them to robustly possess many-one complete sets is the existence of a recursive enumeration of machines covering the languages in the class.[b] However several natural classes, such as UP, R, and BPP, do not have any obvious recursive enumeration of machines covering the class. This raises the possibility that these classes may not robustly possess complete sets.

Sipser showed that R and NP $\cap$ coNP do not robustly possess many-one complete languages [35]. Hartmanis and Hemachandra showed that UP—unambiguous polynomial time (Section 2)—does not robustly possess many-one complete languages, and noted that if UP does have complete languages then UP has complete languages with an unusually simple form—the intersection of SAT with a set in P [18]. In related work, Regan [30] proved that a class $\mathcal{C}$ that is closed downward under $\leq_m^p$ and admits padding has a constructively valid programming system $[Q_k]_{k=1}^{\infty}$ over a sufficiently strong logic F if and only if $\mathcal{C}$ has a complete set under $\leq_m^p$. It is also known that these classes tend to have different positive relativization [21] results than classes that robustly possess complete sets.

One way of strengthening the above theorems would be to show that these classes do not robustly possess complete sets even with respect to reducibilities more flexible than many-one reductions, e.g., k-truth-table, positive truth-table, truth-table, and ultimately Turing reductions [26]. This would show that the structure of these classes differs markedly from that of P, NP, PSPACE, and so on. Gurevich showed that NP $\cap$ coNP has many-one complete languages if and only if it has Turing complete languages ([17], see also [20]). Ambos-

[a] Throughout this paper, we are concerned only with polynomial-time reducibilities.

[b] That is : let $\{T_i\}$ be any standard naming of Turing machines; there exists a recursively enumerable set A such that (1) PSPACE $= \{L(T_i) : i \in A\}$ and (2) each $T_i, i \in A$, runs in polynomial space.

Spies's elegant generalization of this states that for any class $\mathcal{C}$ closed under Turing reductions, $\mathcal{C}$ has Turing complete sets if and only if $\mathcal{C}$ has many-one complete sets [3]. In particular, it follows, from the result of Sipser [35], that NP $\cap$ coNP does not robustly possess Turing complete sets [17]. Similarly, since $\mathrm{P}^{\mathrm{BPP}} = \mathrm{BPP}$ [39], from [18]'s proof that BPP does not robustly possess many-one complete sets it follows that BPP does not robustly possess Turing complete sets.

Since UP $\cap$ coUP and ZPP are closed downward under Turing reductions ([39] for the ZPP case), Ambos-Spies's result mentioned yields: UP $\cap$ coUP and ZPP have Turing complete sets if and only if they have m-complete sets.

However, Ambos-Spies's result does not apply to R, UP, FewP, FewP $\cap$ coFewP, or to other classes not known to be closed under Turing reductions.

Furthermore, the technique used to show that R and UP do not robustly possess many-one complete languages was an indirect proof [35,18], that does not seem to generalize to Turing completeness.

In fact, classes such as R, UP, and FewP, have a complex structure. Machines for these classes must incorporate a promise (e.g., never having more than one accepting computation path), and thus these classes have been referred to as "promise classes" [21]. This paper shows that this promise structure precludes such promise classes as R, UP, and FewP from robustly possessing even Turing complete sets; our proofs exploit the promise-induced limited combinatorial control of probabilistic and nondeterministic machines to corrupt candidates for Turing completeness. Indeed, the promises are so exacting that quite large classes do not robustly contain sets that are hard for these classes (or even for subclasses of these classes).

Section 3 proves that FewP does not robustly possess Turing hard sets for UP $\cap$ coUP. Section 4 proves that IP $\cap$ coIP does not robustly possess Turing hard sets for ZPP.[c] It immediately follows from the above results that ZPP, R, coR, UP $\cap$ coUP, UP, FewP $\cap$ coFewP, FewP, and IP $\cap$ coIP do not robustly

[c] Recently, Shamir [34,27] has shown that IP=PSPACE. Thus in the real world IP = coIP = IP $\cap$ coIP = PSPACE. It follows that Theorem 4.1 does not hold for $A = \emptyset$, i.e., IP $\cap$ coIP, does have Turing hard languages for ZPP, R, coR and BPP. Thus Theorem 4.1 gives another example of oracle result which does not hold in the real world (though some of its corollaries may still be true in the real world). However, IP is not robustly equal to PSPACE [11]. Shamir's non-relativizing technique is not known to apply to classes other than those having to do with interactive proofs (though sometimes the connection to interactive proofs is somewhat disguised [9]), and in particular is not known to apply to BPP. Theorem 4.1 thus suggests the inability of a certain broad body of techniques to resolve the problem of whether, in the real world, BPP has Turing hard sets for ZPP, R, and coR.

possess Turing complete sets.

2 Preliminaries

Let $\mathcal{N}$ denote the set of natural numbers. Σ is an alphabet set, usually $\{0,1\}$. ϵ denotes the empty string. A language is a subset of Σ^*. For two sets L_1 and L_2, $L_1 \triangle L_2$ denotes the set $(L_1 - L_2) \cup (L_2 - L_1)$. $\emptyset$ denotes the empty set. $M_0, M_1, \ldots$ denotes some standard enumeration of polynomial-time deterministic (oracle) Turing machines. $N_1, N_2, \ldots$ denotes some standard enumeration of polynomial-time nondeterministic (oracle) Turing machines. $B_1, B_2, \ldots$ denotes some standard enumeration of polynomial-time probabilistic (oracle) Turing machines [13]. We assume that the running time of machine M_i (N_i) ((B_i)) is bounded by deterministic (nondeterministic) ((probabilistic)) time $r_i(n) = n^i + i$. P denotes the class of all languages accepted by some polynomial-time deterministic Turing machine [22]. NP denotes the class of languages accepted by polynomial-time nondeterministic Turing machines and coNP denotes the class of languages whose complements are in NP [22]. $L(M)$ denotes the language accepted by the machine M. $L(M^A)$ denotes the language accepted by the oracle machine M with the oracle A [4,22].

$\overline{A}$ denotes the complement of A, i.e., $\Sigma^* - A$. χ_A denotes the characteristic function of A. $|x|$ denotes the length of x. $A^{\leq n}$ denotes the set of strings in A with length at most n. $\langle \cdot, \cdot \rangle$ denotes a standard one-to-one, polynomial time computable and polynomial time invertible pairing of natural numbers (see [5, 31]). Similarly, $\langle \cdot, \cdot, \cdot \rangle$ denotes a standard one-to-one, polynomial time computable and polynomial time invertible encoding of triples of natural numbers.

We now review the definitions of various complexity classes discussed in this paper. Implicitly, the relativized version of each of these complexity classes is defined by allowing the nondeterministic (probabilistic) machine(s) of the definition access to some oracle.

Definition 2.1 *[36]*

(Unambiguous Polynomial Time) UP = $\{L$: *there is a nondeterministic polynomial-time Turing machine N such that $L = L(N)$, and for all x, the computation of $N(x)$ has at most one accepting path*$\}$. *We say that a machine N is* categorical *if it has at most one accepting path for every input.*

coUP $= \{L : \overline{L} \in \text{UP}\}$.

UP captures the power of unambiguous computation; UP is the class of problems that have (for some NP machine) unique witnesses. That is, if there is an NP machine N accepting L and for every input x the computation of $N(x)$ has at most one accepting path (i.e., N is a categorical machine), then we say $L \in$ UP.

Recently, UP has come to play a crucial role in both cryptography and structural complexity theory. In cryptography, Ko and Grollmann and Selman have shown that one-way functions[d] exist if and only if P $\neq$ UP [23,16], and one-way functions whose range[e] is in P exist if and only if P $\neq$ UP $\cap$ coUP [16]. Thus, we suspect that P $\neq$ UP because we suspect that one-way functions exist.

Curiously—in light of the results in this paper and [18]—Ko has shown that the *operator* version of UP does have complete sets [23].

The following definition is a generalization of the class UP where we allow the nondeterministic machine to have at most polynomially many accepting paths.

Definition 2.2 *[1,2]*

FewP $= \{L :$ *there is a nondeterministic polynomial-time Turing machine* N *and a* j *such that* $L = L(N)$*, and for all* x*, the computation of* $N(x)$ *has at most* $|x|^j + j$ *accepting paths*$\}$.

coFewP $= \{L : \overline{L} \in \text{FewP}\}$.

We say that N_i^A is FewP^A-like iff $(\forall x)[$ the number of accepting paths of N_i^A on input x is at most $|x|^i + i]$. Note that we are not claiming that it is easy to determine whether N_i^A is FewP^A-like. The following proposition merely reflects the fact that in many natural enumerations each machine in the enumeration essentially appears infinitely often, give or take vacuously padding the machine with unreachable states.

Proposition 2.3 *There exists an enumeration* $N_0', N_1', \ldots$ *of nondeterministic Turing machines such that, for all* i*, run time of* N_i' *(on inputs of length* n*) is bounded by* $n^i + i$ *and*
FewP $= \{L : (\exists i)[L = L(N_i')$ *and* $(\forall x)[$ *number of accepting paths of* N_i' *on* x *is* $\leq |x|^i + i]]\}$.

[d] A function f is *honest* if $(\exists k)(\forall x)[|f(x)|^k + k \geq |x|]$. A *one-way function* is a total, single-valued, one-to-one, honest, polynomial time computable function f such that f^{-1} (which will be a partial function if range$(f) \neq \Sigma^*$) is not computable in polynomial time [16].

[e] *Range*$(f) = \{f(i) : i \in \Sigma^*\}$.

Thus we assume, without loss of generality, that our standard enumeration is one such enumeration. Note that Proposition 2.3 holds robustly. In our relativizations, this will allow us to, for each possible oracle machine, instantly discard machine N_i if we notice that N_i^A is not FewP^A-like in the relativized world, A, we construct (rather than having to explicitly pair each machine with every fewness bound).

Definition 2.4 *[13]*

(Random Polynomial Time) $\mathrm{R} = \{L : $ *there is a probabilistic polynomial-time Turing machine* B *such that* $L = L(B)$*, and for all* x*, either* $B(x)$ *has no accepting paths, or* $B(x)$ *accepts with probability* $\geq 1/2\}$.

$\mathrm{coR} = \{L : \overline{L} \in \mathrm{R}\}$.

Without loss of generality we can assume that the computation tree of a machine accepting a language in R is a full binary tree with paths through the tree corresponding to sequences of coin tosses [5].

R, random polynomial time, is the complexity class that captures the power of probabilistic computation with one-sided error. When an R machine accepts, it is always correct; however, when it rejects it may be incorrect. In relativized worlds, the computational powers of random polynomial time and of unambiguous polynomial time are incomparable, and have both been studied in detail [29, 12,6,10]. We say that B_i^A is R^A-like, iff $(\forall x)[$ either B_i^A on x has no accepting paths or the probability that B_i^A accepts x is at least $1/2]$.

Definition 2.5 *[13]* $\mathrm{ZPP} = \mathrm{R} \cap \mathrm{coR}$.

The class of languages that are in ZPP are exactly those languages which can be solved in expected polynomial time [13].

Definition 2.6 *[13]*

(Bounded Probabilistic Polynomial Time) $\mathrm{BPP} = \{L :$ *there is a probabilistic polynomial time Turing machine* B *such that,* $[x \in L \Rightarrow B(x)$ *accepts with probability* $\geq 2/3]$ *and* $[x \notin L \Rightarrow B(x)$ *accepts with probability* $\leq 1/3]\}$.

BPP is a complexity class that captures the power of probabilistic computation with *bounded* two sided error.

We now discuss interactive proofs [15]. Consider a language L and an input x. Suppose a *prover*, P, is trying to convince a polynomial-time, probabilistic, *verifier* V, that x indeed belongs to L. During this process, the verifier may ask

certain questions to the prover. We assume that the prover P is deterministic in the sense that the answer provided by the prover, on any question, depends only on x, the question asked, and the set of earlier questions asked. However we do not place any resource bound on the prover. Indeed P itself may be a non-recursive procedure. After the interaction, the verifier may accept or reject the input. For the verifier to *accept* the language L, we would like that in cases when x belongs to L, the prover should be able to convince the verifier, with high probability, that x belongs to L. However, if x does not belong to L, then no prover should be able to convince the verifier, with non-negligible probability, that $x \in L$.

Definition 2.7 *We say that a verifier V accepts L iff:*
$(\forall x \notin L)(\forall P)[Prob(V \text{ with prover } P \text{ accepts } x) \leq 1/3]$, *and*
$(\exists P)(\forall x \in L)[Prob(V \text{ with prover } P \text{ accepts } x) \geq 2/3]$.

Note that for a particular verifier V, there can be at most one L that V accepts. We denote by $L(V)$ the language, if any, that the verifier V accepts.

Definition 2.8 *[15]*
$\mathrm{IP} = \{L : (\exists V)[V \text{ accepts } L]\}$.
$\mathrm{coIP} = \{L : \overline{L} \in \mathrm{IP}\}$.

We say that V_i^A is IP_i^A-like iff there exists a language L such that V_i^A accepts L.

Note that a machine may be FewP-like (R-like, IP-like) with respect to some oracles while not being FewP-like (R-like, IP-like) with respect to other oracles; these are properties of machine/oracle pairs.

We let $V_0, V_1, \ldots$ be a standard enumeration of all polynomial time verifiers. We assume that the runtime of verifier V_i, on inputs of length n, is bounded by $n^i + i$.

It follows immediately from the definitions that $(\forall A)[\mathrm{R}^A \subseteq \mathrm{BPP}^A \subseteq \mathrm{IP}^A \cap \mathrm{coIP}^A]$.

For background, we first define Turing reductions and completeness in the real (unrelativized) world.

Definition 2.9 *(see [26,5])*

1. $L_1 \leq_T^p L_2$ *if* $L_1 \in \mathrm{P}^{L_2}$.

2. *L is $\leq_T^p$-hard for $\mathcal{C}$ if every set in $\mathcal{C}$ Turing reduces to L (i.e., $(\forall S \in \mathcal{C})[S \leq_T^p L]$). In addition if $L \in \mathcal{C}$ then we say that L is $\leq_T^p$-complete for $\mathcal{C}$.*

If we wish to discuss Turing completeness in relativized worlds, we must address the key question: are the Turing reductions allowed access to the oracle? Definitions 2.10.2 and 2.10.3 answer this question "yes" and "no," respectively. For the following definition $\mathcal{C}$ stands for one of R, UP, or FewP.

Definition 2.10

1. *$L_1 \leq_T^{p,\,A} L_2$ if $L_1 \in \mathrm{P}^{L_2 \oplus A}$.*
2. *L is $\leq_T^{p,\,A}$-hard for $\mathcal{C}^A$ if $(\forall S \in \mathcal{C}^A)[S \leq_T^{p,\,A} L]$. In addition if $L \in \mathcal{C}^A$ then we say that L is $\leq_T^{p,\,A}$-complete for $\mathcal{C}^A$.*
3. *L is $\leq_T^p$-hard for $\mathcal{C}^A$ if $(\forall S \in \mathcal{C}^A)[S \leq_T^p L]$. In addition if $L \in \mathcal{C}^A$ then we say that L is $\leq_T^p$-complete for $\mathcal{C}^A$.*

We suggest that Definition 2.10.2 above is the natural notion of relativized Turing completeness. Adopting it, we prove that there exist oracles A and B such that ZPP^A, R^A, $\mathrm{IP}^A \cap \mathrm{coIP}^A$ have no $\leq_T^{p,A}$-complete sets and UP^B, FewP^B have no $\leq_T^{p,B}$-complete sets. However, for purposes of completeness results, the different notions of relativized Turing reductions stand or fall together.

Lemma 2.11

1. For any oracle A, and class $\mathcal{C}$: $[\mathrm{FewP}^A$ *has $\leq_T^{p,\,A}$-hard sets for $\mathcal{C}$ if and only if it has $\leq_T^p$-hard sets for $\mathcal{C}$].*

2. For any oracle A, and class $\mathcal{C}$: $[\mathrm{UP}^A$ *has $\leq_T^{p,\,A}$-hard sets for $\mathcal{C}$ if and only if it has $\leq_T^p$-hard sets for $\mathcal{C}$].*

3. For any oracle A, and class $\mathcal{C}$: $[\mathrm{R}^A$ *has $\leq_T^{p,\,A}$-hard sets for $\mathcal{C}$ if and only if it has $\leq_T^p$-hard sets for $\mathcal{C}$].*

4. For any oracle A, and class $\mathcal{C}$: $[\mathrm{IP}^A \cap \mathrm{coIP}^A$ *has $\leq_T^{p,\,A}$-hard sets for $\mathcal{C}$ if and only if it has $\leq_T^p$-hard sets for $\mathcal{C}$].*

This is true since if B is $\leq_T^{p,\,A}$-hard for $\mathcal{C}$ then $B \oplus A$ is $\leq_T^p$-hard for $\mathcal{C}$.

The difference between Definitions 2.10.2 and 2.10.3 is essentially the difference between "full" (2.10.2) and "partial" (2.10.3) relativization discussed in [25] and [32, Section 9.3]. [25] describes how this distinction has had a crucial effect on recent research asking whether all NP-complete sets are polynomially

isomorphic [24,14,19]. However, Lemma 2.11 indicates that in our study of Turing completeness, we need not be concerned with this distinction.

3 Robust Completeness and Classes of Limited Ambiguity

This section shows that FewP does not robustly possess Turing hard sets for $\mathrm{UP} \cap \mathrm{coUP}$. It follows immediately that there is an oracle A such that UP^A and FewP^A lack complete sets with respect to $\leq_T^{p,A}$, as well as with respect to all reductions less flexible than $\leq_T^{p,A}$, such as truth-table reductions [26], bounded truth-table reductions [26], etc.

Theorem 3.1 *There is a recursive oracle A such that FewP^A contains no $\leq_T^{p,A}$-hard languages for $\mathrm{UP}^A \cap \mathrm{coUP}^A$.*

Corollary 3.2 *There is a recursive oracle A such that $\mathrm{UP}^A \cap \mathrm{coUP}^A$, UP^A, $\mathrm{FewP}^A \cap \mathrm{coFewP}^A$, and FewP^A contain no $\leq_T^{p,A}$-complete languages.*

Proof of Theorem 3.1: We wish to construct A such that for no $L \in \mathrm{FewP}^A$ is $\mathrm{UP}^A \cap \mathrm{coUP}^A \subseteq \mathrm{P}^L$, which suffices by Lemma 2.11. Each L in FewP^A is, by definition, accepted by a nondeterministic machine N_i^A, which has its number of accepting paths on any input bounded by $n^i + i$ (see Proposition 2.3). Our goal is to show that for each i, either:

1. N_i^A has more than $|x|^i + i$ accepting paths on some input x, or
2. $(\exists \hat{L}_i)[\hat{L}_i \in \mathrm{UP}^A \cap \mathrm{coUP}^A$ and $\hat{L}_i \notin \mathrm{P}^{L(N_i^A)}]$.

The second condition says that some $\mathrm{UP}^A \cap \mathrm{coUP}^A$ language does not Turing reduce to $L(N_i^A)$.

Let $\hat{L}_i = \{1^n : (\exists k)[(n = (p_i)^k) \wedge (\exists y)[|y| = n \wedge 1y \in A]]\}$, where p_i is the ith prime.

Let requirement $R_{\langle i,j \rangle}$ be

$R_{\langle i,j \rangle}$: $(\exists x)[x \in \hat{L}_i \iff x \in L(M_j^{L(N_i^A)})]$.

In the construction below we will ensure that for each i either:

Req 1 N_i^A has more than $|x|^i + i$ accepting paths on some input x, or

Req 2 $(\forall k)[\mathrm{card}(\{y : y \in A \wedge |y| = (p_i)^k + 1\}) = 1] \wedge (\forall j)[$ requirement $R_{\langle i,j \rangle}$ is satisfied $]$.

Note that this is sufficient to ensure that A satisfies the required properties. If **Req 2** is satisfied, then clearly $\hat{L}_i \notin \mathrm{P}^{L(N_i^A)}$. Also the first clause of **Req 2** makes sure that for each $n = p_i^k$, there exists exactly one string y, of length $n+1$, which belongs to A; thus for $\hat{L}'_i = \{1^n : (\exists k)[(n = (p_i)^k) \wedge (\exists y)[|y| = n \wedge 0y \in A]]\}$, where p_i is the ith prime, we have

(a) $\hat{L}_i \in \mathrm{UP}^A$,
(b) $\hat{L}'_i \in \mathrm{UP}^A$,
(c) $\hat{L}'_i \cup \hat{L}_i = \{1^n : (\exists k)[n = (p_i)^k]\}$, and
(d) $\hat{L}_i \cap \hat{L}'_i = \emptyset$.

Thus we have $\hat{L}_i \in \mathrm{UP}^A \cap \mathrm{coUP}^A$.

For each $\langle i, j\rangle$, we will seek to find a way of extending the oracle so as to make N_i^A non-FewP^A-like. Failing this, we will argue that we can choose our oracle in such a way as to determine the answers to all oracle queries made by M_j, and still have the flexibility to diagonalize against $\hat{L}_i$. This step is a combinatorial argument that machines with *few* accepting paths that do not trivially accept must reject on an overwhelming number of oracle extensions.

In stage $\langle i, j\rangle$, we either make N_i^A non-FewP^A-like by adding strings of length $(p_i)^k + 1$, for some k, to A or without violating the first clause in **Req 2** , satisfy requirement $R_{\langle i,j\rangle}$.

Let $A_{\langle i,j\rangle}$ denote the set of strings determined to be in A constructed before stage $\langle i, j\rangle$. $n_{\langle i,j\rangle}$ denotes the length, such that for each string of length at most $n_{\langle i,j\rangle}$, membership question (in A) has been decided before stage $\langle i, j\rangle$.

Let $A_0 = \emptyset$ and $n_0 = 0$ (we let $\epsilon \notin A$). We will have $A = \bigcup_{\langle i,\, j\rangle} A_{\langle i,\, j\rangle}$. Go to stage 0.

Stage $\langle i, j\rangle$:

1. If $i = 0$ or $j = 0$ or
   if N_i^A has already been made non-FewP^A-like, then set
   $$A_{\langle i,j\rangle+1} = A_{\langle i,j\rangle} \;\&\; n_{\langle i,j\rangle+1} = n_{\langle i,j\rangle},$$
   and go to stage $\langle i, j\rangle + 1$.
2. Let $n = (p_i)^l$ be so large that:
   (i) $n > n_{\langle i,j\rangle}$, and
   (ii) $2^n > [r_j(n)] \cdot [r_i(r_j(n))] \cdot [r_i(r_j(n)) + 1] \cdot [r_i(r_j(n)) + 2]/2$.
   (Recall that $r_j(n) = n^j + j$, the run time of the j^{th} machine.)

3. Let $B = \{0^m : n_{\langle i,j\rangle} < m \leq r_i(r_j(n)) \wedge m \neq n+1\}$.
4. If there exists a set $S \subseteq \{0,1\}^{n+1}$ such that $N_i^{A_{\langle i,j\rangle} \cup S \cup B}$ is non-FewP^A-like on some string of length at most $r_j(n)$ then

   let $A_{\langle i,j\rangle+1} = A_{\langle i,j\rangle} \cup S \cup B$ & $n_{\langle i,j\rangle+1} = r_i(r_j(n))$,

   and go to stage $\langle i,j\rangle + 1$.
5. Else

   Run machine M_j on 1^n using oracle set $L(N_i^{A_{\langle i,j\rangle} \cup B})$.
6. If M_j accepts in step 5 above then let $z, |z| = n$ be such that M_j on input 1^n using oracle set $L(N_i^{A_{\langle i,j\rangle} \cup B \cup \{0z\}})$ still accepts. (We will argue below that such a z indeed exists.) Let

   $A_{\langle i,j\rangle+1} = A_{\langle i,j\rangle} \cup B \cup \{0z\}$ & $n_{\langle i,j\rangle+1} = r_i(r_j(n))$,

   and go to stage $\langle i,j\rangle + 1$.

   (Note that here $1^n \notin \hat{L}_i$; thus $R_{\langle i,j\rangle}$ is satisfied).
7. Else (the computation in step 5 rejects)

   Let z, $|z| = n$ be a string such that M_j on input 1^n using oracle set $L(N_i^{A_{\langle i,j\rangle} \cup B \cup \{1z\}})$ still rejects. (We will argue below that such a z indeed exists.)

   Let

   $A_{\langle i,j\rangle+1} = A_{\langle i,j\rangle} \cup B \cup \{1z\}$ & $n_{\langle i,j\rangle+1} = r_i(r_j(n))$ and

   go to stage $\langle i,j\rangle + 1$.

   (Note that here $1^n \in \hat{L}_i - L(M_j^{L(N_i^A)})$. Thus requirement $R_{\langle i,j\rangle}$ has been satisfied).

End stage $\langle i,j\rangle$

Note that if we never find a way of making N_i^A non-FewP^A-like, then $\hat{L}_i \in \mathrm{UP}^A \cap \mathrm{coUP}^A$ (because the above procedure puts exactly one string at each length important to $\hat{L}_i$) and $(\forall j)$[requirement $R_{\langle i,j\rangle}$ is satisfied] (note that, the definition of B in step 3, ensures that, for lengths that are not used in the diagonalization in steps 5–7, the first clause in **Req 2** is satisfied). Thus the requirement **Req 2** is satisfied. On the other hand, if we do find a way of making N_i^A non-FewP^A-like, then **Req 1** is satisfied (even though we do not need $\hat{L}_i$ to be in $\mathrm{UP}^A \cap \mathrm{coUP}^A$ in this case, our construction leaves $\hat{L}_i$ finite in this case and thus in $\mathrm{UP}^A \cap \mathrm{coUP}^A$). Thus we have met requirements that are sufficient to ensure that FewP^A has no $\leq_T^{p,A}$-hard languages for $\mathrm{UP}^A \cap \mathrm{coUP}^A$.

We need to argue that a z (for steps 6 and 7) can indeed be selected. Let n, B be as in stage $\langle i,j\rangle$ (in which the construction reaches step 6 (7)). We claim that there is a string z, $|z| = n$, such that the computation of M_j on input 1^n using oracle set, $L(N_i^{A_{\langle i,j\rangle}\cup B})$ queries exactly the same strings, getting exactly the same answers, as does the computation of M_j on input 1^n using oracle set, $L(N_i^{A_{\langle i,j\rangle}\cup B\cup\{0z\}})$ $(L(N_i^{A_{\langle i,j\rangle}\cup B\cup\{1z\}}))$. To see that this is the case, suppose that the computation of M_j on input 1^n using oracle set $L(N_i^{A_{\langle i,j\rangle}\cup B})$ queries strings $x_1, x_2, \ldots, x_k$ that are in $L(N_i^{A_{\langle i,j\rangle}\cup B})$, and queries strings $y_1, y_2, \ldots, y_m$ that are not in $L(N_i^{A_{\langle i,j\rangle}\cup B})$. For $1 \le r \le k$, reserve all strings, of length $n+1$, which are queried by the accepting path of $N_i^{A_{\langle i,j\rangle}\cup B}$, on input x_r, for $\overline{A}$ (there are at most $k\cdot[r_i(r_j(n))]$ such strings). For y such that $|y| \le r_j(n)$, let $S_y = \{w : N_i^{A_{\langle i,j\rangle}\cup B\cup\{w\}}$ accepts $y \wedge |w| = n+1\}$. We reserve all strings in $S_{y_r}, 1 \le r \le m$ for $\overline{A}$ (by Corollary 3.3 below $\mathrm{card}(\bigcup_{1\le r\le m} S_{y_r}) \le m\cdot[r_i(r_j(n))]\cdot[r_i(r_j(n))+1]\cdot[r_i(r_j(n))+2]/2$). Since $2^n > r_j(n)\cdot[r_i(r_j(n))]\cdot[r_i(r_j(n))+1]\cdot[r_i(r_j(n))+2]/2$, not all strings of form $0\{0,1\}^n$ and $1\{0,1\}^n$ are reserved for $\overline{A}$. Thus appropriate z exists.

We now bound the size of S_y as defined above. In Corollary 3.3 below we argue that size of S_y is bounded, if, N_i^A could not be made non-FewP^A-like in step 4 of the above construction. This Corollary uses a combinatorial lemma (Lemma 3.4) proved below.

Corollary 3.3 *For y such that $|y| \le r_j(n)$ and $N_i^{A_{\langle i,j\rangle}\cup B}$ rejects y, $\mathrm{card}(S_y) \le [r_i(r_j(n))]\cdot[r_i(r_j(n))+1]\cdot[r_i(r_j(n))+2]/2$.*

Proof of Corollary 3.3: Let $m = \mathrm{card}(S_y)$ and $r, r' = r_i(r_j(n))+1$. Also let $C_1, C_2, \ldots, C_m$ be the m elements of S_y, and D_k be the set of strings queried in the (lexicographically least) accepting path of $N_i^{A_{\langle i,j\rangle}\cup B\cup\{C_k\}}(y)$. Note that $C_k \in D_k$ (since $N_i^{A_{\langle i,j\rangle}\cup B}(y)$ has no accepting paths). Clearly conditions (a) and (c) of Lemma 3.4 below are satisfied. To show that (b) is satisfied suppose, by way of contradiction, that there exists a subset T of $\{1, 2, \ldots, m\}$ of size r' such that $(\forall k', k'' \in T)[k' \ne k'' \Rightarrow C_{k'} \notin D_{k''}]$. Now, for each $k, k' \in T$, $k' \ne k$, since $C_{k'} \notin D_k$, the lexicographically least accepting path of $N_i^{A_{\langle i,j\rangle}\cup B\cup C_k}(y)$ is an accepting path for $N_i^{A_{\langle i,j\rangle}\cup B\cup\{C_{k''} : k''\in T\}}(y)$. Moreover each of these paths are distinct (since $C_k \in D_k - D_{k'}$, for distinct k, k'). Thus N_i, on input y, with oracle $A_{\langle i,j\rangle}\cup B\cup\{C_k : k \in T\}$ has at least r' accepting paths. This is a contradiction to the fact that N_i could not be made non-FewP^A-like. Thus C, D satisfy the conditions of Lemma 3.4, and hence $\mathrm{card}(S_y) \le [r_i(r_j(n))]\cdot$

$[r_i(r_j(n))+1]\cdot[r_i(r_j(n))+2]/2$. □

□ (Theorem 3.1)

Lemma 3.4 *Let r, r' be given. Let $C_k, 1 \leq k \leq m$, be strings and $D_k, 1 \leq k \leq m$, be sets of strings. If C_k, D_k satisfy a),b) and c) then $m \leq (r'-1)\cdot r(r+1)/2$.*

a) $(\forall k,l)[k \neq l \Rightarrow C_k \neq C_l]$,

b) $(\forall T \subseteq \{1, 2, \ldots, m\} : card(T) = r')[(\exists k', k'' \in T)[k' \neq k'' \wedge C_{k'} \in D_{k''}]]$.

c) $(\forall k)$[*Cardinality of D_k is at most $r-1$*].

Proof of Lemma 3.4: Let $P(r,r') = max\{m$: exists a sequence $C_1, \ldots, C_m$ of strings and a sequence $D_1, \ldots, D_m$ of sets of strings, satisfying clauses (a) to (c) in the lemma $\}$. We prove by induction on n that $P(n,n') \leq (n'-1)\cdot n(n+1)/2$. Clearly, $P(1,n') = n'-1$. Suppose $P(n,n') \leq (n'-1)\cdot n(n+1)/2$. We prove that $P(n+1,n') \leq (n'-1)(n+1)(n+2)/2$. Let C, D be such that conditions of the lemma are satisfied with $r = n+1, r' = n'$. Now consider a maximal set $T \subseteq \{1, 2, \ldots, m\}$, such that, $[(\forall k', k'' \in T : k' \neq k'')[C_{k'} \notin D_{k''}]]$. Clearly, $\mathrm{card}(T) \leq n'-1$.

Also for all $k \in \{1, 2, \ldots, m\} - T$, either

(1) C_k is in $D_{k'}$ for some k' in T or

(2) $C_{k'}$ is in D_k for some k' in T

Let $T' = \{k : k \in \{1, 2, \ldots, m\} - T \wedge \neg(\exists k' \in T)[C_k \in D_{k'}]\}$. Since, there are atmost $n * \mathrm{card}(T)$ elements, $k \in \{1, 2, \ldots, m\} - T$, for which there exists a $k' \in T$ such that $C_k \in D_{k'}$, cardinality of T' is atleast $m - (n+1) * \mathrm{card}(T)$.

Now define new sequences C' and D' as follows. Let $i_1, i_2, \ldots, i_{\mathrm{card}(T')}$ be such that $\{i_1, i_2, \ldots, i_{\mathrm{card}(T')}\} = T'$. For $j \in \{1, 2, \ldots, \mathrm{card}(T')\}$, let $C'_j = C_{i_j}$ and $D'_j = D_{i_j} - \{C_k : k \in T\}$. It is easy to see that D'_j is a proper subset of D_{i_j}. Note that C' and D' now satisfy the inductive hypothesis with $r = n$, and $r' = n'$. From the above analysis we have, $m \leq (n+1) * \mathrm{card}(T) + \mathrm{card}(T') \leq (n+1) * (n'-1) + P(n,n') \leq (n'-1)(n+1)(n+2)/2$. The lemma follows. □

An anonymous referee has pointed out that Corollary 3.3 also follows from the following claim, which gives a slightly weaker bound than that of Lemma 3.4 but is easier to prove.

Claim 3.5 *Suppose $C_1, \ldots, C_m$ and $D_1, \ldots, D_m$ satisfying a), b), c) as in Lemma 3.4 are given. Then for some constant c, $m \leq crr'(r'-1)$.*

Proof of Theorem 3.1 can easily be extended to prove that Few [8], a generalization of FewP, does not robustly posses Turing hard languages for UP $\cap$ coUP (see also [37]).

4 Robust Completeness and Probabilistic Classes

In this section we prove that IP $\cap$ coIP does not robustly possess Turing hard sets for ZPP. It immediately follows that there exist relativized worlds in which ZPP, R, IP $\cap$ coIP and BPP do not have Turing complete sets.

Theorem 4.1 *There is a recursive oracle A such that* $\mathrm{IP}^A \cap \mathrm{coIP}^A$ *does not contain any* $\leq_T^{p,A}$*-hard sets for* ZPP^A.

Corollary 4.2 *There is a recursive oracle A such that* ZPP^A, R^A, coR^A, $\mathrm{NP}^A \cap \mathrm{coNP}^A$ *and* $\mathrm{IP}^A \cap \mathrm{coIP}^A$ *contain no* $\leq_T^{p,A}$*-complete languages.*

Proof of Theorem 4.1: We wish to construct A such that for no $L \in \mathrm{IP}^A \cap \mathrm{coIP}^A$ is $\mathrm{ZPP}^A \subseteq \mathrm{P}^L$ (this suffices by Lemma 2.11). For each $L \in \mathrm{IP}^A \cap \mathrm{coIP}^A$ there exist V_a and V_r such that V_a^A is IP^A-like, V_r^A is IP^A-like, $L = L(V_a^A)$ and $\overline{L} = L(V_r^A)$. Our goal is to show that for each i, j, either:

1. $[V_i^A$ is not IP^A-like $\bigvee V_j^A$ is not IP^A-like $\bigvee [L(V_i^A) \neq \overline{L(V_j^A)}]]$ OR
2. $(\exists \hat{L}_i \in \mathrm{ZPP}^A)[\hat{L}_i \notin \mathrm{P}^{L(V_i^A)}]$.

Let $\hat{L}_i = \{1^n : (\exists k)[(n = (p_i)^k) \wedge (\exists y)[|y| = n \wedge 1y \in A]]\}$, where p_i is the ith prime.

Let requirement $R_{\langle i,\, j,\, k\rangle}$ be

$R_{\langle i,\, j,\, k\rangle}$: $(\exists x)[x \in \hat{L}_i \not\Longleftrightarrow x \in L(M_k^{L(V_i^A)})]$.

For $b \in \{0,1\}$ let $\overline{b}$ denote $1 - b$. In the construction below we will ensure that for each i, j either:

Req 3 $[[V_i^A$ is not IP^A-like $] \bigvee [V_j^A$ is not IP^A-like $] \bigvee [L(V_i^A) \neq \overline{L(V_j^A)}]]$ OR

Req 4

$[(\forall k)(\exists b \in \{0,1\})[\mathrm{card}(\{y : by \in A \wedge |y| = (p_i)^k\}) = 0 \wedge \mathrm{card}(\{y : \overline{b}y \in A \wedge |y| = (p_i)^k\}) > 2^{(p_i)^k - 1}]$ and $(\forall k)[$ requirement $R_{\langle i,j,k\rangle}$ is satisfied $]]$.

Note that this is sufficient to ensure that A satisfies the required properties.

For each $\langle i,\, j,\, k\rangle$, we will seek to find a way of extending the oracle to make either V_i^A non-IP^A-like or V_j^A non-IP^A-like or $L(V_i^A) \neq \overline{L(V_j^A)}$. Failing this, we

will argue that $R_{\langle i,j,k\rangle}$ can be satisfied. This step is a combinatorial argument exploiting the IP^A-like properties of V_i^A and V_j^A.

In stage $\langle i, j, k\rangle$, we either:
satisfy **Req 3** by adding strings of length $(p_i)^l + 1$, for some l, to A or without violating the first clause of **Req 4** , satisfy requirement $R_{\langle i, j, k\rangle}$.

Let $A_{\langle i, j, k\rangle}$ denote the set A constructed before stage $\langle i, j, k\rangle$. Let $n_{\langle i,j,k\rangle}$ denote the length, such that for each string of length at most $n_{\langle i,j,k\rangle}$, membership question (in A) has been decided before stage $\langle i, j, k\rangle$.

Let $A_0 = \emptyset$ and $n_0 = 0$ (we let $\epsilon \notin A$). We will have $A = \bigcup_{\langle i, j, k\rangle} A_{\langle i, j, k\rangle}$. Go to stage 0.

Stage $\langle i, j, k\rangle$:

1. If $i = 0$ or $j = 0$ or $k = 0$ or

   if V_i^A has already been made non-IP^A-like or V_j^A has already been made non-IP^A-like or for some x it is already known that $x \notin L(V_i^A) \triangle L(V_j^A)$ then let

   $$A_{\langle i,j,k\rangle+1} = A_{\langle i,j,k\rangle} \ \&\ n_{\langle i,j,k\rangle+1} = n_{\langle i,j,k\rangle},$$

   and go to stage $\langle i, j, k\rangle + 1$.

2. Otherwise, let $n = (p_i)^l$ be so large that:

   (i) $n > n_{\langle i,j,k\rangle}$, and

   (ii) $2^n > 6 \cdot [r_k(n)] \cdot [r_i(r_k(n)) + r_j(r_k(n))]$.

3. Let $B = \{0\{0,1\}^m : n_{\langle i,j,k\rangle} \le m < r_i(r_k(n)) + r_j(r_k(n)) \wedge m \ne n\}$.

4. If there exists a set $S \subseteq \{0,1\}^{n+1}$ such that [$V_i^{A_{\langle i,j,k\rangle} \cup B \cup S}$ or $V_j^{A_{\langle i,j,k\rangle} \cup B \cup S}$ is non-IP^A-like on some string of length at most $r_k(n)$], or [for some string of length at most $r_k(n)$, $x \notin (L(V_i^{A_{\langle i,j,k\rangle} \cup B \cup S}) \triangle L(V_j^{A_{\langle i,j,k\rangle} \cup B \cup S}))$], then

   let $A_{\langle i,j,k\rangle+1} = A_{\langle i,j,k\rangle} \cup S \cup B$, & $n_{\langle i,j,k\rangle+1} = r_i(r_k(n)) + r_j(r_k(n))$,

   and go to stage $\langle i, j, k\rangle + 1$.

5. Else if $M_k^{L(V_i^{A_{\langle i,j,k\rangle} \cup B})}$ accepts 1^n then let S be a maximal subset of $0\{0,1\}^n$ such that $M_k^{L(V_i^{A_{\langle i,j,k\rangle} \cup B \cup S})}$ still accepts 1^n.

   (We will argue below that card$(S) > 2^n/2$)

   Let $A_{\langle i,j,k\rangle+1} = A_{\langle i,j,k\rangle} \cup S \cup B$ & $n_{\langle i,j,k\rangle+1} = r_i(r_k(n)) + r_j(r_k(n))$,

   and go to stage $\langle i, j, k\rangle + 1$.

   (Note that here requirement $R_{\langle i,j,k\rangle}$ is satisfied.)

6. Else, let S be a maximal subset of $1\{0,1\}^n$ such that $M_k^{L(V_i^{A_{\langle i,j,k\rangle}\cup B\cup S})}(1^n)$ still rejects.

   (We will argue below that $\text{card}(S) > 2^n/2$)

   Let $A_{\langle i,j,k\rangle+1} = A_{\langle i,j,k\rangle} \cup S \cup B$ & $n_{\langle i,j,k\rangle+1} = r_i(r_k(n)) + r_j(r_k(n))$,

   and go to stage $\langle i,j,k\rangle + 1$.

   (Note that here requirement $R_{\langle i,j,k\rangle}$ is satisfied.)

End stage $\langle i,j,k\rangle$.

We claim that for steps 5 and 6, $\text{card}(S) \leq 2^{n-1}$ is not possible. We prove this for step 5 (the proof for step 6 is similar). Let T be the set of questions asked by M_k with the oracle $A_{\langle i,j,k\rangle} \cup B \cup S$. Let P_i be the prover corresponding to $V_i^{A_{\langle i,j,k\rangle}\cup B\cup S}$ and P_j be the prover corresponding to $V_j^{A_{\langle i,j,k\rangle}\cup B\cup S}$, which make the verifiers accept their respective languages. Now, since S is maximal, for each string w in $0\{0,1\}^n - S$, there exists a string, $w_q \in T$, such that, $w_q \in L(V_i^{A_{\langle i,j,k\rangle}\cup B\cup S}) \not\Longleftrightarrow w_q \in L(V_i^{A_{\langle i,j,k\rangle}\cup B\cup S\cup\{w\}})$. Thus, if $w_q \in L(V_i^{A_{\langle i,j,k\rangle}\cup B\cup S})$ then $V_i^{A_{\langle i,j,k\rangle}\cup B\cup S}$ with prover P_i must query w with probability $\geq 1/3$ (since probability of acceptance of w_q by V_i changes from $\geq 2/3$ to $\leq 1/3$ when w is added to the oracle). Similarly, if $w_q \in L(V_j^{A_{\langle i,j,k\rangle}\cup B\cup S})$ then $V_j^{A_{\langle i,j,k\rangle}\cup B\cup S}$ with prover P_j must query w with probability $\geq 1/3$ (since probability of acceptance of w_q by V_j changes from $\geq 2/3$ to $\leq 1/3$ when w is added to the oracle). Since, V_i (respectively V_j) queries at most $r_i(r_k(n))$ strings (respectively $r_j(r_k(n))$ strings) in each path, V_i (on input w_q) can query at most $3r_i(r_k(n))$ strings (respectively V_j can query at most $3r_j(r_k(n))$ strings) with probability $\geq 1/3$. Thus there can be at most $\text{card}(T)\cdot 3\cdot[r_i(r_k(n)) + r_j(r_k(n))]$ strings such as w. Thus $2^n - \text{card}(S) \leq 3\cdot r_k(n)\cdot[r_i(r_k(n)) + r_j(r_k(n))]$, which along with the conditions for n in step 2, implies $\text{card}(S) > 2^n/2$.

Now note that if we never find a way of satisfying **Req 3** (in step 4) then $\hat{L}_i \in \text{ZPP}^A$ (because of the number of strings placed into the oracle by the construction, at steps 5 and 6 and in B at each stage, at each length important to $\hat{L}_i$) and $(\forall j)$[requirement $R_{\langle i,j,k\rangle}$ is satisfied]. Thus the requirement **Req 4** is satisfied. Thus we have met requirements that are sufficient to ensure the theorem. □

Class ($\mathcal{C}$)	$\mathcal{C}$ does not robustly possess many-one complete sets	$\mathcal{C}$ does not robustly possess Turing complete sets
NP $\cap$ coNP	[35]	[17]
IP $\cap$ coIP	Corollary 4.2	Corollary 4.2
BPP	[18]	[18] plus [3]
R	[35]	Corollary 4.2
coR	[35]	Corollary 4.2
ZPP	Corollary 4.2	Corollary 4.2
FewP	Corollary 3.2	Corollary 3.2
FewP $\cap$ coFewP	Corollary 3.2	Corollary 3.2
UP	[18]	Corollary 3.2
coUP	[18]	Corollary 3.2
UP $\cap$ coUP	Corollary 3.2	Corollary 3.2

Table 1: Results on Robust Completeness

5 Conclusion

This paper showed that many promise classes do not robustly possess Turing complete sets, and, indeed, even much bigger classes do not contain hard sets for promise classes. Our proofs exploit the combinatorial limitations of machines that attempt to be FewP-like or IP-like. Table 1 summarizes the results of this and earlier papers on complete sets for promise classes. It remains an open problem at present whether there exist relativized worlds in which R or UP or FewP have Turing complete languages but not many-one complete languages; our intuition is that such worlds exist. Relatedly, Watanabe and Tang [38] have shown that if certain conditions hold then m-complete sets and T-complete sets differ in PSPACE. Also, there is a relativized world in which the boolean hierarchy contains bounded truth-table complete sets but not k-truth-table complete sets [7].

6 Acknowledgments

We are very grateful to Osamu Watanabe for discussing with us his notion of "sup-UP" semi-completeness. We thank William Gasarch for helpful comments

on the paper. We thank Ken Regan and Joel Seiferas for enjoyable discussions.

References

[1] E. Allender. The complexity of sparse sets in P. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 1–11. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[2] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[3] K. Ambos-Spies. A note on complete problems for complexity classes. *Information Processing Letters*, 23:227–230, 1986.

[4] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[5] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988.

[6] J. Balcázar and D. Russo. Immunity and simplicity in relativizations of probabilistic complexity classes. *Theoretical Informatics and Applications (RAIRO)*, 22(2):227–244, 1988.

[7] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, December 1988.

[8] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23:95–106, 1990.

[9] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.

[10] D. Eppstein, L. Hemachandra, J. Tisdall, and B. Yener. Simultaneous strong separations of probabilistic and unambiguous complexity classes. *Mathematical Systems Theory*, 25(1):23–36, 1992.

[11] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP? *Information Processing Letters*, 28, 1988.

[12] J. Geske and J. Grollmann. Relativizations of unambiguous and random polynomial time classes. *SIAM Journal on Computing*, 16(2):511–519, 1986.

[13] J. Gill. Computational complexity of probabilistic Turing machines. *Siam Journal of Computing*, 6:675–695, 1977.

[14] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 390–397, 1986.

[15] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[16] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–335, 1988.

[17] Y. Gurevich. Algebras of feasible functions. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press, November 1983.

[18] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.

[19] J. Hartmanis and L. Hemachandra. One-way functions and the non-isomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[20] J. Hartmanis and N. Immerman. On complete problems for NP$\bigcap$coNP. In *Automata, Languages, and Programming (ICALP 1985)*, pages 250–259. Springer-Verlag *Lecture Notes in Computer Science #194*, 1985.

[21] L. Hemachandra and R. Rubinstein. Separating complexity classes with tally oracles. *Theoretical Computer Science*, 92(2):309–318, 1992.

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

[23] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37:1–30, 1985.

[24] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report TR83-001, University of Chicago Department of Computer Science, Chicago, IL, 1983.

[25] S. Kurtz, S. Mahaney, and J. Royer. Collapsing degrees. *Journal of Computer and System Sciences*, 1988.

[26] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[27] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the Association of Computing Machinery*, 39(4):859–868, 1992.

[28] S. R. Mahaney. The isomorphism conjecture and sparse sets. In *Computational Complexity Theory*. Proceedings of Symposia in Applied Mathematics, Volume 38, American Mathematical Society, 1988.

[29] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the Association of Computing Machinery*, 29(1):261–268, 1982.

[30] K. W. Regan. Provable complexity properties and constructive reasoning. Manuscript, April 1989.

[31] K. W. Regan. Minimum-complexity pairing functions. *Journal of Computer and System Sciences*, 45(3):285–295, 1992.

[32] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[33] U. Schöning. Robust algorithms: A different approach to oracles. *Theoretical Computer Science*, 40:57–66, 1985.

[34] A. Shamir. IP=PSPACE. *Journal of the Association of Computing Machinery*, 39(4):869–877, 1992.

[35] M. Sipser. On relativization and the existence of complete sets. In *Automata, Languages, and Programming (ICALP 1982)*. Springer-Verlag *Lecture Notes in Computer Science #140*, 1982.

[36] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[37] N. K. Vereshchagin. Relativizable and nonralativizable theorems of polynomial theory of algorithms. *Izvestiya Rossijskoi Akademii Nauk*, 1993.

[38] O. Watanabe and S. Tang. On polynomial-time Turing and many-one completeness in PSPACE. *Theoretical Computer Science*, 97(2):199–215, 1992.

[39] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: An overview. In *Proceedings 1st Structure in Complexity Theory Conference*, pages 383–400. IEEE Computer Society Press, June 1986.