

# Banishing Robust Turing Completeness\*

Lane A. Hemachandra<sup>†</sup>  
Department of Computer Science  
University of Rochester  
Rochester, New York 14627

Sanjay Jain<sup>‡</sup>  
Department of Computer and Information Sciences  
University of Delaware  
Newark, DE 19716

Nikolai K. Vereshchagin  
Department of Mathematical Logic  
Moscow State University  
Moscow, USSR 119899

December 21, 1991

## Abstract

This paper proves that “promise classes” are so fragilely structured that they do not robustly possess Turing-hard sets even in classes far larger than themselves. In particular, this paper shows that  $\text{FewP}$  does not robustly possess Turing-hard sets for  $\text{UP} \cap \text{coUP}$  and  $\text{IP} \cap \text{coIP}$  does not robustly possess Turing-hard sets for  $\text{ZPP}$ . It follows that  $\text{ZPP}$ ,  $\text{R}$ ,  $\text{coR}$ ,  $\text{UP} \cap \text{coUP}$ ,  $\text{UP}$ ,  $\text{FewP} \cap \text{coFewP}$ ,  $\text{FewP}$ , and  $\text{IP} \cap \text{coIP}$  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.,  $\text{R}$ ,  $\text{UP}$ ,  $\text{FewP}$ ) might robustly have Turing complete sets, and extends the range of classes known not to robustly contain many-one complete sets.

---

\*Some of these results were reported at the Symposium on Mathematical Foundations of Computer Science, Carlsbad, Czechoslovakia, September, 1988. This report revises and extends TR-297.

<sup>†</sup>Supported in part by a Hewlett-Packard Corporation equipment grant and the National Science Foundation under grant CCR-8809174/CCR-8996198 and Presidential Young Investigator Award CCR-8957604.

<sup>‡</sup>Supported by 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., [Mah88]). Most common complexity classes—NP, coNP, PSPACE, etc.—have many-one complete sets<sup>1</sup> that help us study them. Sipser noted, however, that some classes may lack complete sets [Sip82]. His paper sparked much research into which classes robustly (i.e., with respect to all oracles [BGS75, Sch85]) 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.<sup>2</sup> 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  $\text{NP} \cap \text{coNP}$  do not robustly possess many-one complete languages [Sip82]. 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 [HH88]. In related work, Regan [Reg89] 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 [HR] and hierarchy [FS89] 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 [LLS75]. This would show that the structure of these classes differs markedly from that of P, NP, PSPACE, and so on. Gurevich showed that  $\text{NP} \cap \text{coNP}$  has many-one complete languages if and only if it has Turing complete languages ([Gur83], see also [HI85]). Ambos-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 [AS86]. In particular, it follows, from the result of Sipser [Sip82], that  $\text{NP} \cap \text{coNP}$  does

---

<sup>1</sup>Throughout this paper, we are concerned only with polynomial-time reducibilities.

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

not robustly possess Turing complete sets [Gur83]. Similarly, since  $P^{BPP} = BPP$  [Zac86], from [HH88]’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 ([Zac86] 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 [Sip82, HH88], that does not seem to generalize to Turing completeness.

In fact, classes such as R, UP, and FewP, have a chaotic 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” [HR]. 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.<sup>3</sup> 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 possess Turing complete sets.

## 2 Preliminaries

Let  $\mathcal{N}$  denote the set of natural numbers.  $\Sigma$  is an alphabet set, usually  $\{0, 1\}$ . A language is a subset of  $\Sigma^*$ . 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.  $M_0, M_1, \dots$  denotes some standard enumeration of polynomial-

---

<sup>3</sup>Recently, Shamir [Sha90, LFKN90] 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. However, IP is not robustly equal to PSPACE [FS88]. 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 [CH91]), and in particular is not known to apply to BPP. Thus one may take Theorem 4.1 as evidence that, in the real world, BPP does not have Turing hard sets for ZPP, R, and  $coR$ .

time deterministic Turing machines.  $N_1, N_2, \dots$  denotes some standard enumeration of polynomial-time nondeterministic Turing machines.  $B_1, B_2, \dots$  denotes some standard enumeration of polynomial-time probabilistic Turing machines [Gil77]. We assume that the running time of machine  $M_i$  ( $N_i$ ) ( $B_i$ ) is bounded by deterministic (nondeterministic) ((probabilistic)) time  $\tau_i(n) = n^i + i$ .  $P$  denotes the class of all languages accepted by some polynomial-time deterministic Turing machine [HU79].  $NP$  denotes the class of languages accepted by polynomial-time nondeterministic Turing machines and  $coNP$  denotes the class of languages whose complements are in  $NP$  [HU79].  $L(M)$  denotes the language accepted by the machine  $M$ .  $L(M^A)$  denotes the language accepted by the oracle machine  $M$  with the oracle  $A$  [BGS75, HU79].

$\bar{A}$  denotes the complement of  $A$ , i.e.,  $\Sigma^* - A$ .  $\chi_A$  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 [BDG88, Reg88]). 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** [Val76]

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

2.  $coUP = \{L : \bar{L} \in UP\}$ .

$UP$  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<sup>4</sup> exist if and only if  $P \neq UP$  [Ko85,GS88], and one-way functions whose range<sup>5</sup> is in  $P$  exist if and only if  $P \neq UP \cap \text{coUP}$  [GS88]. 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 [HH88]—Ko has shown that the operator version of  $UP$  does have complete sets [Ko85].

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** [AR88]

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

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

**Proposition 2.3** *There exists an enumeration  $N'_0, N'_1, \dots$  of nondeterministic Turing machines such that, for all  $i$ , run time of  $N'_i$  is bounded by  $n^i + i$  and*

$\text{FewP} = \{L | (\exists i)[L = L(N'_i) \text{ and } (\forall x)[\text{number of accepting paths of } N'_i \text{ on } x \text{ is } \leq n^i + i]]\}.$

Thus we assume, without loss of generality, that our standard enumeration is one such enumeration. We say that a machine  $N_i^A$  is  $\text{FewP}^A$ -like iff  $(\forall x)[\text{the number of accepting paths of } N_i^A \text{ on input } x \text{ is at most } |x|^i + i]$ . Note that we are not claiming that it is easy to determine whether a machine is  $\text{FewP}$ -like. Rather, Proposition 2.3 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. Note that Proposition 2.3 holds robustly. In our relativizations, this will allow us to, for each possible oracle machine, instantly discard a machine if we notice that it is not  $\text{FewP}$ -like in the relativized world we construct (rather than having to explicitly pair each machine with every fewness bound).

**Definition 2.4** [Gil77]

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

$\text{coR} = \{L : \bar{L} \in R\}.$

---

<sup>4</sup>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  $\text{range}(f) \neq \Sigma^*$ ) is not computable in polynomial time [GS88].

<sup>5</sup> $\text{Range}(f) = \bigcup_{i \in \Sigma^*} f(i).$

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 [BDG88].

$R$ , 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 [Rac82,GG86,BR88,EHTY89]. We say that a probabilistic machine  $B_i^A$  is  $R^A$ -like, iff  $(\forall x)[\text{either } B_i^A \text{ on } x \text{ has no accepting paths or the probability that } B_i^A \text{ accepts } x \text{ is at least } 1/2]$ .

**Definition 2.5** [Gil77]  $ZPP = R \cap coR$ .

The class of languages that are in  $ZPP$  are exactly those languages which can be solved in expected polynomial time [Gil77].

**Definition 2.6** [Gil77]

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

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

We now discuss interactive proofs [GMR89]. 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 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 high probability, that  $x \in L$ .

**Definition 2.7** We say that a verifier  $V$  accepts  $L$  iff:

$(\forall x \notin L)(\forall P)[\text{Prob}(V \text{ with prover } P \text{ accepts } x) \leq 1/3]$ , and  
 $(\exists P)(\forall x \in L)[\text{Prob}(V \text{ with prover } P \text{ accepts } x) \geq 2/3]$ .

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** [GMR89]

$$\text{IP} = \{L : (\exists V)[V \text{ accepts } L]\}.$$

$$\text{coIP} = \{L : \bar{L} \in \text{IP}\}.$$

We say that a verifier  $V_i^A$  is  $\text{IP}_i^A$ -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 one oracle but not with respect to another.

We let  $V_0, V_1, \dots$  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  $(\forall A)[\text{R}^A \subseteq \text{BPP}^A \subseteq \text{IP}^A \cap \text{coIP}^A]$ .

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

**Definition 2.9** (see [LLS75, BDG88])

1.  $L_1 \leq_T^p L_2$  if  $L_1 \in \text{P}^{L_2}$ .
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 R or UP.

**Definition 2.10**

1.  $L_1 \leq_T^{p,A} L_2$  if  $L_1 \in \text{P}^{L_2 \oplus A}$ .
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  $\text{ZPP}^A, \text{R}^A,$

$\text{IP}^A \cap \text{coIP}^A$  have no  $\leq_T^{p,A}$ -complete sets and  $\text{UP}^B$ ,  $\text{FewP}^B$  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  $C$ :  $[\text{FewP}^A \text{ has } \leq_T^{p,A}\text{-hard sets for } C \text{ if and only it has } \leq_T^p\text{-hard sets for } C]$ .
2. For any oracle  $A$ , and class  $C$ :  $[\text{UP}^A \text{ has } \leq_T^{p,A}\text{-hard sets for } C \text{ if and only it has } \leq_T^p\text{-hard sets for } C]$ .
3. For any oracle  $A$ , and class  $C$ :  $[\text{R}^A \text{ has } \leq_T^{p,A}\text{-hard sets for } C \text{ if and only it has } \leq_T^p\text{-hard sets for } C]$ .
4. For any oracle  $A$ , and class  $C$ :  $[\text{IP}^A \cap \text{coIP}^A \text{ has } \leq_T^{p,A}\text{-hard sets for } C \text{ if and only it has } \leq_T^p\text{-hard sets for } C]$ .

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

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 [KMR88] and [Rog67, Section 9.3]. [KMR88] describes how this distinction has had a crucial effect on recent research asking whether all NP-complete sets are polynomially isomorphic [Kur83, GJ86, HH91]. 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  $\text{FewP}$  does not robustly possess Turing hard sets for  $\text{UP} \cap \text{coUP}$ . It follows immediately that there is an oracle  $A$  such that  $\text{UP}^A$  and  $\text{FewP}^A$  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 [LLS75], bounded truth-table reductions [LLS75], etc.

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

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

**Proof of Theorem 3.1** We wish to show that for no  $L \in \text{FewP}^A$  is  $\text{UP}^A \subseteq P^L$ , which suffices by Lemma 2.11. Each  $L$  in  $\text{FewP}^A$  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.  $(\exists \hat{L}_i)[\hat{L}_i \in \text{UP}^A \cap \text{coUP}^A \text{ and } \hat{L}_i \notin \text{P}^{L(N_i^A)}]$ .

The second condition says that some  $\text{UP}^A \cap \text{coUP}^A$  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  $i$ th prime. Let requirement  $R_{\langle i, j \rangle}$  be

$$R_{\langle i, j \rangle}: (\exists x)[x \in \hat{L}_i \not\leftrightarrow x \in L(M_j^{L(N_i^A)})].$$

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

- (\*)  $N_i^A$  has more than  $|x|^i + i$  accepting paths on some input  $x$ , or
- (\*\*)  $(\forall k)[\text{card}(\{y : y \in A \wedge |y| = (p_i)^k + 1\}) = 1] \wedge (\forall j)[\text{requirement } R_{\langle i, j \rangle} \text{ is satisfied}]$ .

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

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 <sup>$A$</sup> -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 <sup>$A$</sup> -like by adding strings of length  $(p_i)^k + 1$ , for some  $k$ , to  $A$  or without violating (\*\*), 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  $N_i^A$  has already been made non-FewP <sup>$A$</sup> -like, then set

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

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

2. Let  $n = (p_i)^k$  be so large that:

- (i)  $n > n_{\langle i, j \rangle}$ , and
- (ii)  $2^n > p(n)$ , where  $p(n) = [r_j(n)] \cdot [r_i(r_j(n))] \cdot [r_i(r_j(n)) + 1] \cdot [r_i(r_j(n)) + 2]/2$ .

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<sup>A</sup>-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

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

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 4 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

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

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).

We must argue that such a  $z$  (for steps 6 and 7) can indeed be selected. In fact, 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  $x_1, x_2, \dots, x_k$  that are in  $L(N_i^{A_{\langle i,j \rangle} \cup B})$ , and queries strings  $y_1, y_2, \dots, y_m$  that are not in  $L(N_i^{A_{\langle i,j \rangle} \cup B})$ . For  $1 \leq r \leq k$ , 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  $\bar{A}$  (there are at most  $k \cdot [r_i(r_j(n))]$  such strings). For  $1 \leq r \leq m$  define  $S_{y_r} = \{z : N_i^{A_{\langle i,j \rangle} \cup B \cup \{z\}} \text{ accepts } y_r \wedge |z| = n+1\}$ . We reserve all strings in  $S_{y_r}, 1 \leq r \leq m$  for  $\bar{A}$  (by Corollary 3.4  $\text{card}(\bigcup_{1 \leq r \leq m} S_{y_r}) \leq m \cdot [r_i(r_j(n))] \cdot [r_i(r_j(n))+1] \cdot [r_i(r_j(n))+2]/2$ ). 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  $\bar{A}$ . Thus appropriate  $z$  exists.

End stage  $\langle i, j \rangle$

Note that if we never find a way of making  $N_i^A$  non-FewP<sup>A</sup>-like, then  $\hat{L}_i \in \text{UP}^A \cap \text{coUP}^A$  (because the above procedure puts at most one string,  $\hat{z}$ , at each length important to  $\hat{L}_i$ ) and  $(\forall j)$ [requirement  $R_{\langle i, j \rangle}$  is satisfied]. Thus the requirement  $(**)$  is satisfied. On the other hand, if we do find a way of making  $N_i^A$  non-FewP<sup>A</sup>-like, then  $(*)$  is satisfied (even though we do not need  $\hat{L}_i$  to be in  $\text{UP}^A \cap \text{coUP}^A$  in this case, our construction leaves  $\hat{L}_i$  finite in this case and thus in  $\text{UP}^A \cap \text{coUP}^A$ ). Thus we have met requirements that are sufficient to ensure that FewP<sup>A</sup> has no  $\leq_T^{p, A}$ -hard languages for  $\text{UP}^A \cap \text{coUP}^A$ . ■

We have to argue that the size of  $S_{v_r}$  as defined in step 7 in the above construction does not get too large. Let  $n, B$  be as defined in stage  $\langle i, j \rangle$ . For  $x$  such that  $|x| \leq r_j(n)$  and  $N_i^{A_{\langle i, j \rangle} \cup B}$  rejects  $x$  let,  $S_x = \{z : N_i^{A_{\langle i, j \rangle} \cup B \cup \{z\}}$  accepts  $x \wedge |z| = n + 1\}$ . In Corollary 3.4 below we argue that size of  $S_x$  is bounded, if,  $N_i^A$  could not be made non-FewP<sup>A</sup>-like in step 4 of the above construction. We need a combinatorial lemma to show that  $S_x$  is not too large. In the lemma think of  $C[k]$ 's as elements of  $S_x$  and  $D[k]$ 's as the set of strings queried in the accepting path of  $N_i^{A_{\langle i, j \rangle} \cup \{C[k]\}}$ .

**Lemma 3.3** *Let  $r, r'$  be given. Let  $C[k], 1 \leq k \leq m$ , be an array of strings and  $D[k], 1 \leq k \leq m$ , be an array of sets of strings. Then 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)  $(\forall T \subseteq \{1..m\} : \text{card}(T) = r')[(\exists k', k'' \in T)[k' \neq k'' \wedge C[k'] \in D[k'']]]$ .
- c)  $(\forall k)[\text{Cardinality of } D[k] \text{ is at most } r - 1]$ .

**Proof of Lemma 3.3** Let  $P(r, r') = \max\{m : \text{exists arrays } C, D \text{ of size } m \text{ as defined above where the cardinality of } D[k] \text{ is bounded by } r - 1\}$ . 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  $T \subseteq \{1..m\}$ , such that,  $(\forall k', k'' \in T : k' \neq k'')[C[k'] \notin D[k'']]$ . Clearly,  $\text{card}(T) \leq n' - 1$ . Thus, for all but (at most)  $(n + 1) * \text{card}(T)$  of  $D[k]$ 's there exists a  $k' \in T$  such that  $C[k'] \in D[k]$ . Thus  $P(n + 1, n') \leq (n + 1)(n' - 1) + P(n, n')$ . Thus we have  $P(n + 1, n') \leq (n' - 1)(n + 1)(n + 2)/2$ . The lemma follows. ■

As a corollary we obtain,

**Corollary 3.4**  $\text{card}(S_x) \leq [r_i(r_j(n))] \cdot [r_i(r_j(n)) + 1] \cdot [r_i(r_j(n)) + 2]/2$ .

**Proof:** Let  $m = \text{card}(S_x)$  and  $r, r' = r_i(r_j(n)) + 1$ . Also let  $C[1], C[2], \dots, C[m]$  be the  $m$  elements of  $S_x$ , and  $D[k]$  be the set of strings queried in the accepting path of  $N_i^{A_{(i,j)} \cup B \cup \{C[k]\}}$ . Clearly conditions (a) and (c) of the lemma are satisfied. (b) is satisfied because  $N_i^A$  seems  $\text{FewP}^A$ -like. Thus  $C, D$  satisfy the conditions of Lemma 3.3. Thus  $\text{card}(S_x) \leq [r_i(r_j(n))] \cdot [r_i(r_j(n)) + 1] \cdot [r_i(r_j(n)) + 2]/2$ . ■

## 4 Robust Completeness and Probabilistic Classes

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

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

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

**Proof** We wish to show that for no  $L \in \text{IP}^A \cap \text{coIP}^A$  is  $\text{ZPP}^A \subseteq P^L$  (this suffices by Lemma 2.11). For each  $L \in \text{IP}^A \cap \text{coIP}^A$  there exist  $V_1$  and  $V_2$  such that  $V_1^A$  is  $\text{IP}^A$ -like,  $V_2^A$  is  $\text{IP}^A$ -like,  $L = L(V_1^A)$  and  $\bar{L} = L(V_2^A)$ . Our goal is to show that for each  $i, j$ , either:

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

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  $i$ th prime.

Let requirement  $R_{(i,j,k)}$  be

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

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

$$(**) [[V_i^A \text{ is not IP}^A\text{-like} \vee [V_j^A \text{ is not IP}^A\text{-like} \vee [L(V_i^A) \neq \overline{L(V_j^A)}]]] \text{ OR}$$

(\*\*\*\*)

$$[(\forall k)(\exists b \in \{0, 1\})[\text{card}(\{y : by \in A \wedge |y| = (p_i)^k\}) = 0 \wedge \text{card}(\{y : \bar{b}y \in A \wedge |y| = (p_i)^k\}) > 2^{(p_i)^k - 1}] \text{ and}$$

$$(\forall k)[\text{requirement } R_{(i,j,k)} \text{ is satisfied }]].$$

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<sup>A</sup>-like or  $V_j^A$  non-IP<sup>A</sup>-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<sup>A</sup>-like properties of  $V_i^A$  and  $V_j^A$ .

In stage  $\langle i, j, k \rangle$ , we either:  
satisfy  $(***)$  by adding strings of length  $(p_i)^k + 1$ , for some  $k$ , to  $A$  or  
without violating  $(***)$ , 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  $(***)$  has already been satisfied, 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  $n = (p_i)^k$  be so large that:

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

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

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

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<sup>A</sup>-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

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

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

5. Elseif  $M_k^{L(V_i^{A_{\langle i, j, k \rangle} \cup B})}$  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  $\text{card}(S) > 2^{n/2}$ )

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

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  $\langle i, j \rangle + 1$ .

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

We claim that for steps 5 and 6,  $\text{card}(S) \leq 2^{n-1}$  is not possible. We prove this for step 5 (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 for each string  $w$  in  $0\{0,1\}^n - S$ , there exists a string,  $w_q \in T$ , such that,  $w_q \in L(V_i^{A_{\langle i,j,k \rangle} \cup B \cup S}) \not\leftrightarrow w_q \in L(V_i^{A_{\langle i,j,k \rangle} \cup B \cup S \cup \{w\}})$ . 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). 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  $\text{card}(S) > 2^n/2$ .

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

Note that if we never find a way of satisfying  $(***)$  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  $(\forall j)[\text{requirement } R_{\langle i,j,k \rangle} \text{ is satisfied}]$ . Thus the requirement  $(***)$  is satisfied. Thus we have met requirements that are sufficient to ensure the theorem. ■

## 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 are FewP-like or IP-like. 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 [WT89] have shown that if certain conditions hold then m-complete sets and T-complete sets differ in PSPACE.

Class ( $\mathcal{C}$ )	$\mathcal{C}$ does not robustly possess many-one complete sets	$\mathcal{C}$ does not robustly possess Turing complete sets
$\text{NP} \cap \text{coNP}$	[Sip82]	[Gur83]
$\text{IP} \cap \text{coIP}$	Corollary 4.2	Corollary 4.2
BPP	[HH88]	[HH88] plus [AS86]
R	[Sip82]	Corollary 4.2
coR	[Sip82]	Corollary 4.2
ZPP	Corollary 4.2	Corollary 4.2
FewP	Corollary 3.2	Corollary 3.2
$\text{FewP} \cap \text{coFewP}$	Corollary 3.2	Corollary 3.2
UP	[HH88]	Corollary 3.2
coUP	[HH88]	Corollary 3.2
$\text{UP} \cap \text{coUP}$	Corollary 3.2	Corollary 3.2

Table 1: Results on Robust Completeness

Also, there is a relativized world in which the boolean hierarchy contains bounded truth-table complete sets but not  $k$ -truth-table complete sets [CGH<sup>+</sup>88]. Table 1 summarizes the results of this and earlier papers on complete sets for promise classes.

## 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

- [AR88] E. Allender and R. Rubinfeld. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.
- [AS86] K. Ambos-Spies. A note on complete problems for complexity classes. *Information Processing Letters*, 23:227–230, 1986.
- [BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988.

- [BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the  $P=?NP$  question. *SIAM Journal on Computing*, 4(4):431–442, 1975.
- [BR88] J. Balcázar and D. Russo. Immunity and simplicity in relativizations of probabilistic complexity classes. *Theoretical Informatics and Applications (RAIRO)*, 22(2):227–244, 1988.
- [CGH<sup>+</sup>88] 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.
- [CH91] J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters*, 38(4):215–219, 1991.
- [EHTY89] D. Eppstein, L. Hemachandra, J. Tisdall, and B. Yener. Probabilistic and unambiguous computation are incomparable. In *Computing and Information: Proceedings of the 1989 International Conference on Computing and Information (ICCI 1989)*, pages 65–70. North-Holland, 1989.
- [FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP? *Information Processing Letters*, 28, 1988.
- [FS89] L. Fortnow and M. Sipser. Probabilistic computation and linear time. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 148–156. ACM Press, May 1989.
- [GG86] J. Geske and J. Grollmann. Relativizations of unambiguous and random polynomial time classes. *SIAM Journal on Computing*, 16(2):511–519, 1986.
- [Gil77] J. Gill. Computational complexity of probabilistic turing machines. *Siam Journal of Computing*, 6:675–695, 1977.
- [GJ86] 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.
- [GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–335, 1988.
- [Gur83] 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.
- [HH88] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.



- [HH91] J. Hartmanis and L. Hemachandra. One-way functions and the non-isomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.
- [HI85] J. Hartmanis and N. Immerman. On complete problems for  $NP \cap coNP$ . In *Automata, Languages, and Programming (ICALP 1985)*, pages 250–259. Springer-Verlag *Lecture Notes in Computer Science #194*, 1985.
- [HR] L. Hemachandra and R. Rubinfeld. Separating complexity classes with tally oracles. *Theoretical Computer Science*. In press.
- [HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [KMR88] S. Kurtz, S. Mahaney, and J. Royer. Collapsing degrees. *Journal of Computer and System Sciences*, 1988.
- [Ko85] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37:1–30, 1985.
- [Kur83] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report TR83-001, University of Chicago Department of Computer Science, Chicago, IL, 1983.
- [LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 2–10. IEEE Computer Society Press, October 1990.
- [LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.
- [Mah88] 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.
- [Rac82] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the Association of Computing Machinery*, 29(1):261–268, 1982.
- [Reg88] K. Regan. Minimum-complexity pairing functions. Technical Report MSI TR 88-92, Cornell University, Ithaca, NY, September 1988.
- [Reg89] K. Regan. Provable complexity properties and constructive reasoning. Manuscript, April 1989.
- [Rog67] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.
- [Sch85] U. Schöning. Robust algorithms: A different approach to oracles. *Theoretical Computer Science*, 40:57–66, 1985.

- [Sha90] A. Shamir.  $IP=PSPACE$ . In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 11–15. IEEE Computer Society Press, October 1990.
- [Sip82] 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.
- [Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.
- [WT89] O. Watanabe and S. Tang. On polynomial time turing and many-one completeness in  $PSPACE$ . In *Proceedings 4th Structure in Complexity Theory Conference*, pages 15–23. IEEE Computer Society Press, 1989.
- [Zac86] 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.