# Simultaneous Strong Separations of Probabilistic and Unambiguous Complexity Classes *

David Eppstein † Lane A. Hemachandra ‡ James Tisdall §
Bülent Yener ¶

**Abstract**

We study the relationship between probabilistic and unambiguous computation, and provide strong relativized evidence that they are incomparable. In particular, we display a relativized world in which the complexity classes embodying these paradigms of computation are mutually immune. We answer questions formulated in—and extend the line of research opened by—Geske and Grollman [15] and Balcázar and Russo [3].

## 1 Introduction: Why Compare Computational Paradigms?

Many complexity classes have been defined in recent years to characterize the computational powers of natural approaches to computation. However,

*Some of these results were announced at the 1989 International Conference on Computing and Information, Toronto, Canada.

†Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. Research performed in part while at Columbia University, supported in part by NSF grants DCR-85-11713 and CCR-86-05353.

‡Department of Computer Science, University of Rochester, Rochester, NY 14627. Research performed in part while at Columbia University. Research supported in part by the National Science Foundation under grant CCR-8809174/CCR-8996198 and a Presidential Young Investigator Award, and by a Hewlett-Packard Corporation equipment grant.

§Department of Computer Science, University of Michigan, Ann Arbor, MI 48108. Research performed in part while at AT&T Bell Laboratories and Columbia University.

¶Computer Science Department, Columbia University, New York, NY 10027. Research supported in part by the Center for Telecommunications Research, Columbia University.

our prolific creation of complexity classes has outpaced our progress in interrelating these classes. Investigating relativized comparisons of complexity classes is one fundamental way of augmenting our current knowledge [27, 34] about possible class inclusions and oracle separations. Even the *failure* to find relativized separations often leads to the discovery of important new class inclusions and collapses; thus the collapse of the strong exponential hierarchy [21], the inclusion of FewP in parity polynomial time [11], and the inclusion of $\mathrm{P}^{\mathrm{NP}[\log]}$ in probabilistic polynomial time [5, 32] were all found due to failed separations.

In this paper, we compare probabilistic computation with unambiguous computation and prove strong (mutual immunity) separations.

## 2 Basic Paradigms and Previous Work

In this section, we discuss three paradigms of computation—probabilistic computation, unambiguous computation, and unique computation—and the classes that model them.

Probabilistic Turing machines were studied by Gill [16]. Depending on the exact definition of acceptance and the bounds put on errors, various probabilistic complexity classes can be obtained. In this paper, we will focus on the class R, random polynomial time, and the class BPP, bounded-error probabilistic polynomial time. Both these classes, though perhaps not "feasible" in the sense of deterministic polynomial time, are "feasible" in practice, as a Turing machine with a coin can accept languages from these classes with very low error probability.

**Definition 1 ([16, 2])**

1. BPP *is the class of languages recognized by polynomial-time probabilistic Turing machines whose error probability is bounded above by some positive constant* $\epsilon < 1/2$.

2. R *is the class of languages accepted by polynomial time probabilistic Turing machines that have zero error probability for inputs not in the language, and error probability bounded by some* $\epsilon < 1/2$ *for inputs in the language.*

It follows immediately from the definitions that $\mathrm{P} \subseteq \mathrm{R} \subseteq \mathrm{BPP} \cap \mathrm{NP}$, and Lautemann [25] and Sipser [31] showed that $\mathrm{BPP} \subseteq \Sigma_2^{\mathrm{P}} \cap \Pi_2^{\mathrm{P}}$.

Unambiguous computation was introduced by Valiant as a moderate form of nondeterminism.

**Definition 2 ([33])**

1. *A nondeterministic Turing machine is* unambiguous *if, for every input, the machine has at most one accepting computation (accepting path).*

2. UP *is the class of languages accepted by polynomial-time unambiguous Turing machines.*

Unambiguous polynomial time falls between P and NP; P⊆UP⊆NP. There is growing evidence that UP may lack complete sets, even with respect to Turing reductions [22, 20]. Unambiguous computation is related to and motivated by cryptography; Grollmann and Selman have shown that P≠UP if and only if one-way functions exist [17].

A related concept is that of unique acceptance. Blass and Gurevich [7] defined a class that they called UNIQUE SOLUTION or US, which models the sets accepted by nondeterministic Turing machines that by definition accept if and only if they have exactly one accepting path.

**Definition 3 ([7])** US= $\{L|$ *there is a polynomial predicate* $P$ *and an integer* $k$ *such that for all* $x$*:* $x \in L$ *if and only if there is exactly one element in the set* $\{y|\, P(x, y) \wedge |y| \leq |x|^k\}\}$.

Though UP machines can never have more than one accepting path, US machines can—this simply causes them to reject. Immediate relationships are: US is coNP-hard and UP⊆US∩NP [7].

A further generalization of US that has been studied is the Counting Hierarchy [10, 18]. The usual definition of the counting hierarchy is as follows:

**Definition 4** *Given a nondeterministic polynomial time machine* $M$ *and a set of integers* $S$*, let* $L(M, S)$ *be the strings that, when input to machine* $M$*, cause the number of accepting paths of* $M$ *to be a member of* $S$*. Then* CH *is the class of languages* $L(M, S)$ *where* $S$ *is either finite or cofinite.*

By abuse of notation we will call such a pair $(M, S)$ a *counting machine.* The reader will note that there is no explicit hierarchy obvious in this definition. However such a hierarchy can be constructed, and we will need it for

our later proofs. Given a finite set $S$, let $m(S)$ denote the largest number in $S$; given a cofinite set let $m(S)$ denote the largest number not in $S$.

**Definition 5** $\mathrm{CH}_i$ *is the class of languages* $L(M, S)$ *where* $S$ *is either finite or cofinite, and where* $m(S) < i$.

The following facts then follow easily.

- If $i \leq j$ then $\mathrm{CH}_i \subset \mathrm{CH}_j$.
- $\mathrm{CH} = \bigcup_i \mathrm{CH}_i$.
- $\mathrm{NP} \subset \mathrm{CH}_1 = \mathrm{NP} \cup \mathrm{coNP}$.
- $\mathrm{US} \subset \mathrm{CH}_2 = \mathrm{US} \cup \mathrm{coUS}$.

For a complexity class $\mathcal{C}$, we use co$\mathcal{C}$ to denote $\{A | \bar{A} \in \mathcal{C}\}$. For classes whose notion of relativization [23, 1] is defined in a standard way, we use $\mathcal{C}^A$ to denote class $\mathcal{C}$ relativized with oracle set $A$. Informally, this means that the machines involved have access to oracle $A$ (a hypothetical unit cost subroutine for the set $A$).

The philosophy behind relativization is simple: we relativize to explore the limits of current mathematical proof techniques, and to certify plausibilities about complexity classes. Informally, if statement $S$ fails in some relativized world, we can never hope to prove $S$ in the real world by any relativizable proof technique. Since almost all standard proof techniques relativize, relativized results set tight limits on the current power of mathematical proofs. However, theorems that fail to relativize do exist (see [19]).

Finally, we say an infinite set $S$ is $\mathcal{C}$-immune if $S$ contains no infinite subset that is a member of $\mathcal{C}$ [28]. We say a class $\mathcal{C}_1$ is $\mathcal{C}_2$-immune if $\mathcal{C}_1$ contains a $\mathcal{C}_2$-immune set.

There is a large literature on relativization. Among the results that inspired this paper are the following. Relativized relations between US and NP were found by Blass and Gurevich [7] and have been generalized [10]. With respect to a random oracle, it has long been known that US$\neq$P=R=BPP [6] and Kurtz, Mahaney, and Royer have recently shown that P$\neq$UP with probability one relative to a random oracle ([24]; see also [29]).[1] Rackoff initially relativized unambiguous and probabilistic classes to separate them from NP, and to separate them from P [27]. Geske and Grollmann expanded on this to

[1] The result was first asserted by Beigel in [4], but the proof has been retracted. Beigel has alternate proofs [Beigel, personal communication, 1989].

*simultaneously* separate probabilistic classes from both NP and from P, and to *simultaneously* separate unambiguous classes from NP and from P [15]. They introduced some immunity results to this setting, most notably showing that there is a relativized world in which UP is R-immune. Balcázar and Russo [3] continued this work and found many immunity results between pairs of probabilistic complexity classes, and between probabilistic and nondeterministic complexity classes. Balcázar and Russo concluded by proposing as an area for further research the study of immunity results for unambiguous nondeterministic Turing machines [3, p. 243]. We pursue this study, and thus extend the series of results described above.

# 3 Mutually Incomparibility and Mutual Immunity

Our goal is to give relativized evidence that probabilistic and unambiguous classes are incomparable. Incomparability results between deterministic and nondeterministic complexity classes with differing time bounds have been studied by Gasarch, Homer, and Lischke [14, 26, 13]. However, relativized incomparability results are not the strongest type of incomparibility result. Gasarch proved a stronger type of separation between NP and bounded versions of deterministic exponential time [13], and we will also focus on obtaining such "strong" separation results.

**Definition 6** *Complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are* relativizably incomparable *(written $\mathcal{C}_1 \,||\, \mathcal{C}_2$) if there is an oracle set $A$ such that $\mathcal{C}_1^A \not\subseteq \mathcal{C}_2^A$ and $\mathcal{C}_1^A \not\supseteq \mathcal{C}_2^A$.*

**Definition 7**

1. *$\mathcal{C}_1$ and $\mathcal{C}_2$ are* mutually immune *if there exist languages $L_1 \in \mathcal{C}_1$ and $L_2 \in \mathcal{C}_2$ such that $L_1$ is $\mathcal{C}_2$-immune and $L_2$ is $\mathcal{C}_1$-immune.*

2. *We write $\mathcal{C}_1 <> \mathcal{C}_2$ if there exists an oracle $A$ such that $\mathcal{C}_1^A$ and $\mathcal{C}_2^A$ are mutually immune.*

One must be careful. Merely showing that a language $L$ is not in a certain complexity class $\mathcal{C}$ does not guarantee that it is $\mathcal{C}$-immune [6, 30]. For example, there are relativized worlds in which the boolean closure of NP—the so-called boolean hierarchy—contains sets that are not in $\mathrm{D^P}$; nonetheless, every infinite set in the boolean hierarchy has an infinite $\mathrm{D^P}$ subset [9]. Thus immunity may be a more demanding requirement than mere separation.

Our main result is that a mutual immunity relation holds between probabilistic computation and unambiguous computation in a relativized world, and such a relation also holds between probabilistic computation and unique computation.

**Theorem 1**

- BPP<> $\mathcal{C}$ *for* $\mathcal{C}$ *equal to any of the following language classes:* UP, coUP, UP∩coUP, NP, coNP, NP∩coNP, US, coUS, US∩coUS, CH, *and* BH.
- coR<> $\mathcal{C}$ *for* $\mathcal{C}$ *equal to any of the following language classes:* UP, coUP, NP, coUS, *and* UP∩coUP.

Mutual immunity implies mutual incomparibility, so these relations also give us a number of mutual incomparibility results. The dual mutual incomparibilities obtained by replacing each complexity class $\mathcal{C}$ by co$\mathcal{C}$ in the above relations then also follow as corollaries; e.g., R $||$ coNP. However note that mutual immunity does not dualize so trivially.

The proofs rely on interlacing the priorities of the two immunity constructions implicit in each part of the theorem while maintaining appropriate invariants in the oracle; the underlying immunity constructions themselves are novel.

Geske and Grollmann [15] proved the following strong separation: there is a relativized world in which UP contains an R-immune set. An immediate corollary of theorem 1 is a strengthening of their result.

**Corollary 1** *There is an oracle* $A$ *such that* $\mathrm{UP}^A$ *is* $\mathrm{BPP}^A$*-immune.*

# 4 Proof of Mutual Immunity

The immunity results with BPP in theorem 1 follow from the obvious containments among language classes, together with the following result:

**Theorem 2** *There exist languages* $L_1$ *and* $L_2$ *and oracle* $A$ *such that*

- $L_1$ *is in* $\mathrm{UP}^A \cap \mathrm{coUP}^A$.
- $L_1$ *is* $\mathrm{BPP}^A$*-immune.*
- $L_1$ *is infinite.*

- $L_2$ *is in* $\mathrm{BPP}^A$.
- $L_2$ *is* $\mathrm{CH}^A$*-immune.*
- $L_2$ *is infinite.*

**Proof:** We build our oracle $A$ in stages. After each stage $i$, for some number $\ell_i$ we will have fixed all answers to queries of length less than or equal to $\ell_i$, and left undefined all answers to queries of length greater than $\ell_i$. At stage $i+1$ we will be considering the behavior of a finite number of nondeterministic polynomial time machines on a particular input string of length $\ell_i$. Therefore, the number $\ell_i$ will be chosen so that all queries made by the machines under consideration will be fixed. In particular $\ell_0 = 1000$, and $\ell_{i+1} = 2^{\ell_i}$ should suffice for our constructions. We denote the partial oracle constructed in stage $i$ by $A_i$.

Let $f(j)$ be the number of strings of length $j \leq \ell_i$ in $A_i$. Then we maintain the following invariants:

- For each $j$, either $f(3j) \leq 2^j$ or $f(3j) \geq 2^{3j-1}$.
- For each $j$, $f(3j+1) + f(3j+2) = 1$.

We define the languages required by the theorem as follows:

- $L_1 = \{0^j | f(3j+1) = 1\}$.
- $L_2 = \{0^j | f(3j) \geq 2^{3j-1}\}$.

Then clearly, if the invariants are maintained, $L_1$ is in $\mathrm{UP}^A \cap \mathrm{coUP}^A$. To see that $L_2$ is in $\mathrm{BPP}^A$, let an oracle machine choose, on input $j$, a random string of length $3j$, and accept if the string is in the oracle. If $j$ is in $L_2$, the machine will accept with probability $1/2$. If $j$ is not in $L_2$, the machine will accept with probability much less than $1/4$. These probabilities can be amplified by making multiple random choices, to match any separation desired in the definition of $BPP$.

Number all the possible CH oracle machines by $C_1$, $C_2$, etc., with the requirement that the time taken by machine $C_i$ on input $x$ is at most $|x|^{\log i}$, and the machine is in $\mathrm{CH}_i$. Number the probabilistic polynomial time oracle machines $B_1$, $B_2$, etc., with the same time requirement. Note that machine $B_i$ may or may not be a BPP machine, depending on the choice of oracle; in particular we require each BPP machine (relative to a given oracle) to accept all input strings with probability either at least $2/3$ (in which case

the string is in the language accepted by the machine) or at most 1/3 (in which case the string is rejected by the machine). BPP machines with other probability bounds can easily be made into this form, so this restriction involves no loss of generality.

**Even Stages** In stages $i$ where $i$ is even, we deal with the requirement that $L_2$ be CH-immune, yet infinite. All possible strings of $L_1$ fixed by choices made in this stage will be excluded from $L_1$; therefore this stage cannot affect the immunity results for $L_1$. Similarly, all possible strings of $L_2$ fixed by the choices made in this stage are excluded with the possible exception of $0^{\ell_{i-1}}$.

We say that a machine is *eliminated* if it is forced by the oracle choices already fixed to accept a string not in $L_2$. We require that eventually every CH machine either becomes eliminated, or accepts a finite language (thus failing to accept an infinite subset of $L_2$). Let $g = i/4$; note that $g < \ell_{i-1}$. At stage $i$, for $i$ even, we examine machines $C_1$, $C_2$, ..., $C_g$. Let $M_1$, $M_2$, ..., $M_j$ be those machines not already eliminated.

Note that for each machine $M_k$, in $\mathrm{CH}_c$ (where by construction $c \leq \ell_{i-1}$), the total number of possible oracle entries seen on $c$ different nondeterministic paths for input $0^{\ell_{i-1}}$ is at most $c\ell_{i-1}^{\log \ell_{i-1}} = c2^{\log^2 \ell_{i-1}} < 2^{\ell_{i-1}}/\ell_{i-1}$.

Let $\mathcal{A}$ be the class of possible extensions of $A_{i-1}$, such that

- All strings of the form $\{0^{3n+2}\}$, for $\ell_{i-1} < 3n + 2 \leq \ell_i$ are included in all oracles in $\mathcal{A}$.
- All other strings of lengths from $\ell_{i-1}+1$ to $3\ell_{i-1}-1$ and from $3\ell_{i-1}+1$ to $\ell_i$ are excluded from all oracles in $\mathcal{A}$.
- Either $f(3\ell_{i-1}) \leq 2^{\ell_{i-1}}$ or $f(3\ell_{i-1}) \geq 2^{3\ell_{i-1}-1}$.

Such oracles satisfy the invariants required above. Let $\mathcal{A}'$ be the subclass of $\mathcal{A}$ with the further requirement that $f(3\ell_{i-1}) \leq 2^{\ell_{i-1}}$. I.e., $\mathcal{A}'$ consists of those oracles for which $0^{\ell_{i-1}}$ is not in $L_2$.

If there is an oracle $A'$ in $\mathcal{A}'$ such that any machine $M_k$ accepts string $0^{\ell_{i-1}}$, then we choose one such oracle and set $A_i$ to be all strings of $A'$ no longer than $2^{\ell_{i-1}}$; $M_k$ does not have time to query a string longer than this. Then $M_k$ is eliminated; the newly fixed oracle adds no strings to $L_2$ so if any other machine accepts string $0^{\ell_{i-1}}$, that machine is also eliminated.

In the other case, for each oracle $A'$ in $\mathcal{A}'$, no machine $M_k$ accepts $0^{\ell_{i-1}}$. Some machines may perhaps accept strings not of this form, in which

case they become eliminated; we do not concern ourselves further with this possibility. No machines are eliminated in at least 1/2 of the even stages, because only the first $i/4$ machines are considered.

Then we wish to find an oracle $A$ such that no machine $M_k$ accepts $0^{\ell_{i-1}}$, and yet the string is in $L_2$. We do this by restricting the class of oracles under consideration in stages. In each stage, we force one of the machines to reject, while still including oracles for which $0^{\ell_{i-1}}$ is in $L_2$. After we have completed $j$ such stages, we will have found the oracle that causes all the machines to reject, as we desired.

Then let $\mathcal{A}_0 = \mathcal{A}$. We will define oracle classes $\mathcal{A}_1 \subset \mathcal{A}_0$, $\mathcal{A}_2 \subset \mathcal{A}_1$, etc, until we compute $\mathcal{A}_j$. We maintain the invariants that:

- Each class $\mathcal{A}_k$ is formed by restricting the behavior of oracles in the class on a set of strings. In particular, for each $k$ there are two sets of strings $S_k$ and $S'_k \subset S_k$. Then $\mathcal{A}_k$ consists of all oracles $A$ in $\mathcal{A}$ such that $A \cap S_k = S'_k$. In English, the oracles in $\mathcal{A}_k$ are required to include all strings in $S'_k$, and they are required to exclude all strings in $S_k - S'_k$.

- $S_k$ is a superset of $S_{k-1}$; $S'_k \cap S_{k-1} = S'_{k-1}$. Therefore, $\mathcal{A}_k \subset \mathcal{A}_{k-1}$. In other words, once we fix a string in or out of the oracles, we never take back our choice.

- The sets $S_k$ are small; in particular $|S_k - S_{k-1}| \leq k2^{\ell_{i-1}}/\ell_{i-1}$ therefore $\mathcal{A}_k$ contains some oracles in $\mathcal{A}'$, and some oracles not in $\mathcal{A}'$.

- None of machines $M_1$, $M_2$, ..., $M_k$ accept input string $0^{\ell_{i-1}}$ for any oracle in $\mathcal{A}_k$. The choice of $S_k$ will cause this to be true for $M_k$; then the second invariant will cause it to remain true in later stages.

Assume we have defined $\mathcal{A}_{k-1}$ satisfying the invariants above. We now show how to find $\mathcal{A}_k$ maintaining the invariants. In particular, we must find a small set of strings to add to $S_k$ and $S'_k$ that force $M_k$ to reject string $0^{\ell_{i-1}}$.

Let machine $M_k$ be in class $\mathrm{CH}_c$, for $c \leq \min(i/4, \ell_{i-1})$. If there is an oracle in $\mathcal{A}_{k-1}$ such that $M_k$ has at least $c$ accepting paths, on input $0^{\ell_{i-1}}$, there are two cases: (1) $M_k$ accepts when it finds that many accepting paths. Then it must accept on the oracle in $\mathcal{A}_{k-1} \cap \mathcal{A}'$ formed by fixing the queries made on $c$ such paths, and answering all remaining unfixed queries negatively; but this contradicts the assumption that for all $A' \in \mathcal{A}'$, $M_k$

rejects. (2) $M_k$ rejects when it finds that many accepting paths; then set $\mathcal{A}_k$ to be the oracle class formed by fixing the queries made on $c$ accepting paths. In terms of our notation, we add all queries made to $S_k$, and we add only those queries answered positively to $S'_k$. Then $M_k$ must reject for any oracle in $\mathcal{A}_k$.

If there is no oracle that causes $M_k$ to have $c$ or more accepting paths, Then some number $d < c$ is the maximum number of accepting paths on input $0^{\ell_{i-1}}$ for machine $M_k$, maximized over all oracles in $\mathcal{A}_{k-1}$. Pick some oracle $A'$ achieving this maximum, and let $\mathcal{A}_k$ be the class formed by fixing the queries made on the accepting paths of $M_k$ for oracle $A'$. I.e., we add those strings to $S_k$, and we add the strings in $S_k \cap A'$ to $S'_k$. Then for all oracles in $\mathcal{A}_k$, machine $M_k$ has at least those $d$ accepting paths, and it could not have more or else $A'$ would not have maximized $d$. Since $\mathcal{A}_k$ contains oracles in $\mathcal{A}'$, and since we could not eliminate $\mathcal{A}_k$, $M_k$ must reject when it sees $d$ accepting paths; therefore it must reject for all oracles in $\mathcal{A}_k$.

Now we let $\hat{A}$ be any oracle in $\mathcal{A}_j \cap (\mathcal{A} - \mathcal{A}')$, and let $A_i$ be the strings of $\hat{A}$ no longer than $\ell_i$; no machine $M_k$ has time to query a string longer than this. Then each $M_k$ rejects input string $0^{\ell_{i-1}}$ for oracle $A_i$, but this string is in $L_2$.

The result of the stage is that either $0^{\ell_{i-1}}$ is not in $L_2$, but some $M_k$ accepts it, or $0^{\ell_{i-1}}$ is in $L_2$, and all $M_k$ reject it; the second case happens infinitely often. If we do not add any strings to $L_2$ in the odd stages, the conclusion must be that $L_2$ is infinite, but that any machine that is never eliminated can accept only finitely often (in particular it can only accept in those stages before it becomes one of the machines in the list $C_1, C_2, \ldots C_g$).

**Odd Stages** The outline for the odd stages is similar to that for the even stages. In this case, a machine is eliminated when it is caused by the oracle choices fixed so far to have probability greater than $2/3$ of accepting a string not in $L_1$, or when there is an input string that causes it to have probability between $1/3$ and $2/3$ of accepting. Here $M_1, M_2, \ldots M_k$ are the uneliminated BPP machines in the list $B_1, B_2, \ldots B_g$ for $g = \min(i/4, \ell_{i-1})$.

We let $\mathcal{A}$ be the class of oracle extensions such that

- All strings of the form $\{0^{3n+2}\}$, for $\ell_{i-1} < 3n + 2 \leq \ell_i$ and $3n + 2 \neq 3\ell_{i-1} + 2$ are included in all oracles in $\mathcal{A}$.
- All other strings of lengths from $\ell_{i-1} + 1$ to $3\ell_{i-1}$ and from $3\ell_{i-1} + 3$ to $\ell_i$ are excluded from all oracles in $\mathcal{A}$.

- $f(3\ell_{i-1}+1) + f(3\ell_{i-1}+2) = 1$.

Thus we exclude all strings with lengths from $\ell_{i-1}/3$ to $\ell_i/3$ from membership in $L_1$ and $L_2$, except that $0^{\ell_{i-1}}$ may be in $L_1$. We let $\mathcal{A}^+$ be the oracles in $\mathcal{A}$ for which $0^{\ell_{i-1}}$ is in $L_1$, and $\mathcal{A}^- = \mathcal{A} - \mathcal{A}^+$.

If there is an oracle in $\mathcal{A}^-$ such that one of the $M_k$ accepts, or such that some $M_k$ has probability between 1/3 and 2/3 of accepting some string, we fix the strings in that oracle up to length $\ell_i$, to create $A_i$. Then $M_k$ is eliminated, and no strings have been added to $L_1$.

Otherwise, no machine can be eliminated; this second case happens infinitely often since $g < i/4$. Let oracle $A'$ be the oracle meeting all the requirements of class $\mathcal{A}$ except that it does not contain any strings of lengths $3\ell_{i-1}+1$ and $3\ell_{i-1}+2$. $A'$ does not satisfy our invariants, but we now show that the behavior of the machines with oracle $A'$ is in some sense close to their behavior with a certain oracle in $\mathcal{A}^-$ and another oracle in $\mathcal{A}^+$. Therefore if no machine could be eliminated by the oracle in $\mathcal{A}^-$, no machine will accept string $0^{\ell_{i-1}}$ with the oracle in $\mathcal{A}^+$.

If we run all machines $M_1$, $M_2$, $\ldots M_j$ with input string $0^{\ell_{i-1}}$ and oracle $A'$, then the total *expected* number of queries made by all machines together is at most $g\ell_{i-1}^{\log g} < 2^{\log \ell_{i-1} \log\log \ell_{i-1}} < 2^{\ell_{i-1}}$. The numbers of strings of lengths $3\ell_{i-1}+1$ and $3\ell_{i-1}+2$ are $2^{3\ell_{i-1}+1}$ and $2^{3\ell_{i-1}+2}$ respectively. Therefore the average probability of a string of those lengths being queried by at least one machine is at most $2^{-2\ell_{i-1}-1}$. Let $q$ and $r$ be the strings of lengths $2^{3\ell_{i-1}+1}$ and $2^{3\ell_{i-1}+2}$ respectively least likely to be queried. Then their probabilities of being queried are at most this average. The probability of at least one of $q$ or $r$ being queried by at least one machine $M_k$ is then bounded by the sum of their individual probabilities; this adds to at most $2^{-2\ell_{i-1}}$.

Let oracle $A^q$ be formed by adding string $q$ to $A'$, and similarly let $A^r$ be formed by adding $r$ to $A'$. Then $A^q$ is in $\mathcal{A}^+$, and $\mathcal{A}^r$ is in $\mathcal{A}^-$. We now show that, if no machine could have been eliminated by oracle $A^r$, then each machine $M_k$ rejects string $0^{\ell_{i-1}}$ given oracle $A^q$.

The probability that $M_k$ queries one of strings $q$ and $r$ on input $0^{\ell_{i-1}}$, and can therefore distinguish oracles $A^q$ and $A^r$, is at most the probability that any of the $j$ machines queries one of the two strings, which was bounded above by $2^{-2\ell_{i-1}}$. Therefore the probability that $M_k$ accepts input $0^{\ell_{i-1}}$ for oracle $A^q$ differs from the probability of acceptance for oracle $A^r$ $2^{-2\ell_{i-1}} < 1/6$. But since $M_k$ could not be eliminated, it accepts the input for $A^r$ with probability at most 1/3. Therefore it accepts the input for $A^q$ with

probability at most 1/2, and thus either rejects the input or is eliminated.

We create $\mathcal{A}_i$ by fixing all strings in $A^q$ of length at most $\ell_i$. No machine $M_k$ can reach strings longer than this, so each machine must behave the same for any oracle in class $\mathcal{A}_i$ as it does for $A^q$. In particular, it must reject input $0^{\ell_i - 1}$ even though we have caused that string to be in language $L_1$.

Therefore again, either we eliminate one of the $M_k$, or (infinitely often) we extend $L_1$ without letting any of the $M_k$ accept. Therefore $L_1$ is made infinite and BPP-immune. □

Given a class $\mathcal{C}$ of oracle machines, and an oracle $A$, we define $\mathcal{C}^A$ to be the class of languages accepted by machines in the class using the given oracle.

**Corollary 2** *For any class $\mathcal{C}$ of oracle machines such that, for all oracles $A$, $\mathrm{UP}^A \cap \mathrm{coUP}^A \subset \mathcal{C}^A \subset \mathrm{CH}^A$, $\mathrm{BPP} <> \mathcal{C}$.*

In particular this proves the claims of immunity with BPP made in theorem 1.

We should note that, since all languages used in our mutual immunity proofs are subsets of $0^*$, the preceeding and subsequent results also apply to the analogous exponential time computation classes (see [12]).

It only remains to prove our claims of immunity with R. Again, these follow from a single result:

**Theorem 3** *There exist languages $L_1$ and $L_2$ and an oracle $A$ such that*

- *$L_1$ is in $\mathrm{UP}^A \cap \mathrm{coUP}^A$.*
- *$L_1$ is $\mathrm{BPP}^A$-immune.*
- *$L_1$ is infinite.*
- *$L_2$ is in $\mathrm{coR}^A$.*
- *$L_2$ is $\mathrm{coUS}^A$-immune.*
- *$L_2$ is infinite.*

**Proof:** We let $L_1$ and $L_2$ be the same languages as in the proof of theorem 2. However, we must now force $L_2$ to be in coR. To do this, we restrict our attention to oracles $A$ satisfying the following conditions, which are stricly stronger than the conditions given in the proof of theorem 2.

- For each $j$, $f(3j+1) + f(3j+2) = 1$.
- For each $j$, either $f(3j) \leq 2^j$ or $f(3j) = 2^{3j}$. I.e., either all queries of length $3j$ are answered positively, or most of them are answered negatively.

Again we construct our oracle $A$ in stages. In the odd stages, we satisfy the requirements that $L_1$ be infinite and $\mathrm{BPP}^A$-immune. These stages are identical to those in the proof of theorem 2, so we omit their description here.

In the even stages, we maintain a list of coUS machines. We must either eliminate a machine on the list by causing it to accept some input string while not adding that string to $L_2$ or we must add a string to $L_2$ not recognized by any uneliminated machine on the list. The string will as usual be $0^{\ell_{i-1}}$.

As in the previous proof let $\mathcal{A}$ be the class of extensions of oracle $A_{i-1}$ meeting our restrictions, and with all strings of lengths other than $3\ell_{i-1}$ included or excluded in a fixed way so as to cause the corresponding input strings not to be in $L_1$ or $L_2$. Let $U_1, U_2, \ldots U_g$ be the first $g$ coUS-machines, for $g = \min(i/4, \ell_{i-1})$, and let $M_1, M_2, \ldots M_j$ be the set of those machines not already eliminated.

Let $\hat{A}$ be the oracle constructed from $A_{i-1}$ by including all strings of length $3\ell_{i-1}$, and omitting all other length strings as before. Then $\hat{A}$ is the unique oracle in $\mathcal{A}$ for which string $0^{\ell_{i-1}}$ is in $L_2$. Let $\mathcal{A}'$ be $\mathcal{A} - \hat{A}$.

If any oracle $A'$ in $\mathcal{A}'$ causes some machine $M_k$ to accept (either by having no accepting paths, or by having two or more accepting paths) fix $A_i$ to be the reachable strings in $A'$ as before. This eliminates $M_k$ while keeping input string $0^{\ell_{i-1}}$ out of language $L_2$.

The other case is that all machines $M_k$ reject string $0^{\ell_{i-1}}$ for all oracles in $\mathcal{A}'$, and therefore no machine can be eliminated. This case must happen infinitely often because $g < i/4$. We now show that, in this case, all machines $M_k$ also reject string $0^{\ell_{i-1}}$ for oracle $\hat{A}$; therefore we can add the string $0^{\ell_{i-1}}$ to language $L_2$ without adding it to the languages accepted by any of the machines.

We fix our attention on some uneliminated machine $M_k$. First note that, for input $0^{\ell_{i-1}}$ and oracle $\hat{A}$, machine $M_k$ has zero or one accepting paths; if it had more than one, we could fix the query answers made on two such paths, and remove all other strings of length $3\ell_{i-1}$ from the oracle, giving an oracle in $\mathcal{A}'$ for which $M_k$ accepts input string $0^{\ell_{i-1}}$. We now show that $M_k$

must have exactly one accepting path, and therefore does not accept string $0^{\ell_{i-1}}$.

The structure of this section of the proof is similar to a number of previous results in the relativization literature, starting with Baker et al. [1] and Rackoff [27]. In those cases one wishes to test whether a machine accepts or rejects a string, given an oracle; this is done by iteratively determining the membership of a small set of strings in the oracle, until the set contains all the queries made by the machine. Here we already know all strings in the oracle, but we use the same iterative structure to show that the machine in fact rejects the given string.

By assumption $M_k$ has exactly one accepting path for input $0^{\ell_{i-1}}$ for each oracle in $\mathcal{A}'$. For each oracle $A' \in \mathcal{A}'$, let $q(A')$ be the set of queries of length $3\ell_{i-1}$ made in the one accepting path of $M_k$. We can assume without loss of generality that, for any two oracles $X$ and $Y$, $|q(X)| = |q(Y)| = c$ for some $c$ bounded by the running time of $M_k$; $M_k$ can make and then ignore the results of some dummy queries if necessary to make this happen.

Let $X_0$ be the oracle in $\mathcal{A}'$ formed by excluding all strings of length $3\ell_{i-1}$. Let $X_1$ be formed from $X_0$ by including all strings of length $3\ell_{i-1}$ queried in $q(X_0)$; i.e. we change all answers to queries in $q(X_0)$ to positive. Repeating this process, let $X_i$ be formed by changing all answers to queries in $q(X_{i-1})$ to positive. We repeat this process at most $c$ times, so each oracle $X_i$ contains at most $c(c+1) \leq \ell_{i-1}^{\log \ell_{i-1}} \leq 2^{\ell_{i-1}}$ strings of length $3\ell_{i-1}$; therefore all such oracles are in $\mathcal{A}'$.

Then for each $X_i$, $i \geq 1$ it must be that $q(X_i)$ and $q(X_0)$ are non-disjoint; otherwise we could form an oracle in $\mathcal{A}'$ for which $M_k$ had two accepting paths, by fixing the answers to queries in $q(X_0)$ to be those of oracle $X_0$, fixing those of $q(X_i)$ to those of oracle $X_i$, and answering all other queries negatively. This would contradict the assumption that no machine could be eliminated.

Further, each $q(X_i)$, $i \geq 2$ must be non-disjoint with $q(X_1) - q(X_0)$, if $q(X_1) - q(X_0)$ is non-empty; otherwise, we could form an oracle from which $M_k$ has two accepting paths, by answering the queries in $q(X_0)$ positively, the answers in $q(X_1) - q(X_0)$ negatively, and all other queries in $q(X_i)$ as they are set in oracle $X_i$.

Inductively, each $q(X_i)$ must be non-disjoint with each non-empty set of the form $q(X_j) - \bigcup_{k<j} q(X_k)$, for each $j < i$. Otherwise, we could form two accepting paths, by setting the queries in $q(X_i) \cap q(X_j)$ to all positive, matching the answers to those queries both in $X_i$ and $X_j$, and setting the

remaining answers in $q(X_i)$ and $q(X_j)$ to the answers given by $X_i$ and $X_j$ respectively.

Therefore, for some $i \leq c$, we must have $q(X_i) \subset \bigcup_{j<i} q(X_j)$, for otherwise $q(X_c)$ would have to have points in $c$ non-empty disjoint sets, contradicting the assumption that $|q(X_i)| \leq c$. By construction oracle $X_i$ contains all strings in $\bigcup_{j<i} q(X_j)$; it follows that all strings in $q(X_i)$ are also in $X_i$; that is, all queries of length $3\ell_{i-1}$ made on the accepting path of machine $M_k$ for input string $0^{\ell_{i-1}}$ and oracle $X_i$ are answered positively. Therefore that accepting path must also exist for oracle $\hat{A}$, which answers all queries positively. So $M_k$ has an accepting path, and must reject string $0^{\ell_{i-1}}$ for oracle $\hat{A}$.

Therefore, we can set partial oracle $A_i$ to be the reachable queries of $\hat{A}$, and we will be adding a string to $L_2$ that all machines $M_k$ must reject. □

**Corollary 3** *For any class $\mathcal{C}$ of oracle machines such that, for all oracles $A$, $\mathrm{UP}^A \cap \mathrm{coUP}^A \subset \mathcal{C}^A \subset \mathrm{coUS}^A$, $\mathrm{coR}^{<>} \mathcal{C}$.*

In particular this holds for $\mathcal{C}$ taken to be UP, coUP, UP∩coUP, NP, and coUS, as claimed in theorem 1.

# 5 Conclusions and Open Problems

This paper gives strong evidence that the paradigm of probabilistic computation is incomparable with both the unambiguous computation paradigm and the unique computation paradigm. In particular, we proved mutual immunity results.

We have left a number of possible results unproven. For instance, it might be possible to extend the immunity results claimed in theorem 3 to immunity between coR and $CH^+$, the class of languages accepted by counting machines that accept when they have many accepting paths (NP and coUS machines are instances of this, accepting when they have more than 1 or 2 paths respectively).

Can even stronger separations be proven? In particular, can the "ultimate" result—a probability one mutual immunity result—be obtained? It follows from a very recent advance that the answer is "no." Kurtz, Mahaney, and Royer [24] have recently shown that P≠UP with probability one relative to a random oracle. Since it is well known that US≠P=BPP with probability one [6], "ultimate" results of the form above are impossible for BPP versus US, or for BPP versus UP.

## Acknowledgements

We are grateful to Eric Allender for some enjoyable conversations.

## References

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

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

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

[4] R. Beigel. On the relativized power of additional accepting paths. In *Proceedings 4th Structure in Complexity Theory Conference*, IEEE Computer Society Press, pages 216–224, June 1989.

[5] R. Beigel, L. Hemachandra, and G. Wechsung. On the power of probabilistic polynomial time: $\mathrm{P}^{\mathrm{NP}[\log]} \subseteq$PP. In *Proceedings 4th Structure in Complexity Theory Conference*, pages 225–227. IEEE Computer Society Press, June 1989.

[6] C. Bennett and J. Gill. Relative to a random oracle $A$, $\mathrm{P}^A \neq \mathrm{NP}^A$ with probability 1. *SIAM J. Comput.*, 10:96–113, 1981.

[7] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

[8] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *28th Annual IEEE Symp. Foundations of Computer Science*, October 1987.

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

[10] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM J. on Comput.*, 18(1):95–111, February 1989.

[11] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, to appear.

[12] D. Eppstein and L. Hemachandra. Upward translation and immunity. In preparation.

[13] W. Gasarch. Oracles for deterministic versus alternating classes. *SIAM J. Comput.*, 16(4):613–627, 1987.

[14] W. Gasarch and S. Homer. Relativizations comparing NP and exponential time. *Information and Control*, 58:88–100, 1983.

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

[16] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675–695, December 1977.

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

[18] T. Gundermann and G. Wechsung. Counting classes with finite acceptance types. *Computers and Artificial Intelligence* 6(5):395–409, 1987.

[19] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the European Association for Theoretical Computer Science*, 27:40–49, October 1985.

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

[21] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences* 39(3):299–322, 1989.

[22] L. Hemachandra. Structure of complexity classes: Separations, collapses, and completeness. In *Mathematical Foundations of Computer Science 1988, Proceedings of the 13th Symposium*, pages 59–72. Springer-Verlag Lecture Notes in Computer Science #324, August/September 1988.

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

[24] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *21st ACM Symposium on Theory of Computing*, pages 157–166, May 1989.

[25] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 14:215–217, 1983.

[26] G. Lischke. Oracle constructions to prove all possible relationships between relativizations of P, NP, NEL, EP, and NEP. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 32:257–270, 1986.

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

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

[29] S. Rudich. Limits on the provable consequences of one-way functions. Ph.D. thesis, University of California at Berkeley, November 1988.

[30] U. Schöning and R. Book. Immunity, relativization, and nondeterminism. *SIAM J. Comput.*, 13:329–337, 1984.

[31] M. Sipser. A complexity theoretic approach to randomness. In *15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[32] S. Toda. On probabilistic computations with a restricted access to NP oracles. Manuscript, Feb 1989.

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

[34] 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.