INFORMATIQUE THÉORIQUE ET APPLICATIONS

JOSÉ L. BALCÁZAR
DAVID A. RUSSO

Immunity and simplicity in relativizations of probabilistic complexity classes

Informatique théorique et applications, tome 22, n° 2 (1988), p. 227-244.

<http://www.numdam.org/item?id=ITA_1988__22_2_227_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 22, n° 2, 1988, p. 227 à 244)

IMMUNITY AND SIMPLICITY IN RELATIVIZATIONS OF PROBABILISTIC COMPLEXITY CLASSES (*)

by José L. BALCÁZAR ([1]) and David A. RUSSO ([2])

Communicated by J. DIAZ

Abstract. – *The existence of immune and simple sets in relativizations of the probabilistic polynomial time bounded classes is studied. Some techniques previously used to show similar results for relativizations of P and NP are adapted to the probabilistic classes. Using these results, an exhaustive settling of all possible strong separations among these relativized classes is obtained.*

Résumé. – *On étudie les relativisations des classes de complexité probabiliste polynômiale. On adapte aux classes probabilistes des techniques déjà utilisées pour établir des résultats similaires pour les relativisations de P et NP. On obtient à partir de ces résultats une classification de toutes les propriétés de séparation forte pour ces classes relativisées.*

INTRODUCTION

Immunity and simplicity in relativizations of complexity classes have been studied by Homer and Maass [7], Schöning and Book [13], and Balcázar [3]. Previous work on these concepts in the complexity-theoretic setting, without mentioning relativizations, appears in Flajolet and Steyaert [6]. The purpose of this paper is to translate these results to the relativizations of the probabilistic complexity classes.

The probabilistic classes *ZPP*, *R*, *BPP* and *PP* were introduced by Gill [5]. The class *R* is denoted *VPP* there. We follow here the notation *R* as in Adleman and Manders [1]. These classes can be defined by focusing on

(*) Received June 1986, revised June 1987
This research was supported in part by a grant from the U.S.A.-Spanish Joint Committee for Educational and Cultural Affairs, and by the National Science Foundation under Grants MCS80-11979 and MCS83-12472. This research was performed while the first author visited the Department of Mathematics, University of California at Santa Barbara.
([1]) Facultat d'Informàtica, Universitat Politécnica de Catalunya, 08028 Barcelona, Spain.
([2]) Department of Mathematics, University of California at Santa Barbara, California 93106, U.S.A.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 88/02 227 18/$3.80/

the number of accepting and rejecting computations of nondeterministic polynomial time machines. If each nondeterministic step is considered as a random event (a coin toss), and acceptance is defined in terms of the probability of finding an accepting computation, the above-mentioned classes arise naturally by bounding the error probability in different ways.

We show that certain modifications of known techniques for relativizing complexity classes apply to the probabilistic classes. Our constructions, based on the theorems of Baker, Gill, and Solovay [2], are similar to the "slow diagonalizations" of Schöning and Book [13] and Balcázar [3], and yield sets in different probabilistic classes that are immune with respect to other classes. As a consequence, strong separations of the classes under consideration are obtained. A strong separation among two classes is a separation witnessed by an infinite set in one class with no infinite subset in the other class.

It is known from the work of Baker, Gill, and Solovay [2] that for any PSPACE-complete set E, $P(E)=P\text{SPACE}(E)$, which implies that all of the classes studied here collapse to $P(E)$. On the other hand, oracles separating different classes can be constructed by diagonalization; some of our results are of this type. Having a question about nonrelativized classes answered in different ways for different relativizations is often used as evidence for the difficulty of finding an answer to the unrelativized case. This is because almost all currently known methods for proving equalities and inequalities of complexity classes remain valid under relativization. This suggests that the study of the relativizations of these classes might provide a better understanding of the properties of the corresponding computation models.

Considering together all our constructions, we can show that every strong separation consistent with the known results can be achieved in the appropriate relativization. For instance, it holds by definition that $R \subseteq NP \cap BPP$ (see [15]). However, it is not known if this inclusion is proper. We show that no proof solving this problem can relativize: the oracles C and D, constructed in section 4, have the properties that $R(C) \subset NP(C) \cap BPP(C)$ and $R(D)=NP(D) \cap BPP(D) \subset NP(D)$. Here "$\subset$" denotes strict inclusion.

In order to obtain these results, two kinds of modifications must be made to the slow diagonalizations. In some cases, instead of constructing the relativization "one word at a time", the words must be decided "a lot at a time" to preserve the acceptance probability. In other cases, a problem arises since probabilistic machines may query exponentially broad spaces of the oracle set, and therefore its result might depend of "many" queried words. This problem arises also in [3], but the same solution cannot be applied here.

To solve it, a definition of "critical string" is given, so that the result of a probabilistic machine only depends essentially on these critical strings; then, a combinatorial argument is presented that ensures the existence of a small number of these critical strings. Our argument is similar to that of Rackoff [10], although we found it independently.

The paper is organized as follows. Definitions, notation and some basic results are stated in section 1. Section 2 contains a result that uses a variation of the techniques of [13] for strongly separating the smallest among the above-mentioned probabilistic classes, *ZPP*, from the class *P*. In section 3 we present a result that uses a technique based in [3], obtaining a strong separation of *NP* from co-*R*. Many consequences follow from this result. The most important new ideas are introduced in section 4, where more careful applications of the immunity technique are presented. The results are summarized in section 5, and it is argued that they completely settle all possible relationships among the classes studied.

PRELIMINARIES

Our computational model is the nondeterministic oracle multitape Turing machine, with input alphabet $\Sigma=\{0,1\}$. Decisional problems are assumed to be coded as subsets of Σ^*. For undefined notions see [8] and [12].

We denote by $|x|$ the length of $x\in\Sigma^*$, and by Σ^n the set of all $x\in\Sigma^*$ with $|x|=n$. We denote $\Sigma^{\leqq n}=\bigcup\{{}^i|\,0\leqq i\leqq n\}$. A total ordering is defined on Σ^* by letting $x<y$ if and only if $|x|<|y|$ or $|x|=|y|$ and $x<y$ in the lexicographic ordering.

For a subset L of Σ^*, the complement of L, i.e. Σ^*-L, is denoted by $\bar{L}$. The symmetric difference $L_1\Delta L_2$ is defined by $L_1\Delta L_2=(L_1\cap\bar{L}_2)\cup(L_2\cap\bar{L}_1)$.

Let C be a class of subsets of Σ^*. A set L is *C-immune* if and only if L is infinite and no infinite subset of L is a member of C. A set L is *C-simple* if and only if $L\in C$ and its complement $\bar{L}$ is C-immune.

We assume the existence of easily computable tupling functions, i.e., bijections from $(\Sigma^*)^n$ to Σ^*, which we denote with angular brackets: $\langle\ldots\rangle$.

A recursive enumeration of the polynomially clocked deterministic (nondeterministic) oracle Turing machines is assumed, and denoted by P_i (NP_i, respectively). The set of all words accepted by P_i (NP_i, respectively) under oracle A is denoted $L(P_i, A)$ ($L(NP_i, A)$ respectively). Recall that for $x\in L(NP_i, A)$ it is enough to have one accepting computation. Without loss of generality we assume that polynomials p_i are simultaneous bounds for the

running times of both P_i and NP_i. Notice that there are infinitely many machines that reject every input no matter what the oracle is; i. e., there exist infinitely many M such that for all A, $L(M, A)=\varnothing$. This fact will be used later.

We will focus on the following complexity classes:

$P(A)$ is the class of sets accepted by some P_i with oracle A;

$NP(A)$ is the class of sets accepted by some NP_i with oracle A;

co-$NP(A)$ is the class of complements of sets in $NP(A)$;

$PP(A)$ is the class of sets "probabilistically accepted" by some NP_i with oracle A, where a string x is probabilistically accepted by NP_i if and only if more than half of the computations of NP_i accept x;

$BPP(A)$ is the class of sets probabilistically accepted by some NP_i with oracle A with bounded error probability [5]; this is equivalent to saying that for each x either more than one half of the computations accept x (and x is considered accepted) or less than one-fourth the computations accept x (and x is considered rejected);

$R(A)$ is the class of sets probabilistically accepted by some NP_i with oracle A with one-sided errors: for each x either more than half the computations accept or no computation accepts;

co-$R(A)$ is the class of complements of sets in $R(A)$;

$ZPP(A)$ is $R(A) \cap$ co-$R(A)$; characterizations of $ZPP(A)$ in terms of the number of accepting or rejecting computations may be found in [5] and [11];

PSPACE(A) is the class of sets accepted by any Turing machine in polynomial space with oracle A; we assume that the polynomial space bound holds for the query tape.

The relative inclusion structure of all of these class is depicted in figure 1. Each arrow indicates that the first class is included in the second. Most of the inclusions presente in figure 1 are known to be either proper inclusions

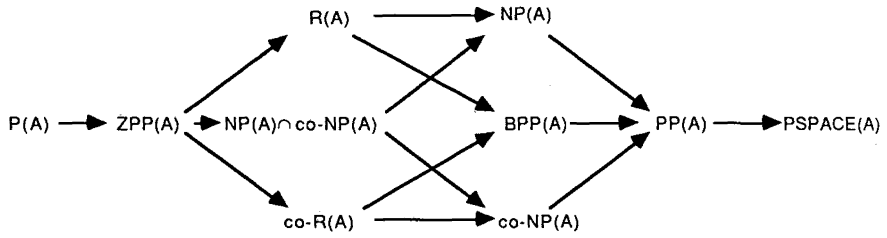

Figure 1

or equalities for the appropriately constructed oracles. In the unrelativized case, all are open problems.

Let A be any oracle set. Suppose that the machine M is a probabilistic machine that witnesses the fact that $L(M, A) \in ZPP(A)$. Then we say that M is a "ZPP machine" under oracle A. Similarly, we apply the shorthands "R machine under A" and "BPP machine under A" to probabilistic machines M witnessing the fact that $L(M, A) \in R(A)$ or $BPP(A)$ respectively.

A set A is PSPACE-complete if and only if $A \in P$SPACE and any set in PSPACE is reducible to A in polynomial time; for related definitions see [8] and [12]. The following result has been pointed out in the introduction, and is due to Baker, Gill, and Solovay [2].

PROPOSITION 1.1: *If A is PSPACE-complete then $P(A) = P\text{SPACE}(A) = P\text{SPACE}$.*

Relative to such a set, polynomial space computable functions are computable in polynomial time (we assume that the polynomial space bound holds for the output tape). We state this fact in the following theorem, which follows from essentially the same prefix-searching argument applied in [14] to the unrelativized classes P and NP.

Fact 1.2. $P(A) = P\text{SPACE}(A)$ if and only if any function computable relative to A in polynomial space is computable relative to A in polynomial time.

THE "IMMUNITY" TECHNIQUES

We show in this section the tightest possible result for P-immunity in the context of the probabilistic polynomial time classes. We construct an oracle A such that a $P(A)$-immune set exists in $ZPP(A)$, the smallest of the classes considered here.

In fact, the proof is close to the basic immunity construction in [13], the key difference being that the "test language" $L(A)$ that diagonalizes over $P(A)$ must be accepted with oracle A by a "ZPP" machine. During the process, we make sure that each machine P_i in an enumeration of all polynomial time clocked deterministic oracle machines is "spoiled". A machine P_i is spoiled as soon as we guarantee that $L(P_i, A)$ is not an infinite subset of $L(A)$.

THEOREM 2.1: *There is a recursive oracle A such that $ZPP(A)$ has a $P(A)$-immune set.*

Proof: The set A will be constructed in such a way that it satisfies a very strong condition. Namely, for each n, either no word of length n in A begins with 0 and less than half the words of length n beginning with 1 are in A, or more than half the words of length n beginning with 0 and all the words of length n beginning with 1 are in A.

It can be seen that if A satisfies this requirement then the set

$$L(A)=\{0^n \mid \exists y, \ |y|=n, \text{ such that } 0y\in A\}$$

can be defined as well as

$$L(A)=\{0^n \mid \forall y \text{ with } |y|=n, \ 1y\in A\}$$

and that a "ZPP" machine accepts it with oracle A. We now construct A in stages. In stage n, the set A_n is defined in an attempt to spoil a "P" machine whose index is in the set R_n. The oracle set A is defined as the union $\bigcup_{n\geq 0} A_n$.

Stage 0: Let $A_0=\varnothing$, $R_0=\varnothing$, and $k(0)=0$.

Stage n: Add a new index to be considered, $R_n=R_{n-1}\cup\{n\}$;
let $k(n)$ be a "sufficiently large" integer; more precisely,
$k(n)=\min\{m \mid \sum_{i\leq n} p_i(m)<2^{m-1}$ and $\max\{p_i(k(n-1)) \mid i<n\}<m\}$;
initially define $A_n=A_{n-1}\cup\{1z \mid |z|=k(n)\}$;
if there is a $j\in R_n$ such that $0^{k(n)}\in L(P_j, A_n)$
then
let $j(n)$ be the least such j;
define A_n so that $0^{k(n)}$ is not a member of the "test language" [i. e., spoil the machine with index $j(n)$] by setting
$A_n=A_n-\{1z \mid |z|=k(n)$, $1z$ was not queried by $P_{j(n)}$ on $0^{k(n)}\}$;
let $R_n=R_{n-1}-\{j(n)\}$;
else
define A_n so that $0^{k(n)}$ is a member of the test language
by setting $A_n=A_n\cup\{0z \mid |z|=k(n)$, $0z$ was not queried by any P_j on $0^{k(n)}\}$;
end if.

The conditions imposed on $k(n)$ guarantee that the oracle meets the conditions stated above. This is so since the condition $\sum_{i\leq n} p_i(k(n))<2^{k(n)-1}$ ensures that more than half the words of length $k(n)$ are never queried by a machine P_j on input $0^{k(n)}$, and therefore adding to A or restraining from A sets of words as in the construction yields A fulfilling these conditions. On the other hand, the condition $\max\{p_i(k(n-1)) \mid i<n\}<k(n)$ ensures that modifications performed at stage n do not disturb the computations of machines P_j at previous stages, and therefore the computations of P_j, $j\in R_n$, on input $0^{k(n)}$ are the same with oracle A_n as with oracle A.

Notice that all of the infinitely many indices of machines for which $L(M, A)=\varnothing$ for all A are never deleted from R_n. This implies that the "else" case occurs infinitely often. Each time the "else" case occurs one more word enters $L(A)$, and so $L(A)$ is finite.

Suppose that there exists an infinite subset C of $L(A)$ such that for some j, $C=L(P_j, A)$. Since C is infinite and $C\subseteq L(A)$, there are infinitely many n such that $0^{k(n)}\in C$. The index j was put into R_j at stage j. Since there are only finitely many indices less than j, there is a stage $m\geq j$ such that $0^{k(m)}\in L(P_j, A_{m-1})$. However, at this stage A_m is defined so that $0^{k(m)}\notin L(A_m)$, and by the second condition on $k(n)$, $0^{k(m)}\notin L(A)$. Hence, $C\not\subset L(A)$, contradicting our assumption. Thus, $L(A)$ is $P(A)$-immune. $\diamond$

Since ZPP is included in all of the classes, we obtain the following corollary.

COROLLARY 2.2: *There is a recursive oracle A such that all of the following classes have a $P(A)$-immune set:*

$$R(A), \quad \text{co-}R(A), \quad NP(A)\cap \text{co-}NP(A),$$
$$NP(A), \quad \text{co-}NP(A), \quad BPP(A), \quad PP(A).$$

A similar construction of an oracle A exhibiting a $P(A)$-immune set in $R(A)$ appears in corollary 2.2 of [4].

THE "SIMPLICITY" TECHNIQUES

The constructions in [3] leading to simple sets for NP and other complexity classes are based also on the slow diagonalizations of [13]. It is possible to extend these results to some interesting ones relating NP and R.

Again we consider the strongest possible result. Clearly, the smallest class among the classes considered here in which an NP-immune set can lie is co-R.

THEOREM 3.1: *There is a recursive oracle B such that* co-$R(B)$ *has an $NP(B)$-immune set.*

Proof: The oracle B will be constructed in such a way that for each length n either no words of this length are in B or more than half the words are in B. This implies that the "test language"

$$L(B)=\{0^n \mid \forall x \text{ with } |x|=n,\ x\in B\}$$

is in co-$R(B)$.

Now, a slow diagonalization allows us to construct B so that for any infinite set L in $NP(B)$ some word in L is not in $L(B)$.

Stage 0: Let $B_0 = \Sigma^*$, $R_0 = \varnothing$ and $k(0) = 0$.

Stage n: Add a new index to be considered, $R_n = R_{n-1} \cup \{n\}$;
let $k(n)$ be a "sufficiently large" integer; more precisely,
$k(n) = \min \{m \mid \sum_{i \leq n} p_i(m) < 2^{m-1}$ and $\max \{p_i(k(n-1)) \mid i < n\} < m\}$;
initially define $B_n = B_{n-1} - \Sigma^{k(n)}$ so that $0^{k(n)}$ is not in the "test language";
if there is a $j \in R_n$ such that $0^{k(n)} \in L(NP_j, B_n)$
then
let $j(n)$ be the least such j;
fix an accepting computation of $NP_{j(n)}$ on $0^{k(n)}$;
spoil the machine with index $j(n)$ by defining
$B_n = B_n \cup \{$all words of length $k(n)$ not queried in the fixed computation$\}$;
let $R_n = R_{n-1} - \{j(n)\}$;
end if.

By the first condition, the set of words not queried includes at least half the words of length $k(n)$. Hence, the oracle B meets the conditions stated above and $L(B) \in$ co-$R(B)$. By the second condition on $k(n)$, the computations at earlier stages are preserved (i. e., machines spoiled in earlier stages remain spoiled).

Again no index of the empty set is ever deleted from R_n. Since there are infinitely many indices of the empty set, infinitely often the "else" case occurs and no word of length $k(n)$ remains in B. Hence $L(B)$ is infinite.

If a set in $NP(B)$ consists of infinitely many words of the form $0^{k(n)}$ then at some stage a word of this kind is found and put out of $L(B)$ by adding words to B. Thus, no infinite set in $NP(B)$ can be included in $L(B)$, and $L(B)$ is $NP(B)$-immune. ◇

The inclusion relations between the classes considered lead to the following

COROLLARY 3.2: *There is a recursive oracle B such that*

(i) *an $R(B)$-simple set exists;*

(ii) *an $NP(B)$-immune set exists in $BPP(B)$;*

(iii) *an $NP(B)$-immune set exists in $PP(B)$.*

THE "CRITICAL STRING CONTROL" TECHNIQUES

The previous sections use adaptations of known techniques of [13] and [3]. More care is taken in the number of words that enter or leave the oracle at each stage of the construction, but most ideas work correctly because the

"base" classes over which the constructions diagonalize are the same: P and NP, respectively.

However, when we are interested in performing a diagonalization over, say, R, the control on the number of queries which an "R" machine can make becomes more complicated. Preserving an accepting computation is easy: only polynomially many queries have to be controlled. However, preventing the creation of accepting computations is difficult for nondeterministic machines: potentially all of the strings have to be controlled.

Fortunately the restriction which R places on the class of machines over which we diagonalize allows us to control only the "important" queries. We will show that controlling only polynomially many words is enough to either preserve all the computations or destroy the particular character of the machines. The existence of this polynomial bound indicates a rather severe restriction on the access to the oracle by "R" machines.

DEFINITION 4.1: A word w is *R-critical* for the word x with respect to the machine M and the oracle A if and only if the following holds:

(i) either none of the computations of M with oracle A accept x, or at least half of these computations accept;

(ii) either none of the computations of M with oracle $A \Delta \{w\}$ accept x, or at least half of these computations accept;

(iii) $x \in L(M, A)$ if and only if $x \notin L(M, A \Delta \{w\})$.

Thus, w is critical for x if the fact of whether $x \in L(M, A)$ is entirely dependent on whether w is in A or not, and in either case the machine behaves correctly on x, i. e., it does not lose its "R" character.

A weaker concept of "critical" appears in Rackoff [10], although we defined ours independently. Rackoff's definition is harder to work with. In fact, our definition of "critical" allows us to recover and generalize an easier, but erroneous, proof that appeared in an earlier version of [10] (namely in [9]).

LEMMA 4.2: *Let M be a nondeterministic oracle Turing machine with a polynomial time bound p, and an oracle A. Then for all inputs x there are at most $2p(|x|)$ R-critical strings for x with respect to M and A.*

Proof: Any critical word w for x must be queried on at least half of the computation paths of M on input x, because either half the computations accept with oracle A and no computation accepts with oracle $A \Delta \{w\}$, or vice versa. Let c be the number of critical words. Consider a square matrix in which each row corresponds to the list of queries made during a computation of M on input x. At least $c\,2^{p(|x|)-1}$ entries exist in the whole matrix,

since each of the c critical queries must be queried in at least $2^{p(|x|)-1}$ computations. But each computation path of M queries A less than $p(|x|)$ times; the dimensions of the matrix are at most $p(|x|)$ by $2^{p(|x|)}$, and hence $c\,2^{p(|x|)-1} \leqq p(|x|)\,2^{p(|x|)}$. ◇

LEMMA 4.3: *For each nondeterministic oracle Turing machine M and oracle A, the function that gives for each x the encoding of the set of R-critical words for x (with respect to M and A) is computable in polynomial space with oracle A.*

Proof: By systematically searching through the words $w \in \Sigma^*$ up to length $p(|x|)$, simulating M using oracle $A \Delta \{w\}$ and counting the number of accepting computations, we can construct the set of critical words in polynomial space. ◇

Now we can use this concept of "critical" to diagonalize over R. Again, we consider the strongest possible result, in the context of the classes considered here.

THEOREM 4.4: *There is a recursive oracle C such that there is a $R(C)$-immune set in $NP(C) \cap$ co-$R(C)$.*

Proof: The set C will be constructed so that for each n either some word of length $2n$ and no word of length $2n+1$ is in C, or no word of length $2n$ and more than half the words of length $2n+1$ are in C. Under such a condition, the set

$$L(C) = \{0^n \mid \exists x, |x| = 2n, \text{ such that } x \in C\}$$

has as complement the set

$$\overline{\{0\}^*} \cup \{0^n \mid \exists x, |x| = 2n+1, \text{ such that } x \in C\}$$

and furthermore the complement of $L(C)$ belongs to $R(C)$.

We construct the oracle C so that C meets the condition above and $L(C)$ is $R(C)$-immune.

Stage 0: Let $C_0 = \{x \in \Sigma^* \mid |x| \text{ is odd}\}$, $R_0 = \varnothing$ and $m_0 = 0$.

Stage n: Add a new index to be considered, $R_n = R_{n-1} \cup \{n\}$;
let $k(n)$ be "sufficiently large", in particular,
$k(n) = \min\{m \mid \sum_{i \leqq n} p_i(m) < 2^{2m-1}$ and $\max\{p_i(k(n-1)) \mid i < n\} < m\}$;
initially define $C_n = C_{n-1} - \{x \mid |x| = 2k(n)+1\}$;
if there is an index $j \in R_n$ such that $0^{k(n)} \in L(NP_j, C_n)$
then
let j_n be the least such j;
fix an accepting computation of NP_j on $0^{k(n)}$;
spoil the machine with index j_n by defining

$C_n = C_n \cup \{$all words of length $2k(n)+1$ not queried in the fixed computation$\}$;
let $R_n = R_{n-1} - \{j_n\}$;
else
let w_n be the least word of length $2k(n)$ that is not critical for $0^{k(n)}$ with respect to any NP_j, $j \in R_n$, and C_n;
define C_n so that $0^{k(n)}$ is a member of the test language by setting $C_n = C_n \cup \{w_n\}$;
end if.

The fact that $L(C)$ is infinite follows from arguments similar to the ones used in sections 2 and 3.

In order to show that every machine accepting an infinite set in $R(C)$ is "spoiled", first note that a machine whose index leaves R_n at some stage is spoiled, whether or not it is an "R" machine for oracle C. Suppose that j does not leave R_n at any stage, i. e., $j \neq j_n$ for all n. Then there exists an n_0 such that $j_n > j$ for all $n > n_0$, and hence $0^{k(n)} \notin L(NP_j, C_{n-1})$ for all $n > n_0$.

We want to show that if $L(NP_j, C) \subseteq L(C)$, and NP_j is an "R" machine under oracle C, then $L(NP_j, C)$ is finite, i. e., $0^{k(n)} \notin L(NP_j, C)$ for all $n > n_0$. The construction guarantees that if $0^{k(n)} \notin L(NP_j, C_n)$ then $0^{k(n)} \notin L(NP_j, C)$. Hence it is sufficient to show that if $L(NP_j, C) \subseteq L(C)$ then $0^{k(n)} \notin L(NP_j, C_n)$ for all $n > n_0$.

Thus, consider any stage n with $n > n_0$ and assume $L(NP_j, C) \subseteq L(C)$, where NP_j is an "R" machine under C. Suppose first that the condition at the "if" statement is true. Since $0^{k(n)}$ is kept out of $L(C)$, it cannot be in $L(NP_j, C)$.

Now suppose that the condition at the "if" statement is false. The word chosen for w_n is not critical for $0^{k(n)}$ with respect to NP_j and $C_{n-1} - \{x \mid |x| = 2k(n)+1\}$. However, conditions (i) and (ii) of the definition of critical word hold, since the machine rejects $0^{k(n)}$ under $C_{n-1} - \{x \mid |x| = 2k(n)+1\}$ and is an "R" machine under C. So, condition (iii) must fail, and therefore the machine still rejects $0^{k(n)}$ under oracle C.

Thus, if NP_j is an "R" machine under oracle C and $L(NP_j, C) \subseteq L(C)$ then $L(NP_j, C)$ must be finite, and so $L(C)$ is $R(C)$-immune. ◇

The inclusion relations between the classes yield the following corollary.

COROLLARY 4.5: *There is a recursive C such that $ZPP(C) \subset R(C)$ and an $R(C)$-immune set exists in $BPP(C) \cap NP(C) \cap$ co-$NP(C)$.*

Observe that this corollary implies that $R(C) \subset BPP(C) \cap NP(C)$. In corollary 2.2 of [4], the relationship of R with the class U of sets accepted by unambiguous nodeterministic Turing machines [14] is discussed, and a construction similar to that of our theorem 4.4 is presented.

The polynomial bound on the number of critical strings is useful not only for separating, but also for collapsing classes. Constructions in [10] making $P=R \subset NP$ become easier with our definition. An interesting result is shown in the next theorem, strengthening those in [10] in two ways: first by collapsing all $BPP(D)$ to $P(D)$, and second by separating $P(D)$ and $NP(D)$ via a $P(D)$-immune set in $NP(D)$.

Some definitions are still needed. We begin by adapting the definition of critical to the *BPP* case, in an intuitively simple manner:

DEFINITION 4.6: A word w is *BPP-critical* for the word x with respect to the machine M and the oracle A if and only if the following holds:

(i) either less than 1/4 or more than 1/2 of the computations of M on input x under oracle A accept;

(ii) either less than 1/4 or more than 1/2 of the computations of M on input x under oracle $A \Delta \{w\}$ accept;

(iii) x is accepted by less than 1/4 of the computations of M under oracle A if and only if x is accepted by more than 1/2 of the computations of M under oracle $A \Delta \{w\}$.

This definition is the immediate analog of the definition of R-critical in the setting of *BPP*. The following lemmas have analogous proofs to the ones for R-critical words.

LEMMA 4.7: *Let M be a nondeterministic oracle Turing machine with a polynomial time bound p, and an oracle A. Then for all inputs x there are at most $4p(|x|)$ BPP-critical strings for x with respect to M and A.*

LEMMA 4.8: *For each nondeterministic polynomial time oracle Turing machine M and oracle A, the function that gives for each x the encoding of the set of BPP-critical words for x (with respect to M and A) is computable in polynomial space with oracle A.*

We now present the construction of oracle D. The facts about D are established after the construction. The oracle D will be a slight variation of an oracle E for which $P(E)=P\text{SPACE}(E)$. The variation consists of a set of words that are added so that the set $L(D)=\{0^n \mid \exists x, |x|=n, \text{ such that } x \in D\}$ diagonalizes out of $P(D)$ [in fact, it becomes $P(D)$-immune], but the diagonalization is so "tiny" that for any "*BPP*" machine we are able to compute the "important" (i. e., critical) added words and reduce it to operate under oracle E.

Throughout the construction the set R_n contains two types of indices. Odd indices, $j=2i+1$, remain in R_n unless it is possible to "spoil" the behavior

of NP_i, making it a non-*BPP* machine; even indices, $j=2i$, remain in R_n until we are able to ensure that $L(P_i, D) \not\subseteq L(D)$, as in the construction in Section 2. This construction will yield an oracle D so that $BPP(D)=P(D) \neq NP(D)$, the latter being witnessed by a $P(D)$-immune set in $NP(D)$.

Now let E be a PSPACE-complete set. Without loss of generality, we assume that no word of even length is in E. Inductively define $e(0)=2$, $e(n+1)=2^{2^{e(n)}}$. Construct D as follows.

Stage 0: Let $D_0=E$, $R_0=\varnothing$, and $r_0=0$.
Stage n: if $2^{e(n)} < \sum_{2j \in E_{n-1}} p_j(e(n))$
then let $D_n=D_{n-1}$, $R_n=R_{n-1}$, $r_n=r_{n-1}$, and go to next stage; *else*
add a new index to be considered, $R_n=R_{n-1} \cup \{r_{n-1}\}$;
let $r_n=r_{n-1}+1$;
for each j *in* R_n *loop*
if $j=2i+1$ *and* there is a word x such that
$e(n-1)<\log(|x|)<p_i(|x|)<e(n+1)$ on wich NP_i with
oracle D_{n-1} is not a "*BPP*" machine
then
let $D_n=D_{n-1}$ and $R_n=R_n-\{j\}$;
exit from loop and proceed with next stage;
comment NP_j is not *BPP* under D: no longer care required;
else
if $j=2i$ and $0^{e(n)} \in L(P_i, D_{n-1})$
then
let $D_n=D_{n-1}$ and $R_n=R_n-\{j\}$;
exit from loop and proceed to next stage;
comment diagonalization over P_i accomplished;
end if
end if;
end loop;
if no j was found satisfying the conditions above
then
let w_n be a word of length $e(n)$ such that no machine
P_i queried the string w_n on input $0^{e(n)}$;
let $D_n=D_{n-1} \cup \{w_n\}$;
comment make $L(D)$ infinite
end if
end if.

Some lemmas will lead to our final result.

LEMMA 4.9: *The function that gives for each x the BPP-critical words for x with respect to machine* NP_i, *and oracle E is computable in polynomial time relative to E.*

Proof: This function is computable in polynomial space relative to E, and $P(E)=P\text{SPACE}(E)$. Apply Fact 1.2. ◇

As no words of even length are in E, we assume without loss of generality that no such words are queried in the computation of the critical words. This

implies that the oracle D suffices for this computation, i. e. the function that gives for each x the *BPP*-critical words for x with respect to machine NP_i, and oracle E is computable in polynomial time relative to D.

Our next lemma shows that a "*BPP*" machine working under oracle D on a fixed input x can be transformed quickly into a "*BPP*" machine on the same input, working under oracle E. It formalizes and develops some ideas close to those in the proof of theorem 6 in [2].

LEMMA 4.10: *Let NP_i be a "BPP" machine under oracle D. Then there is a function f_i computable in polynomial time with oracle D such that for each x the machine with index $f_i(x)$ runs in time $O(|x|.p_i(|x|))$ and either*

(i) *more than half the computations accept x in both NP_i under D and $M_{f_i(x)}$ under E, or*

(ii) *less than 1/4 of the computations accept x in both NP_i under D and $M_{f_i(x)}$ under E.*

Proof: The function $f_i(x)$ computes, for each x, an index of a nondeterministic Turing machine which accepts (rejects) x using oracle E, if and only if M_i accepts (rejects) x using oracle D. This is accomplished by adding a finite "look-up" table to M_i and altering M_i so that it consults the look-up table prior to querying the oracle. In this way, $M_{f_i(x)}$ will only need the oracle E for its computations since the important portions of $D-E$ will be coded into the finite look-up table.

More precisely, let n_0 be such that NP_i is a "*BPP*" machine under D_n for $n>n_0$ (i. e., no odd index less than $2i+1$ is deleted from R_n at stage $n>n_0$). Consider the following algorithm.

```
function f_i
  input x;
  find n such that e(n-1)<log(|x|)<p_i(|x|)<e(n+1);
  if no such n exists or n<n_0 then
    check whether x is accepted by more than half the
    computations of NP_i under D by looking up in a finite table;
    output an index of a machine that always accepts or always
    rejects, accordingly;
  else
    compute D_{n-1}-E by querying D about all the words of length
    e(j), j<n;
    "patch" this portion of the oracle into a look-up table in NP_i
    and let i_0 be the resulting index;
    compute the BPP-critical words for x with respect to
    machine i_0 and oracle D_{n-1};
    query D about the BPP-critical words of length e(n);
    if no one of them is in D then
      output i_0
    else
      comment there is exactly one of them
      "patch" D_n-E=(D_{n-1}-E) ∪ {the BPP-critical word in D}
```

into a table in NP_i and let i_1 be the resulting index;
output i_1;
end if
end if.

By the construction of D and our choice of $n > n_0$, M_i is a "*BPP*" machine for input x under oracles D_{n-1} and D_n. Suppose d is a word contained in $D_n - D_{n-1}$. Notice that d is the only word of its length in D. If d is critical for x with respect to M_i under D_{n-1} then we will be able to compute all of $D_n - E$, and the patched machine with index i_1 will work correctly. If, on the other hand, d is not critical for x with respect to M_i under D_{n-1} then condition (iii) in the definition of *BPP*-critical must fail. Hence, either condition (i) or condition (ii) of the lemma is satisfied. Of course, if $D_n - D_{n-1}$ is empty then again we are able to compute all of $D_n - E$, and the machine with index i_0 will work properly.

The computation by "brute force" of $D_{n-1} - E$ is linear in $|x|$, since $e(n-1) < \log(|x|)$. By the previous lemma, it is possible to compute the *BPP*-critical words in polynomial time, and so f_i is computable in polynomial time. ◇

LEMMA 4.11: *The set*

$$K(E) = \{\langle x, i, 0^n \rangle \mid \textit{more than } 1/2 \textit{ of the computations of } NP_i \textit{ on } x \textit{ with oracle } E \textit{ accept within } n \textit{ steps}\}$$

is in $P(E)$.

Proof: Immediate since $K(E) \in P\,\mathrm{SPACE}(E) = P(E)$. ◇

Again, we may assume that a machine P_K computes $K(E)$ deterministically in polynomial time without querying words of even length. Thus, $L(P_K, D) = K(E)$. We are ready for our last result.

THEOREM 4.12: *There is a recursive oracle* D *for which* $BPP(D) = P(D)$ *and a* $P(D)$*-immune set exists in* $NP(D)$.

Proof: The construction of the set D has been presented before. The set $L(D) \in NP(D)$ is $P(D)$-immune, as can be easily shown following the same arguments as in our previous results.

We will show that $BPP(D) = P(D)$. Let NP_i be any "*BPP*" machine under D, and let f_i be the corresponding polynomial time computable function described in Lemma 4.10. Let $q_i(|x|)$ be the polynomial time bound of the machine with index $f_i(x)$ on input x. The following algorithm uses the oracle

D to decide in polynomial time whether more than half the computations of NP_i under oracle D accept x.

input x;
compute $f_i(x)$;
if $\langle x, f_i(x), 0^{q_i(|x|)} \rangle \in L(P_K, D)$ *then accept else reject*

The correctness of this algorithm follows from the previous lemmas. ◇

COROLLARY 4.13: *There is a recursive* D *such that* $R(D)=P(D)$ *and a* $P(D)$*-immune set exists in* $NP(D)$.

It should be noticed that a similar proof using R-critical words instead of BPP-critical words yields this corollary without collapsing all of $BPP(D)$ to $P(D)$.

Finally, from the fact that $R(D)=BPP(D)=P(D)\subset NP(D)$ we obtain:

COROLLARY 4.14: *There is a recursive* D *such that* $BPP(D)\neq NP(D)$ *but* $R(D)=BPP(D)\cap NP(D)$.

CONCLUSIONS

Let us mention some interesting remarks to the results shown in the previous sections. First, note that several conclusions of the kind "no proof solving such and such open problem relativizes" follow from our results. For instance, as stated in the introduction, no proof settling the question "is $R=NP\cap BPP$?" can relativize: for the oracle C the answer is no, but for the oracle D the answer is yes. Some other similar statements can be derived, although generally there is no need of showing a strong separation to derive them.

Secondly, notice that from the results proven here all possible strong separations can be derived, in the sense that for any two classes $C1$ and $C2$ chosen among P, ZPP, R, NP, BPP, and PP, either $C1$ is always contained in $C2$ in every relativization, or there is an oracle for which a strong separation holds. All the strong separations are withessed by a $C1$-immune set in $C2$, with the only exception that the strong separation of ZPP from R is witnessed by a ZPP-co-immune set.

We show this in the diagram of figure 2. The letters labeling each arrow indicate the oracles which exhibit a set in the target class which is immune with respect to the source class.

Finally, note the crucial role of the concept of critical string in the diagonalizations and collapses presented in theorems 4.4 and 4.12. The statement in

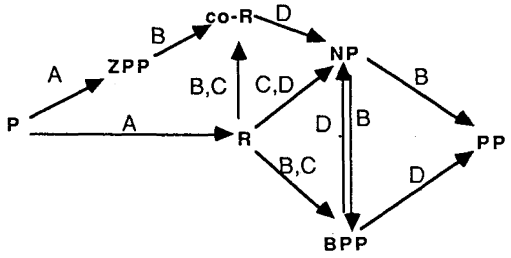

Figure 2

the introduction of section 4, that probabilistic bounded-error oracle machines have inherent restrictions in their use of the information provided by the oracle, has its formal counterpart in lemmas 4.2 and 4.7. We have shown that the polynomial bound on the number of critical strings allows us both to diagonalize against the probabilistic complexity classes and to decide them in relativized deterministic polynomial time. Possibly, this idea can be useful in the study of other complexity classes. Thus, whether the definition of critical (or a similar one) may play an interesting role in other contexts like, for instance, the class U of sets decided by unambiguous nondeterministic machines (see [14] and [4]), is a question that we consider worth to investigate.

REFERENCES

1. L. ADLEMAN and K. MANDERS, *Reducibility, Randomness, and Intractability*, Proc. 9th A.C.M. Symp. Theory of Computing, 1977, pp. 151-163.
2. T. BAKER, J. GILL and R. SOLOVAY, *Relativizations of the P=? NP question*, S.I.A.M. J. Computing, Vol. 4, 1975, pp. 431-442.
3. J. L. BALCÁZAR, *Simplicity, Relativizations and Nondeterminism*, S.I.A.M. J. Computing, Vol. 14, 1985, pp. 148-157.
4. J. GESKE and J. GROLLMAN, *Relativizations of Unambiguous and Random Polynomial Time Classes*, S.I.A.M. J. Computing, Vol. 15, 1986, pp. 511-519.
5. J. GILL, *Computational Complexity of Probabilistic Turing Machines*, S.I.A.M. J. Computing, Vol. 6, 1977; pp. 675-695.
6. Ph. FLAJOLET and J. M. STEYAERT, *Une généralization de la notion d'ensemble immune*, R.A.I.R.O., Vol. 8, R-1, 1974, pp. 37-48.
7. S. HOMER and W. MAASS, *Oracle Dependent Properties of the Lattice of NP Sets*, Theoret. Comput. Sci., Vol. 24, 1983, pp. 279-289.
8. J. E. HOPCROF and J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.
9. Ch. RACKOFF, *Relativized Questions Involving Probabilistic Algorithms*, Proc. 10th A.C.M. Symp. Theory of Computing, 1978, pp. 338-342.

10. Ch. RACKOFF, *Relativized Questions Involving Probabilistic Algorithms*, J. Assoc. Comput. Math., Vol. 29, 1982, pp. 261-268.
11. D. A. RUSSO and S. ZACHOS, *Positive Relativizations of Probabilistic Polynomial Time*, Submitted for publication.
12. U. SCHÖNING, *Complexity and Structure*, Lecture Notes in Computer Science, Vol. 211, Springer-Verlag, 1986.
13. U. SCHÖNING and R. BOOK, *Immunity, Relativizations, and Nondeterminism*, S.I.A.M. J. Computing, Vol. 13, 1984, pp. 329-337.
14. L. VALIANT, *Relative Complexity of Checking and Evaluating*, Inf. Proc. Letters Vol. 5, 1976, pp. 20-23.
15. S. ZACHOS, *Probabilistic Quantifiers, Adversaries, and Complexity Classes: an Overview*, Proc. Conference on Structure in Complexity Theory, Lecture Notes in Computer Science, Vol. 223, Springer-Verlag, 1986, pp. 383-400.