STRONG SEPARATION OF THE BOOLEAN HIERARCHY OVER RP

by

Danilo Bruschi
Deborah Joseph
Paul Young

Computer Sciences Technical Report #847

May 1989

STRONG SEPARATION FOR THE BOOLEAN HIERARCHY OVER RP*

DANILO BRUSCHI
Dipartimento Scienze dell'Informazione, Università degli Studi - Milano
Computer Sciences Department, University of Wisconsin - Madison

DEBORAH JOSEPH
Computer Sciences Department, University of Wisconsin - Madison

PAUL YOUNG
Computer Science Department, University of Washington
Computer Sciences Department, University of Wisconsin - Madison

---

Recently, boolean hierarchies over NP and over RP (denoted BH and RBH respectively) have been introduced in complexity theory. They have proved useful in classifying natural problems which are not easily classified using standard time and space complexity classes. In this paper we are particularly interested in the structural properties of these hierarchies and relationships between the various boolean hierarchies.

Establishing the most significant relationships between BH, RBH and other complexity classes would imply solving some of the major open problems in complexity theory. To date the only significant relations known are: $NP \subseteq BH \subseteq \Delta_2^p$, $RP \subseteq RBH \subseteq BPP \subseteq PP$, and $RBH \subseteq BH$. Essentially nothing is known about the fine structure of BH or RBH.

In [BJSY89] an oracle X is constructed for which both BH^X and RBH^X have an infinite number of proper levels. Further each level of RBH is properly contained in the corresponding level of BH, and RBH is properly contained in PP. In this paper we further explore these constructions. We prove that some of these separations are "strong" separations. That is, they can be witnessed by sets that cannot be "approximated" by sets in the smaller class: the separating sets are *immune* to sets from the smaller class.

Specifically, we prove that the separations between RBH, P, PP and each level of BH, can be witnessed by immune sets.

Contents

Categories and Subject Descriptors: F.1.3 [Theory of Computation]: complexity classes.
General Terms: Structural complexity theory.
Additional Key Words and Phrases: boolean hierarchy, NP, RP.

Date: May 12, 1989

---

* The work of the first author is supported in part by Ministero della Pubblica Istruzione, through "Progetto 40%: Algoritmi e Strutture di Calcolo." The work of the second author is supported by the National Science Foundation under grant DCR-8520597. The work of the third author is supported by a Brittingham Visiting Professorship in the Computer Sciences Department at the University of Wisconsin-Madison.

Authors' 1989 address: Computer Sciences Department, University of Wisconsin, 1210 West Dayton St., Madison, WI 53706.

1. Introduction

Understanding the computational complexity of natural problems is one of computer science's major research concern. For cases in which precise upper or lower bounds on the complexity of a problem seem difficult to obtain, it may be possible to obtain a good estimate of the problem's "difficulty" by showing it complete for some complexity class. Unfortunately this is not a universal method, in fact there exist problems (for example the "graph isomorphism" problem) that to date cannot be classified as complete for any complexity class, and there exist complexity classes that do not have complete problems.

The first classes introduced in complexity theory were quite coarse and were primarily based on restricted uses of deterministic and nondeterministic time and space. As a result for many interesting problems standard deterministic and nondeterministic classes are not adequate to provide a precise characterization of complexity. Looking for complexity classes that either better capture the computational complexity of these problems or for which these problems are complete has led to the introduction of many new complexity classes, for example various probabilistic complexity classes, circuit classes, difference classes, and bounded query classes. We will be concerned with some of these.

In 1982 Papadimitriou and Yannakakis [PaYa84], studying the complexity of languages associated with NP optimization problems, proved that the language:

$$\{< G, k > : \text{ the maximum clique in the graph G is of size } k\},$$

is complete for D^p, a complexity class that lies between NP and Δ_2^p.[1]

On the basis of this result, researchers began to investigate the possibility of defining additional complexity classes between NP and Δ_2^p. Recently [WeWa85, CaHe86, KöSchWa87, BJSY89] have shown that we can define over NP , or over RP , potentially infinite hierarchies which lie below Δ_2^p. These hierarchies, called boolean hierarchies (or difference hierarchies), are an extension to complexity classes of a notion introduced by Hausdorff in 1914 in the context of descriptive set theory and adapted to recursive enumerable sets by Eršov [Er68a,Er68b,Er69].

Boolean hierarchies are typically formed by taking at the base level a class of sets which is closed under union and intersection but not complement, and then forming the $k-th$ level of the hierarchy by iterating closure under k operations involving complement. In this paper we are particularly interested in structural properties and relations among the boolean hierarchy over NP (which in the following will be denoted by BH) introduced in [WeWa85, CaHe86, KöSchWa87] and in the boolean hierarchy over RP (which in the following will be denoted by RBH), introduced in [BJSY89].

Establishing the most significant relations among BH, RBH and other complexity classes would imply the solution of some of the major open problems in complexity theory. To date the only significant relations

[1] More precisely D^p is the class of all languages that can be expressed as the intersection of a language in NP and of a language in $coNP$, or equivalently that can be decided by an NP machine that can access an NP oracle exactly once.

known are: $NP \subseteq BH \subseteq \Delta_2^p$, $RP \subseteq RBH \subseteq BPP \subseteq PP$, and $RBH \subseteq BH$. Essentially nothing is known about the fine structure of BH or RBH. Given this situation, a reasonable first attempt to understand what structure and relations concerning BH and RBH might exist in the "real world" is to use oracle constructions. Although oracle constructions will not give definitive information, such constructions can be helpful for gaining a better understanding of the problem and its difficulty.

Some oracle results for BH and RBH are already known. For example, in [CaHe86], oracles A and B have been built such that BH^A is a proper infinite hierarchy, and BH^B extends exactly k levels. In the same paper relations between BH and Δ_2^p and between BH and the counting classes are settled in a relativized environment. In [BJSY89] an oracle X is built for which both BH^X and RBH^X have infinitely many proper levels; further each level of RBH is properly contained in the corresponding level of BH, and RBH is properly contained in PP. In this paper we continue to explore the latter constructions. In particular we extend the proofs given in [BJSY89] to prove that these separations can be witnessed by sets that cannot be "approximated" by sets in the smaller class; such separations are called "strong" separations. We define this notion more formally below.

Results on strong separations were introduced in complexity theory by Bennet and Gill [BeGi81]. In a certain sense, if relativization results give some evidence to support a conjecture, strong separations can be interpreted as further evidence for the conjecture. Strong separation results are usually introduced for the following reasons. Oracle constructions are generally obtained by diagonalization so that an oracle A and a language L_A are built such that, given two complexity classes C and D, $L_A \in C^A - D^A$. The fact that L_A is built by diagonalization ensures that L_A differs from every set in D^A at least infinitely often. This is not enough to ensure that infinite subsets of L_A are not in D^A. That is, although L_A is not in D^A it could happen that a significant and "nontrivial" portion of L_A is contained in D^A. To avoid this criticism, one can exhibit oracles that not only ensure the separation between two relativized complexity classes, but also ensure that the set which witnesses this difference does not contain infinite subsets belonging to the "smaller" class. Sets with this structure are well known in recursion theory and are generally referred to as *immune* sets. Formally, a *C-immune* set is an infinite set which contains no infinite subset belonging to C.[2]

Separation results witnessed by immune or simple sets are called "strong separations," and their proofs generally require a somewhat more sophisticated diagonalization technique called "slow diagonalization," or "wait and see " ([To 86]).

In this paper we prove that the entire boolean hierarchy over RP can be separated from P, from each level of BH, and from PP by immune sets. The constructions given here can be interspersed with the construction given in [BJSY89] to obtain infinite boolean hierarchies over NP and over RP.

---

[2] Another concept for oracle constructions borrowed from recursion theory, but which we do not need, is that of a *simple* set. A *C-simple* set is a set belonging to C whose complement is *C-immune*.

2. Basic Definitions

First we describe some of the notation used in this paper. We assume that the reader is familiar with definitions of standard complexity classes. For a detailed description of these we refer the reader to [Hopcroft and Ullman, 1979], or to [Balcázar et. al., 1986].

For all of our constructions we will use the usual two character alphabet Σ, with $\Sigma^{=n}$ denoting the set of all strings of length n.

Several definitions of the boolean hierarchy over NP have been given. Subsequently all have been shown to be equivalent. We use the following characterization: the i-th level of BH is denoted by $NP(i)$ and is defined by

$$NP(0) = P,$$
$$NP(i+1) = \{L_1 - L_2 : L_1 \in NP, L_2 \in NP(i)\}.$$

For RBH the definition is similar: the i-th level of RBH is denoted by $RP(i)$ and is defined by

$$RP(0) = P,$$
$$RP(i+1) = \{L_1 - L_2 : L_1 \in RP, L_2 \in RP(i)\}.$$

We fix enumerations $\{P_i\}_{i\in N}$, $\{NP_i\}_{i\in N}$, of polynomial time-bounded deterministic and non-deterministic oracle Turing machines respectively. We also fix an enumeration $\{BH_i\}_{i\in N}$ of polynomial time bounded machines for the boolean hierarchy over NP. For example a BH machine might be something like a $NP \wedge coNP$ machine. Among the NP_i machines we will distinguish machines as having a nondeterministic (NP) acceptance criterion or an (RP) acceptance criterion, i.e. RP machines are those whose computation trees have, for each input, either more than half accepting computations or else all rejecting computations. We will refer to nondeterministic oracle Turing machines with an RP acceptance criterion as "RP machines". We denote the corresponding oracle machines that use oracle X by P_i^X, NP_i^X and BH_i^X. We assume without loss of generality that the polynomial $p_i(n) = n^i + i$ bounds the running times of P_i^X, NP_i^X and BH_i^X. We denote the languages accepted by the machines P_i^X by $L(P_i^X)$, the languages accepted by the machines NP_i^X by $L(NP_i^X)$ and the languages accepted by the machines BH_i^X by $L(BH_i^X)$.

Following [Pa83], we use R^n to denote the "random" quantifier. Thus $R^n x P(x)$ means that more than half the strings, x, of length n satisfy the relation $P(x)$. We will also use the restricted universal quantifier $\forall^n$, where $\forall^n x P(x)$ means that all strings, x, of length n satisfy the relation $P(x)$, and the restricted existential quantifier $\exists^n$, where $\exists^n x P(x)$ means that there exists at least one string x, of length n which satisfies the relation $P(x)$.

3. Strong Separation Results for RBH

[CaHe86] answers some questions concerning the membership of certain types of immune sets at various levels of BH. Analogous questions can be answered, using the same arguments adopted in [CaHe86], with respect to RBH. In particular it can be shown that no sets in RBH can be RP bi-immune or $coRP$ bi-immune,[3] and that no sets in RBH can be $RP(2)$-immune. However the question of whether there exists some set in RBH which is RP-immune is instead unsolved.

Here we show a relativized world where it is possible to obtain a stronger result, in particular we will show the existence of an oracle X such that there is a language $L_X \in RP^X(2)$ which is $coNP^X(2)$-immune.[4] The method used to obtain this result is essentially the same used by Cai and Hemachandra to obtain the corresponding result for BH. The main difference being the fact that we have to construct an oracle X such that there is a language L_X which, respect to that considered in [CaHe86], has to satisfy the further requirement of being accepted by a probabilistic machine. This implies the construction of an oracle that has to satisfy more stringent constraints and so a bit more sophisticated oracle construction technique is used.

Theorem 1. *There exists an oracle X separating the entire boolean hierarchy over RP such that for all $k \geq 2$, $RP^X(k)$ contains a $coNP^X(2)$ immune set.*

Proof: It is sufficient to build an oracle X such that some set in $RP^X(2)$ is simultaneously RP^X-immune and $coRP^X$-immune. More precisely, we build an oracle X for which it is possible to construct a language $L_X \in NP^X(2) \cap RP^X(2)$, but such that $\overline{L_X}$ has a nonempty intersection with each infinite language in NP^X or in $coNP^X$. In this manner any $coNP^X(2)$ machine (and therefore any $coRP^X(2)$ machine) which accepts an infinite language accepts at least one string in the complement of L_X and so cannot accept some infinite subset of L_X.

Consider the language:

$$L_X = \{0^{2m} : (R^m x)[x \in X] \ AND \ (\forall^{m+1} x)[x \notin X]\}.$$

Clearly, $L_X \in NP^X(2)$. In order to ensure that L_X is in $RP^X(2)$ *our oracle X will be forced to obey the following simple language construction constraints*, for all sufficiently large m :

either no string of length m is in X OR more than half the strings of length m are in X.

We now show how to build, in successive stages, an oracle X satisfying the above constraints and such that for each index n' there is some Stage n which guarantees that some associated string $0^{h(n)}$ is in $\overline{L_X}$ *if and only if* it is accepted by the n'^{th} $coNP^X_{n'}(2)$ machine. To achieve this goal at each Stage n we keep

[3] We recall that a set S is C bi-immune if both S and $\overline{S}$ are C-immune.

[4] Since by definition $coRP \subseteq coNP$, and for each $k > 2$, $RP^X(2) \subseteq RP^X(k)$, our result easily implies that $\forall k \geq 2$, $RP^X(k)$ is $coRP^X(2)$-immune.

a finite list S of $k \leq n$ $coNP(2)$ machines and we verify that if some of the machines contained in the list accept the associated string $0^{h(n)}$, then an accepting computation of those machines is fixed and the oracle changed so that $0^{h(n)}$ is in $\overline{L_X}$. [5] If none of the machines contained in the list accept the string $0^{h(n)}$, then the oracle will be designed so that $0^{h(n)} \in L_X$. In this manner at the end of the construction of X, for all n', no infinite subset of L_X can be accepted by the $coNP^X_{n'}(2)$ machine, which implies that L_X is $coNP^X(2)$-immune.

We now describe Stage n of the construction if it is designed for "spoiling" the k machines contained in S. To this end, consider a $coRP(2)$ machine which accepts a language given by the union of the k $coRP(2)$ languages accepted by the machines contained in S. More formally, we consider the machines NP_{m_1} and NP_{m_2} such that:

$$L(\overline{NP}_{m_1} \ OR \ NP_{m_2}) = \bigcup_{1 \leq t \leq k} L(coNP_{i_t}(2)).$$

Stage n for "spoiling" the machine $\overline{NP_{m_1}}$ OR NP_{m_2} :

a) Choose $h(n)$ large enough that all strings already put into X or marked for holding out of X at earlier stages of the construction have length less than $h(n)$. Also choose $h(n)$ large enough that, in each path of the computation $0^{h(n)}$ the nondeterministic machines contained in the stack query at most 1/4 of the strings of length $h(n)$. (Thus $h(n)$ is chosen large enough to avoid any conflicts with earlier assignments of strings to X or to $\overline{X}$.) For all m such that $h(n-1)+1 < m < h(n)$, using whatever criteria are appropriate with respect to the various language constraints, either place all strings of length m which are not marked for holding out of X into X or else place no strings of length m into X.

We now take X to be the set of strings placed into X during earlier stages and during Step (a) of Stage n, and then we try to extend X to a larger oracle by trying various extensions, $X \cup W(n)$, of X :

b) If there exists a set of strings $W(n)$ which does not contain strings previously marked for holding out of X and such that $0^{h(n)} \in NP^{X \cup W(n)}_{m_2}$,

then

c) fix and freeze an accepting computation path of $NP^{X \cup W(n)}_{m_1}(0^{h(n)})$. Call all queries made on this path which receive *negative* replies *critical*. Mark for holding out of X all critical strings and put into X all strings of length $h(n)$ and $h(n)+1$ which are not critical. (Obviously, this puts more than three quarters of the strings of length $h(n)$ and $h(n)+1$ into X. This both satisfies our language construction constraints for L_X and forces $0^{h(n)}$ into $\overline{L_X}$.)

---

[5] What generally happens in these cases is that we can satisfy the requirement relative to i-th machine in a Stage $n' \geq i$, so that we have to wait for an opportune stage to diagonalize against a given machine, hence the name "wait and see" or "slow diagonalization."

d) Delete from the list S all the machines which accept the string $0^{h(n)}$.

else

e) If there exists a set of strings $W(n)$ which does not contain strings previously marked for holding out of X and strings of length $h(n)+1$, such that $0^{h(n)} \in NP_{m_1}^{X \cup W(n)}$,

then

f) Fix and freeze an accepting computation path of $NP_{m_1}^{X \cup W(n)}(0^{h(n)})$. Call all queries made on this path which receive *negative* replies *critical*. Mark for holding out of X all critical strings and put into X all strings of length $h(n)$ which are not critical. (Obviously, this puts no strings of length $h(n)+1$ into X and puts more than three quarters of the strings of length $h(n)$ into X. This satisfies our language construction constraints for L_X and forces $0^{h(n)}$ into L_X.)

g) Insert the next $coNP(2)$ machine into the list S.

else

h) Mark for holding out of X all critical strings of length $h(n)+1$ and delete from S all the machines which accept the string $0^{h(n)}$. (In fact at this point we know that if we do not place into X strings of length $h(n)+1$, then $0^{h(n)}$ is accepted by some of the machines on the list, and also satisfies our language construction constraints for L_X and forces $0^{h(n)}$ into $\overline{L_X}$.

End of Stage n.

To complete the proof we must show that L_X is infinite and that each infinite set in $coNP^X(2)$ contain at least one string of $\overline{L_X}$

The fact that L_X is infinite follows from the observation that at any moment the list S contains only a finite number of machines so that Step (d) or Step (h) can be executed consecutively only a finite number of times, thus Step (f) and Step (g) are executed infinitely often. Hence L_X cannot be finite.

We now show that each infinite set in $coNP^X(2)$ has a nonempty intersection with $\overline{L_X}$. Note that we only need to worry about machines which accept languages in $\{0\}^*$. If $coNP_i^X(2)$ accepts infinitely many strings of the form $0^{h(n)}$, then there are must exist a stage j during which we diagonalize against $coNP_i^{X_j}(2)$, (where X_j denotes the content of the oracle at Stage j). On the other hand if there exists a stage k such that for each stage $j \geq k$ we are not able to diagonalize against $coNP_i^{X_j}(2)$, then we can conclude that $|L(coNP_i^X(2)) \cap L_X|$ is finite. ∎

4. Strong Separation between RBH and P

It is unclear the extent to which bounded probabilistic complexity classes like BPP and RP differ from P. On the one hand, on intuitive grounds it seems strongly believable that $P \neq RP$ and $P \neq BPP$, but on the other hand, there is the surprising result obtained in [BeGi81] that, with respect to a random oracle X, $P^X = BPP^X$ with probability one.

In this section we produce evidence that perhaps the bounded probabilistic classes should differ from P by showing that there are relativized worlds in which ZPP^X (i.e. $RP^X \cap coRP^X$) contains a P^X-immune set. This result immediately implies that each level of the boolean hierarchy over RP^X can be strongly separated from P^X.

Theorem 2. *There exists an oracle X such that some infinite set in ZPP^X is P^X-immune.*

Proof: We show how to construct an oracle X and an infinite language L_X in $ZPP^X - P^X$ that does not contain infinite subsets that can be recognized in polynomial time. The construction is slightly different with respect to the previous one because of different constraints that the oracle X has to respect to ensure that L_X is contained in $ZPP^X - P^X$. In particular the technique that we use is an adaptation of slow diagonalization to the technique used in [BaGiSo75] to separate P^X from $NP^X \cap coNP^X$.

Consider the language:

$$L_X = \{0^m : (R^m x)[x \in X]\}.$$

Clearly, $L_X \in RP^X$. In order to ensure that L_X is in $RP^X \cap coRP^X$, *our oracle X will be forced to obey the following simple language construction constraints*, for all sufficiently large m:

(more than half the strings of length m are in X) AND (no string of length $m+1$ is in X)

OR

(no string of length m is in X) AND (more than half the strings of length $m+1$ are in X).

We now show how to construct an oracle X which respects the above constraint and such that L_X is infinite and contains no infinite subsets in P^X. The strategy used to build this oracle is essentially the same as that used in the preceding section.

Stage n :

a) This step is exactly like the Step (a) of the construction used to prove Theorem 1, except that now the list S contains deterministic machines.

b) Verify whether among the machines contained in the stack there is some machine which accepts $0^{h(n)}$, using as oracle the strings put into X at earlier stages. Call the strings that each machine in S queries in its computation on $0^{h(n)}$ *critical strings*.

c) If there is some machine which accepts,

then

d) Mark for holding out of X all critical strings, and place into X all strings of length $h(n)+1$ which are not critical strings. (Obviously, this places at least three quarters of the strings of length $h(n)+1$ into X and forces $\overline{L_X} \cap L(P_i^X) \neq \emptyset$.)

e) Delete from the list S all of the machines which accept the string $0^{h(n)}$.

else

f) (Comment: $0^{h(n)}$ is rejected by all the machines in the stack.) Place into X all strings of length $h(n)$ which are not critical strings. (Obviously, this places at least three quarters of the strings of length $h(n)$ into X and forces $0^{h(n)} \in L_X$.)

g) Insert the next P machine into the list S.

End of Stage n.

The same arguments as those used in the proof of Theorem 1 can be used here to show that L_X is infinite and that no infinite subset of L_X can be recognized by a P machine which accepts an infinite language. Thus, we conclude that L_X is P -immune. ∎

5. Strong Separations between RBH and BH

Probabilistic computations with bounded error have always been considered a good alternative to nondeterministic computations. Unfortunately many languages contained in nondeterministic classes seem not to be recognizable using probabilistic algorithms with bounded error. In this section we support this evidence, showing the existence of a relativized world in which $NP^X \cap coNP^X$ contains a language which is RBH -immune.

The proofs used in this and in the next section diagonalize against RBH machines so that we face the problem that the subtree of accepting (or rejecting) computations of such machines may contain exponentially many queries. Thus the techniques used in the preceding proofs, based on the fact that the acceptance of a word by a machine depends on a polynomial number of queries, must be readdressed. To overcome this difficulty we will use the notion of a critical set with respect to an RBH machine M, an oracle X and an input x, introduced in [Ra82] as generalized in [BJSY89]. [6]

Informally, a set of strings W is critical with respect to an RP machine M an oracle X and an input x, if when W is added to X or subtracted from X, it modifies the input/output behavior of the machine $M^X(x)$, while maintaining M an RP machine. By this we mean that $x \in L(M^X)$ *if and only if* $x \notin L(M^{X \triangle W})$ while $M^{X \triangle W}$ remains an RP machine. (Here $\triangle$ denotes the symmetric difference of the sets X and W; i.e $X \triangle W = (X - W) \cup (W - X)$.)

A *non-critical set* relative to an RP machine M, an oracle X and an input x, is a set W such that either $x \in L(M^X)$ *if and only if* $x \in L(M^{X \triangle W})$ while $M^{X \triangle W}$ remains an RP machine or else $M^{X \triangle W}$ is no longer an RP machine.

The fundamental property of critical sets for RP machines is stated in the following lemma proved in [BJSY89], which generalizes a lemma proved in [Ra82].

[6] More specifically in [Ra82] a method is introduced to diagonalize against single RP machines using the notion of critical string. In [BJSY89] the problem is instead to diagonalize against a k-tuple of RP machines; for this case the notion of critical string is too weak, and the more general notion of critical set has been introduced.

Lemma. *Let M be an $RP(k)$ machine with input x, oracle X and running time bounded by a polynomial $p(n)$. For each t-tuple of nonempty disjoint sets $\{X_0, \cdots, X_t\}$ with $t \geq (2p(|x|)+1)^k$ there exists at least one $X_i \in \{X_0, \cdots, X_t\}$ which is a non-critical set with respect to $M^X(x)$, i.e. such that:*

- $M^{X \triangle X_i}(x) = M^X(x)$ *or*
- $M^{X \triangle X_i}(x)$ *is not a RP (k) machine.*

Theorem 3. *There exists an oracle X such that some infinite set in $NP^X \cap coNP^X$ is RBH -immune.*

Proof: We show how to construct an oracle X and an infinite language L_X in $(NP^X \cap coNP^X) - RBH^X$ that does not contain infinite subsets that can be recognized by an RBH machine.

Consider the language:

$$L_X = \{0^m : (\exists^m x)[x \in X]\}.$$

Clearly, $L_X \in NP^X$. In order to ensure that L_X is in $NP^X \cap coNP^X$, *our oracle X will be forced to obey the following simple language construction constraint*, for all sufficiently large m:

(at least one string of length m is in X) AND (no string of length $m+1$ is in to X)

OR

(no string of length m is in X) AND (at least one of the strings of length $m+1$ is in X).

We now show how to construct an oracle X which respects the above constraint and such that L_X is infinite and contains no infinite subsets in RBH^X. The strategy used to build this oracle is again slow diagonalization.

Stage n:

a) First choose $h(n)$ large enough that all strings already put into X or marked for holding out of X at earlier stages of the construction have length less than $h(n)$. Also, choose $h(n)$ large enough that the number of critical *strings* with respect to all the machines in S, the oracle X at previous stage and input $0^{h(n)}$, is smaller than the number of strings of length $h(n)$. Also, choose another value $h'(n)$ large enough that, for all X', all machines in S query only strings of length less than $h'(n)$. For all m, $h(n)+1 < m < h'(n)$, using whatever criteria are appropriate with respect to the various language constraints, either place one string of length m into X or place no strings of length m into X.

b) Using for oracle X the strings put into X at earlier stages and during Step (a) of Stage n, verify whether there exists some machine in the stack which accepts the input $0^{h(n)}$ with a random acceptance criterion.

c) If there are machines of this type,

then

d) Choose the one with the smallest index i, remove it from the stack. Put into X a non-critical string of length $h(n)+1$. Go to the next stage. (Since there are an exponential number of strings of length

$h(n)+1$ and the lemma states the existence of a non-critical string for each polynomial number of strings of a given length, we are sure of the existence of a non-critical string, so we can execute this operation. Obviously, none of the strings of length $h(n)$ is put into X and this forces $0^{h(n)} \notin L_X$.)

else

e) (Comment: $0^{h(n)} \notin L(BH_{n'}^X)$.) Place into X a non-critical string of length $h(n)$. (Obviously this forces one of the strings of length $h(n)$ into X and forces $0^{h(n)} \in L_X$.)

End of Stage n.

It is easy to verify that we have forced the oracle X to obey the language constraint, and that L_X is infinite and has no infinite subsets accepted by a BH machine whose acceptance criterion is a random one.

∎

6. Strong Separations between RBH and PP

In this section we prove the existence of a world were bounded error probabilistic computations are strongly separated from unbounded error ones. More formally:

Theorem 4. *There exists an oracle X such that some infinite set in PP^X is RBH -immune.*

Proof: Since the proof is virtually the same as that of Theorem 3, we will only outline the proof. We force the oracle X *to respect the following language construction constraint:*

$$|X \cap \{x : |x| = m\}| \geq (2^{m-1}) + 1$$

OR

$$|X \cap \{x : |x| = m\}| \leq (2^{m-1}) - 1.$$

For such an oracle X, consider the language:

$$L_X = \{0^m : (R^m x)[x \in X]\}.$$

We easily see that if X respects the above constraint, then $L_X \in PP^X$.

The main difference between the constraints used in this construction and those used in the previous construction is the number of strings of length m which must be forced into the oracle. In this case we must put into the oracle at most $half - 1$, or at least $half + 1$ of the total number of strings of length m. To satisfy this requirement, at the beginning of each Stage n of the oracle construction, adopting the usual strategy described in Step (a) of the preceding constructions, we choose an appropriate value $h(n)$ [7] and

[7] In particular $h(n)$ should be chosen large enough that besides satisfying the usual requirements, the number of critical *pairs* in the computation for each of the machines in S on input $0^{h(n)}$ is less than the number of disjoint *pairs* of strings of length $h(n)$, i.e. $2^{h(n)-1}$.

place $2^{h(n)-1}-1$ strings of length $h(n)$ into X. Then we proceed to consider the machines in the list S, on input $0^{h(n)}$. If one of these machines accepts, we do nothing, so that $0^{h(n)} \in \overline{L_X}$. On the other hand if all machines contained in S reject, then the lemma assures us that, if $h(n)$ is large enough then there exists at least one *pair* of non-critical strings that we can put into the oracle. Obviously, this forces half + 1 of the strings of length $h(n)$ into X and forces $0^{h(n)} \in L_X$.

Clearly in this manner we force the oracle X to obey the constraint which keeps $L_X \in PP^X$, and for n' we have forced $\overline{L_X} \cap L(RBH^X_{n'}) \neq \emptyset$. Thus, if for each index n', some Stage $n_{n'}$ is forced to be the stage for "spoiling" the n'^{th} $BH^X_{n'}(k)$ machine, then we will have forced each RBH machine to accept at least one string of $\overline{L_X}$. Since it can be easily verified that L_X is also infinite, we have built a set in PP^X which is also RBH^X-immune. ∎

7. Bibliography

[BaGiSo 75] T. Baker, J. Gill and R. Solovay, "Relativizations of the $P = NP$ question," *SIAM J Computing*, v 4 (1975), pp 431-442.

[BeGi 81] C. Bennet and J. Gill, "Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1," *SIAM J Computing*, v 10 (1981), pp 96-113.

[BJSY 89] Danilo Bruschi, Deborah Joseph, Meera Sitharam and Paul Young, "Generalized boolean hierarchies and the boolean hierarchy over RP," *Tec. Rep. # 809, Dep. Computer Science, Univ. Wisc.*, (1989).

[CaHe 86] Jin-yi Cai and Lane Hemachandra, "The boolean hierarchy: hardware over NP", *Structure in Complexity Conference*, (1986), pp 105-124.

[Er 68a] Yu Ershov, "A hierarchy of sets, I," *Algebra and Logic*, v7 n1 (1968), pp 25-43.

[Er 68b] Yu Ershov, "A hierarchy of sets, II," *Algebra and Logic*, v7 n4 (1968), pp 15-47.

[Er 69] Yu Ershov, "A hierarchy of sets, III," *Algebra and Logic*, v9 (1969), pp 20-31.

[KöSchWa 87] J. Köbler, U. Schöning and K. Wagner, "The difference and truth-table hierarchies for NP," *RAIRO*, v21 (1987), pp 419-435.

[Pa 83] C. Papadimitriou, "Games against nature," *Proceedings of the 24th IEEE Foundations of Computer Science Conference*, (1983), pp 446-450.

[PaYa 84] C. Papadimitriou and M. Yannakakis "The complexity of facets (and some facets of complexity)," *J Computer and Systems Science*, v28 (1984), pp 224-259.

[To 86] L. Torenvliet, "Structural concepts in relativised hierarchies," *Doctoral Dissertation Univ. Amsterdam*, (1986).

[Ra 82] C. Rackoff, "Relativized questions involving probabilistic algorithms," *J ACM*, v29 (1982), pp 261-268

[WeWa 85] G. Wechsung and K. Wagner, "On the boolean closure of *NP* ," *Proceedings of the Conference of the Fundamentals of Computation Theory*, Lecture Notes in Computer Science 199 (1985), pp 485-493