On the Power of Deterministic Reductions to $C_=P$

Frederic Green [1]
Department of Mathematics and Computer Science
Clark University
Worcester, Massachusetts 01610
fgreen@clarku.bitnet

September 15, 1991

[1]Research supported by a grant from the Dirección General de Investigación Científica y Técnica (DGICYT), Spanish Ministry of Education, while the author was visiting the Facultat d'Informàtica, Universitat Politècnica de Catalunya, Barcelona.

Abstract

The counting class $\mathrm{C}_{=}\mathrm{P}$, which captures the notion of "exact counting", while extremely powerful under various nondeterministic reductions, is quite weak under polynomial-time deterministic reductions. We discuss the analogies between NP and co-$\mathrm{C}_{=}\mathrm{P}$, which allow us to derive many interesting results for such deterministic reductions to co-$\mathrm{C}_{=}\mathrm{P}$. We exploit these results to obtain some interesting oracle separations. Most importantly, we show that there exists an oracle A such that $\oplus\mathrm{P}^A \not\subseteq \mathrm{P}^{\mathrm{C}_{=}\mathrm{P}^A}$ and $\mathrm{BPP}^A \not\subseteq \mathrm{P}^{\mathrm{C}_{=}\mathrm{P}^A}$. Therefore, techniques that would prove that $\mathrm{C}_{=}\mathrm{P}$ and PP are polynomial-time Turing equivalent, or that $\mathrm{C}_{=}\mathrm{P}$ is polynomial-time Turing hard for the polynomial-time hierarchy, would not relativize.

1 Introduction

The class $C_=P$ (see section 2 for precise definitions) is an extremely powerful counting class which captures the notion of "exact counting". It can be characterized by nondeterministic machines which accept if and only if the number of accepting paths is exactly equal to a given number. The power of $C_=P$ can be seen from the following facts:

Facts:
(i) $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{NP}^{\mathrm{C_=P}}$.
(ii) $\mathrm{C_=P}^{\mathrm{PH}} \subseteq \mathrm{BP} \cdot \mathrm{C_=P}$.

Fact (i) is a consequence of Toda's theorem [18] that $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}}$ combined with a theorem of Torán [22] which states $\mathrm{NP}^{\mathrm{PP}} = \mathrm{NP}^{\mathrm{C_=P}}$. It is significant since it states that $C_=P$ is hard for the polynomial hierarchy under nondeterministic reductions. Fact (ii) was proved by Toda and Ogiwara [20], and in a stronger form by Tarui [16]. It is significant since it says that $C_=P$ is hard for the polynomial hierarchy under randomized reductions.

The purpose of this paper is to observe that although $C_=P$ is quite powerful under nondeterministic or randomized reductions, nevertheless under deterministic reductions it appears to be quite weak. Specifically, we address primarily the following questions:

(1) How powerful are polynomial-time Turing reductions to $C_=P$?
(2) How powerful is a constant number of queries to $C_=P$?

In attempting to answer these questions, we find that in many respects the classes NP and *co*-$C_=P$ (or alternatively $C_{\neq}P$) are similar. In particular, NP has a complete problem, and is closed under union, intersection, polynomial-time disjunctive and conjunctive reductions, and polynomial-time nondeterministic many-one reductions. The class $C_{\neq}P$ has all of these properties. It turns out that many proofs about restricted Turing reductions to NP, the Boolean hierarchy over NP, and related notions, depend only on these properties. Thus many interesting results about NP can be easily translated into analogous results about $C_{\neq}P$. For example, we find that the closure of $C_=P$ under polynomial time truth-table reductions is exactly as powerful as the closure under polynomial-time Turing reductions with logarithmically many queries (i.e., $\mathrm{P}_{tt}^{\mathrm{C_=P}} = \mathrm{P}_{O(log(n))-\mathrm{T}}^{\mathrm{C_=P}}$; see theorem 5 and the remarks preceeding it). However, our main interest is in exploiting these results in order to answer the above questions.

Some insight into question 2 is gained immediately by examining the analogies between NP and $C_=P$. Using these analogies we find here that the query and Boolean hierarchies are just as closely intertwined for $C_=P$ as they are for NP [4]. We also find that the proof of Kadin [13] and Chang and Kadin [11] which shows that if the Boolean hierarchy over NP collapses then the polynomial hierarchy collapses, also works for the Boolean hierarchy over $C_=P$. Thus if the Boolean hierarchy over $C_=P$ collapses to some finite level, we find that the polynomial hierarchy *relative to PP* collapses to the Δ_2 level *relative to PP*. This links a collapse of the query hierarchy over $\mathrm{C_=P}$ to a collapse of the polynomial hierarchy relative to PP. Beigel, Chang and Ogiwara [5] have independently proved a stronger version of this theorem, using techniques somewhat different than those of Chang and Kadin. Note that this is in sharp contrast to the query hierarchy over PP, which collapses to PP [6]. These results are discussed in section 3.

Nothing is known about question 1, not even oracle separations, and up until now there has been no reason (other than intuition) to believe that $\mathrm{P}^{\mathrm{C_=P}}$ is any less powerful than P^{PP}. The intuition is quite strong, however. With queries to PP, there is a well-known binary search algorithm which enables us to count the number of accepting paths in nondeterministic computations. A PP oracle provides us pairs of the form $\langle x, y\rangle$ such that $f(x) \geq y$ and $f \in \#\mathrm{P}$, from which it is possible to determine the value of $f(x)$ by binary search on y. On the other hand, a $\mathrm{C_=P}$ oracle only provides information about the graph of a #P function, that is, pairs of the form $\langle x, y\rangle$ such that $f(x) = y$. It seems obvious that it would be impossible to determine the value of f from this information (in polynomial time) in the absence of nondeterminism. Thus for example, although it is well known that $\oplus\mathrm{P} \subseteq \mathrm{P}^{\mathrm{PP}}$, it is not at all clear that $\oplus\mathrm{P} \subseteq \mathrm{P}^{\mathrm{C_=P}}$. Indeed, it does not seem likely that any of the classes $\oplus$P, PP, or PH are in $\mathrm{P}^{\mathrm{C_=P}}$.

In this paper, we obtain oracle separations of $\mathrm{P}^{\mathrm{C_=P}}$ and P^{PP}. Specifically, we construct an oracle A relative to which $\oplus$P and BPP are not contained in $\mathrm{P}^{\mathrm{C_=P}}$ (see section 5). A direct consequence of the separation of BPP from $\mathrm{P}^{\mathrm{C_=P}}$ are separations of both PP and $\Sigma_2^p \cap \Pi_2^p$ from $\mathrm{P}^{\mathrm{C_=P}}$. The constructions are based on circuit lower bounds, building on a result of Gundermann, Nasser and Wechsung [12], as well as a new characterization of $\mathrm{P}^{\mathrm{C_=P}}$ (easily proved using the analogies between $\mathrm{C_{\neq}P}$ and NP). The circuit lower bounds are for depth-2 circuits consisting of a single "equals" gate over AND-gates (called "EQ circuits"), and are actually lower bounds on the fanin of the AND-gates rather than lower bounds on the size of the circuits.

We finally turn to oracle results relating to question 2. Gundermann, Nasser and Wechsung [12] obtained an oracle separation of the Boolean hierarchy over $\mathrm{C_=P}$. Since the Boolean hierarchy is intertwined with the query hierarchy, in some relativized world, $k+1$ queries to $\mathrm{C_=P}$ are more powerful than k. Here we consider the question of whether $k+1$ questions to *NP* cannot be answered by k questions to $\mathrm{C_=P}$. In fact we find that in some relativized world, there are sets that are recognizable with $k+1$ queries to NP that cannot be recognized with k queries to $\mathrm{C_=P}$. One may regard this as a generalization of the result of Torán [21] that relative to some oracle, NP is not contained in $\mathrm{C_=P}$. In order to do this we again adapt the technique of [12] (section 4) to appropriate circuit problems. In section 6 an oracle is constructed which separates every level of the Boolean hierarchy over NP from a level of the Boolean hierarchy over $\mathrm{C_=P}$. Clearly this simultaneously separates the query hierarchies over NP and $\mathrm{C_=P}$. The resulting construction for the Boolean hierarchy over NP is simpler than existing ones [9] although it apparently is not powerful enough to obtain random oracle results (see [8]).

2 Preliminaries

We assume the reader is familiar with complexity classes such as P, NP, Σ_k^p and PH (see, e.g., [3]). Let N be a polynomial time-bounded nondeterministic Turing machine. Then $\#acc_N(x)$ denotes the number of accepting paths of N on input x. #P is the class of functions f such that there exists a polynomial time-bounded nondeterministic machine N such that for all x, $f(x) = \#acc_N(x)$.

The class $\mathrm{C_=P}$ [23] is defined to be the set of languages L such that there exist functions $f \in \#\mathrm{P}$, $t \in \mathrm{FP}$ and for all x, $x \in L$ if and only if $f(x) = t(x)$. The notation co-$\mathrm{C_=P}$ denotes the class of sets whose complements are in $\mathrm{C_=P}$. We will denote co-$\mathrm{C_=P}$ alternatively by $\mathrm{C_{\neq}P}$. PP is similarly defined, but with "$f(x) = t(x)$" replaced by "$f(x) \geq t(x)$".

It was recently remarked in [12] that in the definitions above one can replace the function $t \in \mathrm{FP}$ by a function $g \in \#\mathrm{P}$, and still obtain the same classes. We can furthermore assume without loss of generality that, in the resulting alternative definition of $\mathrm{C_=P}$, $f(x) \geq g(x)$ for all x.

$\oplus\mathrm{P}$ is defined as the set of languages L such that there exists a function $f \in \#\mathrm{P}$ and for all x, $x \in L$ if and only if $f(x)$ is odd.

We denote by ESAT the standard complete language [23] for $\mathrm{C_=P}$: ESAT $= \{\langle \mathcal{F}, n\rangle | \mathcal{F}$ is a Boolean formula with exactly n satisfying truth assignments $\}$. Obviously $\overline{\mathrm{ESAT}}$ is complete for $\mathrm{C_{\neq}P}$.

We now present other definitions that will be important in later sections. Let A and B be sets. We say $A \leq_m^{\mathrm{NP}} B$ if and only if there exist a polynomial p and a function $f \in \mathrm{FP}$ such that for any x, $x \in A$ if and only if $(\exists z, 1 \leq |z| \leq p(|x|))(f(\langle x, z\rangle) \in B)$. For any complexity class $\mathcal{C}$, the notation $\exists\mathcal{C}$ denotes the class of all sets L such that there exists a polynomial p and a set $B \in \mathcal{C}$ such that for all x, $x \in L \Leftrightarrow (\exists z, |z| \leq p(|x|))(\langle x, z\rangle \in B)$. A set A is *conjunctive truth-table reducible* to a set B, (symbolically $A \leq_{c-tt}^p B$) if there exists a function $f \in FP$ such that for any x, $f(x) = \langle q_1, ...q_k\rangle$, and $x \in A$ iff $(\forall i, 1 \leq i \leq k)(q_i \in B)$. For any complexity class $\mathcal{C}$, $\mathrm{P}^{\mathcal{C}}_{k-\mathrm{T}}$ denotes the class of sets polynomial time Turing reducible to $\mathcal{C}$ with no more than k queries, $\mathrm{P}^{\mathcal{C}}_{tt}$ the class of sets polynomial-time truth-table reducible to $\mathcal{C}$, and $\mathrm{P}^{\mathcal{C}}_{O(f(n))-\mathrm{T}}$ the class of sets Turing reducible to $\mathcal{C}$ with $O(f(n))$ queries. The *query hierarchy* over $\mathcal{C}$ is defined as $\bigcup_{k=1}^{\infty} \mathrm{P}^{\mathcal{C}}_{k-\mathrm{T}}$.

Let $\mathcal{C}$ be a complexity class. Following [9], we define the *Boolean hierarchy over* $\mathcal{C}$ inductively as follows: Let $\mathrm{BH}_1(\mathcal{C}) = \mathcal{C}$, and, for all $k > 1$,

$$\mathrm{BH}_{2k}(\mathcal{C}) = \{L | L = L_1 \cap \overline{L}_2, L_1 \in \mathrm{BH}_{2k-1}(\mathcal{C}), L_2 \in \mathcal{C}\},$$

$$\mathrm{BH}_{2k+1}(\mathcal{C}) = \{L | L = L_1 \cup L_2, L_1 \in \mathrm{BH}_{2k}(\mathcal{C}), L_2 \in \mathcal{C}\}.$$

Finally, $\mathrm{BH}(\mathcal{C}) = \bigcup_{k=1}^{\infty}(\mathrm{BH}_k(\mathcal{C}))$. Boolean hierarchies over general complexity classes have been studied previously in [2].

The Boolean hierarchy can be defined in many different ways. We will make use of the following two characterizations which hold for complexity classes $\mathcal{C}$ that are closed under union, which will be the case for the classes we consider in this paper. The first is a convenient normal form, and the second shows that for such classes the Boolean hierarchy is the same as the difference hierarchy [9].

Proposition 1 *Suppose $\mathcal{C}$ is closed under union. For any $k > 1$,*

$$\mathrm{BH}_{2k}(\mathcal{C}) = \bigcup_{i=1}^{k} L_i$$

where for each i, $L_i = L_{i1} \cap \overline{L}_{i2}$, $L_{i1}, L_{i2} \in \mathcal{C}$, and

$$\mathrm{BH}_{2k+1}(\mathcal{C}) = (\bigcup_{i=1}^{k} L_i) \cup L_{k+1}$$

where the L_i, $1 \leq i \leq k$, are as above, and $L_{k+1} \in \mathcal{C}$.

Proposition 2 *Suppose $\mathcal{C}$ is closed under union. Then for all $k > 1$,*

$$\mathrm{BH}_k(\mathcal{C}) = \{L | L = L_1 \cap L_2, \overline{L}_1 \in \mathrm{BH}_{k-1}(\mathcal{C}), L_2 \in \mathcal{C}\}.$$

where

3 Analogies Between NP and $\mathrm{C}_{\neq}\mathrm{P}$

It has been known for some time that $\mathrm{C}_{=}\mathrm{P}$ is closed under intersection ([22], [23]). In [22] it was also proved that $\mathrm{C}_{=}\mathrm{P}$ is closed under unbounded cartesian product which implies that it is closed under (polynomial-time) conjunctive truth-table reductions. The proof of this fact is similar to the proof that $\mathrm{C}_{=}\mathrm{P}$ is closed under intersection. The closure of $\mathrm{C}_{=}\mathrm{P}$ under union was proved in [12]. Again, using essentially the same proof as for the closure under union it is easy to see that $\mathrm{C}_{=}\mathrm{P}$ is closed under polynomial-time disjunctive truth-table reductions. Translating these results into the context of $\mathrm{C}_{\neq}\mathrm{P}$, we find that $\mathrm{C}_{\neq}\mathrm{P}$ is closed under union, intersection, and both disjunctive and conjunctive polynomial-time truth-table reductions. That $\mathrm{C}_{\neq}\mathrm{P}$ is closed under $\leq_m^{\mathrm{NP}}$-reductions is a simple observation, and is well-known. For completeness, we give a proof here.

Proposition 3 $\mathrm{C}_{\neq}\mathrm{P}$ *is closed under $\leq_m^{\mathrm{NP}}$-reductions.*

Proof: Let $L \in \mathrm{C}_{\neq}\mathrm{P}$ and suppose $L' \leq_m^{\mathrm{NP}} L$. We will show that $L' \in \mathrm{C}_{\neq}\mathrm{P}$. We know that there exist functions $f, g \in \#\mathrm{P}$ that for any w, $f(w) \geq g(w)$ and $w \in L$ if and only if $f(w) \neq g(w)$. Since $L' \leq_m^{\mathrm{NP}} L$, there exists a polynomial p and a function $h \in \mathrm{FP}$ such that $x \in L' \Leftrightarrow (\exists z, |z| \leq p(|x|))(f(h(\langle x, z\rangle)) \neq g(h(\langle x, z\rangle))$. Since for any w, $f(w) \geq g(w)$, the predicate $(\exists z, |z| \leq p(|x|))(f(h(\langle x, z\rangle)) \neq g(h(\langle x, z\rangle))$ is true if and only if $\sum_z f(h(\langle x, z\rangle)) \neq \sum_z g(h(\langle x, z\rangle))$. But then using standard techniques we can implement these sums in terms of #P machines, i.e., there exist nondeterministic, polynomial time bounded machines N_1' and N_2' such that for all x, $x \in L' \Leftrightarrow \#acc_{N_1'}(x) \neq \#acc_{N_2'}(x)$. Hence $L' \in \mathrm{C}_{\neq}\mathrm{P}$. ∎

For the most part, we make use of this fact in the following form, which says that $C_{\neq}P$ is closed under existential quantification.

Corollary 4 $\exists C_{\neq}P = C_{\neq}P$.

Now many interesting results can be derived using analogous proofs for NP, with $C_{\neq}P$ playing the role of NP. For the first result, recall that truth-table reductions to NP are exactly as powerful as Turing reductions with logarithmically many queries, i.e., $P^{NP}_{O(log(n))-T} = P^{NP}_{tt}$ ([7], [10]). Here we find the same is true of $C_{=}P$. This result has also been found by Toda [19]. The result and the proof reported here were obtained independently [1]. The proof we give is typical of those that use the closure properties shared by NP and $C_{\neq}P$.

Theorem 5 $P^{C_{=}P}_{O(log(n))-T} = P^{C_{=}P}_{tt}$.

Proof: The inclusion from left to right is straightforward: the query tree of a $P^{C_{=}P}_{O(log(n))-T}$ machine can be written down in polynomial time, and a polynomial size truth table can be constructed from the query tree.

The inclusion from right to left follows from an argument similar to the one used to prove $P^{NP}_{tt} \subseteq P^{NP}_{O(log(n))-T}$, aided by corollary 4. Let M^A be a $P^{C_{=}P}_{tt}$ machine, where (without loss of generality) $A \in C_{\neq}P$ (e.g., $A = \overline{\text{ESAT}}$), and M^A is bounded by the polynomial p. On input x, $|x| = n$, suppose M produces the queries $S = \{q_1, q_2, ...q_{p(n)}\}$. We will show how to simulate the rest of the computation of M^A using logarithmically many queries to $C_{\neq}P$. First, we show that with logarithmically many queries to $C_{\neq}P$, we can determine the number l of q_i's such that $q_i \in \overline{\text{ESAT}}$. Consider a nondeterministic machine N that, on input $\langle S, k\rangle$, guesses k elements of S and then accepts iff all the guessed strings are elements of $\overline{\text{ESAT}}$. Using the closure of $C_{\neq}P$ under polynomial-time conjunctive reductions, it is easy to see that N is an $\exists C_{\neq}P$ machine. By corollary 4 we can simulate N by some $C_{\neq}P$ machine N'. Note that by binary search on k, since $0 \leq k \leq p(n)$, we can determine l with log many queries. We now know both l, the number of positive answers to the queries in S, as well as $p(n) - l$, the number of negative answers. With one extra query to $C_{\neq}P$ we can simulate M^A. We construct a nondeterministic machine N'' which guesses l elements of S, and rejects if any one of them is in ESAT. If all the guessed elements are in $\overline{\text{ESAT}}$, then we know the set of queries with positive answers and with negative answers. Using this information, simulate M^A directly, accepting if and only if M^A accepts. It is easy to see that N'' is an $\exists C_{\neq}P$ machine, and therefore, again using corollary 4, a $C_{\neq}P$ machine. Hence with $O(log(n))$ queries to $C_{\neq}P$ we can simulate M^A. ∎

It is well known that NP = co-NP if and only if PH = NP. A similar phenomenon occurs if $C_{=}P$ is closed under complement.

[1] Subsequently Ogiwara [15] has substantially generalized this result, exhibiting sufficient conditions for any complexity class $\mathcal{C}$ to obey $P^{\mathcal{C}}_{tt} = P^{\mathcal{C}}_{O(log(n))-T}$. He has also proved the very nice result that $C_{=}P$ is closed under positive Turing reductions (and exhibits sufficient conditions for any class to have this property).

Corollary 6 $C_=P = C_{\neq}P$ *if and only if* $PH^{PP} = C_=P$.

Proof: The "if" part is clear. Then suppose $C_=P = C_{\neq}P$. Torán [22] has proved that $NP^{PP} = NP^{C_=P} = \exists C_=P$. Hence if $C_=P = C_{\neq}P$, $NP^{PP} = \exists C_=P = \exists C_{\neq}P = C_{\neq}P = C_=P$. ∎

We conclude this section with some remarks on the Boolean and query hierarchies. Not surprisingly, many of the properties of BH(NP) are shared by BH($C_=P$). For example, the levels of BH($C_=P$) have complete problems analogous to those for the levels of BH(NP). Another important example for this paper is that, just as in the case of NP, there is a tight intertwining relationship between the Boolean and query hierarchies over $C_=P$. The proof of this fact follows Beigel's proof for NP [4]. In fact it was pointed out in [4] that the proof for this theorem only depends on the existence of a complete problem, and closure under intersection, union and $\leq_m^{NP}$-reducibility.

Theorem 7 *For all* $k \geq 1$, $BH_{2^k-1}(C_=P) \cup \text{co-}BH_{2^k-1}(C_=P) \subseteq P^{C_=P}_{k-T} \subseteq BH_{2^k}(C_=P) \cap \text{co-}BH_{2^k}(C_=P)$.

It is natural to ask if the Boolean hierarchy over $C_=P$ collapses, or, equivalently, if the query hierarchy over $C_=P$ collapses to some finite level. (Corollary 6 represents a first step in this direction.) As mentioned in the introduction, up to now the only known separation is a relativized one (see [12] and section 6). In contrast for NP, structural relationships between the Boolean hierarchy over NP and the polynomial hierarchy are known. It was proved by Kadin that if the Boolean hierarchy collapses to a finite level then PH collapses to $P^{NP^{NP}}$ [13],[11]. Stated more precisely, if for any k, $BH_k(NP) = \text{co-}BH_k(NP)$ then $PH \subseteq \Delta_2^{NP}$ (in fact, Chang and Kadin show the much sharper consequence $PH \subseteq BH_k(NP^{NP})$). Another way of saying this is that if for any k, $BH_k(NP) = \text{co-}BH_k(NP)$, then $PH^{NP} \subseteq BH_k(NP^{NP})$. We observe here that Chang and Kadin's proof of this fact carries over directly if we put $C_{\neq}P$ in place of NP. Thus if we replace the NP's in the hypothesis $BH_k(NP) = \text{co-}BH_k(NP)$ with $C_=P$'s, we can replace the oracle NP's in the conclusion with $C_=P$, that is the conclusion becomes $PH^{C_=P} \subseteq BH_k(NP^{C_=P})$. Since $NP^{PP} = NP^{C_=P}$ and $PH^{PP} = PH^{C_=P}$ [22], we find the following

Theorem 8 *If for any* k, $BH_k(C_=P) \subseteq \text{co-}BH_k(C_=P)$ *then* $PH^{PP} \subseteq BH_k(NP^{PP}) \subseteq P^{NP^{PP}}_{k-T}$.

Independently, Beigel, Chang and Ogiwara [5] have made the same observation, but they present their own simpler proof of this, using a "mind change" technique that allows them to derive a stronger result.

Thus making use of theorem 7 we have,

Corollary 9 *If for any* k, $P^{C_=P}_{(k+1)-T} \subseteq P^{C_=P}_{k-T}$, *then* $PH^{PP} \subseteq P^{NP^{PP}}$.

The proof of theorem 8 exploits the analogy between $C_{\neq}P$ and NP. Using the terminology of [13], Corollary 4 allows us to do "oracle replacement" in nondeterministic computations when it is possible to find "small $C_{\neq}P$ machines" for $C_=P$. The "hard string/easy string" argument similarly holds with $\overline{\text{ESAT}}$ playing the role of SAT. The reason their proof works is that the only essential properties of NP that are used are the existence of a complete problem, and closure under $\leq_m^{\text{NP}}$-reductions and conjunctive reductions, all of which are also properties of $\text{C}_{\neq}\text{P}$.

Rather than reproduce the full proof of Chang and Kadin, we will indicate how it works by briefly sketching the proof for the following special case, analogous to the example proved in their paper: If $\text{BH}_2(\text{C}_=\text{P}) \subseteq \text{co-BH}_2(\text{C}_=\text{P})$ then $\text{PH}^{\text{PP}} \subseteq \text{P}_{2-\text{T}}^{\text{NP}^{\text{PP}}}$. First note that the following sets are $\leq_m^p$-complete for $\text{BH}_2(\text{C}_=\text{P})$ and $\text{co-BH}_2(\text{C}_=\text{P})$, respectively:

$$\overline{\text{ESAT}} \ \wedge \ \text{ESAT} = \{\langle x_1, x_2\rangle | x_1 \in \overline{\text{ESAT}} \text{ and } x_2 \in \text{ESAT}\}$$

$$\text{ESAT} \ \vee \ \overline{\text{ESAT}} = \{\langle x_1, x_2\rangle | x_1 \in \text{ESAT} \text{ or } x_2 \in \overline{\text{ESAT}}\}.$$

Now assume $\text{BH}_2(\text{C}_=\text{P}) \subseteq \text{co-BH}_2(\text{C}_=\text{P})$. Then $\overline{\text{ESAT}} \wedge \text{ESAT} \leq_m^p \text{ESAT} \vee \overline{\text{ESAT}}$, where the reduction is via some polynomial time computable function h. A string x is called *hard* if $x \in \text{ESAT}$ and for any y where $|y| \leq |x|$, $\pi_2(h(y,x)) \in \text{ESAT}$. Otherwise x is said to be easy. If a string x is known to be easy, we can determine that $x \in \text{ESAT}$ using a $\text{C}_{\neq}\text{P}$ algorithm: guess a y with $|y| \leq |x|$, and accept if $\pi_2(h(y,x)) \in \overline{\text{ESAT}}$ (actually the algorithm is $\exists\text{C}_{\neq}\text{P}$, but this is the same as $\text{C}_{\neq}\text{P}$). On the other hand, if there exists a hard string z of a given length m, the hard string can be used to find a $\text{C}_{\neq}\text{P}$ algorithm for all of ESAT up to length m. That is, given any string x, $|x| \leq m$, compute $\pi_1(h(x,z))$ and verify that it is in $\overline{\text{ESAT}}$. It is not hard to show that $x \in \text{ESAT} \Leftrightarrow \pi_1(h(x,z)) \in \overline{\text{ESAT}}$. Note that the problem of determining if a string is hard is in $\forall\text{C}_=\text{P} = \text{C}_=\text{P}$, and therefore the problem of determining if there exists a hard string of a given length is in $\text{NP}^{\text{C}_=\text{P}}$. We can now argue that $\text{NP}^{\text{NP}^{\text{C}_=\text{P}}} \subseteq \text{P}_{2-\text{T}}^{\text{NP}^{\text{C}_=\text{P}}}$. Suppose $L \in \text{NP}^{\text{NP}^{\text{C}_=\text{P}}}$, in particular, $L = L(N)$, where N is an NP-machine which makes queries to an $\text{NP}^{\text{C}_=\text{P}}$ oracle and whose run-time is bounded by r. We will now describe a $\text{P}_{2-\text{T}}^{\text{NP}^{\text{C}_=\text{P}}}$ algorithm for L. Suppose we want to know if $x \in L$. Without loss of generality, the queries that N^{NP} makes to $\text{C}_=\text{P}$ have length exactly equal to $p(r(|x|))$, for some polynomial p. With one query to $\text{NP}^{\text{C}_=\text{P}}$ we can determine if there is a hard string of length $p(r(|x|))$. With one last query to $\text{NP}^{\text{C}_=\text{P}}$ we can simulate N using one of the above algorithms to reduce ESAT to $\overline{\text{ESAT}}$, depending on whether or not there is a hard string of length $p(r(|x|))$, as follows. If there is no hard string of length $p(r(|x|))$, we know any query to $\text{C}_=\text{P}$ is easy and we can use the above "easy string" guessing algorithm (and the fact that $\text{NP}^{\text{C}_=\text{P}} = \exists\text{C}_=\text{P}$) to replace the $\text{NP}^{\text{C}_=\text{P}}$ oracle of N with a $\text{C}_{\neq}\text{P}$ oracle. If there is a hard string of length $p(r(|x|))$, we can guess it, verify with a query to $\text{C}_=\text{P}$ that it is indeed hard, and then continue the computation using the "hard string" algorithm to determine if $x \in L$. The latter is an $\text{NP}^{\text{C}_=\text{P}}$ computation. Both the easy and hard cases only require one extra query to $\text{NP}^{\text{C}_=\text{P}}$. Thus $\text{NP}^{\text{NP}^{\text{C}_=\text{P}}} \subseteq \text{P}_{2-\text{T}}^{\text{NP}^{\text{C}_=\text{P}}}$ and therefore $\text{PH}^{\text{C}_=\text{P}} \subseteq \text{P}_{2-\text{T}}^{\text{NP}^{\text{C}_=\text{P}}}$, which implies $\text{PH}^{\text{PP}} \subseteq \text{P}_{2-\text{T}}^{\text{NP}^{\text{PP}}}$.

4 EQ Circuits and Their Limitations

We define a family of circuits closely related to $\text{C}_=\text{P}$. An *EQ circuit of size m, order s and threshold t* over the inputs $X = \{x_1, ..., x_n\}$ consists of a set of AND gates, $c_i, 1 \leq i \leq m$, where each AND gate has fanin at most s and the circuit outputs 1 if and only if $\sigma(X) = t$, where by definition

$$\sigma(X) = \sum_{i=1}^{m} c_i(X).$$

An *NEQ circuit of size m, order s and threshold t* is defined similarly, except that it outputs 1 if and only if $\sigma(X) \neq t$. (Note that the AND-gates in these definitions can have negated variables. An AND gate can have fanin 0 in which case by definition it outputs the constant 1.)

We say that an EQ circuit C on the inputs $\{x_1, ..., x_n\}$ is *computable by a non-negative polynomial of degree d* if there exists a polynomial p of degree d with integer coefficients in the variables $\{x_1, ..., x_n\}$, and for all settings of the x_i's, the following conditions hold: $p(x_1, ..., x_n) \geq 0$, and $C(x_1, ..., x_n) = 1$ if and only if $p(x_1, ..., x_n) = 0$. The notion that an NEQ circuit C is computable by a non-negative polynomial is defined precisely the same, except that $C(x_1, ..., x_n) = 1$ if and only if $p(x_1, ..., x_n) \neq 0$.

Using a technique from [12], it is easy to show that every EQ (resp. NEQ) circuit can be computed by a non-negative polynomial.

Proposition 10 *Let C be an EQ circuit of size m and order s. Then C is computable by a non-negative polynomial p with at most $(2^s m + 1)^2)$ terms and degree at most $2s$.*

Proof: We have that $C = 1$ if and only if

$$\sum_{i=1}^{m} c_i(X) = t$$

where $X = \{x_1, ..., x_n\}$. Then $C = 1$ if and only if

$$(\sum_{i=1}^{m} c_i(X) - t)^2 = 0.$$

Note that the left hand side is always ≥ 0. Rewrite each negated variable $\overline{x}_i$ as $1 - x_i$. Expressing each AND-gate c_i as a product, expand to obtain a sum of terms of the form $\prod_{i \in S} x_i$ (note that $||S|| \leq s$), or the constant term 1. Note that the coefficients of the non-constant terms can be arbitrary, possibly negative integers (not just 0 or 1). This yields at most 2^s terms for each of the m AND-gates. Then perform the square in the above equation. We obtain a sum of terms which are constant or are of the form $w \cdot \prod_{i \in S'} x_i$, where w is an integer. Now note that $||S'|| \leq 2s$. Thus the sum is a polynomial p of degree $\leq 2s$ with at most $(2^s m + 1)^2$ terms, and integer coefficients. We have that $p \geq 0$, and $p = 0$ if and only if $C = 1$.

A similar argument holds for NEQ. ∎

We use the following lemma to establish the relativized separations. It is the same as lemma 30 in [12], translated into the context of circuits. In addition, we generalize the lemma so that it can be applied to Boolean functions which are symmetric with respect to certain (disjoint) subsets of the input variables, a generalization of the usual definition of symmetric function. More precisely, in the following, for any subset $S \subseteq \{x_1, ..., x_n\}$ we define, for any truth assignment to the x's, $y(S) = ||\{j|(x_j \in S) \wedge (x_j = 1)\}||$. For some Boolean function f over the variables $\{x_1, ..., x_n\}$, suppose there exist disjoint subsets $S_1, S_2, ..., S_k \subseteq \{x_1, ..., x_n\}$ such that if we put $y_j = y(S_j)$ for each $j, 1 \leq j \leq k$, then f can be expressed as a function g of the numbers y_j, that is, there is a function g such that $(\forall x_1, ..., x_n)(f(x_1, ..., x_n) = g(y_1, ..., y_k))$. We then say that f is *symmetric* in the subsets $S_1, ..., S_k$ via the function $g(y_1, ..., y_k)$.

Tarui ([17], proposition 1) proves a similar lemma for $k = 1$ using algebraic techniques. He proves that a polynomial of degree d which equals 0 for more than d values of y_1 identically vanishes for all y_1.

Lemma 11 *Let $s, k, n \in \mathbb{N}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function over the variables $\{x_1, ..., x_n\}$ which is symmetric in the subsets $S_1, ..., S_k \subseteq \{x_1, ..., x_n\}$ via the function $g(y_1, ..., y_k)$. Let $Y_1, Y_2, ..., Y_k$ respectively denote the sets of values of $y_1, ..., y_k$ such that $g(y_1, ..., y_k) = 1$, and suppose that for all $i, 1 \leq i \leq k$, $||Y_i|| \geq s$. Let C be an EQ circut of order $< s/2$ such that $f(x_1, ..., x_n) = 1 \Rightarrow C = 1$. Then for any input setting, $C = 1$.*

Proof: By proposition 10, we know that C is computable by a non-negative polynomial q of degree less than s. Let q consist of terms c_i, $i = 1, ..., m$, each of degree at most $s - 1$. Let x denote $\{x_1, ..., x_n\}$. Thus q is of the form $\sum_{l=1}^{m} w_l c_l(x)$ where the w_l's are integers. We know $C = 1$ if and only if $q(x) = 0$. Also from proposition 10 we can assume that $q(x) \geq 0$ for any setting of the inputs. We know for any choice of y_j's such that $g(y_1, ..., y_k) = 1$, that $q(x) = 0$. Let us compute the sum of $q(x)$ over input settings x for any fixed $y_j, 1 \leq j \leq k$. Let the notation "$x : y$" denote that the assignments x can vary given fixed values for the y_j's. One can reverse the order of the sums: $\sum_{x:y} q(x) = \sum\limits_{l=1}^{m} w_l \sum_{x:y} c_l(x)$. Let $S_{lj} = \{i | x_i \in S_j$ and x_i is an input to $c_l\}$ and $s_{lj} = ||S_{lj}||$. Thus for each l, $c_l(x) = \prod_{j=1}^{k}(\prod_{i \in S_{lj}} x_i)$. Letting $n_j = ||S_j||$ where $1 \leq j \leq k$, it is not hard to show that

$$\sum_{x:y} c_l(x) = \prod_{j=1}^{k} \binom{n_j - s_{lj}}{y_j - s_{lj}}.$$

Observe that, by hypothesis, $\sum_{j=1}^{k} s_{lj} < s$, so that the quantity

$$\prod_{j=1}^{k} y_j!(n_j - y_j)! \prod_{j=1}^{k} \binom{n_j - s_{jl}}{y_j - s_{jl}} = \prod_{j=1}^{k} [(n_j - s_{jl})! \prod_{i=0}^{s_{jl}-1} (y_j - i)]$$

is a polynomial in each individual y_j of degree $< s$. Therefore the quantity

$$\prod_{j=1}^{k} y_j!(n_j - y_j)! \sum_{x:y} q(x)$$

is a polynomial $p(y_1, ..., y_k)$ of degree $< s$ in each y_j. Since for any x such that $g(y_1, ..., y_k) = 1$ we also have $q(x) = 0$, it follows from the definition of $p(y_1, ..., y_k)$ that $p(y_1, ..., y_k) = 0$ for any such x. Conversely, note that since $q(x) \geq 0$ for any x, if for any choice of $y_1, ..., y_k$ we have $p(y_1, ..., y_k) = 0$, then for all x with this choice of $y_1, ..., y_k$, it follows that $q(x) = 0$ and therefore $C = 1$.

We will now show that for all possible values of $y_1, ..., y_k$, $p(y_1, ..., y_k) = 0$. This will prove that for all input settings, $C = 1$.

We know by hypothesis that

$$(\forall y_1 \in Y_1)(\forall y_2 \in Y_2)...(\forall y_k \in Y_k)(g(y_1, ..., y_k) = 1).$$

where for all $i, 1 \leq i \leq k$, $||Y_i|| \geq s$. Thus from the above argument, it follows that

$$(\forall y_1 \in Y_1)(\forall y_2 \in Y_2)...(\forall y_k \in Y_k)(p(y_1, ..., y_k) = 0).$$

Fix any $y_1, y_2, ..., y_{k-1}$ to the values $\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_{k-1}$ respectively where $\mathrm{y}_1 \in Y_1, \mathrm{y}_2 \in Y_2, ...$ and $\mathrm{y}_{k-1} \in Y_{k-1}$ (the Roman y's denote that these values are fixed). Then $p(\mathrm{y}_1, ..., \mathrm{y}_{k-1}, y_k)$ is a polynomial in y_k of degree $< s$. However, $g(\mathrm{y}_1, ..., \mathrm{y}_{k-1}, y_k) = 1$ for $\geq s$ values of y_k, and hence $p(\mathrm{y}_1, ..., \mathrm{y}_{k-1}, y_k) = 0$ for this many values of y_k. Therefore $p(\mathrm{y}_1, ..., \mathrm{y}_{k-1}, y_k) = 0$ for all values of y_k, $0 \leq y_k \leq ||S_k||$. Proceeding inductively in this fashion, we find that

$$(\forall y_1, 0 \leq y_1 \leq ||S_1||)...(\forall y_k, 0 \leq y_k \leq ||S_k||)(p(y_1, ..., y_k) = 0).$$

This proves the lemma. ▮

Note that we can replace "EQ" with "NEQ" and "$C = 1$" with "$C = 0$" in the above lemma and that it remains true.

Lemma 11 will be used to eliminate EQ (respectively, NEQ) circuits by showing that they have constant values.

5 Separations of $\mathrm{P}^{\mathrm{C_=P}}$ from P^{PP}

As explained in the introduction, intuitively PP appears to be more powerful than $\mathrm{C_=P}$ in deterministic reductions, even though it is no more powerful than $\mathrm{C_=P}$ in nondeterministic reductions. We show here that any proofs that Turing reductions to $\mathrm{C_=P}$ are as powerful as Turing reductions to PP would, at least, not relativize. These separations leave open the possibility that, while P^{PP} machines can count (via the binary search technique mentioned in the introduction), $\mathrm{P}^{\mathrm{C_=P}}$ perhaps cannot.

First we need an alternative characterization of $\mathrm{P}^{\mathrm{C}_=\mathrm{P}}$ in terms of the extended Boolean hierarchy over $\mathrm{C}_=\mathrm{P}$. Following Wagner [24], we define, for any complexity class $\mathcal{C}$ and for any bounding function b, the class $\mathcal{C}(b)$ as follows: $A \in \mathcal{C}(b)$ if and only if there exists a $B \in \mathcal{C}$ such that for any x, for all z where $1 \le z \le b(|x|)$, $\chi_B(\langle x, z\rangle) \le \chi_B(\langle x, z-1\rangle)$, and $x \in A$ if and only if $max\{z | 0 \le z \le b(|x|) - 1 \text{ and } \langle x, z\rangle \in B\}$ is odd. $\mathcal{C}(2^{poly})$ is defined as $\bigcup_{c\in\mathbb{N}} \mathcal{C}(2^{n^c})$. It was shown in [24] that $\mathrm{NP}(2^{poly}) = \mathrm{P}^{\mathrm{NP}}$. We can prove the analogous result here.

Theorem 12 $\mathrm{C}_{\neq}\mathrm{P}(2^{poly}) = \mathrm{P}^{\mathrm{C}_=\mathrm{P}}$.

Proof: ($\subseteq$): Let $A \in \mathrm{C}_{\neq}\mathrm{P}(2^{poly})$ and let B be as in the definition above (with $\mathcal{C} = \mathrm{C}_{\neq}\mathrm{P}$), and b the bounding function (of the form $2^{p(n)}$). Define the set $S = \{\langle x, y\rangle | 0 \le y \le b(|x|), (\exists z, 0 \le z \le b(|x|))(z > y \wedge \langle x, z\rangle \in B)\}$. By binary search, we can find the maximum y such that $\langle x, y\rangle \in S$ in time $p(|x|)$, using S as an oracle. We can easily tell if the maximum y is odd. Finally, note that $S \in \exists\mathrm{C}_{\neq}\mathrm{P} = \mathrm{C}_{\neq}\mathrm{P}$. Then $A \in \mathrm{P}^{\mathrm{C}_=\mathrm{P}}$.

($\supseteq$): Let $L \in \mathrm{P}^{\mathrm{C}_=\mathrm{P}}$ via machine M which makes queries to a set $A \in \mathrm{C}_{\neq}\mathrm{P}$. Define the set B' as follows. $B' = \{\langle x, z\rangle | z = r_1\#r_2\#...r_{p(|x|)}\#a, a \in \{0,1\}, M$, using query answers $r_1, ..., r_{p(|x|)}$, produces queries $q_1, ..., q_{p(|x|)}$, and $(M \text{ accepts } x \Leftrightarrow a = 1) \wedge (\forall i)(r_i = 1 \Rightarrow q_i \in A)\}$. Using the fact that $\mathrm{C}_{\neq}\mathrm{P}$ is closed under conjunctive truth-table reductions, it is easy to see that $B' \in \mathrm{C}_{\neq}\mathrm{P}$. Furthermore, M accepts x if and only if the maximum z such that $\langle x, z\rangle \in B'$ is odd. Thus

$$\begin{aligned} x \in L &\Leftrightarrow max\{z | \langle x, z\rangle \in B'\} \neq 0 \bmod 2 \\ &\Leftrightarrow max\{z | (\exists w \ge z)\langle x, w\rangle \in B'\} \neq 0 \bmod 2 \\ &\Leftrightarrow max\{z | \langle x, z\rangle \in B\} \neq 0 \bmod 2 \end{aligned}$$

where $B = \{\langle x, z\rangle | (\exists w \ge z)(\langle x, w\rangle \in B')\}$. Now since $B' \in \mathrm{C}_{\neq}\mathrm{P}$, $B \in \exists\mathrm{C}_{\neq}\mathrm{P} = \mathrm{C}_{\neq}\mathrm{P}$. Furthermore, $\chi_B(\langle x, z\rangle) \le \chi_B(\langle x, z-1\rangle)$. This proves the theorem. ▮

Observe that the statement $max\{z | \langle x, z\rangle \in B\} \neq 0 \bmod 2$ is equivalent to the statement $\sum_{z=0}^{b(|x|)} \chi_B(\langle x, z\rangle) \neq 0 \bmod 2$, because of the monotonicity of the χ_B's in z. Thus the separation results can be understood in terms of a simple and highly constrained circuit model.

Definition 13 Let c_i, $1 \le i \le m$, be a set of NEQ circuits, over the inputs $\{x_1, ..., x_n\}$, each of order s, with the property that $c_i = 1 \Rightarrow c_{i-1} = 1$. A $\mathrm{P}^{\mathrm{C}_=\mathrm{P}}$*-circuit of size* m *and order* s is a circuit which outputs 1 if and only if $||\{i | 1 \le i \le m, c_i = 1\}||$ is odd.

Proposition 14 *Let* M *be a* $\mathrm{P}^{\mathrm{C}_=\mathrm{P}^A}$*-machine. Then there is a polynomial* q *such that for any input* x*, there exists a* $\mathrm{P}^{\mathrm{C}_=\mathrm{P}}$*-circuit* C *of order* $q(|x|)$ *whose inputs are of the form* $\chi_A(z)$ $(|z| \le p(|x|))$*, such that* $C = 1$ *if and only if* M *accepts* x.

Proof: By theorem 12 and the remarks following it, we can assume there is a set $B \in \mathrm{C}_{\neq}\mathrm{P}^A$ and a polynomial p such that for any x, M accepts x if and only if

$$\sum_{z=0}^{2^{p(|x|)}} \chi_B(\langle x, z\rangle) \neq 0 \bmod 2.$$

Let N be a $\mathrm{C}_{\neq}\mathrm{P}^A$ machine with time bounded by polynomial r, recognizing B, so that for some $t \in \mathrm{FP}$, for any x, $x \in B$ if and only if $\#acc_N(x) \neq t(x)$. Note that for some polynomial q, $r(|\langle x, z\rangle|) \leq q(|x|)$, so that computation paths of N on input $\langle x, z\rangle$ are no longer than $q(|x|)$. Because N correctly recognizes B, it is clear that if N accepts on input $\langle x, z\rangle$ then N also accepts on input $\langle x, z-1\rangle$. Consider the computation of N on an input $\langle x, z\rangle$. In a standard fashion, define a new machine N' which simulates N as follows. Whenever N would query a string to A, N' guesses an answer and records both the query and the guess. When N' arrives in an accept state of N, it attempts to verify the answers it guessed by making queries to A in a series of universal moves; refer to the resulting subtree as a "verification subtree". Note that the verification subtrees are no larger than $q(|x|)$. Also note that N' may have as many as $2^{2q(|x|)}$ computation paths leading to verification subtrees (the upper bounded applying if N made queries in every step of every computation path). Now we can think of the computation tree of N' on input $\langle x, z\rangle$ as an NEQ circuit which (for any given x) we will call c_z. The inputs to c_z are values of the characteristic function of A for various strings (i.e., of the form $\chi_A(w)$). The verification subtrees correspond to AND-gates with fanin $q(|x|)$. We eliminate the contribution of computation paths along which no queries were made by adjusting the threshold. If a_{nq} is the number of accepting paths of N along which no queries were made, then the threshold for c_z will be $t(x) - a_{nq}$. Thus for any input to N of the form $\langle x, z\rangle$, c_z outputs 1 if and only if N accepts. Furthermore, each c_z is of order $q(|x|)$.

Evidently M accepts x if and only if

$$\sum_{z=0}^{2^{p(|x|)}} c_z \neq 0 \bmod 2$$

and furthermore $c_z = 1 \Rightarrow c_{z-1} = 1$. This defines a $\mathrm{P}^{\mathrm{C=P}}$-circuit of order $q(|x|)$. Note that this order is polylogarithmic in the number of inputs to the circuit, since the number of inputs (which are of the form $\chi_A(w)$ where $|w| \leq q(|x|)$) is exponential in $|x|$. ∎

In the proof of the following we make use of lemma 11, restricted to the case of $k = 1$, that is, for ordinary symmetric functions. We say a family of circuits $\{C_n\}$ has *polylog order* if there exists a polynomial q such that for all n, C_n has order $q(log(n))$.

Lemma 15 *Let $s \in \mathbb{N}$, $s \geq 1$, and $f : \{0,1\}^n \to \{0,1\}$ be a symmetric Boolean function such that if we put $y = ||\{i | x_i = 1\}||$, then $f(x_1, ..., x_n) = 1$ for $\geq s$ values of y and $f(x_1, ..., x_n) = 0$ for $\geq s$ values of y. Then no $\mathrm{P}^{\mathrm{C=P}}$-circuit C of order $< s/2$ can compute f.*

Proof: Let $C(x_1, ..., x_n)$ be a $\mathrm{P}^{\mathrm{C=P}}$-circuit of order $< s/2$ with NEQ subcircuits $c_i, 1 \leq i \leq m$, which computes $f(x_1, ..., x_n)$. We will show that for all input settings, $C = 0$. Suppose the number of NEQ subcircuits m is odd (the proof when m is even is the same, as will be obvious). Then whenever $f(x) = 0$, we must have $c_m = 0$, since if $c_m = 1$ all the c_i's with $i < m$ will also have to be 1, and hence we would have $C = 1$. By hypothesis, for $\geq s$ values of y, $f(x) = 0$. Since c_m is of order $< s/2$ but is zero for $\geq s$ values of y, we can conclude from lemma 11 that $c_m = 0$ for all input settings. This eliminates c_m from consideration. Consider the remaining $m-1$ gates. Since $m-1$ is even, if $f(x) = 1$ we must have $c_{m-1} = 0$, otherwise an even number of NEQ circuits would yield 1. c_{m-1} can be eliminated just as c_m, since $f(x) = 1$ for $\geq s$ values of y. Proceeding inductively, we find that all the c_i's equal 0 for all input settings, and therefore $C = 0$ for all input settings. ▮

The parity function on n inputs, $\oplus(x_1, ..., x_n) =$, is defined to be 1 if and only if $||\{i|x_i = 1\}||$ is odd.

Lemma 16 *For any $n \geq 1$, no $\mathrm{P}^{\mathrm{C=P}}$-circuit of order $< n/4$ can compute the parity function on n inputs.*

Proof: $\oplus(x_1, ..., x_n)$ is a symmetric function which is 1 for $\geq n/2$ values of $y = ||\{i|x_i = 1\}||$ and is 0 for $\geq n/2$ values of y. Applying lemma 15, the result is immediate. ▮

In the proof of the following, as well as in the following section, s_i^n denotes the i^{th} string of length n. We note that Tarui [17] has independently obtained the theorem as well as theorem 19 using slightly different techniques. In place of the concept of $\mathrm{P}^{\mathrm{C=P}}$-circuits, Tarui uses the concept of small-depth decision trees.

Theorem 17 *There exists an oracle A such that $\oplus\mathrm{P}^A \not\subseteq \mathrm{P}^{\mathrm{C=P}^A}$.*

Proof: Define the test language $L(A) = \{1^n |$ there are an odd number of strings in A of length $n\}$. Then $L(A) \in \oplus\mathrm{P}^A$ for any A. Thus for any n, $1^n \in L(A) \Leftrightarrow \oplus(\chi_A(s_1^n), \chi_A(s_2^n), ..., \chi_A(s_{2^n}^n)) = 1$. An A can be constructed in stages such that $L(A) \notin \mathrm{P}^{\mathrm{C=P}^A}$. Let $M_i, i \in \mathbb{N}$ be an enumeration of $\mathrm{P}^{\mathrm{C=P}}$-machines. At stage i, we will choose an $n_i \in \mathbb{N}$. We know that from proposition 14, there is a polynomial q_i such that we can find a $\mathrm{P}^{\mathrm{C=P}}$-circuit of order $q_i(n_i)$ which outputs 1 if and only if M_i accepts 1^{n_i}. We choose n_i to be $\geq$ its value in the previous stage and sufficiently large that $q_i(n_i) < 2^{n_i - 2}$. Using lemma 16 one can easily show that it is possible to add strings to A of length n_i such that $1^{n_i} \in L(A) \Leftrightarrow M_i$ rejects 1^{n_i}. Strings not of length n_i which M_i queries at this stage, unless fixed at earlier stages, are not put in A. We then set n_{i+1} to the smallest integer greater than any of the strings queried by M_i and proceed to stage $i+1$. ▮

Using the same technique it is also possible to prove that in some relativized world the polynomial hierarchy is not contained in $\mathrm{P}^{\mathrm{C=P}}$. In fact we can construct an oracle such that BPP is not contained in $\mathrm{P}^{\mathrm{C=P}}$, which provides a simultaneous proof that there is an oracle such that neither $\Sigma_2^p \cap \Pi_2^p$ nor PP are contained in $\mathrm{P}^{\mathrm{C_=P}}$. For this purpose we define a function related to BPP. Define the "strict majority" function on n inputs $T_{3/4}$ as follows:

$$T_{3/4}(x_1, ..., x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n x_i \geq \frac{3}{4}n, \\ 0 & \text{if } \sum_{i=1}^n x_i \leq \frac{1}{4}n, \\ ? & \text{otherwise.} \end{cases}$$

We say a circuit C *computes* $T_{3/4}$ if $T_{3/4}(x_1, ..., x_n) = 1 \Rightarrow C(x_1, ..., x_n) = 1$ and $T_{3/4}(x_1, ..., x_n) = 0 \Rightarrow C(x_1, ..., x_n) = 0$.

Lemma 18 *For any $n \geq 1$, no $\mathrm{P}^{\mathrm{C_=P}}$-circuit of order $< n/8$ can compute the strict majority function on n inputs.*

Proof: $T_{3/4}(x_1, ..., x_n)$ is a symmetric function which is 1 for $\geq n/4$ values of $y = ||\{i|x_i = 1\}||$ and is 0 for $\geq n/4$ values of y. The result follows from lemma 15. ∎

Theorem 19 *There exists an oracle A such that $\mathrm{BPP}^A \not\subseteq \mathrm{P}^{\mathrm{C_=P}^A}$.*

Proof: Define the test language $L(A) = \{1^n|$ at least 3/4 of the strings of length n are in $A\}$. Clearly, for any A such that for each length n, either 3/4 of the strings of that length are in A or no more than 1/4 are in A, then $L(A) \in \mathrm{BPP}^A$. For such A's, $T_{3/4}(\chi_A(s_1^n), \chi_A(s_2^n), ..., \chi_A(s_{2^n}^n)) = 1 \Leftrightarrow 1^n \in L(A)$ and $T_{3/4}(\chi_A(s_1^n), \chi_A(s_2^n), ..., \chi_A(s_{2^n}^n)) = 0 \Leftrightarrow 1^n \notin L(A)$. The proof is a stage construction as in theorem 17. Let $M_i, i \in \mathbb{N}$ be an enumeration of $\mathrm{P}^{\mathrm{C_=P}}$-machines. Analogously to the proof of theorem 17, using lemma 18 one can easily show that A can be constructed such that for each i, for some n_i, $1^{n_i} \in L(A) \Leftrightarrow M_i$ rejects 1^{n_i}, and furthermore $T_{3/4}(\chi_A(s_1^{n_i}), \chi_A(s_2^{n_i}), ..., \chi_A(s_{2^{n_i}}^{n_i})) = 1 \Leftrightarrow 1^{n_i} \in L(A)$ and $T_{3/4}(\chi_A(s_1^{n_i}), \chi_A(s_2^{n_i}), ..., \chi_A(s_{2^{n_i}}^{n_i})) = 0 \Leftrightarrow 1^{n_i} \notin L(A)$. ∎

Corollary 20 *There exists an oracle A such that $\Sigma_2^{p,A} \cap \Pi_2^{p,A} \not\subseteq \mathrm{P}^{\mathrm{C_=P}^A}$ and $\mathrm{PP}^A \not\subseteq \mathrm{P}^{\mathrm{C_=P}^A}$.*

Proof: It is well known that $\mathrm{BPP} \subseteq \mathrm{PP} \cap \Sigma_2^p \cap \Pi_2^p$ via a proof that relativizes (e.g., [14]). ∎

6 An Oracle Interlocking the Query Hierarchies Over NP and $\mathrm{C_=P}$

In this section we construct an oracle which gives an optimal separation of the Boolean and query hierarchies over NP and $\mathrm{C_=P}$. This represents a technical improvement of the oracle of Gundermann, Nasser and Wechsung [12].

It is possible to reduce this separation to circuit lower bounds. We now give this reduction.

Define the function $f_{2k,n}$ in $2kn$ variables $X = \{x_{ij} | i = 1..n, j = 1..2k\}$ as follows:

$$f_{2k,n}(X) = \bigvee_{j=1}^{k} [(\bigvee_{i=1}^{n} x_{i,(2j-1)}) \wedge (\bigwedge_{i=1}^{n} \overline{x}_{i,2j})]$$

Similarly, we define $f_{2k+1,n}(X \cup \{x_{i,2k+1}\})$ as

$$f_{2k+1,n}(X \cup \{x_{i,2k+1}\}) = f_{2k,n}(X) \vee (\bigvee_{i=1}^{n} x_{i,2k+1})$$

Observe that $f_{k,n}$ is symmetric in the subsets $S_j = \{x_{ij} | 1 \leq i \leq n\}$, where $1 \leq j \leq k$, via the following function g: Setting $y_j = ||\{i | x_{ij} = 1\}||$, when k is even, $g(y_1, ..., y_k) = 1$ if and only if for some odd $j \leq k-1$ we have $y_j > 0$ and $y_{j+1} = 0$. When k is odd, $g(y_1, ..., y_k) = 1$ if and only if $y_k > 0$ or for some odd $j \leq k-2$ we have $y_j > 0$ and $y_{j+1} = 0$.

A $\mathrm{BH}_k^{\mathrm{C=P}}$ circuit is defined inductively as follows. A $\mathrm{BH}_1^{\mathrm{C=P}}$ circuit is an EQ circuit. A $\mathrm{BH}_{2k}^{\mathrm{C=P}}$ circuit is a circuit of the form $C_1 \vee C_2$ where C_1 is a $\mathrm{BH}_{2k-1}^{\mathrm{C=P}}$ circuit and C_2 is an NEQ circuit. A $\mathrm{BH}_{2k+1}^{\mathrm{C=P}}$ circuit is a circuit of the form $C_1 \wedge C_2$ where C_1 is a $\mathrm{BH}_{2k}^{\mathrm{C=P}}$ circuit and C_2 is an EQ circuit. A $\mathrm{BH}_k^{\mathrm{C=P}}$ circuit of order s is one in which the base EQ and NEQ circuits are all of order s.

The main theorem which allows a complete separation of BH(NP) from BH($\mathrm{C_=P}$) is the following. It is here that we use the full power of lemma 11, with arbitrary k.

Theorem 21 *For any $n \geq 1$ and $k \geq 1$, no $\mathrm{BH}_k^{\mathrm{C=P}}$ circuit C of order $< n/2$ can compute $\{f_{k,n}\}$.*

Proof: For convenience in the proof we drop the n subscript in $f_{k,n}$. The proof is by induction. For the base case, suppose f_1 is computable by a $\mathrm{BH}_1^{\mathrm{C=P}}$ circuit, i.e., an EQ circuit C^0, of order $< n/2$. Define $y_1 = ||\{i | x_{i,1} = 1\}||$. We know that if $y_1 > 0$ then $C^0 = 1$. Thus $C^0 = 1$ for n values of y_1, and hence by lemma 11, for all input settings $C^0 = 1$. However then $C^0 = 1$ when $f_1 = 0$, a contradiction.

For the inductive step, we first consider even levels. Suppose by hypothesis that no $\mathrm{BH}_{2k-1}^{\mathrm{C=P}}$-circuit of order $< n/2$ can compute f_{2k-1}. Suppose however that there exists a $\mathrm{BH}_{2k}^{\mathrm{C=P}}$-circuit C of order $< n/2$ to compute f_{2k} correctly. C is of the form $C_1 \vee C_2$, where C_1 is a $\mathrm{BH}_{2k-1}^{\mathrm{C=P}}$-circuit and C_2 is an NEQ circuit, both of order $< n/2$. Let $y_j = ||\{i | x_{ij} = 1\}||$, as in the discussion preceeding this theorem. Also note from that discussion that whenever $y_j > 0$ for all EVEN j, $f_{2k} = 0$. Since for all j, $0 \leq y_j \leq n$, the function $g(y_1, ..., y_{2k})$ is zero for n values of $y_1, ...y_{2k}$, respectively. Now whenever $g(y_1, ...y_{2k}) = 0$, $C_2 = 0$. Then by lemma 11, for all input settings $C_2 = 0$. We can thus ignore C_2. By setting $y_{2k} = 0$, we find that C_1 computes f_{2k-1} correctly, which contradicts the induction hypothesis.

The argument for odd values $2k+1$ is similar to the base case. We assume by hypothesis that no $\mathrm{BH}_{2k}^{\mathrm{C=P}}$-circuit of order $< n/2$ can compute f_{2k}, but that there exists a $\mathrm{BH}_{2k+1}^{\mathrm{C=P}}$-circuit of order $< n/2$ of the form $C_1 \wedge C_2$ which computes f_{2k+1} correctly. We define as before $y_j = ||\{i | x_{i,j} = 1\}||$, $1 \leq j \leq 2k+1$, now noting that whenever $y_{2k+1} > 0$ we have

$f_{2k+1} = 1$. As before we find that for all input settings $C_2 = 1$, and by setting $y_{2k+1} = 0$ we conclude that C_1 computes f_{2k} correctly, which is a contradiction. ▮

The next proposition establishes the relationship between $\mathrm{BH}_k(\mathrm{C_{=}P}^A)$ and $\mathrm{BH}_k^{\mathrm{C_{=}P}}$-circuits.

Proposition 22 *Let $k \geq 1$. For any $\mathrm{BH}_{2k-1}(\mathrm{C_{=}P}^A)$ machine M, there is a polynomial q such that for any input x there exists a $\mathrm{BH}_{2k-1}^{\mathrm{C_{=}P}}$-circuit C of order $q(|x|)$ with inputs $\chi_A(w)$, $|w| \leq q(|x|)$, and $C = 1$ if and only if M accepts x. Similarly, for any co-$\mathrm{BH}_{2k}(\mathrm{C_{=}P}^A)$ machine M, there is a polynomial q such that for any input x there exists a $\mathrm{BH}_{2k}^{\mathrm{C_{=}P}}$-circuit C of order $q(|x|)$ with inputs $\chi_A(w)$, $|w| \leq q(|x|)$, and $C = 1$ if and only if M accepts x.*

Proof: The proof is by induction. For $k = 1$, it is easy to show (as in the proof of proposition 14) that for any $\mathrm{C_{=}P}^A$ machine M and input x, there exists an EQ circuit C such that M accepts x if and only if $C = 1$. The order of the circuit is polynomial in $|x|$. (Note that since there are exponentially many inputs of the form $\chi_A(w)$, the order is polylog in the number of inputs to C.)

Suppose the proposition is true for some $k \geq 1$. Consider any co-$\mathrm{BH}_{2k}(\mathrm{C_{=}P}^A)$-machine M. By proposition 2, there exist a $\mathrm{BH}_{2k-1}(\mathrm{C_{=}P}^A)$-machine M_1 and a $\mathrm{C_{\neq}P}$-machine M_2 such that for any input x, M accepts x if and only if either M_1 or M_2 accept x. By the induction hypothesis, we can replace M_1 with a $\mathrm{BH}_{2k-1}^{\mathrm{C_{=}P}}$-circuit C_1 of order polynomial in $|x|$. By an argument similar to the base case, we can replace M_2 with an NEQ circuit C_2 of order polynomial in $|x|$. This yields a circuit of the form $C_1 \vee C_2$ which outputs 1 if and only if M accepts x. $C_1 \vee C_2$ is a $\mathrm{BH}_{2k}^{\mathrm{C_{=}P}}$ circuit of order polynomial in $|x|$. The argument that we can find an appropriate $\mathrm{BH}_{2k+1}^{\mathrm{C_{=}P}}$-circuit for any $\mathrm{BH}_{2k+1}(\mathrm{C_{=}P}^A)$-machine is similar, again using proposition 2. ▮

We are now in a position to explain how the oracle separations follow from theorem 21. Define, for each k, the test languages $L_k(A)$ related to the functions $f_{k,n}$ as follows.

$$L_{2k}(A) = \{1^n | \bigvee_{i=1}^{k} [(\exists z, |z| = n)(\langle z, 2i-1\rangle \in A) \wedge (\forall z, |z| = n)(\langle z, 2i\rangle \notin A)]\}$$

$$L_{2k+1}(A) = \{1^n | 1^n \in L_{2k}(A) \vee [(\exists z, |z| = n)(\langle z, 2k+1\rangle \in A)]\}$$

It is clear from proposition 1 that for all k and A, $L_k(A) \in \mathrm{BH}_k(\mathrm{NP}^A)$. Furthermore, for any x, $x \in L_k(A) \Leftrightarrow f_{k,2^n}(X(A)) = 1$ where the arguments $X(A)$ for $f_{k,2^n}$ are defined by $x_{i,j} = \chi_A(\langle s_i^n, j\rangle)$.

Combining these observations with theorem 21 proposition 22, and a stage construction such as in theorem 17, we conclude that for some A, for all even k, $L_k(A) \notin$ co-$\mathrm{BH}_k(\mathrm{C_{=}P}^A)$, and for all odd k, $L_k(A) \notin \mathrm{BH}_k(\mathrm{C_{=}P}^A)$. This proves

Theorem 23 *There exists an oracle A such that for odd k, $\mathrm{BH}_k(\mathrm{NP}^A) \not\subseteq \mathrm{BH}_k(\mathrm{C_{=}P}^A)$ and for even k, $\mathrm{BH}_k(\mathrm{NP}^A) \not\subseteq$ co-$\mathrm{BH}_k(\mathrm{C_{=}P}^A)$.*

Then using theorem 7, we have,

Corollary 24 *There exists an oracle A such that $\mathrm{P}^{\mathrm{NP}^A}_{(k+1)-\mathrm{T}} \not\subseteq \mathrm{P}^{\mathrm{C_=P}^A}_{k-\mathrm{T}}$.*

7 Open Problems

Oracle separations are naturally not satisfying in these times of techniques that do not relativize (see, e.g., [1] and references therein), and thus the results of this paper raise more questions than they answer. It would be interesting to see sharper results along the lines of Theorem 8, in particular using techniques which exploit *differences* rather than similarities between NP and $\mathrm{C_{\neq}P}$ (indeed, it is possible that Theorem 8 is trivial since it is possible with our current state of knowledge that $\mathrm{P^{PP}}$ =PSPACE). Similarly, some plausible separation of $\mathrm{P^{C_=P}}$ from $\mathrm{P^{PP}}$ without oracles would be very interesting: Does the hypothesis $\mathrm{P^{C_=P}} = \mathrm{P^{PP}}$ have any dramatic consequences?

Acknowledgements

I wish to thank the members of the Departament de Llenguatges i Sistemes Informàtics, UPC Barcelona, where this work was done, for their hospitality, with special thanks to José Balcázar for helping to make the visit possible and for valuable suggestions and comments on the paper. I thank Jacobo Torán for suggestions as well as many discussions on this subject, and Mitsunori Ogiwara and Seinosuke Toda for sharing their results with me. Conversations with Antoni Lozano and Gerd Wechsung are also gratefully acknowledged. I am grateful to two anonymous referees who pointed out some errors in the original version of this paper and made many suggestions which greatly improved the presentation.

References

[1] L. Babai, "E-Mail and the Unexpected Power of Interaction", *5th Annual Conference on Structure in Complexity Theory* (1990) 30-44.

[2] A. Bertoni, D. Bruschi, D. Joseqh, M. Sitharam, and P. Young, "Generalized Boolean Hierarchies and Boolean Hierarchies over RP", *Proceedings of the 7th Conference on Fundamentals of Computation Theory*, Springer-Verlag 1989, Lecture Notes in Computer Science 380.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity Theory I*, Volume II of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, New York, 1988.

[4] R. Beigel, "Bounded Queries to SAT and the Boolean Hierarchy", Johns Hopkins Tech Report 87-8 (1988), to appear in Theoretical Computer Science.

[5] R. Beigel, R. Chang, and M. Ogiwara, "A Relationship between Difference Hierarchies and Relativized Polynomial Hierarchies", manuscript, January, 1991.

[6] R. Beigel, N. Reingold and D. Spielman, "PP is Closed Under Intersection", STOC 1991.

[7] S. R. Buss and L. Hay, "On Truth-table Reducibility to SAT and the Difference Hierarchy Over NP", *Proceedings of the 3rd Conference on Structure in Complexity Theory* (1988) 224-233.

[8] J.-y. Cai "Probability One Separation of the Boolean Hierarchy", *4th Annual Symposium on Theoretical Aspects of Computer Science*, Springer-Verlag Lecture Notes in Computer Science 247 (1987) 148-158.

[9] J.-y. Cai, T. Gunderman, J. Hartmanis, L. A. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, "The Boolean Hierarchy I: Structural Properties", *SIAM Journal of Computing*, **17** (1988) 1232-1252.

[10] L. A. Hemachandra, "The Strong Exponential Hierarchy Collapses", *Journal of Computer and System Science* (1989) 299-322.

[11] R. Chang and J. Kadin, "The Boolean Hierarchy and the Polynomial Hierarchy: a Closer Connection", *5th Annual Conference on Structure in Complexity Theory* (1990) 169-178.

[12] T. Gundermann, N. A. Nasser and G. Wechsung, "A Survey on Counting Classes", *5th Annual Conference on Structure in Complexity Theory* (1990) 140-153.

[13] J. Kadin, "Restricted Turing Reducibilities and the Structure of the Polynomial Time Hierarchy", Ph.D. thesis, Cornell University, February 1988.

[14] C. Lautenmann, "BPP and the Polynomial Hierarchy", *Information Processing Letters* **17** (1983) 215-217.

[15] M. Ogiwara, "Generealized Theorems on Relationships Among Reducibility Notions to Certain Complexity Classes", manuscript, April 1991.

[16] J. Tarui, "Randomized Polynomials, Threshold Circuits, and the Polynomial Hierarchy", *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science* (1991) 238-250.

[17] J. Tarui, "Degree Complexity of Boolean Functions and Its Applications to Relativized Separations", to appear in *6th Annual Conference on Structure in Complexity Theory* (1991).

[18] S. Toda, "On the computational power of PP and $\oplus$P", *Proceedings 30th IEEE Symposium on Foundations of Computer Science* (1989) 514-519.

[19] S. Toda, "On Polynomial-Time Truth-Table Reducibility to $C_=P$ Sets", Colloquium, Department of Computer Science, University of Chicago, October 26, 1990.

[20] S. Toda and M. Ogiwara, "Counting Classes Are as Hard as the Polynomial- Time Hierarchy", to appear in *6th Annual Conference on Structure in Complexity Theory* (1991).

[21] J. Torán, "Structural Properties of the Counting Hierarchy", Ph.D. thesis, Facultat d'Informatica de Barcelona, 1988.

[22] J. Torán, "An Oracle Characterization of the Counting Hierarchy", *3rd Annual Conference on Structure in Complexity Theory* (1988) 213 - 223.

[23] K. Wagner, "Compact Descriptions and the Counting Polynomial Time Hierarchy", *Acta Informatica* **23** (1986) 325-356.

[24] K. Wagner, "Bounded Query Classes", *SIAM Journal of Computing* **19** (1990) 833-846.