GENERALIZED BOOLEAN HIERARCHIES AND BOOLEAN HIERARCHIES OVER RP[1]

ALBERTO BERTONI – Università degli Studi di Milano

DANILO BRUSCHI – Università degli Studi di Milano and University of Wisconsin

DEBORAH JOSEPH – University of Wisconsin

MEERA SITHARAM – University of Wisconsin

PAUL YOUNG – University of Washington and University of Wisconsin

1. INTRODUCTION.

In this paper we study the complexity hierarchy formed by taking the Boolean closure of sets in RP, the class of sets decidable in random polynomial time. This hierarchy, which we call the *Boolean hierarchy over RP* and denote by RBH, is analogous to the difference hierarchy for *r.e.* sets studied by Ershŏv ([Er 68a 68b 69]) and to the Boolean hierarchy over NP studied by ([BuHa 88], [CGHHSWW 88], [CGHHSWW 89], [CaHe 86], [Ka 88], [KöSchWa 87], [PaYa 84], [Wa 86 87a 87b 88], [WeWa 85]). RBH lies above RP and below BPP. It is of particular interest because so little is known about sets that might be in $BPP - RP$. In fact, since Adleman and Huang ([AdHu 87]) showed that primality testing lies in RP, (and hence in ZPP), there have been no natural candidates for the class $BPP - RP$. Thus, the examples we give for the Boolean hierarchy over RP should help renew the belief that the classes BPP and RP are truly different.

In Section 3 of this paper we give a uniform proof of the following metatheorem:

- Boolean hierarchies over arbitrary complexity classes satisfy the same definitional equivalences as the Boolean hierarchy over NP. Moreover, many of the properties of polynomial time machines that make a constant number of queries to an oracle in NP (adaptively and non-adaptively) generalize to machines that query an oracle from an arbitrary complexity class, C. For instance, C could be Random Polynomial time, (RP), *FewP*, Uniform *RNC*, the nondeterministic exponential time classes $NEXP^{linear}$ and $NEXP^{poly}$, or the recursively enumerable sets, (RE).

We establish this very general theorem in Section 3 by generalizing many of the theorems for the Boolean hierarchy over NP to Boolean hierarchies over fairly arbitrary complexity classes C. In doing so, we give a careful treatment of *extended* Boolean hierarchies over fairly arbitrary classes. Some of the proofs for the hierarchy over NP generalize trivially. However, some of the proofs over NP use the existence of complete sets or other special properties of NP, and these proofs require modification for classes such as RP.

Knowing that Boolean hierarchies over most complexity classes have similar definitional invariants and relationships to bounded query classes, we begin our studies of the Boolean hierarchy over RP in Section 2 with several natural examples of sets in the hierarchy:

---

[1] This work was supported in part by the Ministero della Pubblica Istruzione, through "Progetto 40%: Algoritmi e Strutture di Calcolo," the National Science Foundation under grant DCR-8402375, and by the Wisconsin Alumni Research Foundation under a Brittingham Visiting Professorship.

The last four authors' 1988-89 address is: Computer Sciences Department, University of Wisconsin, 1210 West Dayton St., Madison, WI 53706, U.S.A.

- Fermat showed that every integer can be represented as the sum of at most four integer squares. From recent work of Bach, Miller, Rabin, and Shallit ([BaMiSh 86], [RaSh 88]), it follows that the set $\{n : n$ is perfect and is the sum of two integer squares$\}$ is also in *RP,* and that the set $\{n : n$ is perfect and is the sum of three integer squares$\}$ is in RP. Combining these results, we can see that the set $\{n : n$ is perfect and is the sum of three integer squares but not a sum of two integer squares$\}$ is in the second level of the Boolean hierarchy over RP. Similarly, $\{n : n$ is perfect and is the sum of four integer squares but not a sum of three integer squares, or n is perfect and is the sum of two integer squares$\}$ is in the third level of the Boolean hierarchy over RP.
- Schwartz proved that checking whether a polynomial p represented as a straight line program is identically zero is in *coRP,* ([Sch 80]). We show that this problem is equivalent to checking whether two straight line programs compute the same natural number, and thus the latter problem is also in *coRP.* However, the problem of deciding whether the polynomial represented by a straight line program is a monomial seems not to be in either RP or coRP, although we *can* show that it is in the second level of the Boolean hierarchy over RP. In fact, we show that if this latter problem is in RP or *coRP,* then the problem of checking whether straight line programs are identically zero and the problem of checking whether two straight line programs compute the same natural number are both in ZPP.

In Section 4 we briefly explore structural properties of the Boolean hierarchy over RP and how truth-table reducibility from self-reducible sets in NP together with membership in BPP can force sets from the Boolean hierarchy over NP into the Boolean hierarchy over RP.

One would like to prove that the Boolean hierarchy over RP is a proper hierarchy and satisfies the set containments "which it should" with respect to the polynomial time hierarchy, other probabilistic classes, and the Boolean hierarchy over NP. This goal is obviously too ambitious since it would prove $P \neq NP$, so in Section 5 we give various oracle constructions including:

- There is an oracle, X, relative to which the obvious proper containments which one *expects* to hold for the regular and extended Boolean hierarchies over RP and for RP *vs* NP do hold with respect to X.

This at least shows that no simple proof should show that the Boolean hierarchy over RP does not behave as we would "expect" it to behave.

Sections 3, 4 and 5 have been written so that they are largely independent. Thus, readers should have no trouble reading these sections in any order.

2. BOOLEAN HIERARCHIES AND THE BOOLEAN HIERARCHY OVER RP.

In this section, we focus our attention on the Boolean hierarchy over RP. We begin with a discussion of basic definitions and properties of Boolean hierarchies followed by some examples of sets which are in the Boolean hierarchy over RP. Later, in Section 4, we will discuss characterizations of the Boolean hierarchy over RP using machine-acceptors and probabilistic quantifiers, and investigate some of the structural properties of the hierarchy.

Boolean hierarchies are typically formed by taking classes of sets which are closed under union and intersection but not complementation, and then forming the Boolean closure of the class by iterating closure under operations involving complementation. We start with basic definitions. Where possible, we use the notation from Wagner's survey, [Wa 88a].

Definition 2.0.1. *For each k, let h'_k be a k-ary Boolean function. For any set Y, let c_Y denote the characteristic function of Y. Then for any collection of sets C*

$$h'_k[C] \quad =_{def} \quad \{\ X : \exists X_1, \ldots, X_k \in C \ \ [c_X(x) \ = \ h'_k(c_{X_1}(x), \ldots, c_{X_k}(x))\ ]\ \}. \qquad (*)$$

That is, each set in $h'_k[C]$ is a specific Boolean combination, h'_k, of the finite collections of sets taken k at time from the class C. Regular Boolean hierarchies are typically obtained by taking some reasonable collection of Boolean functions, $\{h'_k,\}$ which are uniformly specifiable from k, and which are so chosen that the containments

$$h'_1[C] \ \subseteq \ h'_2[C] \ \subseteq \ \ldots \ \subseteq \ h'_i[C] \ \ldots$$

are obvious. True hierarchies are obtained when the containments turn out to be proper. The following are some of the standard choices for h'_k.

Definition 2.0.2.

(i) The **Difference hierarchy** was defined over NP by Köbler and Schöning ([Kö 85], [KöScWa 87]), and was denoted by them as $Diff_k \ =_{def} \ f_k[NP]$, where

$$f_k(x_1, \ldots, x_k) \ = \ x_1 \oplus \ldots \oplus x_k \qquad \text{(parity)}.$$

(ii) The **Wechsung-Wagner hierarchy** over NP as defined by Wechsung and Wagner ([WeWa 85]) was denoted by them as $g_k[NP]$, where

$$g_k(x_1, \ldots, x_k) \ = \ (\neg x_1 \wedge x_2) \vee \ldots \vee (\neg x_{k-1} \wedge x_k)$$

for k even, and

$$g_k(x_1, \ldots, x_k) \ = \ (\neg x_1 \wedge x_2) \vee \ldots \vee (\neg x_{k-2} \wedge x_{k-1}) \vee \neg x_k$$

for k odd.

(iii) The **Hausdorff hierarchy** $g'_k[C]$, is $g_k[C]$ with the additional requirement that for all i, j such that $j > i$, $x_i = 1 \Rightarrow x_j = 1$. I.e.,

$$g'_k \ = \ g_k \bigwedge_{1 \le j \le k} x_j \to x_{j+1}.$$

Hausdorff ([Ha 78]) showed that $g'_k[D] = g_k[D]$ for all classes D which are closed under union and intersection.

(iv) The **Boolean hierarchy** as defined over NP by Cai and Hemachandra ([CaHe 86]) was denoted by them as $NP[k] \ =_{def} \ h_k[NP]$, where

$$h_k(x_1, \ldots, x_k) \ = \ (\ldots((x_1 \wedge \neg x_2) \vee x_3) \ldots) \vee x_{k-1}) \wedge \neg x_k$$

for k even, and

$$h_k(x_1, \ldots, x_k) \ = \ (\ldots((x_1 \wedge \neg x_2) \vee x_3) \ldots) \wedge \neg x_{k-1}) \vee x_k$$

for k odd.

In each case, these sequences of truth-tables can be used to define hierarchies over arbitrary base classes in addition to NP. It is known, (see e.g. Corollary 3.21 and Definition 3.22) that the four collections of Boolean functions defined in (i) - (iv) generate the same Boolean hierarchy over NP. Thus the notation $NP[k]$ is standard for all four definitions. Our primary goal in this paper is to study these classes when the base class is RP.

2.1 THE BOOLEAN HIERARCHY OVER RP

Definition 2.1.1. A set S is in *RP if there is a nondeterministic polynomial time Turing machine, M, and a positive fraction $1/c$ such that*

$$x \in S \implies M(x) \text{ accepts on at least } 1/c \text{ of its paths}$$
$$x \notin S \implies M(x) \text{ rejects on all of its paths.}$$

The machine M is called an *RP-acceptor* for S, and its acceptance criterion is said to have *one-sided* error probability. The complement class of RP is denoted $coRP$, and the intersection of RP and $coRP$ is the class ZPP. Obviously from the above definition, RP is contained in NP.

The class BPP, or Bounded error Probabilistic Polynomial time, is given by

Definition 2.1.2. *$S \in BPP$ if there exists a nondeterministic polynomial time Turing machine, M, and positive fractions $1/c, 1/d$, with $(1-(1/d)) < 1/c$, such that*

$$x \in S \Rightarrow M(x) \text{ accepts on at least } 1/c \text{ of its paths}$$
$$x \notin S \Rightarrow M(x) \text{ rejects on at least } 1/d \text{ of its paths.}$$

The machine M is called a *BPP-acceptor* for S, and its acceptance criterion is said to have a *two-sided* error probability. Notice from this definition that BPP is closed under complements, and that RP and $coRP$ are subsets of BPP. Following work of Sipser, Gacs and Lautemann have independently shown that BPP is contained in the $\Sigma_2^P \cap \Pi_2^P$ level of the Meyer-Stockmeyer hierarchy, ([La 83], [Si 83]).

Ko ([Ko 82]) and Zachos ([Za 86]) introduced a polynomial time hierarchy over RP (denoted RH) which is analogous to the Meyer-Stockmeyer hierarchy over NP. Zachos showed that the entire random polynomial time hierarchy, RH, is contained in BPP. Recalling our motivation for studying the structure of $BPP - RP$, it should be noted that there are no known natural candidates for sets in $\Sigma_2^{RP} - RP$. This is partly because Σ_2^{RP}-acceptors have awkward acceptance criteria which natural problems do not seem to satisfy. In fact, Zachos and Heller have attempted to prove that RH collapses to $\Sigma_2^{RP} \cap \Pi_2^{RP}$ $(= \Delta_2^{RP})$.

The Boolean hierarchy over RP can now be defined using any of the standard formulas from Definition 2.0.2. In Section 3 we will show that the definition of the regular Boolean hierarchy over RP remains invariant if one uses any of the standard Boolean functions given in Definition 2.0.2. This justifies the following definition.

Definition 2.1.3. *Let h'_k be any one of the standard Boolean functions from Definition 2.0.2. Define*

$$RP[k] = h'_k[RP] = \{X : \exists X_1, \ldots, X_k \in RP \ [x \in X \iff h'_k(c_{X_1}, (x) \ldots, c_{X_k}(x))]\}.$$

The complement class of $h'_k[RP]$ will be denoted as $coh'_k[RP]$.

2.2 EXAMPLES OF LANGUAGES IN RBH.

Here we give natural examples of languages which lie in the Boolean hierarchy over RP. These languages are of particular interest because they represent natural problems which are in BPP but do not seem to lie in RP.

In [SoSt 77], Solovay and Strassen showed that primality testing is in $coRP$. Recently Adleman and Huang, ([AdHu 87]), have shown that primality testing is also in RP, and hence primality testing is now known to be in ZPP. As a consequence problems such as determining whether a number is perfect and determining whether a number is a Carmichael number that were previously conjectured to be in $BPP - RP$ are now known to be in RP. However, we can use the fact that the set of perfect numbers is in RP to give examples of problems that are in the Boolean hierarchy over RP but seem not to be in RP.

Our first examples use Fermat's well-known result that every integer can be represented as the sum of at most four integer (possibly zero) squares. Combining Fermat's result with recent work of Bach, Miller, Rabin, and Shallit ([BaMiSh 86], [RaSh 88]) shows that the set $\{n : n$ is perfect and is the sum of two integer squares$\}$ is in RP and that the set $\{n : n$ is perfect and is the sum of three integer squares$\}$ is in RP. These results together give that

(i) $\{n : n$ is perfect and is the sum of three integer squares but is not a sum of two integer squares $\}$ is in $RP[2]$.

(ii) $\{n : n$ is perfect and is the sum of four integer squares but is not a sum of three integer squares $\}$ $\cup$ $\{n : n$ is perfect and is the sum of two integer squares $\}$ is in $RP[3]$.

Currently there are no known techniques that place either of these two sets in RP. [2]

A second, and rather different, collection of problems involving polynomials that are represented by straight line programs can also be shown to be in RBH. Schwartz has shown that checking whether a polynomial p, represented as a straight line program, is identically zero is a $coRP$ problem ([Sch 80]). More formally,

Definition 2.2.1. *A straight line program Φ is an ordered sequence of instructions $I_1, \cdots, I_n$ such that an individual instruction I_k has one of the following forms:*

$x_k \leftarrow x_j + x_s$; $x_k \leftarrow x_j - x_s$; $x_k \leftarrow x_j x_s$; $x_k \leftarrow 0$; $x_k \leftarrow 1$;

where x_k, x_j, x_s are indexed variables and $j, s < k$.

For a straight line program Φ, we will let x_n denote the variable with greatest index contained in Φ, let N_Φ denote the number contained in the variable x_n at the end of the computation of Φ, and let $l(\Phi)$ denote the number of instructions of Φ. If we allow instructions of the form: $x \leftarrow z$; where z is a formal variable, then Φ describes a polynomial $P_\Phi(z)$ of degree at most $2^{l(\Phi)}$.

With this notation we can now formally describe our next set of problems.

Problem 2.2.2. (Schwartz).
Instance : A straight line program, Φ_1, with a formal variable.
Question: Is P_{Φ_1} identically null?

Problem 2.2.3.
Instance : Two straight line programs, Φ_1 *and* Φ_2.
Question: Is N_{Φ_1} equal to N_{Φ_2}?

As mentioned above, Schwartz has shown that Problem 2.2.2 is in $coRP$. Obviously, Problem 2.2.3 can be polynomially many-one reduced to Problem 2.2.2, so it too is in $coRP$. A somewhat less obvious proof shows that Problem 2.2.2 can be reduced to Problem 2.2.3.

Next we consider a problem involving straight line programs that is in RBH, but seems not to be in RP or $coRP$. To state the problem, for a polynomial $p(z) = \sum_{k=0}^{n} a_k z^k$, we will denote the coefficient of z^k in $p(z)$ by $[z^k]p(z)$.

Problem 2.2.4.
Instance : A straight line program, Φ, with a formal variable.
Question: Does Φ represent a monomial? I.e., does there exist exactly one k such that $[z^k]P_\Phi$ is not equal to 0?

Using the following lemma we can show that Problem 2.2.4 is in $RP[2]$.

Lemma 2.2.5. *For each continuous function p, the following two sentences are equivalent:*
(i) $\exists\alpha\exists\beta[p(x) = \alpha x^\beta$ *and* $\alpha \neq 0]$, *and*
(ii) $\forall x[p(1)p(x^2) = p^2(x)$ *and* $p(1) \neq 0]$.

Now given a straight line program Φ which represents a polynomial $P_\Phi(z)$, to verify whether $P_\Phi(z)$ is a monomial we do the following:

---

[2] Of course, we should also point out that it is not known that there are infinitely many perfect numbers (although this is widely believed to be the case), and even if there are infinitely many perfect numbers, either of the sets described above could be empty, although many number theorists believe that this is unlikely.

(i) Compute a straight line program Φ_1 such that $N_{\Phi_1} = P_\Phi(1)$;
(ii) Compute a straight line program Φ_2 such that $P_{\Phi_2}(x) = N_{\Phi_1} P_\Phi(z^2) - P_\Phi^2(z)$;
(iii) If $N_{\Phi_1} \neq 0$ and $P_{\Phi_2} = 0$, then accept; else reject.

Given Φ, Steps (i) and (ii) can be computed in polynomial time. Step (iii) can be computed with one query to an RP oracle and with one independent query to a coRP oracle. Thus Problem 2.2.4 is in $RP[2]$.

Next we give some evidence that it is unlikely that Problem 2.2.4 is either in RP or in $coRP$.

Theorem 2.2.6. *If Problem 2.2.4 is in $RP \cup coRP$, then Problems 2.2.2 and 2.2.3 are each in ZPP.*

Proof: To prove this it is adequate to show that if Problem 2.2.4 is in RP (or in coRP), then Problem 2.2.3 and its complement are in RP (and in coRP). Notice that verifying that the number N_Φ, represented by the straight line program Φ, is not equal to zero is equivalent to verifying that N_Φ is a monomial. In fact: $N_\Phi \neq 0$ if and only if N_Φ is a monomial, since every integer greater than zero is a monomial.

On the other hand, verifying that N_Φ is equal to zero is equivalent to verifying that the program $\Phi_1 = 1 + \Phi * z$ is a monomial (since Φ_1 is a monomial only if $\Phi * z$ is null). In fact: P_{Φ_1} is a monomial *if and only if* $N_\Phi = 0$.

This shows Problem 2.2.3 and its complement are in RP, and thus Problem 2.2.3 is in ZPP. The same reasoning is applied for the case in which Problem 2.2.4 can be solved with a coRP algorithm. ∎

We should also comment that by generalizing Problem 2.2.2 to multivariate polynomials we have obtained problems that lie in higher levels of the Boolean hierarchy over RP.

3. BASIC PROPERTIES OF GENERALIZED BOOLEAN HIERARCHIES.

As mentioned in Section 2, Boolean hierarchies are typically formed by taking classes of sets which are closed under union and intersection but not complementation, and then forming the Boolean closure of the class by iterating closure under operations involving complementation.

As early examples, Boolean hierarchies were studied by Hausdorff in 1914 in the context of descriptive set theory ([Ha 78]), and the Boolean hierarchy over the recursively enumerable sets was defined and studied by Eršov ([Er 68a], [Er 68b], [Er 69]) who in [Er 68b] noted the connections between its transfinite extension through the recursive ordinals and the work done by Putnam, [Pu 65], on "mind changes" of recursive sequences. Together, the work of Putnam on "mind changes" of recursive sequences and of Eršov on extensions of Boolean hierarchies through recursive ordinals shows that the use of either truth-tables or "mind-changes" provides characterizations of the class $\Sigma_2^0 \cap \Pi_2^0$ in the arithmetic hierarchy. Independent of the work of Putnam and Eršov, Boolean hierarchies were also studied by Addison in the context of their applications in logic, ([Ad 65]).

Boolean hierarchies over NP have been defined in various ways, for example, by Papadimitriou and Yanakakis ([PaYa 84]), by Wechsung and Wagner ([WeWa 85]), by Cai and Hemachandra ([CaHe 86]), and by Köbler and Schöning ([Kö 85], [KöSchWa 87]). Most of the results of these latter papers are contained in journal form in [CGHHSWW 88] and [CGHHSWW 89]. For regular Boolean hierarchies over NP, the various definitions used by these authors have all been proven equivalent ([KöSchWa 87]). In [Wa 88], Wagner surveys these equivalences for the regular Boolean hierarchy over NP. He also gives a definition of an extended Boolean hierarchy over NP. In [Wa 87a], Wagner proves the equivalence of these various definitions for the extended Boolean hierarchy over NP. Although our primary interest is in the Boolean hierarchy over RP, in this section we build on this earlier work to show how standard properties of Boolean hierarchies and extended Boolean hierarchies can be generalized to give extended Boolean hierarchies over quite general complexity classes.

Thus, in this section we lay the foundations for discussing both regular and extended Boolean hierarchies over quite general complexity classes, and over RP in particular. Our goal here is to instantiate, as thoroughly as possible, the following general observation:

Observation 3.1. *All regular Boolean hierarchies over fairly arbitrary complexity classes, C, satisfy the same definitional equivalences as the regular Boolean hierarchy over NP. Furthermore, at least at lower levels, all extended Boolean hierarchies satisfy these same definitional equivalences. Moreover, general proofs of these equivalences show that many of the properties that relate the number of parallel and sequential calls to an oracle in NP follow from uniform constructions which apply to machines that query an oracle from these quite general complexity classes. Classes, C, for which these results apply include not only classes like NP and RP, but also such classes as Uniform RNC, FewP, the nondeterministic exponential time classes $NEXP^{linear}$ and $NEXP^{poly}$, the recursively enumerable sets, standard classes in the polynomial time hierarchy, standard classes in the arithmetic hierarchy, etc.*

To justify this observation, we will give generalizations of many of the results that have been proved for Boolean hierarchies and for extended Boolean hierarchies over NP. Wagner's survey paper ([Wa 88]) contains an excellent review of such results over NP, including proofs, and, as mentioned earlier, some of these proofs generalize trivially. Our development here relies heavily on Wagner's survey paper. However, many of the proofs Wagner gives use properties that are specific to NP, such as nondeterministic "guessing" and the existence of complete sets for NP. Such proofs obviously require modification.

The various basic definitions for building regular Boolean hierarchies were given in Section 2. Definitions 2.0.1 and 2.0.2 are critical to an understanding of this section and we ask the reader to review them here.

Each of the four classes of Boolean formulas used in Definition 2.0.2 can be used to define a Boolean hierarchy over *arbitrary* base classes, in addition to hierarchies over NP and over RP. For Boolean hierarchies over NP these various definitions have all been shown to be equivalent, ([KöSchWa 87], [Wa 87a]). As we shall see, Wagner ([Wa 88]) has used the proof of these equivalences to find a "normal form" for the ordinary Boolean hierarchy over NP, and he has used this normal form as a basis for a definition of an extended Boolean hierarchy over NP. However, from our point of view, following a suggestion of Cai and Hemachandra, a more natural way to define an extended Boolean hierarchy might be to take any of the above families of functions $\{h'_k\}$ and then build classes above any of the finite levels of the Boolean hierarchy by taking (slowly growing) functions which majorize any of the finite levels of the hierarchy. Just as one defines $h'_k[C]$ using definition (*) to obtain finite levels of the Boolean hierarchy, one might hope to define higher levels of an extended Boolean hierarchy — for example, by defining classes like

$$h'_{log}[C] \quad =_{def} \quad \{\ X : \exists X_1, \ldots, X_{log(|x|)} \in C \ \ [c_X(x) \ = \ h'_{log}(c_{X_1}(x), \ldots, c_{X_{log(|x|)}}(x))\ ]\ \}. \qquad (**)$$

Notice that for finite levels of the Boolean hierarchy, one builds the k^{th} level of the Boolean hierarchy by using k *fixed* sets $X_1, \ldots, X_k$ from the base level to build each set at the k^{th} level. However, to make an extension beyond finite levels, one must somehow make sense of what it means to use *infinitely* many sets $X_1, \ldots, X_{log(|x|)}$ from the base level to build each set at an extended level in the definition (**). To do so is very much in the spirit of the approach suggested by Cai and Hemachandra. However, when working over classes like NP which have complete sets, one can get the sequence of sets to substitute into (**) by using a uniform reduction of the sequence to a complete set. For example, Cai and Hemachandra have suggested using sets uniformly reducible to SAT for the substitutions, X_i.

When complete sets do not exist, one clearly needs some (efficient) representation of the base sets $X_1, \ldots, X_{log(|x|)}$ which are chosen from the base class C when defining the set X in (**). That is, the representation of the X_i must be given in terms of some *a priori* representation, or indexing, of the sets in the base class, C.

Preliminary Definitions and Detailed Notations.

In building extended Boolean hierarchies over arbitrary complexity classes we obviously will also build finite Boolean hierarchies, but notational issues which are obvious for the finite case become more delicate in the extended case. In this section we review definitions and explain the notational conventions which we will use.

We begin by observing that a *finite,* or *concrete* truth-table is one which is syntactically given in some nice canonical form, for example, like the Boolean formulas of Definition 2.0.2. We leave the reader to supply their own canonical definition of finite truth-tables. Note that there can be enough flexibility in how one defines a concrete "truth-table" that for the purposes of this paper one could just as well mean "Boolean circuit" and all results presented here would still go through. We note that if T is a syntactically given truth-table with length $|T|$, then NV_T, which is the number of (distinct) variables appearing in T must satisfy $NV_T \leq |T|$.

For our purposes, we will be interested, not just in the forms for truth-tables, but also in various *orderings* of the variables of the truth tables. A *concrete ordering* of a finite truth-table with k variables is just a nice, syntactically given, permutation of the set $\{1, 2, \ldots, k\}$. Again, we leave the reader to supply their own definition, but we observe that any permutation of $\{1, 2, \ldots, k\}$ can always be realized in a nice syntactic way with a complete table of (bit) length roughly $k * log(k)$, (although no such nice polynomial bound exists for truth-tables with k variables).

For any order O of the variables $x_1, x_2, \ldots, x_k$ and for any value $1 \leq i \leq k$, we define $Ord_O(i) =_{def} O^{-1}(i)$; thus $Ord_O(i)$ is intuitively just the "position" of the variable x_i in the ordering of the variables under the order O. Orders will be used to define "mind-changes" of truth-tables, which in turn are used as a means to find homeomorphisms between truth-tables. Such homeomorphisms are necessary to establish uniqueness and equivalence of the classes based on these truth-tables.

Definition 3.2. *Let T be any finite truth-table with k variables and associated ordering O of $\{1, 2, \ldots, k\}$. We denote the length k vectors of all zeroes and of all ones by $\vec{0}_k$ and $\vec{1}_k$, respectively. We denote the function which counts the number of one's in a Boolean vector, $\vec{b}$, by $\#_1(\vec{b})$. Suppose that $\vec{a} = \langle a_1, \ldots, a_k \rangle$ and $\vec{b} = \langle b_1, \ldots, b_k \rangle$ are Boolean vectors with all $a_j \leq b_j$. We say that T changes its mind from $\vec{a}$ to $\vec{b}$ if $T(\vec{a}) \neq T(\vec{b})$. In addition we let $\mathbf{MC_{T,O}}(\vec{\mathbf{b}}, \mathbf{i})$ denote the number of mind changes of T which have been obtained just after i bits are flipped from 0 to 1 when transforming $\vec{0}$ to $\vec{b}$, where the transformation is accomplished one bit at a time, always flipping a "0" to a "1" in the order specified by the ordering O. We frequently abbreviate $MC_{T,O}(\vec{b}, \#_1(\vec{b}))$ to simply $MC_{T,O}(\vec{b})$. By convention, for $j \geq \#_1(\vec{b})$, we define $\mathbf{MC_{T,O}}(\vec{\mathbf{b}}, \mathbf{j})$ to be simply $\mathbf{MC_{T,O}}(\vec{\mathbf{b}}, \#_\mathbf{1}(\vec{\mathbf{b}}))$. Perhaps the most important thing to notice about mind changes is that for any order O*

$$T(b_1, \ldots, b_k) = T(0, \ldots, 0_k) + MC_{T,O}(\vec{b}) \mod (2).$$

Since in calculating $MC_{T,O}(\vec{b})$, we always are interested in the pairs $\langle O, \vec{b} \rangle$, it will be useful to combine such pairs as follows:

Definition 3.3. *A concrete partial ordering is a function τ which maps some number, j, of elements of $\{1, 2, \ldots, k\}$ to 0 and the remaining $k - j$ elements of $\{1, 2, \ldots, k\}$ in a one-one fashion to $\{1, 2, \ldots, k - j\}$. (A concrete ordering is then just the special case in which $j = 0$.) For any concrete partial ordering τ we denote by $\vec{a}_\tau$ that Boolean vector which has its j^{th} bit, $\vec{a}_{\tau,j}$, equal to 0 if $\tau(j) = 0$ and $\vec{a}_{\tau,j}$ equal to 1 if $\tau(j) > 0$. We shall use the notation $MC_{T,\tau}(\vec{a}_\tau)$ in the obvious manner. That is, $MC_{T,\tau}(\vec{a}_\tau)$ is the number of mind changes in going from $\vec{0}$ to $\vec{a}_\tau$ in the order determined by (the nonzero portion of) τ.*

It is not hard to see that there are at most $2^{k*log(k)}$ concrete partial orderings of $\{1, 2, \ldots, k\}$. For our purposes, these must be coded into strings or integers in some "nice" way, so that the encodings are

"easily" recognized. Equally important, we require of these codings that all of the concrete partial orderings of $\{1, 2, \ldots, k\}$ be coded as strings or integers less than $2^{k*log(k)}$. We leave details to the reader.

To build truth-tables other than finite, bounded-truth-tables, one needs some class of functions which are used to build concrete truth-tables of increasing length. Traditionally, one also needs to calculate values on which to evaluate the truth-tables. For example, a standard definition of *a polynomially computable truth-table* is a pair $\langle T, f\rangle$ of polynomially computable functions, T and f. On input x, T produces a concrete, finite, truth-table, T_x, of some number of variables. We denote this number of variables by $NV_T(x)$. The truth-table $\langle T, f\rangle$ is said to reduce the set A to the set B, $(A \leq_{tt} B)$, if for all x,

$$c_A(x) = T_x(c_B(f(x,1), c_B(f(x,2), \ldots, c_B(f(x, NV_T(x))).$$

In the remainder of this section, we wish to develop a *general* theory of Boolean hierarchies and of extended Boolean hierarchies. Thus, we wish to consider hierarchies obtained by unbounded truth-tables produced, not just by polynomially computable functions, but also by functions chosen from other classes of functions — for example, log space functions, or functions computable in NC, or functions computable in exponential time, or primitive recursive functions. Our results will thus apply to fairly general ways of producing functions and working with complexity classes. To achieve this, our functions will be chosen from some fairly arbitrary class, Q. The following definition explains what minimal sets of conditions this base class, Q, of functions must satisfy. For the moment, the reader may want to observe that the class Q might be the class of functions computable in polynomial time, but it might just as well be the class of functions computable in log space. Still smaller classes will also do, as will larger classes of functions.

Definition 3.4. *A minimal class of functions is any class of functions, Q, which:*
(i) is closed under substitution (composition);
(ii) contains a (numerical) successor function, which we denote by $+1$;
(iii) contains the constant function, $Z(x) = 0$;
(iv) contains the usual pairing functions $\langle x_1, \ldots, x_n\rangle$ and projection functions, $P_i^n(\langle x_1, \ldots, x_i, \ldots, x_n\rangle) = x_i$;
(v) contains the function $Test(x, y, z, w)$ which returns z if $x \leq y$ and returns w if $x > y$;
(vi) contains the characteristic function of the predicate which tests, of a string y and an input x, whether y represents a concrete partial ordering of $\{1, 2, \ldots, x\}$.

If Q is any minimal class of functions, a subset $B \subseteq Q$ is called bounded if $f \in B$ implies that $2^{f*log(f)}$ is bounded by some function in Q.

Note that in any example in which we explicitly take Q to be the set of all functions computable in polynomial time or in log space, if we wish we may simply choose

$$B = \{f : f \in Q \ \& \ \exists \text{ a polynomial } p \text{ such that } f(x) \leq p(|x|)\}.$$

In this case we'll also have $f \in Q$ and $|f(x) * log(f(x))| < (k * log(|x|))^2$, so that trivially $2^{f*log(f)}$ is bounded by a function in Q.

Definition 3.5. *Let Q be any minimal class of functions,*
(i) A Q-truth-table is a function $T \in Q$ which on input x, produces T_x, a concrete finite truth-table, $T_x(x_1, x_2, \ldots, x_{NV_T(x)})$ with variables $x_1, x_2, \ldots, x_{NV_T(x)}$. For T to be a Q-truth-table, we also require that
  (a) the function $\lambda x T_x(0, \ldots, 0_{NV_T(x)})$ be in the function class Q,
  (b) the function $\lambda x, \tau \ MC_{T_x,\tau}(\vec{a}_\tau)$, where the variable τ ranges over all concrete partial orderings of $\{1, 2, \ldots, NV_T(x)\}$, should be in the function class Q, and
  (c) the function $NV_T(x)$, which simply counts the number of variables in T_x be in the subclass B.

(ii) For sets A and D, we write $A \leq_{tt}^{Q} D$ and say that A is Q-truth-table reducible to D with $NV_T(x)$ (parallel or non-adaptive queries to D) if there exists a truth-table $T \in Q$ and a function $f \in Q$ such that

$$c_A(x) \;=\; T_x(f(x,1), f(x,2), \ldots, f(x, NV_T(x))).$$

In this case we sometimes write $A \leq_{\{NV_T(x)\}-tt}^{Q} D$.

Definition 3.6. *Let Q be any minimal class of functions.*

(i) We say that a set A is Q-bounded-search reducible to a set D and we write $A \leq_{bs}^{Q} D$, if c_A is in the smallest class of functions which contains the characteristic function c_D of D, contains all functions in Q, is closed under composition, and is closed under minimalizations which are bounded in output by functions in the class B.

(ii) If $A \leq_{bs}^{Q} D$ and if on input x the "program" which does the reduction evaluates c_D at most $n(x)$ times, we sometimes write $A \leq_{\{n(x)\}-bs}^{Q} D$.

We have already remarked that, for any reasonable representation of truth-tables, for any truth-table T, $NV_T(x) \leq |T_x|$, so that $|2^{NV_T(x)}| \leq |T_x| + 1$. In most reasonable complexity classes of functions, and certainly for those at least as strong as log space, if we can compute $f(x)$, then the only thing that stops us from computing $2^{f(x)}$, and hence placing 2^f in Q, is that $2^{f(x)}$ is too long to write out. The relations $|2^{NV_T(x)}| \leq |T_x| + 1$ and $T \in Q$, thus motivate the requirement that $NV_T \in B$. Furthermore, in any complexity class of functions at least as strong as log space, one can always, given a concrete truth-table, calculate both the number of variables in the truth-table and the value $T_x(0, \ldots, 0_{NV_T(x)})$. Thus, in classes of functions like log space which are a little bit more than minimal, the functions NV_T will *automatically* be in the bounded subclass B and the function $\lambda x T_x(0, \ldots, 0_{NV_T(x)})$ will *automatically* be in the class Q if the function T which produces the concrete truth-table is in the class Q. The situation with respect to the function $\lambda x, \tau\ MC_{T_x,\tau}(\vec{a}_\tau)$ is only slightly more difficult. The truth-table T_x must itself be produced by a log space calculation and $NV_T(x) \leq |T_x|$. Thus from the order τ we can successively produce the sequence of vector substitions into T_x and evaluate the truth-table as we do the substitions. Clearly the count of mind changes can be maintained within log space. Thus in reasonable complexity classes of functions, Q and certainly for those at least as strong as log space, the only necessary requirement for a truth-table to be a Q truth-table is that the function T_x be a function in Q; conditions (a), (b), and (c) are all superflous. However, as we shall see, the special truth-tables based on Definition 2.0.2, have such nice properties that the caluculation of "mind-changes" may be possible by *ad hoc* methods even when the underlying functional class Q is not strong enough to permit the successive substitutions just described within logspace.

In the remainder of this section, when we refer to a *truth-table* we *always* have in mind a Q-truth-table for some fixed minimal class of functions Q. When we mean to specify a concrete, finite, truth-table instead of an arbitrary Q-truth-table, we will *always* make this clear. Unless otherwise specified, the class Q with bounded subclass B may now be thought of as remaining fixed throughout this section.

Definition 3.7. *Let Q be any minimal class of functions, and let T be any Q-truth-table. Let O be a function which on input x produces a (concrete) permutation O_x of the set $\{1, 2, \ldots, NV_T(x)\}$. We extend the definition of the the function Ord_O preceding Definition 3.2 to the ordering function O by defining $Ord_O(x,i) =_{def} O_x^{-1}(i)$, which gives the "position" of the i^{th} variable of T_x in the ordering O_x. We similarly extend Definition 3.2 to define mind changes for unbounded Q-truth-tables by defining the mind-change function $MC_{T,O}$ by*

$$MC_{T,O}(x, \vec{b}, i) = MC_{T_x,O_x}(\vec{b}, i),$$

where the reader will recall that MC_{T_x,O_x} gives the number of mind changes of truth-table T_x in going from $\vec{0}_{NV_T(x)}$ to $\vec{b}$ by flipping i bits in the order dictated by the order O_x.

(i) We say that the ordering function O is a Q-ordering of the truth-table T if the functions $\lambda x, i\ Ord_O(x,i)$ and $\lambda x, i\ MC_{T,O}(x, \vec{1}, i)$ are in Q.

(ii) For any Q-truth-table T and associated Q-ordering O we define the "maximal mind changes" on input x as follows:

$$MaxMC_{T,O}(x) =_{def} max_{\{\vec{b}:\ |\vec{b}|\ =\ NV_T(x)\}}\{MC_{T,O}(x, \vec{b}, NV_T(x))\}.$$

Just as with the function NV_T, and the function $\lambda x, \tau\ MC_{T_x,\tau}(\vec{a}_\tau)$, required to be in Q for the definition of Q truth-tables, if we are dealing with a class Q of functions which is a little bit more than minimal, for example with a class containing all log space computable functions, if we can compute both T_x and O_x within the class Q, then it will *automatically* follow that both of the functions Ord_O and $MC_{T,O}$ are in Q. Furthermore, since $MC_{T,O} \leq NV_T$, $MC_{T,O}$ will automatically be in the subclass B if it is in Q. Thus in Definitions 3.5 and 3.7, the *explicit* assumptions that the functions NV_T, Ord_O, $T_x(0,\ldots,0_{NV_T(x)})$ and the various mind-change functions be in the classes B or Q are only needed for *very* minimal classes of functions Q. *For classes of functions Q with reasonable computational power and for any reasonable choice of the subclass B, there is no way to avoid having these functions be in Q or in B.*

With this background, let us return to the problem of building extended truth-tables based on any of the special families $\{h'_k\}$, $(k = 1,2,3,\ldots)$ of k-ary Boolean functions from Definition 2.0.2.

Example 3.8. (a) *Let $\lambda k\{h'_k\}$ be any of the sequences of k-ary truth-tables of Definition 2.0.2 and suppose that Q is any minimal class of functions powerful enough that, from the value (string) 2^k we can produce the formula h'_k by a function in the class Q. (For example, demanding that Q contain all log space computable functions is much more than adequate.) Let t be any integer valued function in the bounded subclass B. First observe that the number of variables in $h'_{t(x)}$ is simply $t(x)$ and that $h'_{t(x)}(\vec{0})$ is always identically 0. These functions are always in Q. Thus, h'_t will be a Q-truth-table provided that the function $\lambda x, \tau MC_{h'_t,\tau}(\vec{a}_\tau)$ is also in the class Q. For minimal function classes Q at least as large as log space, we have seen that this condition will always hold, but for very minimal function classes this may depend both on the particular choice of h' and the class Q. As the simplest example, for the case where h' is the parity function of Definition 2.0.2 (i), all that is necessary to make h'_t a Q-truth table is that the class Q be strong enough, given a concrete partial truth-table τ, to count the number of 1's in $\vec{a}_\tau$ since this gives the mind changes for the parity function.*

(b) *Now for any of these four choices of h' consider the identity ordering, $O_x(i) = i$. It is easy to see that*

$$MC_{h'_t,O}(x, \vec{1}, i) = i, \quad \text{for } i \leq t(x)$$

(and of course $MC_{h'_t,O}(x, \vec{1}, i) = t(x)$ for $i \geq t(x)$). Furthermore, $Ord_O(x,i) = i$. Thus the functions $\lambda x, i\ Ord_O(x,i)$ and $\lambda x, i\ MC_{T,O}(x, \vec{1}, i)$ are trivially in Q. Thus for any minimal collection of functions Q and any function $t \in B$ the identity ordering is trivially a Q-ordering for the Q-truth-table h'_t for any choice of h'_t.

In the following section, given *any* Q-truth-table T, we will explain how to define $T[C]$, the T^{th} level of the extended Boolean hierarchy over C. For different truth-tables T and T', we will be interested in knowing whether $T[C] \subseteq T'[C]$. For the case $C = NP$ where the class Q is taken to be all functions with finite range, the best published result we know on how mind changes affect hierarchy containment and uniqueness of classes is the following "max-max" theorem:

Theorem, ([Wagner 88, Thm 8.2 & Cor 8.3]). *Take C to be NP. Take Q to be all functions with a finite range. Then, if T and T' are Q-truth-tables such that*

$$max_{\{O \in Q\}}\{MaxMC_{T,O}(x)\} \quad < \quad max_{\{O \in Q\}}\{MaxMC_{T',O}(x)\},$$

then $T[C] \subseteq T'[C]$.

From work in [Wa 87a] a similar result clearly holds for extended hierarchies over NP.

Our general goal in the remainder of this section is to see how far we can generalize Wagner's theorem to extended hierarchies over arbitrary classes C and arbitrary minimal function classes Q.

Building Extended Boolean Hierarchies over Arbitrary Complexity Classes.

We begin with the observation that most *finite* Boolean hierarchies that are of interest are built from base classes (such as NP or RP) that are closed under finite intersection, union, and reduction by functions computable in polynomial time. In order to extend these notions, we will need to extend the finite closures under union and intersection to include closure under appropriate slowly growing, but infinite, unions and intersections. As discussed earlier, for classes that have complete sets, one can hope to simply code all of the sets which one hopes to substitute in some formal version of equations like (**) into the complete set, and then use only the complete set in defining extended hierarchies. But in the general case, making all suitable substitutions of sets into arbitrary (infinite) truth-tables requires some suitable *indexing* of the base class, C.

Definition 3.9. *An indexed collection of sets, $I : S \to C$, is a collection of sets C, together with a function I mapping some subset S of Σ^* onto C and a decomposition of C into $C = \bigcup C_k$, $C_k \subseteq C_{k+1}$. We use the notation X_j for the set $I(j)$ and the notation S^k for the set of indices $I^{-1}[C_k]$.*

Intuitively, for machine based complexity classes, the set S is normally chosen to be just some canonical set of machines which accept all of the members of C, and then X_j is just the set accepted by machine j.[3] C_k is normally just the collection of sets recognized by the machines in this machine class computing within a time bound of some fixed degree, for example of degree at most k. S^k is then just the machines which run within time bound of the same size. For example, for NP, S^k might be the set of machines running nondeterministically within time bound $|x|^k$. For another application, we might take S_k to be machines running in nondeterministic time $2^{k*|x|}$. When dealing with a class like the recursively enumerable sets where uniformity is not an issue, S_k might simply be *all* Turing machines for all k.

Definition 3.10. *Let Q be a minimal collection of functions with B a bounded subset of Q. Let $I : S \to C = \bigcup C_k$ be any indexed collection of sets. We say that C is fully closed under disjunctions (i.e., under unions) if for every k and every function $s \in Q$ with range $s \subseteq S^k$ and every function $t \in Q$ the set X defined by*

$$x \in X \iff \bigvee_{i \leq t(x)} x \in X_{s(i)} \qquad (***)$$

is also in C. We shall abuse this notation when the meaning is clear and write X as $\bigcup_{i \leq t(x)} X_{s(i)}$. We say that C is closed under bounded disjunctions (unions) if for every function $s \in Q$ with range $s \subseteq S^k$ and every function $t \in B$ the set X defined by (***) is also in C. Similar definitions hold for closure under full and bounded conjunctions (i.e., intersections). We will say that a sequence of sets, $\lambda i S_i$ is uniform in C if there is a function $s \in Q$ with range $s \subseteq S^k$ such that $S_i = X_{s(i)}$.

In the special cases where the class Q is the set of functions computable in polynomial time or in log space, we sometimes call full closure under disjunctions simply closure under *exponential* unions, and in this case we call bounded closure under disjunctions closure under *polynomial* unions.

[3] Note that, for base classes like RP, we can not reasonably demand that the set S of machines which define RP be easily decidable, or even decidable at all.

The proofs given below require both uniform closure of *sequences* under intersections and more flexibility in accessing the members of the basic sequence $\{\lambda i X_{s(i)}\}$.

Definition 3.11. *We say that an indexed collection of sets $C = \bigcup C_k$ is uniformly closed under bounded conjunctive reducibilities (from the class Q) if for all functions $f \in Q$, for all k and all functions $s \in Q$ with range $s \in S^k$, and for all functions $t \in B$, there exists a k' and a function $s' \in Q$ with range $s' \subseteq S^{k'}$ such that for all j*

$$x \in X_{s'(j)} \iff \bigwedge_{i \leq t(x)} f(x,i,j) \in X_{s(i)}.$$

We say that the sequence $\{X_{s'(j)}\}$ is uniformly reducible to the sequence $\{X_{s(j)}\}$, and, as defined in 3.10, we often simply say that the sequence $\{X_{s'(j)}\}$ is a uniform sequence in C.

Note that any class which is closed under bounded conjunctive reducibilities is trivially closed under bounded conjunctions.

Definition 3.12. *We say that an indexed collection of sets C which contains at least one non-trivial set*[4] *is uniformly closed if it is closed under both bounded unions and bounded conjunctive reducibilities.*

Example 3.13. *For each uniformly closed class Q in the following examples, the reader should choose a sensible bounded sublcass, B.*

(i) For both the function class Q of all functions computable in Uniform NC and the function class Q of all functions computable in log space, the base class Uniform RNC is uniformly closed.

(ii) For both the function class Q of all functions computable in polynomial time and the function class Q of all functions computable in log space, the base classes $C = NP$, $NEXP^{linear}$, $NEXP^{poly}$, RP, $FewP$, and RE all are uniformly closed. Furthermore, NP, $NEXP^{linear}$, $NEXP^{poly}$, and RE are also fully closed under (exponential) unions.

(iii) For the class Q of functions computable in EXP^{linear}, the base classes $NEXP^{linear}$, $NEXP^{poly}$, and RE are uniformly closed and are fully closed under unions. For both the function class Q of all functions computable in EXP^{linear} and the function class Q of all functions computable in EXP^{poly}, the base classes $NEXP^{poly}$ and RE are uniformly closed and are fully closed under unions.

(iv) For the class Q of total recursive functions, the base class RE of all recursively enumerable sets is uniformly closed and is fully closed under unions.

(v) Take Q to be the class of functions computable in polynomial time and let A be any set in NP. Then the class $C \subseteq NP$ of all sets $\leq_m^P A$ is uniformly closed and is also closed in NP under full unions. If A is in RP, then the same class C is a uniformly closed subset of RP, but we cannot be sure that it is closed in RP under full unions.

Because we are primarily interested in RP, we will be most interested in proving theorems about Boolean hierarchies built over uniformly closed complexity classes. However, we will point out how to strengthen these theorems for classes like NP which are also fully closed under intersections. A major goal will be to delineate differences in Boolean hierarchies built over base classes like RP which are merely uniformly closed and those built over base classes like NP which are also fully closed under unions. As we shall see, the extent to which base classes are closed under more than bounded unions directly influences the ease with which one can build nice extended hierarchies using truth-tables where the number of variables grows faster than logarithmically.

We now give the basic definition for building extended Boolean hierarchies over quite general collections of base classes, C.

[4] I.e., a set which is neither empty nor universal.

Definition 3.14. *Let Q be any minimal collection of functions, and let T be any Q-truth-table. Analogously to the sets defined by (*) for the regular Boolean hierarchy, we build classes in an extended Boolean hierarchy by defining*

$$T^Q[C] \quad =_{def} \quad \{\ X : \exists k,\ \exists\ s \in Q \text{ with } \textit{range}\ s \subseteq S^k \\ [\ c_X(x)\ =\ T_x(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x))\ ]\ \}. \qquad (****)$$

When the context is clear, we will usually drop the superscript Q in $T^Q[C]$.

The following lemma explains why, in building extended Boolean hierarchies, we could limit Definition 3.14 to classes which can be defined without using reductions to the sets $X_{s(i)}$.

Lemma 3.15. *Let Q be any minimal collection of functions and let C be any uniformly closed collection of sets. Let T be any truth-table from the class Q. For any function f in Q, define*

$$\langle T, f\rangle^Q[C] \quad =_{def} \quad \{\ X :\ \exists k,\ \exists\ s \in Q \text{ with } \textit{range}\ s \subseteq S^k \\ [\ c_X(x)\ =\ T_x^Q(c_{X_{s(1)}}(f(x,1)), c_{X_{s(2)}}(f(x,2)), \ldots, c_{X_{s(NV_T(x))}}(f(x, NV_T(x)))\ ]\ \}.$$

Then $\langle T, f\rangle^Q[C] \ \subseteq\ T^Q[C]$.

Proof: Simply define the sequence $S'_j =_{def} \{x : f(x,j) \in X_{s(j)}\}$. Since the class C is closed under bounded conjunctive reducibilities, the sequence $\{\lambda j S'_j\}$ is a uniform sequence in the class C. But

$$T_x(c_{X_{s(1)}}(f(x,1)), c_{X_{s(2)}}(f(x,2)), \ldots, c_{X_{s(NV_T(x))}}(f(x, NV_T(x))) \\ =\ T_x(c_{S'_1}(x), c_{S'_2}(x), \ldots, c_{S'_{NV_T(x)}}(x)).$$

This shows that $\langle T, f\rangle^Q[C] \ \subseteq\ T^Q[C]$. ∎

The following two propositions, which are not used in the remainder of this paper, show that things are not so simple when we instead allow, not just uniform reductions to the base sets $X_{s(i)}$ but instead allow many-one reductions to sets in the T^{th} level of the hierarchy. Recall that a truth-table T is *fixed* if the function $T(x)$ is constant. Our next proposition shows that fixed truth-table classes are closed under many-one reductions.

Proposition 3.16. *Let Q be any minimal collection of functions and let C be any uniformly closed collection of sets. Let T be any fixed truth-table from the class Q. For any function f in Q, define*

$$T_f^Q[C] \quad =_{def} \quad \{\ X' :\ \exists X \in T^Q[C]\ \ [\ X' \leq_m^Q X\ ]\ \}.$$

Then $T_f^Q[C] = T^Q[C]$.

Proof: The containment $T^Q[C] \subseteq T_f^Q[C]$ is obvious. For the reverse containment, suppose that $X' \leq_m^Q X$ via the function $f \in Q$. Much as in the proof of Lemma 3.15, define the sequence $S'_j =_{def} \{x : f(x) \in X_{s(j)}\}$. Since the class C is closed under bounded conjunctive reducibilities, the sequence $\{\lambda j S'_j\}$ is a uniform sequence in the class C. But

$$c_{X'}(x) = c_X(f(x)) =\ T_{f(x)}(c_{X_{s(1)}}(f(x)), c_{X_{s(2)}}(f(x)), \ldots, c_{X_{s(NV_T(f(x)))}}(f(x))) \\ =\ T_{f(x)}((c_{S'_1}(x), c_{S'_2}(x), \ldots, c_{S'_{NV_T(f(x))}}(x)).$$

Since the function T_x is constant for fixed truth-tables, so is the function $NV_T(x)$, and this shows that $X' \in T^Q[C]$. ∎

The next proposition shows that the situation is not so pretty in more general truth-table classes. For example, it shows that sets in standard extended hierarchies which have more than a linear number of variables in their underlying definition can always be many-one reduced to sets which have a *sub*linear number of variables in their underlying definition.[4] A similar phenomena occurs around sets having a logarithmic number of variables in their underlying definition. With a little more work Proposition 3.8 could also be proved with appropriate *positive* truth-table reductions replacing many-one reductions.

Proposition 3.17. *Let Q be any minimal collection of functions and let C be any uniformly closed collection of sets. Let T be any truth-table from the class Q, and let f be any function in Q for which f^{-1} is also in Q. Define $T'_x = T_{f^{-1}(x)}$ (so that $NV_{T'}(x) = NV_T(f^{-1}(x))$ and thus $NV_{T'}(x) < NV_T(x)$ if f is monotonic). Then*

$$X \in T[C] \Longrightarrow X \leq_m^Q X' \in T'[C].$$

Proof: Let $X \in T[C]$. Then

$$c_X(x) \;=\; T_x(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x)).$$

Much as in the preceding proof, define a uniform sequence $\lambda j S'_j$ by $S'_j =_{def} \{f(x) : x \in X_{s(j)}\}$. Define X' by

$$c_{X'}(x) =_{def} \; T'_x(c_{S'_1}(x), c_{S'_2}(x), \ldots, c_{S'_{NV_{T'}(x)}}(x)).$$

Then

$$\begin{aligned} c_X(x) &= T_x(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x)}}(x)) \\ &= T'_{f(x)}(c_{S'_1}(f(x)), c_{S'_2}(f(x)), \ldots, c_{S'_{NV_{T'}(f(x))}}(f(x))). \\ &= c_{X'}(f(x)). \quad \blacksquare \end{aligned}$$

Definition 3.14 gives a quite general method for building Boolean hierarchies over quite general complexity classes, and Definition 3.14 together with Example 3.8 provides a method for generalizing the various definitions for the specific regular Boolean hierarchies given in 2.0.2 to extended hierarchies. We can now ask how these classes relate to the conventional classes of the regular Boolean hierarchy and to each other.

The chief results used in the standard proofs of the equivalence of the various definitions of the Boolean hierarchy over NP and in the proofs that Turing reductions over NP require fewer queries to NP sets than truth-table reductions are the next two basic propositions. The proofs we give are similar in spirit to results for NP which are surveyed in Wagner's paper [Wa 88]. However, as discussed above, some changes are required both because the arbitrary complexity classes which we consider, including RP, need not have the power of non-determinism and need not have complete sets and because exhaustive searches which are possible when $NV_T(x)$ is bounded by a constant, are not possible at higher levels of extended Boolean hierarchies where $NV_T(x) > O(log(|x|))$.

Proposition 3.18. Consider any Q-truth-table T, associated ordering O, and uniformly closed complexity class C. Then for any set $A \in T[C]$, there exists a set D such that
(i) $c_A(x) \;=\; \sum_{i=1}^{NV_T(x)} [c_D(i,x)] \;+\; T_x(0, \ldots, 0_{NV_T(x)}) \mod 2$;
(ii) $c_D(i,x) \;\geq\; c_D(i+1,x)$, and $c_D(NV_T(x)+1, x) = 0$;
(iii) $A \leq_{bs}^Q D$ via a Q-bounded-search reduction which makes at most $\lceil log(NV_T(x)+1) \rceil$ queries to D;
(iv) D is in the closure of C under unions bounded by a function in Q.

---

[4] For example, Proposition 3.17 shows that for any of the truth-table classes arising from Example 3.8, $h'_{|x|^k}[C] \leq_m^Q h'_{\sqrt{|x|}}[C]$. It follows, for example, from Corollary 3.23 that in extended hierarchies over NP and over RP that *all* of the sets in the extended hierarchy can be $\leq_m^P$ reduced to sets which always have $NV_T(x) \leq \sqrt{|x|}$.

Proof: By definition of $T[C]$, the set A is definable by a function $s \in Q$ with *range* $s \subseteq S^k$ such that $c_A(x) = T_x(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x))$. Thus we must prove that

$$T_x(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x)) \quad = \quad \sum_{i=1}^{NV_T(x)} [c_D(i,x)] \quad + \; T_x(0, \ldots, 0_{NV_T(x)}) \mod 2$$

for an appropriate set D. From this formulation, it is clear that to establish (i) it is adequate to define the set D in such way that for every choice of x there is some concrete order τ of $\{1, 2, \ldots, NV_T(x)\}$ such that

$$\sum_{i=1}^{NV_T(x)} [c_D(i,x)] \; = \; MC_{T_x,\tau}(c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x)).$$

We will define D to be

$$D \quad =_{def} \quad \{ \langle i, x \rangle : \text{for some order } \tau \text{ of } \{1, 2, \ldots, NV_T(x)\} \text{ there exist at least } i \text{ mind changes of } T_x$$
$$\text{between } \langle 0, 0, \ldots, 0_{NV_T(x)} \rangle \text{ and } \langle c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x) \rangle \; \}.$$

The definition of the set D makes both (i) and (ii) obvious. Part (iii) follows directly from (i) and (ii) by using bounded minimalization to do a binary search on D.

It remains only to locate the set D in the closure of the class C under unions bounded by members of Q. We begin by observing that for the sequence $X_{s(1)}, X_{s(2)}, X_{s(3)}, \ldots$ which defines the set A, the related sequence $S_1, S_2, S_3, \ldots$ given by

$$S_j =_{def} \; \{1\} \bigcup \{x + 2 : x \in X_{s(j)}\}$$

is easily seen to be a uniform sequence in C.

Define a function f in Q by

$$\begin{aligned} f(j,i,x,\tau) = 0 \qquad & \text{if } MC_{T_x,\tau}(\vec{a}_\tau) < i \quad \textit{or} \quad \tau \text{ is not a concrete} \\ & \qquad \textit{partial} \text{ ordering of } \{1, 2, \ldots, NV_T(x)\} \\ = 1 \qquad & \text{if } MC_{T_x,\tau}(\vec{a}_\tau) \geq i \quad \textit{and} \quad a_j = 0 \quad \textit{and} \\ & \qquad \tau \text{ is a concrete partial } \textit{ordering} \\ = x + 2 \quad & \text{otherwise.} \end{aligned}$$

Next define the sets D_τ by

$$\langle i, x \rangle \in D_\tau \quad \Longleftrightarrow \quad \bigwedge_{j \leq NV_T(x)} f(j,i,x,\tau,) \in S_j.$$

Because the class C is closed under bounded conjunctive reducibilities, the sequence $\lambda \tau D_\tau$ is a uniform sequence in C.

Now note that for each τ if $\langle i, x \rangle \in D_\tau$ then you get at least i mind changes in going from $\vec{0}_{NV_T(x)}$ to $\vec{a}_\tau$ by flipping only bits in the vector $\vec{a}_\tau$, doing so in the order determined by τ. Furthermore, the definition of D_τ guarantees that if $\langle i, x \rangle \in D_\tau$ and $\vec{a}_{\tau,j} = 1$, then $x \in X_{s(j)}$. It follows that

$$MC_{T_x,\tau'}(\langle c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x) \rangle) \quad \geq \quad i$$

for an obvious choice of a (total) concrete ordering τ', so $\langle i, x \rangle \in D$. I.e., for all concrete partial orderings τ we have that $D_\tau \subseteq D$.

But from its definition it is now clear that $D \subseteq \bigcup D_\tau$. (The concrete partial ordering τ witnessing the containment can be nonconstructively chosen so that $\vec{a}_\tau$ happens to be $\langle c_{X_{s(1)}}(x), c_{X_{s(2)}}(x), \ldots, c_{X_{s(NV_T(x))}}(x)\rangle$.) Thus the set D can be defined by

$$\langle i,x\rangle \in D \iff \langle i,x\rangle \in \bigcup_{\tau \text{ is a partial ordering of } \{1,2,\ldots,NV_T(x)\}} D_\tau.$$

Since these concrete permutations are all coded as numbers less than or equal to $2^{NV_T(x)*log(NV_T(x))}$, we have that

$$D = \langle i,x\rangle \in \bigcup_{\{j\,:\, j \leq 2^{NV_T(x)*log(NV_T(x))}\}} D_j.$$

Thus D is in the closure of C under unions bounded by a function in Q. This establishes part (iv). ∎

Part (i) of the following corollary for the special case where $C = NP$ and $Q = P$ is standard in the literature, ([Be 88], [Wa 88], [Wa 87a]).

Corollary 3.19.

(i) Let C be any indexed complexity class, (such as NP), which is uniformly closed and which is also closed under full unions. Then any set, A, which is Q-truth-table reducible to a set $X \in C$ can be Q-bounded-search reduced to some other set $D \in C$ via a bounded-search reduction which makes at most $\lceil log(NV_T(x)+1)\rceil$ queries to D; i.e.,

$$[A \leq^P_{\{NV_T(x)\}-tt} X \;\&\; X \in C] \implies \exists D \in C\ [A \leq^Q_{\{\lceil log(NV_T(x)+1)\rceil\}-bs} D].$$

(ii) Let C be any indexed complexity class, (such as RP), which is uniformly closed. Then any set, A, which is Q-truth-table reducible to a set $X \in C$ via a truth table T with $2^{NV_T(x)*log(NV_T(x))}$ bounded by a function in B can be Q-bounded-search reduced to a set $D \in C$ via a bounded-search reduction which makes at most $log(\lceil NV_T(x)+1\rceil)$ queries to D; i.e.,

$$[A \leq^P_{\{NV_T(x)\}-tt} X \;\&\; X \in C] \implies \exists D \in C\ [A \leq^Q_{\{\lceil log(NV_T(x)+1)\rceil\}-bs} D].$$

Proof: Immediate from Proposition 3.18. ∎

Simple calculations show that meeting the requirement for part (ii) for classes like RP which are based on polynomial time but not closed under full unions can be met by keeping $NV_T(x) \leq O(\frac{log(|x|)}{log(log(|x|))})$.

From our next proposition, it is easily seen that a full converse to Proposition 3.18 also holds provided that the set D is actually in the class C, (as it always must be if C is closed under full unions) *or* if the function $2^{NV_T*log(NV_T)}$ is bounded by a function in B.

Proposition 3.20. *Suppose that*

(i) C is any indexed complexity class which is uniformly closed;
(ii) t is any integer valued function in Q and initval is any Boolean valued function in Q;
(iii) D is any nontrivial set in C such that $c_D(i,x) \geq c_D(i+1,x)$;
(iv) A is any set defined by $c_A(x) = \sum_{i=1}^{t(x)} [c_D(i,x)] + initval(x)$ mod 2.

For any Q-truth-table T, define $adjust(x) =_{def} T_x(0,\ldots,0_{NV_T(x)}) + initval(x)$ mod 2. Then for any Q-truth-table T and any Q-ordering function O for T,

$$[\ t(x) \leq MC_{T,O}(x, \vec{1}_{NV_T(x)}, NV_T(x)) + adjust(x)\ ] \implies A \in T[C].$$

Proof:[5] We use the two functions $Ord_O(x,i)$, which gives the ordering of the i^{th} variable of T_x in the order O_x, and the function $MC_{T,O}(x, \vec{1}_{NV_T(x)}, i)$ which gives the number of mind changes which have occurred just *after* flipping the i^{th} bit in the determination of $MC_{T,O}(x, \vec{1}_{NV_T(x)})$. Because O is a Q-ordering of T, both of these functions must be in the class Q.

The idea of the proof is simple: the truth-table T, with its variables ordered by the ordering O has enough mind changes to do a *mod 2* count of the values $\langle i,x\rangle$ for which the value of $c_D(i,x)$ is "1." Hence, if we substitute

$$c_D(1,x),\ldots,c_D(1,x),c_D(2,x),\ldots,c_D(2,x),\ldots,c_D(t(x),x),\ldots,c_D(t(x),x)$$

into the variables for T_x in the order dictated by O, beginning the substitution of the next new variable each time a mind change occurs, then T_x, with these substitutions, must give a *mod 2* count of $\sum_{i=1}^{t(x)}[c_D(i,x)]$. Since for any vector $\vec{b}$

$$T_x(\vec{b}) = MC_{T,O}(x,\vec{b},NV_T(x)) + T_x(\vec{0}_{NV_T(x)}) \mod 2,$$

doing the proper substitutions should give us the right result, provided the initial values $initval(x)$ and $T_x(\vec{0}_{NV_T(x)})$ are identical. If these two variables are not identical, then we must do enough substitutions to effect one more mind change.

Formally, we proceed as follows. First, we define the function $t'(x) =_{def} t(x) + adjust(x)$. It is easily seen that $t' \in Q$. Next, recall that the function $Ord(O_x,j)$ produces the "position" of the j^{th} variable of T_x in the order determined by O. Thus, if we can force

$$c_D(1,x),\ldots,c_D(1,x),c_D(2,x),\ldots,c_D(2,x),\ldots,c_D(t(x),x),\ldots,c_D(t(x),x) \qquad (\dagger)$$

to be substituted into T_x in this order and then flip the bits in this order, then all of the bit flipping will occur, first in all those places where the Boolean values are assigned the value "1", followed by (attempted, but blocked) bit flipping in all those places where the Boolean values are assigned the value "0." This will force the mind changes in this order for this substitution to reflect the values in the summation $\sum_{i=1}^{t(x)}[c_D(i,x)]$. Of course, if $adjust(x) \neq 0$, then we need to force one more mind change, so we should have used $t'(x)$ instead of $t(x)$ in $(\dagger)$.

To accomplish these substitutions, note that $MC_{T,O}(x, \vec{1}_{NV_T(x)}, Ord_O(x,j)-1)$ will tell us how many mind changes have occurred just *before* the j^{th} bit is flipped. We should be substituting the j^{th} variable during those periods when the mind changes that have been witnessed total $j-1$, and we should quit when

$$MC_{T,O}(x, \vec{1}_{NV_T(x)}, Ord_O(x,j)) > t'(x).$$

Since the set D is not trivial, we may let d_0 be some fixed member of $\overline{D}$. To accomplish the substitutions we have just discussed, we define the function

$$\begin{aligned} f(x,j) &= \langle MC_{T,O}(x, \vec{1}_{NV_T(x)}, Ord_O(x,j)-1)+1, x\rangle \quad \text{if } MC_{T,O}(x, \vec{1}_{NV_T(x)}, Ord_O(x,j)) \leq t'(x)), \\ &= d_0 \qquad\qquad \text{else.} \end{aligned}$$

[5] At the expense of introducing still more notation, we could get a somewhat stronger result here by replacing the trivial vector valued function $\vec{1}_{NV_T(x)}$ by an arbitrary vector valued function $\vec{b(x)}$ in Q satisfying $|\vec{b(x)}| = NV_T(x)$. We shall see from the proof of Corollary 3.22 that *for the applications we have in mind* this is not really a stronger result, except perhaps for very weak minimal function classes, Q. The reader should also note that, given any function $initval$ in Q and a Q-truth-table T of the appropriate number of variables, by properly initializing T one can get a truth-table T' which has *exactly* the same mind change sequence as T and satisfies $T'_x(0,0,\ldots,0_{NV_{T'}(x)}) = initval(x)$. Of course, since in general the classes $T[C]$ are not closed under complements, one should not expect that $T[C] = T'[C]$. Note also that t is not only in Q, but that because $t(x) \leq NV_T(x)$, t is actually bounded by a function in B. For most definitions of B, this will actually force $t \in B$.

By our assumptions on Q and O, the function f is easily seen to be in Q, and we then have that

$$MC_{T,O}(x, \langle c_D(f(1,x)), c_D(f(2,x)), \ldots, c_D(f(NV_T(x),x))\rangle, NV_T(x))$$
$$= \sum_{i=1}^{t(x)} [c_D(i,x)] + adjust(x) \mod 2,$$

and thus that

$$T_x(c_D(f(x,1)), c_D(f(x,2)), \ldots, c_D(f(x, NV_T(x)))) = \sum_{i=1}^{t(x)} [c_D(i,x)] + initval(x) \mod 2.$$

The result now follows from Lemma 3.15. ∎

Propositions 3.18 and 3.20 together show that classes, like NP, which are closed under full unions must have Boolean hierarchies defined by Q-truth-tables T where the classes $T[C]$ are determined solely by the initial values of T together with the number of mind-changes of T induced by a maximal Q-ordering of the variables of T.

These propositions also show that where the class C is fully closed under unions, one need not have complete sets in the base class C in order to give an "indexing free" definition of extended Boolean hierarchies. Following Wagner, ([Wa 88]), one simply takes something like Part (iv) of Proposition 3.20 as the definition of the sets, A, in the extended Boolean hierarchy, but limits the selection of the functions t to those $t \in B$. This definition is equivalent to the definition using truth-tables which we have given, since Proposition 3.20 guarantees that any such set A is in $T[C]$ for any truth-table which has enough mind changes, while Proposition 3.18 guarantees that any $A \in T[C]$ must be definable by some such set D.

Characterizations like those of the preceding two paragraphs also hold for lower levels of extended Boolean hierarchies over classes, like RP, which are not fully closed under unions, but for most applications we've shown above only that the determination of the classes $T[C]$ is uniquely determined by the number of mind changes when the number of the variables is given by functions f for which $2^{f*log(f)}$ is bounded by a function in Q. As stated earlier, in the case of RP, this means, for example, that the number of variables should be at most about $O(\frac{log(|x|)}{log(log(|x|))})$.

In closing this section, we shall make explicit a number of corollaries to Propositions 3.18 and 3.20. These results show that, for many interesting cases, the number of mind changes of truth-tables T and T' (sometimes together with the initial values of T and T') determine whether $T[C] \subseteq T'[C]$. Moreover, we will see that for truth-tables T based on any of the sequences $\{h'_k\}$ of Definition 2.0.2, the antecedents of the following corollaries are easy to establish. Hence, for quite general base classes, C, equivalence and uniqueness of the lower levels of the extended Boolean hierarchies defined by the sequences from Definition 2.0.2 will be directly established by a trivial count of the mind changes together with a trivial check of the initial values of the truth-tables.

Before proceeding, one more bit of notation will prove useful.

Definition 3.21. *Let T and T' be any two Q-truth-tables. Define $adjust_{T,T'}(x) =_{def} T_x(0,\ldots,0_{NV_T(x)}) + T_x(0,\ldots,0_{NV_T(x)}) \mod 2$. (Note that $adjust_{T,T'}(x)$ is always in the function class Q.)*

Corollary 3.22. *Let Q be any minimal collection of functions and let C be any uniformly closed complexity class. Let T and T' be Q-truth-tables, and suppose either that C is closed under full unions or that NV_T is bounded by a function f with $2^{f*log(f)} \in B$. Then, if there exists a Q-ordering O' for T' such that*

$$max_{O \in Q}\{MaxMC_{T,O}(x)\} \leq MC_{T',O'}(x, \vec{1}_{NV_{T'}(x)}, NV_{T'}(x)) + adjust_{T,T'}(x),$$

then

$$T[C] \subseteq T'[C].$$

Proof: This follows directly from Propositions 3.18 and 3.20. ∎

Corollary 3.22 is immediately applicable to the hierarchies based on Example 3.8. Let t be any function in B and let h'_t be the corresponding truth-table built as in Example 3.8. For any such function t, for *any* truth-table T with $NV_T(x) \leq t(x)$ and $T(\vec{0}_{NV_T(x)}) = 0$ $(= h'_t(\vec{0}_{t(x)}))$, we have for *any* ordering O and for the identity ordering O' that

$$MaxMC_{T,O}(x) \leq t(x) = MC_{h'_t,O'}(x, \vec{1}_{t(x)}, t(x)),$$

so $T[C] \subseteq h'_t[C]$. In the case that $T(\vec{0}_{NV_T(x)}) \neq 0$, we draw out our conclusions in more explicit fashion in Corollary 3.23, below.

It should be noted that Parts (ii) and (iii) of Corollary 3.23 establish that for *any* extended hierarchy over *any* complexity class closed under full unions, *any* of the standard methods for defining Boolean hierarchies yield identical hierarchies. For classes like RP where the base complexity class is merely uniformly closed, these basic definitions are shown here to be equivalent only for functions t for which both $NV_{h'(x)}$ is bounded by a function f with $2^{f*log(f)} \in B$. As pointed out earlier, for RP and similar classes, this means that the number of distinct variables should essentially be bounded by $O(log(|x|)*log(log(|x|)))$ if we are to guarantee by our methods that the various possible definitions are all equivalent.

Corollary 3.23.

(i) Let Q be any minimal collection of functions and let C be any uniformly closed complexity class. Let t be any function in B and suppose either that C is closed under full unions or that $2^{t*log(t)}$ is bounded by a function in B. Then for any specific Q truth-table h'_t built from any of the four base classes of Definition 2.0.2 as explained in Example 3.8, if T is any Q-truth-table for which

$$NV_T(x) + T(\vec{0}_{NV_T(x)}) \leq t(x),$$

then

$$T[C] \subseteq h'_t[C].$$

(ii) *Let Q, C, t, and B, be as in Part (i). Then for each of the four specific truth-tables built from the four base classes of Definition 2.0.2 as explained in Example 3.8,*

$$h_t[C] = g_t[C] = g'_t[C] = Diff_t[C].$$

(iii) *By the same argument, the complements of these classes, which we denote as $coh_t[C]$, $cog_t[C]$, $cog'_t[C]$, and $coDiff_t[C]$ are also equal.*

(iv) *For the special case where $C = NP$ and $r \in P$, Wagner defines the r^{th} level of the Boolean hierarchy over C by*

$$C[r] =_{def} \{A : (\exists D \in C)[c_D(x, i+1) \leq c_D(x,i) \ \& \ c_A(x) = \sum_{i=1}^{r(|x|)} [\, c_D(i,x)] \mod 2.\ ]\}$$

Let Q, C, and B, be as in Part (i), let $r \in Q$, and let $t(x) = r(|x|)$. Then for each of the four specific truth-tables built from the four base classes of Definition 2.0.2 as explained in Example 3.8,

$$h_t[C] = g_t[C] = g'_t[C] = Diff_t[C] = C[r].$$

Corollary 3.23 justifies using standard definitions as follows.

Definition 3.24. *Let Q be any minimal collection of functions and let C be any uniformly closed complexity class. Let t be any function in B and suppose either that C is closed under full unions or that $2^{t*log(t)}$ is also in B. Then Q-truth-table h'_t, we let $C[t]$ denote any of the classes $h'_t[C]$ from Example 3.8. (Corollary 3.23 guarantees the invariance of the definition no matter which concrete Q-truth-tables are chosen from the example.)*

In the case of the base classes $C = NP$ or $C = NP$, this yields the notations $NP[k]$, $NP[t]$, $RP[k]$ and $RP[t]$ for constants k and for suitable functions $t \in B$.

In closing we give a final corollary which generalizes known results on the relations among various bounded-query classes over NP. To keep these results in a familiar domain, we return to taking the class Q to be either the set of functions computable in polynomial time or in log space but we otherwise let C be *any* uniformly closed class of sets. Employing standard notation, we let $P^C_{tt}[t]$ denote the class of sets that are reducible to some set in C via some Q-truth-table which has at most $t(x)$ variables, and we let $P^C_T[t]$ denote the class of languages recognizable by a Q-machine which, on input x, makes at most $t(x)$ queries to an oracle in C. We then have:

Corollary 3.25. *Let Q be the class of functions computable in polynomial time or in log space. Let C be an arbitrary uniformly closed complexity class. Let t be any function in Q such that for some integer k $t(x) \leq |x|^k$ if C is fully closed under unions, and such that $t(x) \leq k * [\frac{log(|x|)}{log(log(|x|))}]$ if C is not fully closed under unions. Then*

(i) $P^C_T[\lceil log(t) \rceil] = P^C_{tt}[2^{\lceil log(t) \rceil} - 1]$, provided that $2^{\lceil log(t) \rceil} - 1$ is in the bounded class B.

(ii) $Diff_t[C] \cup coDiff_t[C] \subseteq P^C_{tt}[t] \subseteq Diff_{t+1}[C] \cap coDiff_{t+1}[C]$.

(iii) The $P^C_{tt}[t]$ hierarchy collapses if and only if the $P^C_T[t]$ hierarchy collapses if and only if the $Diff_t$ hierarchy collapses.

Proof: Part (ii) and one of the containments of Part (i) are immediate from Corollary 3.19. The other containment for Part (i) follows directly from the standard proof that $P^C_T[k] \subseteq P^C_{tt}[2^k - 1]$, which is proven by a straightforward induction on k for *arbitrary* classes C, (see, e.g., [Be-88]). Part (iii) follows directly from Parts (i) and (ii). ∎

An appropriate version of Corollary 3.25 also holds when Q is taken to be an *arbitrary* minimal class of functions. In this case $Q^C_{tt}[t]$ is taken to be the class of sets that are reducible to some set in C via some Q-truth-table which has at most $t(x)$ variables, but some care is needed in giving an appropriate definition of $Q^C_T[t]$, and in stating an exact version of Part (i) of the corollary.

We note that Beigel ([Be 88], Section 4.1, Theorems 17 and 18) has briefly considered Boolean hierarchies and bounded query classes over arbitrary collections of sets, C, and with *no* assumptions on C has proved the following results, for all integers k:

(i) $P^C_{tt}[k] \subseteq P^C_T[k] \subseteq P^C_{tt}[2^k - 1]$.

(ii) $Diff_k(C) \subseteq P^C_{tt}[k] \subseteq Diff_{(2^k(2k+3))}[C]$.

(iii) The $P^C_{tt}[k]$ hierarchy collapses if and only if the $P^C_T[k]$ hierarchy collapses if and only if the $Diff_k$ hierarchy collapses.

Beigel's results can obviously be extended from integers k to arbitrary functions, t. However in Corollary 3.25 we have seen that, for constant bounded query classes and for lower levels of functionally bounded query classes, by placing minimal assumptions on the class, C, we obtain sharper results than those obtainable with no conditions on the class C.

4. STRUCTURAL PROPERTIES OF THE BOOLEAN HIERARCHY OVER RP.

In this section, discuss some structural properties specific to the Boolean hierarchy over RP. In the first subsection we give a characterization of the Boolean hierarchy over RP using machine acceptors and probabilistic quantifiers. Since one of our reasons for studying this hierarchy is to investigate the structure of $BPP - RP$, we compare the machines and predicates that define RBH with those that define BPP. For instance, it is obvious that the languages in RBH are recognized by probabilistic machines with bounded two-sided error, however we show that they are recognized by machines of this type that take on a very special form.

In the second subsection, we investigate when self-reducibility and membership in BPP can cause sets in the Boolean hierarchy over NP to lie in the Boolean hierarchy over RP. Our results extend Ko's results which show that self-reducible sets that are both in NP and BPP are in RP, ([Ko 82]).

Machines and Predicates for RBH.

We know that the Boolean hierarchy over RP is contained within BPP and thus that every language in RBH is recognized by a BPP machine. However, we can show that for languages in RBH these machines can be assumed to have a very nice form. We obtain these machines by combining Boolean circuits for the Hausdorff functions of Definition 2.0.2 with simple RP machines in much the same manner that Cai and Hemachandra used "hardware over NP," ([CaHe 86]) to recognize sets in the Boolean hierarchy over NP .

Recall that since RP is uniformly closed, by Corollary 3.21, the class $RP[k]$ is invariant with respect to the Boolean functions used in Definition 2.0.2. We describe machines for $RP[k]$ that are constructed using the Wechsung-Wagner function g_k.

Definition 4.1. *A $g_k[RP]$-acceptor M, for a set $X \in RP[k]$ is a tuple $\langle b^k, M_1, \ldots, M_k \rangle$, where $M_1, \ldots, M_k$ are RP-acceptors, and b^k is a Boolean circuit of k inputs that computes the Boolean function g_k. On input x, the machines M_i non-deterministically guess a computation path and $M_i(x)$ outputs a '1' if that path accepts, and outputs a '0' otherwise. The i^{th} input to the circuit b^k is $M_i(x)$. The machine M "accepts" if and only if the output of b^k is a '1'.*

Although the RP--acceptors used to define a $g_k[RP]$-acceptor all have one-sided error, the $g_k[RP]$-acceptor generally has two-sided error. Nevertheless, a $g_k[RP]$-acceptor has a very nice structure and its two-sided error probability is *strictly* determined by the one-sided error probabilities of its component RP --acceptors.

Proposition 4.2. *Let the set X be any set in $RP[k]$. Then there are RP--acceptors $M_1, \ldots, M_k$ each with one-sided error probability of $1 - (1/c)$ such that the $g_k[RP]$–acceptor, $M = \langle b^k, M_1, \ldots, M_k \rangle$, recognizes X with a two-sided error probability of $1 - (1/c)$.*

Proof: We give the proof for k even; the case when k is odd can be proved similarly.

First recall the definition of Hausdorff's Boolean function g'_k.

$$g'_k(x_1, \ldots, x_k) = g_k(x_1, \ldots, x_k) \bigwedge_{1 \le j \le k} x_j \to x_{j+1}$$

where

$$g_k(x_1, \ldots, x_k) = (\neg x_1 \wedge x_2) \vee \ldots \vee (\neg x_{k-1} \wedge x_k)$$

when k is even.

Since RP is closed under union and intersection, it follows from Hausdorff's result ([Ha 78]) (and from Corollary 3.21) that $g'_k[RP] = g_k[RP]$. Because of the special form of g'_k, we can assume without loss of generality that the sets $X_1, \ldots, X_k$ form a subset chain: $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_k$. Therefore,

$$x \in X \iff x \in \text{ exactly one of the sets } (X_{2i} - X_{2i-1}).$$

Hence

$$\text{Prob}\,[M \text{ makes an error, given } x \in X] \;=\; \text{Prob}[b^k(M_1(x),\ldots,M_k(x)) \;=\; 0, \text{ given } x \in X],$$

which is at most

$$\max_i \text{Prob}[b^k(M_1(x),\ldots,M_k(x)) \;=\; 0, \text{ given } x \in (X_{2i} - X_{2i-1})].$$

To analyze the above quantity we assume that x is in $(X_{2i} - X_{2i-1})$, for some $i \leq k/2$. Then for all $j < i, x \notin X_{2j}$ and since the M_{2j}'s are RP-acceptors they do not make errors on the $\overline{X}_{2j}$'s. Therefore

$$\forall j < i \;\text{Prob}[(\neg M_{2j-1}(x) \wedge M_{2j}(x)) \text{ is true, given } x \in (X_{2i} - X_{2i-1})] \;=\; 0.$$

Hence the event: $[b^k(M_1(x),\ldots,M_k(x)) \;=\; 0, \text{ given } x \in (X_{2i} - X_{2i-1})]$ only occurs when for all $j > 2i$, M_j correctly accepts x and M_{2i} wrongly rejects x. Thus

$$\text{Prob}[b^k(M_1(x),\ldots,M_k(x)) \;= 0, \text{ given } x \in (X_{2i} - X_{2i-1})] \;=$$

$$\text{Prob}[\bigwedge_{2i<j\leq k} (M_j(x) = 1) \wedge (M_{2i}(x) = 0), \text{ given } x \in (X_{2i} - X_{2i-1})]. \qquad (*)$$

Since $\text{Prob}[\bigwedge_{2i<j\leq k}(M_j(x) = 1)]$ is at most $(1/c)^{k-2i}$, and $\text{Prob}[(M_{2i}(x) = 0)]$ is at most $(1 - (1/c))$, the probability of $(*)$ is at most

$$(1/c)^{k-2i} * (1 - (1/c)).$$

It is easy to see that the above error probability is maximized and equals $1 - (1/c)$, when $i = k/2$, i.e. when $x \in (X_k - X_{k-1})$, and the first conjunct in $(*)$, $\bigwedge_{2i<j\leq k}(M_j(x) = 1)$, disappears.

Moreover, by observing that $x \notin X \iff x$ is in exactly one of the sets: X_1, $\overline{X_k}$, and $(X_{2i+1} - X_{2i})$, for $1 \leq i \leq (k/2) - 1$, the above argument extends easily to show that

$$\text{Prob}[M \text{ makes an error, given } x \notin X] \;=\; 1 - 1/c.$$

Thus M has a two-sided error probability of $1 - 1/c$. ∎

Note that $(*)$ shows that the two-sided error probability of a $g'_k[RP]$-acceptor for a set in $RP[k]$ is strictly determined by the independent one-sided error probabilities of a fixed, finite number of RP-acceptors: a property that general BPP-acceptors can not be expected to have.

Next, we give a characterization of $RP[k]$ in terms of probabilistic predicates. The polynomially bounded probabilistic quantifier, "$\exists^+$," which denotes "there exists at least a fixed constant fraction of," together with the regular polynomially-bounded quantifier "$\forall$" were used by Zachos in [Za 86] to characterize BPP and RP-predicates. For instance, a set X is in RP if and only if there exists a polynomially computable predicate P such that

$$x \in X \iff \exists^+ y P(x,y),$$

and

$$x \notin X \iff \forall y P(x,y).$$

Abbreviating, we can characterize sets in RP as "$\exists^+ y\; P(x,y)$ / $\forall y\; \neg P(x,y)$," or just "$\exists^+/\forall$." Similarly, $coRP$ can be characterized as "$\forall/\exists^+$," and BPP as "$\exists^+/\exists^+$." Moreover, by manipulating probabilistic quantifiers, the following non-trivial characterization of BPP was obtained by Zachos and Heller ([ZaHe 84]).

Theorem 4.3. *([ZaHe 84]) Any set X in BPP can be characterized in each of the following ways.*
(i) $\exists^+/\exists^+$, *that is,*

$$x \in X \iff \exists^+ y\; P(x,y),$$

$$x \notin X \iff \exists^+ y\; \neg P(x,y);$$

(ii) $\exists^+\forall/\forall\exists^+$, *that is,*

$$x \in X \iff \exists^+ y \forall z\ Q(x,y,z),$$
$$x \notin X \iff \forall y \exists^+ z\ \neg Q(x,y,z);$$

(iii) $\forall\exists^+/\exists^+\forall$, *that is*

$$x \in X \iff \forall y \exists^+ z\ R(x,y,z),$$
$$x \notin X \iff \exists^+ y \forall z\ \neg R(x,y,z);$$

where P, Q and R are polynomially computable predicates.

The following proposition is a straightforward extension Theorem 4.3 and defines predicates that characterize $RP[k]$.

Proposition 4.4. *Let h'_k be any one of the Boolean functions of Definition 2.0.2. We will assume with out loss of generality that k is even, and we observe that with some renumbering we may assume that the literals with indices $(k/2)+1$ through k are those that appear negated in the expansion of of h'_k. Then,*

(i) A set, X, in $RP[k]$ can be characterized as: $\exists^+\forall h'_k/\exists^+\forall\neg h'_k$, that is

$$x \in X \iff \exists^+ y_1,\ldots,y_{k/2} \forall y_{(k/2)+1},\ldots,y_k\ h'_k(P_1(x,y_1),\ldots,P_k(x,y_k)),$$
$$x \notin X \iff \exists^+ y_1,\ldots,y_{k/2} \forall y_{(k/2)+1},\ldots,y_k\ \neg h'_k(P_1(x,y_1),\ldots,P_k(x,y_k));$$

for some $P_1,\ldots,P_k$ which are polynomially computable predicates.

(ii) the quantifiers that appear in $RP[k]$-predicates can be interchanged freely:

$$\exists^+\forall h'_k/\exists^+\forall\neg h'_k\ =\ \forall\exists^+ h'_k/\forall\exists^+\neg h'_k\ =\ \exists^+\forall h'_k/\forall\exists^+\neg h'_k\ =\ \forall\exists^+ h'_k/\exists^+\forall\neg h'_k.$$

Proof: The proof of (i) is immediate from the definition of general Boolean hierarchies. The proof of (ii) follows by observing that in (i) the quantifiers have independent (non-intersecting) scopes. ∎

Notice that the characterization of BPP-predicates as $\exists^+ y\forall z P(x,y,z)$ / $\forall y\exists^+ z\neg P(x,y,z)$ does not allow such an interchange of quantifiers without changing the polynomially computable predicate P, while such an interchange is possible in the case of $RP[k]$-predicates. This is because, in the case of general BPP-predicates, the universally and probabilistically quantified variables occur in the same polynomially computable predicate, i.e, the quantifiers have intersecting scopes, while $RP[k]$-predicates are completely characterized by k independent RP-predicates, and hence their quantifiers have non-intersecting scopes.

Remarks on Self-reducibility.

Ko proved that if a disjunctive self-reducible set in NP is also in BPP, then it must be in RP, ([Ko 82]). We shall see that this result can be extended to show that many sets which arise naturally in NP[k] can not be in BPP without forcing the collapse of their associated natural "spans" into RP[k].

We begin by considering certain natural operators on sets in NP. For any set S in NP, we denote by $span_S$ any collection of sets of NP which satisfies

(i) $A \in span_S$ *implies* $A \leq^P_T S$

(ii) $A \in span_S$ & S disjunctively self − reducible *implies* $A \leq^P_T S$ & A disjunctively self − reducible.

As natural examples, we might take $span_S$ as the collection of sets $\leq^P_m S$ or we might take $span_S$ as the collection of sets positive truth-table reducible to S. Note that in these latter two cases, if S is complete for NP, then $span_S$ is just NP.

Now let h'_k be any one of the Boolean functions of Definition 2.0.2except that for the Hausdorff hierarchy, g'_k. Let

$$S_k\ =_{def}\ \{\langle x_1,\ldots,x_k\rangle : h'_k(c_S(x_1),\ldots,c_S(x_k))\}.$$

Then S_k corresponds to a "complete" set for the k^{th} level of the Boolean hierarchy built over $span_S$. For example if $S = NP$ then S_k is just a standard complete set for $NP[k]$. For these choices of h'_k, one easily sees

that $S \leq_m^P S_k$. Following our usual practices, for any polynomially computable function t with $t \in B$ we denote the t^{th} level of the extended Boolean hieararchy over $span_S$ by $span_S[t]$. Obviously, $span_S[t] \subseteq NP[t]$.

Proposition 4.5. *Let S be in NP and let t be any polynomially computable function with $t(x) \leq p(|x|)$ for some polynomial p. Suppose that either S or S_k is disjunctively self-reducible. Then, $S_k \in BPP$ or $S \in BPP$ implies that $span_S[t] \subseteq RP[t]$.*

Proof. Since $S \leq_m^P S_k$, we see that if S_k is disjunctively self-reducible, so is S. By hypothesis on $span_S$ this implies that all members of $span_S$ are disjunctively self-reducible. Furthermore, since $S \leq_m^P S_k$, and since every set in $span_S$ is $\leq_T^P$ to S, we have that every set in $span_S$ is $\leq_T^P$ to S_k. But from Zachos, ([Za 86]), membership in BPP is preserved under $\leq_T^P$ reductions. Thus every set in $span_S$ is in BPP. But by [Ko 82] every disjunctively self-reducible set in $BPP \cup NP$ is in RP. Thus $span_S \subseteq RP$, and so trivially $span_S[k] \subseteq RP[k]$.

5. BIBLIOGRAPHY.

[**Ad 65**] J. Addison, "The method of alternating chains," *Symp Theor Models,* North-Holland, (1965), 1-16.

[**AdHu 87**] L. Adleman and M. A. Huang, "Recognizing primes in random polynomial time," *ACM Symp Theory Computing,* (1987), 462-469.

[**BaMiSh 86**] E. Bach, G. Miller and J. Shallit, "Sums of divisors, perfect numbers and factoring," *SIAM J Computing,* 15, (1986), 1143-1154.

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

[**Be 87**] R. Beigel, "Bounded queries to SAT and the Boolean hierarchy," *preprint,* (1987).

[**BHW 89**] R. Beigel, L. Hemachandra and G. Wechsung, "On the power of probabilistic polynomial time: $P^{NP[log]} \subseteq PP$," *Structure in Complexity Conference,* (1989), to appear.

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

[**BBJSY 89**] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam and P. Young, "Generalized Boolean hierarchies and Boolean hierarchies over RP," *Univ Wisconsin, CS Dept Tech Report,* 809 (1989), 1-50.

[**BJY 89**] D. Bruschi, D. Joseph and P. Young, "Strong separations for the Boolean hierarchy over RP," *Univ Wisconsin, CS Dept Tech Report,* 847 (1989), 1-12.

[**BuHa 88**] S.Buss and L. Hay, "On truth-table reducibility to SAT and the difference hierarchy," *Structure in Complexity Conference,* (1988), 224-233.

[**CGHHSWW 88**] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, "The Boolean hiearchy I: structural properties," *SIAM J Comput,* 6 (1988), 1232-1252.

[**CGHHSWW 89**] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, "The Boolean hiearchy II: applications," *SIAM J Comput,* 7 (1989), 95-111.

[**CaHe 86**] J. Cai and L. Hemachandra, "The Boolean hierarchy: hardware over NP," *Structure in Complexity Conference,* (1986), 105-124.

[**Er 68a**] Y. Ershŏv, "A hierarchy of sets, I," *Algebra and Logic,* 7 (1968), 25-43.

[**Er 68b**] Y. Ershŏv, "A hierarchy of sets, II," *Algebra and Logic,* 7 (1968), 15-47.

[**Er 69**] Y. Ershŏv, "A hierarchy of sets, III," *Algebra and Logic,* 9 (1969), 20-31.

[**GeGr 86**] J. Geske and J. Grollmann, "Relativizations of unambiguous and random polynomial time classes," *SIAM J Computing,* 15 (1986), 511-519.

[**GuWe 86**] T. Gundermann and G. Wechsung, "Nondeterministic Turing machines with modified acceptance," *Proc MFCS*, L N CS, 233 (1986), 396-404.

[**Ha 78**] F. Hausdorff, *Set Theory,* Chelsea, 3rd ed., 1978.

[**HiZa 84**] P. Hinman and S. Zachos, "Probabilistic machines, oracles and quantifiers," *Proc Recursion Theory Week,* L. N. Math 1141 (1984), 159-192.

[**Ka 88**] J. Kadin, "The polynomial time hierarchy collapses if the Boolean hierarchy collapses," *Structure in Complexity Conference,* (1988), 278-292.

[**KöSchWa 87**] J. Köbler, U. Schöning and K. Wagner, "The difference and truth-table hierarchies for *NP* ," *RAIRO,* 21 (1987), 419-435.

[**Ko 82**] K. Ko, "Some observations on probabilistic algorithms and *NP*-hard problems," *Inform Proc Letters,* 14 (1982), 39-43.

[**La 83**] C. Lautemann, "*BPP* and the polynomial hierarchy," *Inform Proc Letters,* 17 (1983), 215-217.

[**Pu 65**] H. Putnam, "Trial and error predicates and a solution to a problem of Mostow-ski," *J Sym Logic,* 30 (1965), 49-57.

[**Ra 80**] M. Rabin, "Probabilistic tests for primality," *J Number Theory,* 12 (1980), 128-138.

[**RaSh 88**] M. Rabin and J. Shallit, "Randomized algorithms in number theory," To appear *Comm Pure Appl Math.*

[**Ra 82**] C. Rackoff, "Relativized questions involving probabilistic algorithms," *JACM,* 29 (1982), 261-268.

[**Sch 80**] J. Schwartz, "Fast probabilistic algorithms for the verification of polynomial identities," *JACM,* 27 (1980), 701-717.

[**Si 83**] M. Sipser, "A complexity theoretic approach to randomness," *Proc Symp Theory Comput,* (1983), 330-335.

[**SoSt 77**] R. Solovay and V. Strassen, "A fast Monte-Carlo test for primality," *SIAM J Comput* 6 (1977), 84-85; [Erratum: 7 (1978), 118].

[**Wa 86**] K. Wagner, "More complicated questions about maxima and minima, and some closure properties of *NP*," *ICALP,* L N Comp Sc, 226 (1986), 434-443.

[**Wa 88**] K. Wagner, "Bounded query computations," *Structure in Complexity Conference,* (1988), 260-277.

[**WeWa 85**] G. Wechsung and K. Wagner, "On the Boolean closure of *NP*," *Proc Conf Fundament Comput Theory,* L N Comp Sc, 199 (1985), 485-493.

[**ZaHe 84**] S. Zachos and H. Heller, "A decisive characterization of *BPP*," *Information and Control,* 69 (1986), 125-135.

[**Za 86**] S. Zachos, "Probabilistic quantifiers, adversaries and complexity classes: an overview," *Structure in Complexity Conference,* (1986), 383-400.