INFORMATION AND COMPUTATION 90, 156–177 (1991)

Separating the Low and High Hierarchies by Oracles*

KER-I KO

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794

The relativized low and high hierarchies within NP are considered. An oracle A is constructed such that the low and high hierarchies relative to A are infinite, and for each k an oracle A_k is constructed such that the low and high hierarchies relative to A_k have exactly k levels.

1. INTRODUCTION

The notion of lowness and highness has been a useful tool in classifying the information content in recursion theory as well as in complexity theory. Within the class NP, one may define low sets and high sets based on the following intuition: a set A in NP is *low* if the information content of A is so meager that a nondeterministic oracle machine using A as the oracle does not compute more than when it uses the empty set as the oracle; a set A in NP is *high* if the information content of A is so rich that a nondeterministic oracle machine using A as the oracle computes as much as when it uses an NP-complete set as the oracle. Extending this idea, Schöning (1983) gave the following formal definition of the low and high hierarchies: For each $k \geqslant 0$, a set A in NP is in L_k^P if $\Sigma_k^P(A) = \Sigma_k^P$; and it is in H_k^P if $\Sigma_k^P(A) = \Sigma_{k+1}^P$. It follows immediately that for each $k \geqslant 0$, $L_k^P \subseteq L_{k+1}^P$ and $H_k^P \subseteq H_{k+1}^P$. Therefore, they form two hierarchies inside NP.

The structure of the two hierarchies has a close connection with the structure of the polynomial-time hierarchy. Namely, for each $k \geqslant 0$, $\Sigma_k^P \neq \Pi_k^P$ implies $L_k^P \cap H_k^P = \varnothing$, and $\Sigma_k^P = \Pi_k^P$ implies $L_k^P = H_k^P = NP$. In addition, if the polynomia-time hierarchy is infinite, then there exist sets in NP which are not in the low hierarchy nor in the high hierarchy (Schöning, 1983). This relation reveals a rich structure within NP, assuming that the polynomial-time hierarchy is infinite.

The high hierarchy can be used as a classification of complete sets under various different notions of reducibility. For instance, it is shown in

* Research supported in part by the NSF Grant CCR-8801575.

0890-5401/91 $3.00

Schöning (1983) that H_0^P consists of exactly those sets which are $\leqslant_T^P$-complete for NP, and H_1^P consists of exactly those set which are $\leqslant_T^{SN}$-complete for NP, where $\leqslant_T^{SN}$ is the strong nondeterministic polynomial-time Turing reduction (Long, 1982).

The low hierarchy consists of sets which contain little information when used as oracles, and they are not likely to be complete for NP. For instance, sparse sets and sets solvable in polynomial-time by probabilistic algorithms (i.e., sets in BPP) have been shown not complete for NP unless the polynomial-time hierarchy collapses. Ko and Schöning (1985) gave better quantitative classification of these sets by showing that all sparse sets in NP are in L_2^P and all sets in NP which have polynomial-size circuits (including sets in $NP \cap BPP$) are in L_3^P. More recently, through the use of interactive proof systems, the problem of graph isomorphism is shown to locate in L_2^P (Goldwasser and Sipser, 1986; Schöning, 1987). Thus, not only that the graph isomorphism problem is not $\leqslant_m^P$-complete for NP, it actually cannot be $\leqslant_T^{SN}$-complete for NP, unless the polynomial-time hierarchy collapses to Σ_2^P.

As stated above, the sets L_k^P and sets H_k^P form two hierarchies, and they have a close relation with the polynomial-time hierarchy. However, some basic questions about the structure of the hierarchies remain unanswered. In particular, under what conditions do we have $L_k^P \neq L_{k+1}^P$ and $H_k^P \neq H_{k+1}^P$? We only know that the hierachies collapse if the polynomial-time hierarchy collapses, but do not know whether the converse holds. Furthermore, we do not even know, for instance, whether it is possible that the hierarchies are infinite but L_{k+1}^P collapses to L_k^P for some k.

In view of the difficulty of resolving the above questions about the basic structure of the hierarchies, we investigate the structure of the hierarchies in the relativized world. Relative to a set A, the low and high hierarchies $L_k^P(A)$ and $H_k^P(A)$ are defined as follows: a set $B \in NP(A)$ is in $L_k^P(A)$ if $\Sigma_k^P(B \oplus A) = \Sigma_k^P(A)$, and a set $C \in NP(A)$ is in $H_k^P(A)$ if $\Sigma_k^P(C \oplus A) = \Sigma_{k+1}^P(A)$, where $\oplus$ is the join operation on sets: $X \oplus Y = \{0x \mid x \in X\} \cup \{1y \mid y \in Y\}$. The main results of this paper are the following.

THEOREM A. *There exists an oracle A such that $L_k^P(A) \neq L_{k+1}^P(A)$ and $H_k^P(A) \neq H_{k+1}^P(A)$ for all $k \geqslant 0$.*

THEOREM B. *For every $k \geqslant 0$, there exists an oracle A such that $L_j^P(A) = L_k^P(A)$ for all $j > k$ and $L_j^P(A) \neq L_{j+1}^P(A)$ for all $j < k$ and $H_j^P(A) = H_k^P(A)$ for all $j > k$ and $H_j^P(A) \neq H_{j+1}^P(A)$ for all $j < k$.*

These results show a variety of possible structures of the low and high hierarchies. They do not, however, exhaust all possibilities. In particular, it

would be interesting to find out whether there exist oracles A_k such that $L_j^P(A_k) = L_0^P(A_k) = P$ for all $j \leqslant k$ but $L_{k+1}^P(A_k) \neq L_k^P(A_k)$. We remark that for $k = 1$, such an oracle A exists, as demonstrated by Baker, Gill, and Solovay (1975). However, their proof technique does not seem to generalize to our questions for $k > 1$. On the other hand, for the high hierarchy, we do not even know whether there exists an oracle B such that $H_0^P(B) = H_1^P(B)$ but $H_1^P(B) \neq H_2^P(B)$. We conjecture that such an oracle does exist and construct an oracle B relative to which $H_2^P(B) \neq H_1^P(B)$ and the class $H_1^P(B)$ *almost* collapses to $H_0^P(B)$.

Our proof technique is a modification of the proof technique developed by Yao (1985), Hastad (1987), and Ko (1989) for separating and collapsing relativized polynomial-time hierarchy. The use of this proof technique is necessary as any oracle A separating, for instance, $L_{k+1}^P(A)$ from $L_k^P(A)$ must also separate $\Sigma_{k+1}^P(A)$ from $\Sigma_k^P(A)$.

We give notations and formal definitions in Section 2. Also in Section 2, the lower bound results of Yao (1985) and Hastad (1987) on constant-depth circuits are stated. We prove the separating and collapsing results on the low hierarchy in Section 3 and the corresponding results on the high hierarchy in Section 4. Results on partially collapsing the hierarchies are discussed in Section 5, and some open questions are listed in Section 6.

2. Preliminaries

In this paper, all sets A are sets of strings over the alphabet $\{0, 1\}$. For each string x, let $|x|$ denote its *length*. Let $\{0, 1\}^n$ be the set of all strings of length n. We assume that there is a one-to-one *pairing function* $\langle , ..., \rangle$ such that for all i, $\langle x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n \rangle \leqslant \langle x_1, ..., x_{i-1}, y_i, x_{i+1}, ..., x_n \rangle$ if $x_i \leqslant y_i$. For each set A, let χ_A be its characteristic function. We write $A^{\leqslant n}$ to denote the set $\{x \in A \mid |x| \leqslant n\}$.

We assume that the reader is familiar with oracle Turing machines (TMs) and related complexity classes. In particular, we will work on the relativized complexity classes $P(A)$, $NP(A)$, $\Sigma_k^P(A)$, and $\Pi_k^P(A)$. The relativized polynomial-time hierarchy $PH(A) = \bigcup_{k=0}^{\infty} \Sigma_k^P(A)$ can be characterized by alternating quantifiers. Let $\sigma(A; x)$ be a predicate over a set variable A and a string variable x. We say that $\sigma(A; x)$ is a *P-predicate* if σ is computable in polynomial time by a deterministic oracle machine which uses set A as the oracle and takes string x as the input. Let $k \geqslant 1$. We say $\sigma(A; x)$ is a *Σ_k^P-predicate* if there exist a P-predicate σ' and a polynomial q such that for all sets A and all x with $|x| = n$, $\sigma(A; x)$ is true iff $(\exists y_1, |y_1| \leqslant q(n))(\forall y_2, |y_2| \leqslant q(n)) \cdots (Q_k y_k, |y_k| \leqslant q(n)) \sigma'(A; x, y_1, ..., y_k)$, where $Q_k = \exists$ if k is odd, and $Q_k = \forall$ if k is even. It is well known that a

set B is in $\Sigma_k^P(A)$ iff there exists a Σ_k^P-predicate σ such that for all x, $[x \in B \Leftrightarrow \sigma(A; x)]$. The relativized low and high hierarchies and the basic relations between the two hierarchies and the polynomial-time hierarchy have been given in Section 1.

For any complexity class $\mathscr{C}$, a set $A \in \mathscr{C}$ is $\leqslant_m^P$-*complte* for $\mathscr{C}$ if for every set $B \in \mathscr{C}$, there exists a polynomial-time computable function f such that for all x, $x \in B$ iff $f(x) \in A$. We will use some specific complete sets for these classes. First, we assume a fixed enumeration $\{\sigma_i^k\}$ of all Σ_k^P-predicates. We assume that the ith Σ_k^P-predicate $\sigma_i^k(A; x) \equiv (\exists y_1, |y_1| \leqslant d(n))(\forall y_2, |y_2| \leqslant q(n)) \cdots (Q_k y_k, |y_k| \leqslant q(n))\sigma'(A; x, y_1, ..., y_k)$ has the property that both the length-bounding polynomial q and the runtime of the deterministic oracle TM that computes the predicate σ' are bounded by the ith polynomial p_i.

Define, for each set A, the set $K^k(A)$ to be $\{0^i 1 z 1 0^j \mid \sigma_i^k(A; z)$ holds and $j \geqslant p_i(|z|)\}$. Then, it is obvious that $K^k(A)$ is complete for $\Sigma_k^P(A)$. Furthermore, for any string x, the question of whether $x \in K^k(A)$ depends only on the set $A^{<n} = \{y \in A \mid |y| < |x|\}$, because $x = 0^i 1 z 1 0^j$ implies $j < |x|$ and all queries made in the computation of $\sigma_i^k(A; z)$ must be of length $\leqslant p_i(|z|) \leqslant j < |x|$. (In other words, if B agrees with A on strings of length $< |x|$, then $x \in K^k(A)$ iff $x \in K^k(B)$.)

We will deal with *circuits* of unbounded fanins. In this paper, a circuit is formally defined as a tree. Each interior node of the tree is attached with an AND gate or an OR gate and has an unlimited number of child nodes. It is usually assumed that the gates alternate so that all children of an OR (or AND) gate are AND (or OR, respectively) gates. Each leaf is attached with a constant 0, a constant 1, a variable v, or a negated variable $\bar{v}$. Each circuit computes a boolean function on its variables. In this paper, each variable v is associated with a string $y \in \{0, 1\}^*$ and is denoted by v_y. The *depth* of a circuit is the length of the longest path in the tree. The *size* of a circuit is the number of gates (or, the number of interior nodes) in the tree. The *fanin* of a gate is the number of children of the node. The *bottom fanin* of a circuit is the maximum fanin of a gate of the lowest level in the tree.

Let V be the set of variables occurred in a circuit C. Then a *restriction* ρ of C is a mapping from V to $\{0, 1, *\}$. For each restriction ρ of C, $C\lceil_\rho$ denotes the circuit C' obtained from C by replacing each variable v_x with $\rho(v_x) = 0$ by 0 and each v_y with $\rho(v_y) = 1$ by 1. Assume that ρ' is a restriction of $C\lceil_\rho$. We write $C\lceil_{\rho\rho'}$ to denote $(C\lceil_\rho)\lceil_{\rho'}$. We also write $\rho\rho'$ to denote the combined restriction on C with values $\rho\rho'(x) = \rho(x)$ if $\rho(x) \neq *$ an with values $\rho\rho'(x) = \rho'(x)$ if $\rho(x) = *$. If a restriction ρ of C maps no variable to $*$, then we say ρ is an *assignment* of C. Let ρ be a restriction of C. We say that ρ *completely determines* C if $C\lceil_\rho$ computes a constant function 0 or 1. An assignment ρ of C always completely determines the

circuit C. A specific assignment ρ_A is defined for each set $A \subseteq \{0,1\}^*$: $\rho_A(v_y) = 1$ if $y \in A$ and $\rho_A(v_y) = 0$ if $y \notin A$.

There are some specific circuits which are useful in our proofs. One class of them is the circuits defining the functions f_k^m, which were first defined in Sipser (1983). Our definition of function f_k^m is a little different from that defined in Sipser (1983) and Hastad (1987). Let C_k^m be a depth-k circuit having the following properties:

(a) the top gate of C_k^m is an OR gate,

(b) the fanin of all bottom gates of C_k^m is $\sqrt{m}$,

(c) the fanin of all other gates is m, and

(d) there are $m^{k-1/2}$ variables each occurring exactly once in a leaf in the positive form.

Let the function computed by C_k^m be f_k^m.

One of the most important tools in constructing oracles to separate the polynomial-time hierarchy is to encode the computations of Σ_k^P-oracle machines into constant-depth circuits (Furst, Saxe, and Sipser, 1984; Sipser, 1983). This allows us to replace the often complicated arguments about computation trees of Σ_k^P-machines by simpler lower bound arguments about circuits (Yao, 1985; Hastad, 1987). In the following, we summarize these results.

LEMMA 2.1 (Furst, Saxe, and Sipser, 1984). *Let $k \geq 1$ and let $\sigma_i^k(A; x)$ be the ith Σ_k^P-predicate. Then, for each string x, there exists a circuit C having the following properties*:

(a) *the depth of C is $k+1$,*

(b) *the fanin of each gate in C is $\leq 2^{p_i(|x|)}$,*

(c) *the bottom fanin of C is $\leq 2p_i(|x|)$,*

(d) *the variables of C are strings which are queried in the computation of $\sigma_i^k(A; x)$ for all possible oracles A, and*

(e) *for each set A, $C\lceil_{\rho_A}$ outputs 1 iff $\sigma_i^k(A; x)$ holds.*

The main combinatorial lemma about circuits and relativization is first proved by Yao (1985) and then simplified and improved by Hastad (1987). The basic idea is to find a certain probability distribution such that a depth-2 circuit which is an AND of ORs with small bottom fanins, when applied by a random restriction ρ, has a high probability to be equivalent to a circuit which is an OR of ANDs with small bottom fanins. Furthermore, under the same probability distribution, a circuit computing a $f_k^{2^n}$ function, when applied by a random restriction, has a high probability that

the resulting circuit contains a subcircuit computing a $f_{k-1}^{2^n}$ function. Applying this lemma inductively, we obtain the following lemma which will be the main combinatorial tool for our construction of oracles.

LEMMA 2.2 (KO, 1989). *For every $k \geqslant 2$ there exists a constant n_k such that the following holds for all $n > n_k$. Let $t = n^{\log n}$, $m < 2^t$, and $C_0, C_1, ..., C_m$ be $m+1$ circuits each defining a $f_k^{2^n}$ function, with their variables pairwisely disjoint. Let C be a depth-k circuit of size $\leqslant 2^t$ having bottom fanins $\leqslant t$. Then, there exists a restriction ρ on C such that ρ completely determines C but it does not completely determine any C_i, $0 \leqslant i \leqslant m$.*

Remark. Lemma 2.2 also holds for the case $k = 1$, if C is a depth-2 circuit and is an AND of ORs with bottom fanins $\leqslant t$. It follows from a simple counting argument.

3. RELATIVIZED LOW HIERARCHY

In this section, we construct oracles to separate or collapse the low hierarchy. Oracles separating or collapsing the high hierarchy will be constructed in the next section. Theorems A and B then can be proved by combining these results together.

THEOREM 3.1. *There exists a set A such that for all $k \geqslant 0$, $L_k^P(A) \neq L_{k+1}^P(A)$.*

Proof. We first discuss informally the idea of the construction of set A. In order to prove that $L_k^P(A) \neq L_{k+1}^P(A)$, we need to find a set $B_k(A) \in NP(A)$ such that

(a) $\Sigma_{k+1}^P(B_k(A) \oplus A) = \Sigma_{k+1}^P(A)$ and

(b) $\Sigma_k^P(B_k(A) \oplus A) \neq \Sigma_k^P(A)$.

To satisfy condition (a), we would like to make the set $B_k(A)$ to behave like an empty set as much as possible, and, on the other hand, to satisfy condition (b), we need to make the set $B_k(A)$ to be similar to the set used in, for instance, Baker, Gill, and Solovay (1975) to separate $NP(A)$ from $P(A)$. Since the condition (b) can be satisfied by a delayed diagonalization, we will define set $B_k(A)$ as follows. First, let $e(0) = 1$, and $e(n+1) = 2^{2^{e(n)}}$ for all $n > 0$. Then, for each $k \geqslant 0$, let

$$B_k(A) = \{x \mid |x| = e(\langle k, m \rangle) \text{ for some } m, \text{ and } (\exists y, |y| = |x|)\, 0xy \in A\}.$$

It is obvious that $B_k(A) \in NP(A)$. Also, observe that any Σ_{k+1}^P-machine using oracle $B_k(A) \oplus A$ working on an input x of length far from $e(\langle k, m \rangle)$

for any m is at most as powerful as a Σ_{k+1}^P-machine using oracle A, and so condition (a) is partially satisfied. For some input x of length $e(\langle k, m\rangle)$ for some m, we will perform diagonalization to make a Σ_k^P-machine using oracle $B_k(A) \oplus A$ more powerful than a Σ_k^P-machine using oracle A; in the mean time we will still satisfy condition (a) so that a Σ_{k+1}^P-machine using oracle $B_k(A) \oplus A$ is no more powerful than a Σ_{k+1}^P-machine using oracle A, even if it is working on strings of length close to $e(\langle k, m\rangle)$ for some m. We note that the diagonalization part is similar to the diagonalization for $\Sigma_{k+1}^P(A) \neq \Sigma_k^P(A)$ (since $B_k(A) \in NP(A)$) and the collapsing part is similar to collapsing $\Sigma_{k+2}^P(A) = \Sigma_{k+1}^P(A)$. We use the technique of Ko (1989) (i.e., by Lemma 2.2) to do these two tasks in the same region. Note that for different k_1 and k_2, the use of different diagonalization regions $\{0,1\}^{e(\langle k_1, m\rangle)}$ and $\{0,1\}^{e(\langle k_2, l\rangle)}$ lets us avoid the possible conflict between the requirement (a) for k_1 and the requirement (b) for k_2, which conflict, if occurs, is difficult to resolve even with the tools of Yao (1985), Hastad (1987), and Ko (1989).

Now we give the formal proof. Let the function $e(n)$ and sets $B_k(A)$ be defined as above. We want to make $B_k(A)$ to satisfy both conditions (a) and (b). For condition (a), we first consider a set $D_k(A)$ in $\Sigma_{k+1}^P(A)$. Recall that $K^{k+1}(E \oplus A)$ is a complete set for the class $\Sigma_{k+1}^P(E \oplus A)$. In particular, there exist a polynomial q and a predicate $\sigma \in P$ such that $x \in K^{k+1}(E \oplus A)$ iff $(\exists u_1, |u_1| \leqslant q(|x|)) \cdots (Q_{k+1} u_{k+1}, |u_{k+1}| \leqslant q(|x|))$ $\sigma(E \oplus A; x, u_1, ..., u_{k+1})$. We replace the predicate σ by the predicate τ which behaves as follows:

On input $(x, u_1, ..., u_{k+1})$ and with oracle A, it first finds the integer n such that $2^{e(\langle k,n\rangle)} \leqslant |x| < 2^{e(\langle k,n+1\rangle)}$. Then, it simulates the computation of $\sigma(E \oplus A; x, u_1, ..., u_{k+1})$. During the simulation, each query $1y$ (i.e., query $y \in ? A$) is answered YES iff $y \in A$, and each query $0y$ (i.e., query $y \in ? E$) is answered as follows: if $|y| > e(\langle k, n\rangle)$ then answer NO; if $|y| \neq e(\langle k, m\rangle)$ for any m then answer NO; if $|y| = e(\langle k, m\rangle)$ for some $m \leqslant n$ then answer YES iff $(\exists z, |z| = |y|)\ 0yz \in A$.

Note that if $|y| = e(\langle k, m\rangle)$ for some $m \leqslant n$ then $|y| \leqslant \log |x|$ and so the predicate $[(\exists z, |z| = |y|)\ 0yz \in A]$ can be decided in polynomial time using oracle A. Therefore, the predicate τ is polynomial-time computable, and hence the set $D_k(A) =_{\text{defn}} \{x \mid (\exists u_1, |u_1| \leqslant q(|x|)) \cdots (Q_{k+1} u_{k+1}, |u_{k+1}| \leqslant q(|x|))\tau(A; x, u_1, ..., u_{k+1})\}$ is in $\Sigma_{k+1}^P(A)$. Furthermore, from the definition of $B_k(A)$ and the fact that whether $x \in K^{k+1}(E \oplus A)$ depends only on the set $(E \oplus A)^{\leqslant |x|}$, we have the following relation:

Fact 3.2. If $2^{e(\langle k,n\rangle)} \leqslant |x| < e(\langle k, n+1\rangle)$, then $x \in K^{k+1}(B_k(A) \oplus A)$ iff $x \in D_k(A)$.

Now we divide condition (a) into an infinite number of requirements

$R_{0,k,t}$, $t \geqslant 0$. For integer t such that $e(\langle k, n\rangle) \leqslant t < 2^{e(\langle k,n\rangle)}$ for some n, we let

$$R_{0,k,t}\colon (\forall x, |x| = t)[x \in K^{k+1}(B_k(A) \oplus A) \Leftrightarrow \pi_k(A; x)],$$

where $\pi_k(A; x)$ is the following Σ^P_{k+1}-predicate:

$$\pi_k(A; x) \equiv (\exists u_1, |u_1| = |x|)(\forall u_2, |u_2| = |x|) \cdots (Q_{k+1} u_{k+1}, |u_{k+1}| = |x|) 10^k 1 x u_1 u_2 \cdots u_{k+1} \in A.$$

For integer t such that $2^{e(\langle k,n\rangle)} \leqslant t < e(\langle k, n+1\rangle)$ for some n, we let

$$R_{0,k,t}\colon (\forall x, |x| = t)[x \in K^{k+1}(B_k(A) \oplus A) \Leftrightarrow x \in D_k(A)].$$

Clearly, if requirements $R_{0,k,t}$ are satisfied for all t, then $K^{k+1}(B_k(A) \oplus A) \in \Sigma^P_{k+1}(A)$ and so condition (a) is satisfied.

For condition (b), we first define sets $E_k(A)$ as follows. First, $E_0(A) = B_0(A) \cap \{0\}^*$. For $k > 0$, if k is even, then

$$E_k(A) = \{0^n \mid (\exists m)(\exists j, 0 \leqslant j < k)[kn + j = e(\langle k, m\rangle)] \text{ and } (\exists y_1, |y_1| = n) \cdots (Q_k y_k, |y_k| = n)\, y_1 \cdots y_k 0^j \in B_k(A)\}$$

and if k is odd, then

$$E_k(A) = \{0^n \mid (\exists m)(\exists j, 0 \leqslant j < k)[kn + j = e(\langle k, m\rangle)] \text{ and } (\exists y_1, |y_1| = n) \cdots (Q_k y_k, |y_k| = n)\, y_1 \cdots y_k 0^j \notin B_k(A)\}$$

It is clear that $E_k(A) \in \Sigma^P_k(B_k(A))$ for all $k \geqslant 0$. Since $B_k(A) \in NP(A)$, it follows that $E_k(A)$ is in $\Sigma^P_{k+1}(A)$. Now we divide the condition (b) into an infinite number of requirements $R_{1,k,i}$, $i \geqslant 0$. Recall that $\sigma^k_i(A; x)$ is the ith Σ^P_k-predicate:

$$R_{1,k,i}\colon (\exists n)[0^n \in E_k(A) \Leftrightarrow \sigma^k_i(A; 0^n) \text{ is false}].$$

If requirements $R_{1,k,i}$ are satisfied for all i then $E_k(A)$ is a witness for condition (b).

We will construct set A be stages. Before stage 1, we let $A(1) = A'(1) = \varnothing$. In each stage α, we will define sets $A(\alpha + 1)$ and $A'(\alpha + 1)$ such that $A(\alpha + 1)$ and $A'(\alpha + 1)$ are always disjoint and $A(\alpha + 1)$ is always an extension of $A(\alpha)$ and $A'(\alpha + 1)$ is always an extension of $A'(\alpha)$. We use $A'(\alpha + 1)$ to denote the set of strings reserved for $\bar{A}$, and will eventually define A to be the union of all $A(\alpha)$. (We do not reserve every string in $\bar{A}$; that is, the union of all $A(\alpha)$ is a subset of $\bar{A}$ but not necessarily equal to it.)

At an even stage 2α, $\alpha = \langle k, m \rangle$, we try to satisfy the requirement $R_{1,k,i}$ for the least i for which $R_{1,k,i}$ is not yet satisfied (i.e., the pair $\langle k, i \rangle$ is *uncancelled*). We choose $n = e(\alpha)$ if $k = 0$ and choose $n = \lfloor e(\alpha)/k \rfloor$ and $j = e(\alpha) - kn$, if $k \geq 1$. If $n < n_k$ (n_k is the constant of Lemma 2.2) or $2p_i(e(\alpha)) \geq n^{\log n}$ or $n^{\log n} \geq 2^{e(\alpha)}$ then let $A(2\alpha+1) = A(2\alpha)$ and $A'(2\alpha+1) = A'(2\alpha)$ and go to the next stage. Otherwise, if $n > n_k$ and $2p_i(e(\alpha)) < n^{\log n} < 2^{e(\alpha)}$ then we will satisfy requirement $R_{1,k,i}$ with 0^n as a witness. This part of construction is similar to the construction of the oracle X such that $\Sigma_{k+1}^{P}(X) = \Pi_{k+1}^{P}(X) \neq \Sigma_k^P(X)$ in Ko (1989). We consider three types of circuits:

(1) Let C be the circuit corresponding to the predicate $\sigma_i^k(A; 0^n)$ as described in Lemma 2.1, with a further modification that all variables v_y such that $y \in A(2\alpha) \cup A'(2\alpha)$ or $|y| < e(\alpha)$ are replaced by the values $\chi_{A(2\alpha)}(y)$. Then, C has depth $k+1$, with fanins $\leq 2^{2p_i(e(\alpha))} < 2^{n^{\log n}}$ and bottom fanins $\leq 2p_i(e(\alpha)) < n^{\log n}$. If $A(2\alpha) \subseteq A$, $A'(2\alpha) \subseteq \bar{A}$ and A agrees with $A(2\alpha)$ on strings of length $< e(\alpha)$, then $C\lceil_{\rho_A}$ outputs 1 iff $\sigma_i^k(A; 0^n)$ is true. (If $k = 0$, then we make C an AND of ORs with bottom fanins $\leq n^{\log n}$. This can be done as $\sigma_i^k(A; 0^n)$ is a P-predicate and hence a Π_1^P-predicate.)

(2) Let C_0 be the circuit corresponding to the Σ_{k+1}^P-predicate "$0^n \in E_k(A)$" as described in Lemma 2.1 such that for every set A, $C_0\lceil_{\rho_A} = 1$ iff $0^n \in E_k(A)$. From the definitions of $E_k(A)$ and $B_k(A)$, it is easy to see that all variables in C_0 are of the form v_{0y}, with $|y| = 2e(\alpha)$, such that each variable occurs at most once in C_0. That is, C_0 contains a subcircuit computing a function $f_{k+1}^{2^n}$.

(3) For each w of length $e(\alpha) \leq |w| < p_i(e(\alpha))$, let C_w be the circuit corresponding to the Σ_{k+1}^P-predicate $\pi_k(A; w)$. Then C_w has depth $k+1$, with fanins $\geq 2^{e(\alpha)}$ for all gates, and its variables are of the form v_y, $|y| = (k+2)|w|$ and $y \in 10^k 1\{0,1\}^*$, each occurring in C_w at most once. Thus, each C_w contains a subcircuit computing a $f_{k+1}^{2^n}$ function.

We note that by the condition $2p_i(e(\alpha)) < n^{\log n}$, there are at most $n^{\log n}$ many circuits C_0 and C_w. So, by Lemma 2.2 (and, in the case $k = 0$, by the remark following Lemma 2.2), we can find a restriction ρ on variables such that $C\lceil_\rho$ is completely determined but $C_0\lceil_\rho = C_w\lceil_\rho = *$ for all w, $e(\alpha) \leq |w| < p_i(e(\alpha))$. Since all variables in $C_0\lceil_\rho$ are of the form v_y with $y \in 0\{0,1\}^*$ and all variables in $C_w\lceil_\rho$ are of the form v_y with $y \in 1\{0,1\}^*$, we can find an assignment ρ' on variables in $C_0\lceil_\rho$ such that $C_0\lceil_{\rho\rho'}$ outputs differently from $C\lceil_{\rho\rho'}$ and yet $C_w\lceil_{\rho\rho'} = *$ for all w. We update the sets $A(2\alpha+1) = A(2\alpha) \cup \{y \mid \rho\rho'(v_y) = 1\}$ and $A'(2\alpha+1) = A'(2\alpha) \cup \{y \mid \rho\rho'(v_y) = 0\}$. Note that if $A(2\alpha+1) \subseteq A$ and $A'(2\alpha+1) \subseteq \bar{A}$ then $0^n \in E_k(A) \Leftrightarrow \sigma_i^k(A; 0^n)$ is false and the requirement $R_{1,k,i}$ is satisfied. We cancel pair $\langle k, i \rangle$ and go to the next stage.

At stage $2\alpha+1$, $\alpha=\langle k, m\rangle$, we will satisfy requirements $R_{0,j,t}$ for all j and all t such that $e(\alpha)\leqslant t<e(\alpha+1)$. We divide this stage into $e(\alpha+1)-e(\alpha)$ many substages: $\langle 2\alpha+1, t\rangle$, $e(\alpha)\leqslant t<e(\alpha+1)$. Each substage $\langle 2\alpha+1, t\rangle$ will satisfy requirements $R_{0,j,t}$ for all j.

To begin the construction of substage $\langle 2\alpha+1, e(\alpha)\rangle$, we set $X(e(\alpha))=A(2\alpha+1)$ and $X'(e(\alpha))=A'(2\alpha+1)$. Then, in each substage $\langle 2\alpha+1, t\rangle$, $e(\alpha)\leqslant t<2^{e(\alpha)}$, we determine, for every w of length t, whether $w\in K^{k+1}(B_k(X(t))\oplus X(t))$. Let C_w be the circuit corresponding to the predicate $\pi_k(A; w)$. We find an assignment ρ_w on variables of C_w such that $C_w\lceil_{\rho_w}$ outputs 1 iff $w\in K^{k+1}(B_k(X(t))\oplus X(t))$ and that it is consistent with the sets $X(t)$ and $X'(t)$, in the sense that $\rho_w(v_z)=1$ for all $z\in X(t)$ and $\rho_w(v_z)=0$ for all $z\in X'(t)$. We will show later that such an assignment ρ_w must exist. Let $Y(w)=\{1wz\mid |z|=(k+1)|w|, \rho_w(v_{1wz})=1\}$ and $Y'(w)=\{1wz\mid |z|=(k+1)|w|, \rho_w(v_{1wz})=0\}$. The substage $\langle 2\alpha+1, t\rangle$ is complete after we have done this for all w of length t and let $X(t+1)=X(t)\cup(\bigcup_{|w|=t} Y(w))$ and $X'(t+1)=X'(t)\cup(\bigcup_{|w|=t} Y'(w))$.

In substage $\langle 2\alpha+1, t\rangle$, $2^{e(\alpha)}\leqslant t<e(\alpha+1)$, we do nothing; i.e., let $X(t+1)=X(t)$ and $X'(t+1)=X'(t)$. At the end of substage $\langle 2\alpha+1, e(\alpha+1)-1\rangle$, let $A(2\alpha+2)=X(e(\alpha+1))$ and $A'(2\alpha+2)=X'(e(\alpha+1))$.

The set A is defined to be $\bigcup_{\alpha=1}^{\infty} A(\alpha)$.

First we prove the claim that in substage $\langle 2\alpha+1, t\rangle$, $e(\alpha)\leqslant t<2^{e(\alpha)}$, we can find, for each w of length t, an assignment ρ_w on variables of C_w such that $\rho_w(v_z)=1$ if $z\in X(t)$, $\rho_w(v_z)=0$ if $z\in X'(t)$ and $C_w\lceil_{\rho_w}=1$ iff $w\in K^{k+1}(B_k(X(t))\oplus X(t))$. We note that if v_z is a variable of C_w, $|w|=t$, then $z=10^k1ww'$ for some w' of length $(k+1)t$. So, v_z cannot be a variable of circuit C_u corresponding to any predicate $\pi_j(A; u)$, if $j\neq k$ or $u\neq w$. Thus, we know that if $z\in X(t)\cup X'(t)$ then it must have been added to them in stage 2α in which we defined restrictions ρ and ρ' of circuits C and C_0 and made $\rho\rho'(v_z)=*$. Therefore, we need only to find an assignment ρ_w such that ρ_w is consistent with $\rho\rho'$ and $C_w\lceil_{\rho_w}=1$ iff $w\in K^{k+1}(B_k(X(t))\oplus X(t))$. Since we have defined restrictions ρ and ρ' in such a way that $C_w\lceil_{\rho\rho'}$ is not completely determined, this assignment must exist.

Next we check our requirements. It is not hard to verify that requirements $R_{1,k,i}$ are satisfied for all k and all i. We need to show that requirements $R_{0,k,t}$ are also satisfied for all k and all t. For t and k satisfying $2^{e(\langle k,n\rangle)}\leqslant t<e(\langle k, n+1\rangle)$ for some n, Fact 3.2 shows that requirement $R_{0,k,t}$ is satisfied. So, we need only consider requirements $R_{0,k,t}$ with $e(\langle k,n\rangle)\leqslant t<2^{e(\langle k,n\rangle)}$ for some n; i.e., we need to show that for each w of length t, $w\in K^{k+1}(B_k(A)\oplus A)$ iff $\pi_k(A; w)$. Since the question of whether $w\in K^{k+1}(B_k(A)\oplus A)$ depends only on the set $(B_k(A)\oplus A)^{\leqslant t}$, and, since we do not add any string of length $\leqslant t$ or any string of the form $0yz$, $|y|=|z|\leqslant t$, to A after substage $\langle 2\alpha+1, t\rangle$, we have that $w\in K^{k+1}(B_k(A)\oplus A)$ iff $w\in K^{k+1}(B_k(X(t))\oplus X(t))$, where $X(t)$ is the set

$X(t)$ defined in the substage $\langle 2\alpha+1, t\rangle$. By the relation between circuit C_w and the predicate $\pi_k(A; w)$ and by the fact that in substage $\langle 2\alpha+1, t\rangle$ we have made $C_w\lceil_{\rho_w}$ output 1 iff $w \in K^{k+1}(B_k(X(t)) \oplus X(t))$, we conclude that requirements $R_{0,k,t}$ is satisfied if $e(\langle k, n\rangle) \leqslant t < 2^{e(\langle k,n\rangle)}$ for some n. This completes the proof of the theorem. ▮

THEOREM 3.3. *For every $k \geqslant 1$, there exists an oracle A such that $L_j^P(A) = NP(A)$ for all $j \geqslant k$ and $P_j^P(A) \neq L_{j+1}^P(A)$ for all $j < k$.*

Sketch of Proof. First note that if $\Sigma_k^P(A) = \Sigma_{k+1}^P(A)$ then $L_j(A) = NP(A)$ for all $j \geqslant k$. Therefore, we only need to construct A to satisfy $\Sigma_k^P(A) = \Sigma_{k+1}^P(A)$ and $L_j^P(A) \neq L_{j+1}^P(A)$ for all $j < k$. The proof technique is the same as that of Theorem 3.1. We define function $e(n)$, sets $B_j(A)$, $D_j(A)$, and $E_j(A)$, and predicates $\pi_j(A; x)$ exactly the same as in Theorem 3.1, for all $j < k$. To satisfy the condition that $L_j^P(A) \neq L_{j+1}^P(A)$ for $j < k$, we require the set A to satisfy the requirements $R_{0,j,t}$ and $R_{1,j,i}$ (as stated in the proof of Theorem 3.1) for all $j < k$ and for all t and i. To satisfy the condition that $\Sigma_k^P(A) = \Sigma_{k+1}^P(A)$, we further require the set A to satisfy $R_{0,k,t}$ for all t,

$$R_{0,k,t}\colon (\forall w, |w| = t)[w \in K^{k+1}(A) \Leftrightarrow \pi'_k(A; w)],$$

where $\pi'_k(A; w)$ is the Σ_k^P-predicate

$$\pi'_k(A; w) \equiv (\exists u_1, |u_1| = t) \cdots (Q_k u_k, |u_k| = t)10^k 1 w u_1 \cdots u_k \in A.$$

The construction of set A is similar to that for Theorem 3.1. We only sketch the idea here.

At stage 2α, where $\alpha = \langle j, m\rangle$ for some $j < k$, we try to satisfy requirement $R_{1,j,i}$ for the least i such that the pair $\langle j, i\rangle$ is uncancelled. To do this, we need to consider four types of circuits. Let $n = \lfloor e(\alpha)/j \rfloor$:

(1) Circuit C corresponding to the predicate $\sigma_i^j(A; 0^n)$ (with variables v_y such that $y \in A(2_\alpha) \cup A'(2\alpha)$ or $|y| < n$ replaced by values $\chi_{A(2\alpha)}(y)$),

(2) Circuit C_0 corresponding to predicate "$0^n \in E_j(A)$,"

(3) Circuit C_w, for each w of length $e(\alpha) \leqslant |w| < p_i(e(\alpha))$, corresponding to predicate $\pi_j(A; w)$, and

(4) Circuit C'_w, for each w of length $e(\alpha) \leqslant |w| < p_i(e(\alpha))$, corresponding to predicate $\pi'_k(A; w)$.

It is necessary to find a restriction ρ on variables such that $C\lceil_\rho$ and $C_0\lceil_\rho$ are completely determined and $C\lceil_\rho \neq C_0\lceil_\rho$, but $C_w\lceil_\rho$ and $C'_w\lceil_\rho$ are left undetermined for all w. Note that each circuit C_0 or C_w or C'_w, for any w of length $e(\alpha) \leqslant |w| < p_i(e(\alpha))$, contains a subcircuit computing a $f_{j+1}^{2^n}$

function; and circuit C is of depth $j+1$ and has small bottom fanins. So, it is ready to see from Lemma 2.2 that such a restriction ρ exists as long as n is large enough so that the fanins of C are bounded by $2^{n^{\log n}}$, the bottom fanins of C are bounded by $n^{\log n}$, and the number of circuits C_w and C'_w is bounded by $n^{\log n}$.

Thus, in stage 2α, we can make $C\lceil_\rho \neq C_0\lceil_\rho$, and hence satisfy requirement $R_{1,j,i}$ when sets $A(2\alpha+1)$ and $A'(2\alpha+1)$ are updated accordingly. Furthermore, since we have left all circuits $C_w\lceil_\rho$ and $C'_w\lceil_\rho$ undetermined, and they have pairwisely disjoint variables, we can satisfy requirements $R_{0,j,t}$ and $R_{0,k,t}$ for all $j<k$ and for all t such that $e(\alpha)\leqslant t<e(\alpha+1)$ in the next stage $2\alpha+1$. So, the actions in stage $2\alpha+1$ are almost the same as that in the proof of Theorem 3.1. The only difference is that in substage $\langle 2\alpha+1, t\rangle$, $2^{e(\alpha)}\leqslant t<e(\alpha+1)$, we need to satisfy requirement $R_{0,k,t}$ by considering circuits $C'_{w'}$ for all w' of length t. Note that $w\in K^{k+1}(A)$ depends only on set $A^{\leqslant|w|}$, and hence requirement $R_{0,k,t}$ can be satisfied easily. Since the variables in $C'_{w'}$ are those v_z's with $z\in 10^k1w\{0,1\}^*$, the assignement of these z's into $A(2\alpha+2)$ or $A'(2\alpha+2)$ will not affect the construction of the next stage. ▮

4. Relativized High Hierarchy

Now we show similar results for the high hierarchy. The proofs for the high hierarchy are quite similar to those for the low hierarchy, and we only give sketches.

Theorem 4.1. *There exists a set A such that $H^P_{k+1}(A)\neq H^P_k(A)$ for all $k\geqslant 0$.*

Proof. In order to prove that $H^P_k(A)\neq H^P_{k+1}(A)$, we need to find a set $B_k(A)\in NP(A)$ such that

(a) $\Sigma^P_{k+1}(B_k(A)\oplus A)=\Sigma^P_{k+2}(A)$, and

(b) $\Sigma^P_k(B_k(A)\oplus A)\neq\Sigma^P_{k+1}(A)$.

Define a function e by $e(0)=1$ and $e(n+1)=2^{2^{e(n)}}$. We want to perform diagonalization for condition (b) by strings of length $e(\langle k,m\rangle)$ for some m. For strings far from the diagonalization region, we make $B_k(A)$ to be the same as an $NP(A)$-complete set $K(A)$ so that condition (a) is satisfied. For strings close to the diagonalization region, we make $B_k(A)$ to be the same as the empty set so that condition (b) can be satisfied by the same technique for separating $\Sigma^P_{k+1}(A)$ from $\Sigma^P_k(A)$. In the mean time, to satisfy condition (a) in the diagonalization region, we let $K^{k+2}(A)$ in this region be computable in $\Sigma^P_{k+1}(A)$.

Recall that $K(A)$ is an $NP(A)$-complete set with a special padding included in the inputs: $K(A)=\{0^i1w10^j \mid j \geqslant p_i(|w|)$ and the ith nondeterministic oracle TM N_i accepts w, with oracle A (in $\leqslant p_i(|w|)$ moves)$\}$. Let

$$B_k(A)=\{x \mid 2^{e(\langle k,n-1\rangle)} \leqslant |x| < e(\langle k,n\rangle) \text{ for some } n \text{ and } x \in K(A)\}.$$

Then it is clear that $B_k(A) \in NP(A)$.

To satisfy condition (a), we define a set $D_k(A) \in \Sigma^P_{k+1}(B_k(A) \oplus A)$ as follows. Since $K^{k+1}(X) \in \Sigma^P_{k+1}(X)$, there exist a polynomial q and a P-predicate σ such that $x \in K^{k+1}(X)$ iff $(\exists u_1, |u_1| \leqslant q(|x|)) \cdots (Q_{k+1}u_{k+1}, |u_{k+1}| \leqslant q(|x|))\sigma(X; x, u_1, ..., u_{k+1})$. We replace the predicate σ by the predicate τ which behaves as follows:

On input $(x, u_1, ..., u_{k+1})$ and with oracle $B \oplus A$, it first finds the integer n such that $2^{e(\langle k,n\rangle)} \leqslant |x| < 2^{e(\langle k,n+1\rangle)}$. Then, it simulates the computation of $\sigma(X; x, u_1, ..., u_{k+1})$. During the simulation, each query "$y \in ? X$" is answered as follows: if y is not of the form 0^i1z10^j, with $j \geqslant p_i(|z|)$, then answer NO; otherwise, assume that $y=0^i1z10^j$, $j > p_i(|z|)$, and consider three cases:

Case 1. If $e(\langle k,n\rangle) \leqslant |y| < 2^{e(\langle k,n\rangle)}$, then let $y'=0^i1z10^{j'}$ such that $i+j'+|z|+2=2^{e(\langle k,n\rangle)}$ and answer YES iff $y' \in B$.

Case 2. If $2^{e(\langle k,n\rangle)} \leqslant |y|$ then answer YES iff $y \in B$.

Case 3. If $|y| < e(\langle k,n\rangle)$, then use oracle A to determine whether $y \in K(A)$ and answer YES iff $y \in K(A)$.

Note that in Case 1, $|y'|=2^{e(\langle k,n\rangle)} \leqslant |x|$ and so the question of whether $y' \in B$ can be determined in $|y'| \leqslant |x|$ steps. Also, in Case 3, $|y| < e(\langle k,n\rangle) \leqslant \log |x|$, and so the question of whether $y \in K(A)$ can be determined in only $2^{c|y|} \leqslant |x|^c$ steps for some constant c (because $y=0^i1z10^j$ with $j \geqslant p_i(|z|)$ and so the runtime for the ith nondeterministic machine N_i on z is bounded by $|y|$ and so $N_i(y)$ can be simulated by a deterministic machine in time $2^{c|y|}$). This means that τ is polyomial-time computable.

Now we define set $D_k(A)$ to be $\{x \mid (\exists u_1, |u_1| \leqslant q(|x|)) \cdots (Q_{k+1}u_{k+1}, |u_{k+1}| \leqslant q(|x|))\tau(B_k(A) \oplus A; x, u_1, ..., u_{k+1})\}$. It is clear that $D_k(A)$ is in $\Sigma^P_{k+1}(B_k(A) \oplus A)$. Furthermore, we claim that

FACT 4.2. *If* $2^{e(\langle k,n\rangle)} \leqslant |x| < e(\langle k, n+1\rangle)$ *for some* n, *then* $x \in K^{k+1}(K(A)) \Leftrightarrow x \in D_k(A)$.

Proof. The above algorithm of $\tau(B \oplus A; x, u_1, ..., u_{k+1})$, with $B=B_k(A)$, is intended to be a simulation of $\sigma(K(A); x, u_1, ..., u_{k+1})$. We need only to show that this simulation is correct for x if $2^{e(\langle k,n\rangle)} \leqslant |x| < e(\langle k,n+1\rangle)$.

In Case 1, we replace y by y' such that $y \in K(A)$ iff $y' \in K(A)$ and answer

YES iff $y' \in B_k(A)$. Since $|y'| = 2^{e(\langle k,n\rangle)}$, by the definition of $B_k(A)$, $y' \in K(A)$ iff $y' \in B_k(A)$. So the simulation in this case is correct. In Case 2, since $K^{k+1}(K(A))$ asks query y only if $|y| \leqslant |x|$, we have $2^{e(\langle k,n\rangle)} \leqslant |y| \leqslant |x| < e(\langle k, n+1\rangle)$. Therefore, by the definition of $B_k(A)$, $y \in B_k(A)$ iff $y \in K(A)$, and the simulation in this case is correct. The correctness of Case 3 is trivial. ▮

We now specify our requirements for condition (a). For t satisfying $2^{e(\langle k,n\rangle)} \leqslant t < e(\langle k, n+1\rangle)$ for some n, we require

$$R_{0,k,t}\colon (\forall w, |w| = t)\, w \in K^{k+1}(K(A)) \Leftrightarrow x \in D_k(A).$$

For t satisfying $e(\langle k, n\rangle) \leqslant t < 2^{e(\langle k,n\rangle)}$ for some n, we require

$$R_{0,k,t}\colon (\forall w, |w| = t)\, w \in K^{k+1}(K(A)) \Leftrightarrow w \in \pi_k(A; w),$$

where $\pi_k(A; w)$ is the Σ^P_{k+1}-predicate

$$\pi_k(A; w) \equiv (\exists u_1, |u_1| = |w|) \cdots (Q_{k+1} u_{k+1}, |u_{k+1}| = |w|)$$
$$10^k 1 w u_1 \cdots u_{k+1} \in A.$$

From Fact 4.2 and the fact that $\pi_k(A; w)$ is a Σ^P_{k+1}-predicate, it is clear that if $R_{0,k,t}$ is satisfied for all t then $K^{k+1}(K(A)) \in \Sigma^P_{k+1}(B_k(A) \oplus A)$.

For condition (b), we let

$$E_k(A) = \{0^n \mid (\exists y_1, |y_1| = n) \cdots (Q_{k+1} y_{k+1}, |y_{k+1}| = n) 0 y_1 \cdots y_{k+1} \in A\}.$$

It is obvious that $E_k(A) \in \Sigma^P_{k+1}(A)$. Recall that σ^k_i is the ith Σ^P_k-predicate over a set variable and a string variable. Our requirements for condition (b) are

$$R_{1,k,i}\colon (\exists n)[0^n \in E_k(A) \Leftrightarrow \sigma^k_i(B_k(A) \oplus A; 0^n) \text{ is false}].$$

It is clear that if $R_{1,k,i}$ is satisfied for all i then $E_k(A) \notin \Sigma^P_k(B_k(A) \oplus A)$ and hence $\Sigma^P_{k+1}(A) \not\subseteq \Sigma^P_k(B_k(A) \oplus A)$.

We will construct set A by stages like in the proof of Theorem 3.1. Sets $A(\alpha)$ and $A'(\alpha)$ are used in the same way as in that proof. Before stage 1, we let $A(1) = A'(1) = \varnothing$.

At an even stage 2α, $\alpha = \langle k, m\rangle$ for some m, we try to satisfy the requirement $R_{1,k,i}$ for the least i such that pair $\langle k, i\rangle$ is uncancelled. We choose $n = e(\alpha)$. If $n < n_k$ (n_k is the constant of Lemma 2.2) or $2p_i(n) \geqslant n^{\log n}$ or $n^{\log n} \geqslant 2^n$ then go to the next stage. Otherwise, if $n > n_k$ and $2p_i(n) < n^{\log n} < 2^n$ then we wil satisfy requirement $R_{1,k,i}$ with 0^n as a witness. We consider three types of circuits:

(1) Let C' be the circuit corresponding to the predicate $\sigma_i^k(B_k(A) \oplus A; 0^n)$, as described in Lemma 2.1. Then C has depth $k+1$, fanins $\leqslant 2^{2p_i(n)}$ and bottom fanins $\leqslant 2p_i(n)$. Replace each variable v_{0y} with $|y| < n$ by the value $\chi_{B_k(A(2\alpha))}(y)$ (note that $B_k(A(2\alpha)) = B_k(A)$ if A agrees with $A(2\alpha)$ on strings of length $< n$); replace each variable v_{0y} with $|y| \geqslant n$ by the value 0 (note that $B_k(A) \cap \{x \mid n \leqslant |x| < 2^n\} = \varnothing$); replace each variable v_{1y} with $|y| < n$ or $y \in A(2\alpha) \cup A'(2\alpha)$ by the value $\chi_{A(2\alpha)}(y)$; and then replace all other variables v_{1y} by variable v_y. (For each negated variable $\bar{v}_y$, replace it by the negation of the constants or circuits described above.) Let the resulting circuit by C. Then, for any set A such that $A(2\alpha) \subseteq A$, $A'(2\alpha) \subseteq \bar{A}$ and that A agrees with set $A(2\alpha)$ on strings of length $< n$ then $C\lceil_{\rho_A}$ outputs 1 iff $\sigma_i^k(B_k(A) \oplus A; 0^n)$ holds.

(2) Let C_0 be the circuit corresponding to the predicate "$0^n \in E_k(A)$." Then, C_0 computes a $f_{k+1}^{2^n}$ function, whose variables are of the form v_{0z}, $|z| = (k+1)n$.

(3) For each w of length $n \leqslant |w| < p_i(n)$, let C_w be the circuit corresponding to the predicate $\pi_k(A; w)$. Then C_w computes a $f_{k+1}^{2^n}$ function such that $C_w\lceil_{\rho_A}$ outputs 1 iff $\pi_k(A; w)$ holds. Note that all its variables are of the form 10^k1wz for some z.

Now apply Lemma 2.2 to these circuits and obtain a restriction ρ such that $C\lceil_\rho \neq *$ and $C_0\lceil_\rho = C_w\lceil_\rho = *$ for all w, $n \leqslant |w| < p_i(n)$. Note that $C_0\lceil_\rho$ and $C_w\lceil_\rho$ have disjoint variables, for each w. So, we can find a restriction ρ' such that $C_0\lceil_{\rho\rho'} \neq *$ and $C\lceil_{\rho\rho'} \neq C_0\lceil_{\rho\rho'}$ but $C_w\lceil_{\rho\rho'} = *$ for all w. We define $A(2\alpha+1) = A(2\alpha) \cup \{y \mid \rho\rho'(v_y) = 1\}$ and $A'(2\alpha+1) = A'(2\alpha) \cup \{y \mid \rho\rho'(v_y) = 0\}$, anc cancel pair $\langle k, i\rangle$. This completes stage 2α.

At stage $2\alpha+1$, similarly to the stage $2\alpha+1$ in the proof of Theorem 3.1, we satisfy requirements $R_{0,j,t}$ for all t, $e(\alpha) \leqslant t < e(\alpha+1)$, and for all j. At substage $\langle 2\alpha+1, t\rangle$, we find new restrictions ρ_w for each w of length $n \leqslant |w| < 2^n$ such that $C_w\lceil_{\rho\rho'\rho_w}$ outputs 1 iff $w \in K^{k+1}(K(X(t)))$, where $X(t)$ is the set of strings reserved for A by the beginning of substage $\langle 2\alpha+1, t\rangle$; i.e., $X(t) = A(2\alpha+1) \cup \{y \mid \rho_u(v_y) = 1$ for some u, $e(\alpha) \leqslant |u| < t\}$. These restrictions ρ_w must exist because we have left each $C_w\lceil_{\rho\rho'}$ undetermined in stage 2α and because all of C_w's have pairwisely disjoint variables. We let $A(2\alpha+2) = A(2\alpha+1) \cup \{y \mid \rho_w(v_y) = 1$ for some $w, n \leqslant |w| < 2^n\}$ and $A'(2\alpha+2) = A'(2\alpha+1) \cup \{y \mid \rho_w(v_y) = 0$ for some $w, n \leqslant |w| < 2^n\}$. This completes stage $2\alpha+1$.

Let $A = \bigcup_{\alpha=1}^{\infty} A(\alpha)$. It is not hard to verify by the arguments similar to those in the proof of Theorem 3.1 that all requirements $R_{1,k,i}$ and $R_{0,k,t}$ have been satisfied. In particular, Fact 4.2 implies that requirements $R_{0,k,t}$ are satisfied if $2^{e(\langle k,m\rangle)} \leqslant t < e(\langle k, m+1\rangle)$. For requirements $R_{0,k,t}$ with $e(\langle k, m\rangle) \leqslant t < 2^{e(\langle k,m\rangle)}$, we note that in stage $2\alpha+1$, $\alpha = \langle k, m\rangle$, each circuit C_w, which is corresponding to the predicate $\pi_k(A; w)$, has been

made to output $C_w\lceil_{\rho_A} = 1$ iff $w \in K^{k+1}(K(X(t)))$. So these requirements are all satisfied because we do not add any string of length $\leqslant |w|$ to A after this stage. We leave it to the reader to verify the details. ▮

THEOREM 4.3. *For every $k \geqslant 1$, there exists an oracle A such that $H_j^P(A) = NP(A)$ for all $j \geqslant k$ and $H_j^P(A) \neq H_{j+1}^P(A)$ for all $j < k$.*

Sketch of Proof. The proof is a modification of the proof of Theorem 4.1, in a way similar to the modification of the proof of Theorem 3.1 for Theorem 3.3. We need to construct a set A such that $\Sigma_k^P(A) = \Sigma_{k+1}^P(A)$ (and so $H_j^P(A) = NP(A)$ for all $j \geqslant k$) and $H_j^P(A) \neq H_{j+1}^P(A)$ for all $j < k$.

We define function $e(n)$, sets $B_j(A)$, $D_j(A)$, $E_j(A)$ and predicate $\pi_j(A; x)$, for all $j < k$, as in the proof of Theorem 4.1. For each $j < k$ and each t and i, we also make requirements $R_{0,j,t}$ and $R_{1,j,i}$ exactly the same as in that proof. In addition, we require $R_{0,k,t}$, for all t, as in the proof of Theorem 3.3.

The only thing needs to be checked in the stage construction of set A is the diagonalization step for $R_{1,j,i}$ in stage 2α, where $\alpha = \langle j, m \rangle$ for some $j < k$ and some m. In this stage, we let $n = e(\alpha)$ and consider four types of circuits:

(1) circuit C corresponding to predicate $\sigma_i^j(B_j(A) \oplus A; 0^n)$ with the similar modification as in Theorem 4.1;

(2) circuit C_0 corresponding to predicate "$0^n \in E_j(A)$;"

(3) circuit C_w, for each w of length $n \leqslant |w| < p_i(n)$, corresponding to the predicate $\pi_j(A; w)$; and

(4) circuit C'_w, for each w of length $n \leqslant |w| < p_i(n)$, corresonding to the predicate $\pi'_k(A; w)$.

Note that all the variables in circuits C_0, C_w, C'_w are pairwisely disjoint, and each of these circuits contains a subcircuit computing a $f_{j+1}^{2^n}$ function. Therefore, by Lemma 2.2, we can find a restriction ρ which completely determines the circuits C and C_0 but none of C_w's or C'_w's and $C\lceil_\rho \neq C_0\lceil_\rho$. This restriction gives us the desired diagonalization for requirement $R_{1,j,i}$.

In an odd stage $2\alpha + 1$, $\alpha = \langle k, m \rangle$, we satisfy requirements $R_{0,j,t}$ and $R_{0,k,t}$ for $j < k$ and for the t such that $e(\alpha) \leqslant t < e(2\alpha)$. More precisely, we find for each w, $e(\alpha) \leqslant w < 2^{e(\alpha)}$, ρ_w such that $C_w\lceil_{\rho\rho_w} = 1$ iff $w \in K^{j+1}(K(A))$, and for each w, $e(\alpha) \leqslant w < e(\alpha + 1)$, ρ'_w such that $C'_w\lceil_{\rho\rho'_w} = 1$ iff $w \in K^{k+1}(K(A))$. By the fact that the variables of C_w's and C'_w's are pairwisely disjoint, these restrictions can be found without interference from each other. Thus all requirements can be satisfied. ▮

We now combine the above results and results of Section 3 to prove the main theorems.

THEOREM A. *There exists an oracle A such that $L_k^P(A) \neq L_{k+1}^P(A)$ and $H_k^P(A) \neq {}^P_{k+1}(A)$ for all $k \geqslant 0$.*

Proof. We need to alternate the diagonalizations in the proofs of Theorems 3.1 and 4.1, so that the encoding requirements of one proof does not affect the diagonalization requirements of the other proof. We let $B_{2k}(A)$ be the set $B_k(A)$ defined in Theorem 3.1, and $B_{2k+1}(A)$ be the set $B_k(A)$ defined in Theorem 4.1, with some modification so that the "diagonalization region" of them are pairwisely disjoint. More precisely,

$$B_{2k}(A) = \{x \mid |x| = e(\langle 2k, n\rangle) \text{ for some } n \text{ and } (\exists y, |y| = |x|)\, 0xy \in A\}$$

and

$$B_{2k+1}(A) = \{x \mid 2^{e(\langle 2k+1, n-1\rangle)} \leqslant |x| < e(\langle 2k+1, n\rangle) \text{ for some } n \text{ and } x \in K(A)\}.$$

Then, we construct set A just like what we did in the proofs of Theorems 3.1 and 4.1. In particular, in stage 2α with $\alpha = \langle 2k, m\rangle$ for some m, we satisfy the requirement $R_{1,k,i}$ of Theorem 3.1 for the least i such that pair $\langle 2k, i\rangle$ is uncancelled, and in stage 2α with $\alpha = \langle 2k+1, m\rangle$ for some m, we satisfy the requirement $R_{1,k,i}$ of Theorem 4.1 for the least i such that pair $\langle 2k+1, i\rangle$ is uncancelled. Then it is easy to check that no interference between different requirements may occur. For example, assume that at some stage 2α, $\alpha = \langle 2k+1, m\rangle$ for some m, we want to satisfy requirement $R_{1,k,i}$ of Theorem 4.1 for some i. The diagonalization region is $\{x \mid e(\alpha) \leqslant |x| < 2^{e(\alpha)}\}$. Then, certainly there will be no conflict between this requirement and any requirement $R_{1,j,i'}$ of Theorem 3.1, because they have different diagonalization regions. In addition, there will be no conflict between this requirement and requirement $R_{0,j,t}$ of Theorem 3.1 for any j and any t. This is true because for integer t such that $e(\alpha) \leqslant t < 2^{e(\alpha)}$, we must have $2^{e(\langle 2j,l\rangle)} \leqslant t < e(\langle 2j, l+1\rangle)$ for some l. Thus, for the requirement $R_{0,j,t}$ of Theorem 3.1, we need to make $(\forall x, |x| = t)x \in K^{j+1}(B_{2j}(A) \oplus A) \Leftrightarrow x \in D_{2j}(A)$. (The set $D_{2j}(A)$ here is the set $D_j(A)$ in Theorem 3.1.) But this clearly holds by Fact 3.2. So, the process of diagonalization for requirement $R_{1,k,i}$ of Theorem 4.1 can be done in exactly the same way as in the proof of Theorem 4.1. This shows that the constructions in the two proofs can be made independent of each other and hence the theorem is proven. ∎

THEOREM B. *For every $k \geqslant 0$, there exists an oracle A such that $L_j^P(A) = L_k^P(A)$ for all $j > k$ and $L_j^P(A) \neq L_{j+1}^P(A)$ for all $j < k$ and $H_j^P(A) = H_k^P(A)$ for all $j > k$ and $H_j^P(A) \neq H_{j+1}^P(A)$ for all $j < k$.*

Proof. Using the same argument, we can see that Theorems 3.3 and 4.3 can be combined without interference. We omit the proof. ∎

5. Partial Collapsing Results

We have pointed out in Section 1 that Theorems A and B do not exhasut all possible structures of the low and high hierarchies. For instance, we cannot rule out the possibility that the hierarchies are infinite an yet the kth level and the $(k+1)$th level of the hierarchies coincide for some $k>0$. Constructing oracles to have these properties appears to be very difficult. In this section we show some weak partial collapsing results.

First, we observe that the following interesting result of Baker, Gill, and Solovay (1975) implies a partial collapsing of the low hierarchy.

THEOREM 5.1. *There exists an oracle A such that $P(A)=NP(A)\cap co\text{-}NP(A)\neq NP(A)$. Furthermore, the set A can be constructed such that $A=Q\oplus S$ for some PSPACE-complete set Q and some sparse set S.*

Long and Selman (1986) and Balcázer, Book, and Schöning (1986) have shown that if S is a sparse set then $\Sigma_2^P(S)=\Pi_2^P(S)$ iff $\Sigma_2^P=\Pi_2^P$. This result is easy to see relativizable to any oracle Q.

LEMMA 5.2. *Let Q be a PSPACE-complete set and S a spare set. Then, $\Sigma_2^P(Q\oplus S)=\Pi_2^P(Q\oplus S)$.*

So, we have obtained the following interesting structure of the low hierarchy:

COROLLARY 5.3. *There exists an oracle A such that $P(A)=L_0^P(A)=L_1^P(A)\neq L_2^P(A)=NP(A)$.*

Proof. The theorem follows from Theorem 5.1 and Lemma 5.2 by noting that $L_0^P(A)=P(A)$, $L_1^P(A)=NP(A)\cap co\text{-}NP(A)$, and that $\Sigma_2^P(A)=\Pi_2^P(A)$ implies $L_2^P(A)=NP(A)$. ∎

Unfortunately, as Lemma 5.2 points out, this technique can be used only for collapsing $L_1^P(A)$ to $L_0^P(A)$ and cannot be generalized to $L_k^P(A)$ for $k>1$. Moreover, for the high hierarchy, even the collapsing of $H_1^P(A)$ to $H_0^P(A)$ seems difficult. Using Baker, Gill, and Solovay's proof technique, we are only able to prove a weaker result. For each set $B\in NP(A)$, we say that B' is a *prefix set* of B if there exist a polynomial q and a polynomial-time predicate σ such that $B=\{x\mid(\exists y,\ |y|\leqslant q(|x|))\,\sigma(A;x,y)\}$ and $B'=\{\langle x,u\rangle\mid(\exists v,\ |uv|\leqslant q(|x|))\sigma(A;x,uv)\}$. It is well known that set B is $\leqslant_T^P$-reducible to its prefix sets B' but the converse is not known. On the other hand, every known *natural* NP-complete problem is $\leqslant_T^P$-equivalent to a prefix set.

THEOREM 5.4. *There exists an oracle A such that $H_2^P(A) = NP(A) \neq H_1^P(A)$ and all prefix sets of a set $B \in H_1^P(A)$ are in $H_0^P(A)$.*

Proof. We need to find an oracle A such that $P(A) \neq NP(A)$ and for every set $B \in H_1^P(A)$, $K(A) \in P(B' \oplus A)$, if B' is a prefix set of B. We follow the idea of Theorem 5.1 to construct the oracle A; in particular, we make A to be equal to $Q \oplus S$ for some *PSPACE*-complete set Q and some sparse set S.

Let $\{N_i\}$ be an enumeration of all polynomial-time nondeterministic oracle machines with the runtime of N_i bounded by polynomial p_i. We let $L(N_i, A)$ denote the set accepted by N_i using oracle A. We define the following requirements:

$$R_{0,i}\colon (\exists n)[(\exists x)\, |x| = n \text{ and } 1x \in A \Leftrightarrow N_i^A \text{ accepts } 0^n],$$

$$R_{1,\langle j,k\rangle}\colon (\exists x)[x \in K(A) \Leftrightarrow N_j \text{ accepts } x \text{ with oracle } L(N_k, A) \oplus A].$$

If requirements $R_{0,i}$ are satisfied for all i, then they imply that the set $L(A) = \{0^n \mid (\exists x, |x| = n) 1x \in A\}$ is in $NP(A) - co\text{-}NP(A)$. Our construction will make A satisfy the requirements $R_{0,i}$ for all i.

For any fixed k, if requirements $R_{1,\langle j,k\rangle}$ are satisfied for all j, then $\overline{K(A)} \neq L(N_j, L(N_k, A) \oplus A)$ for all j. This implies that $K(A) \notin co\text{-}NP(L(N_k, A) \oplus A)$, or equivalently, $L(N_k, A)$ is not in $H_1^P(A)$. In the following construction, we will prove that if $L(N_k, A)$ is in $H_1^P(A)$, and hence requirement $R_{1,\langle j,k\rangle}$ is not satisfied for some j, then $K(A) \in P(B_k \oplus A)$ for any set B_k which is a prefix set of $L(N_k, A)$. Therefore, for any k, either $L(N_k, A) \notin H_1^P(A)$ or $B_k \in H_0^P(A)$ for all prefix sets B_k of $L(N_k, A)$.

Define a function e by $e(0) = 1$ and $e(n+1) = 2^{2^{e(n)}}$. Let $A(0) = \{0y \mid y \in Q\}$, where Q is a fixed *PSPACE*-complete set.

For each n, we say $\alpha = 2i$ is *vulnerable* at stage n if $2^{e(n)-1} > p_i(e(n))$ and $\alpha = 2i+1$ is *vulnerable* at stage n if $i = \langle j, k\rangle$ and $p_j(2^{e(n)}) < e(n+1)$ and $p_k(2^{e(n)}) < e(n+1)$. In stage n we consider the least uncancelled vulnerable integer α. If $\alpha = 2i$ then we try to satisfy requirement $R_{0,i}$, and if $\alpha = 2i+1$, then we try to satisfy requirement $R_{1,\langle j,k\rangle}$, where $\langle j, k\rangle = i$. If the requirement corresponding to the least α is not satisfied, then we try to satisfy the requirement corresponding to the next least uncancelled vulnerable α until either we satisfy some requirement and cancel some integer β or until there is no vulnerable integer for stage n.

In the first case when $\alpha = 2i$, then $2^{e(n)-1} > p_i(e(n))$ and we will satisfy requirement $R_{0,i}$. We simulate the computation of $N_i^{A(n)}(0^{e(n)})$. If it rejects then we let $A(n+1) = A(n)$; it it accepts, we search for $1x \in \{0,1\}^{e(n)}$ which is not queried in an accepting computation of $N_i^{A(n)}(0^{e(n)})$ and let $A(n+1) = A(n) \cup \{1x\}$. Then we cancel $\alpha = 2i$.

In the second case when $\alpha = 2\langle j, k\rangle + 1$, then $p_j(2^{e(n)}) < e(n+1)$ and $p_k(2^{e(n)}) < e(n+1)$ and we check whether there exists an x, $2^{e(n-1)} \leqslant |x| < 2^{e(n)}$, such that

$$x \in K(A(n)) \Leftrightarrow N_j(x) \quad \text{rejects with oracle} \quad L(N_k, A(n)) \oplus A(n).$$

If so, we cancel α and let $A(n+1) = A(n)$. If no such x exists, then we declare that we did not satisfy requirement $R_{1,\langle j,k\rangle}$ and continue to consider the next uncancelled α.

We let $A = \bigcup_{n=0}^{\infty} A(n)$. It is clear that every requirement $R_{0,i}$ is eventually satisfied since each 2α is eventually vulnerable. We claim that if $\overline{K(A)} = L(N_j, L(N_k, A) \oplus A)$ then there is a fixed polynomial-time algorithm solving the problem $x \in ? K(A)$ using oracle $B_k \oplus A$ for any set B_k which is a prefix set of $L(N_k, A)$. The algorithm works as follows:

ALGORITHM FOR $K(A)$. Assume that by some stage n_0, all integers less than $\alpha = 2\langle j, k\rangle + 1$ which are to be eventually cancelled are already cancelled, and α is vulnerable at stage n_0. On input x, $|x| > 2^{e(n_0-1)}$, first find n such that $2^{e(n-1)} \leqslant |x| < 2^{e(n)}$. Next, we query A to find all strings in $A(n) - Q$ and form a finite table T such that $A(n) = Q \cup T$. Then, we use oracle Q to decide whether $x \in K(A(n)) = K(Q \oplus T)$ (in polynomial time because Q is *PSPACE*-complete). Now consider two cases:

Case 1. $x \in K(A(n))$. Then, using Q as the oracle we can find an accepting computation of $N_K(x)$ with oracle $A(n)$, where N_K is the nondeterministic oracle machine for $K(A(n))$. Now, using oracle A we check for each query made in this computation whether the answer from A agrees with the answer from $Q \oplus T$. If all answers agree, then this computation is one for $N_k^A(x)$ and we conclude that $x \in K(A)$. If they do not agree for some query y, then y must be the single element in $A(n+1) - A(n)$. Then, we let $T' = T \cup \{y\}$ and answer $x \in K(A)$ iff $x \in K(A(n+1)) = K(Q \oplus T')$ (note that the latter question can be answered in polynomial time using oracle Q).

Case 2. $x \notin K(Q \oplus T)$. Since in stage n, requirement $R_{1,\langle j,k\rangle}$ has been considered but not cancelled, it implies that

$$x \in K(A(n)) \Leftrightarrow N_j \text{ rejects } x \text{ with oracle } L(N_k, A(n)) \oplus A(n).$$

Therefore, we know that N_j accepts x using oracle $L(N_k, A(n)) \oplus A(n))$. We use oracle Q to find an accepting computation of $N_j(x)$ using this oracle. Then we check for all queries made to $A(n)$ whether the answers given by $A(n)$ agree with the answers of A. If not, then we found the single string $y \in A(n+1) - A(n)$ and we can decide $x \in K(A)$ iff $x \in K(A(n+1))$. If $A(n)$ and A agree on all queries, then we consider queries made to

$L(N_k, A(n))$. This time, we use oracle B_k, which is a prefix set of $L(N_k, A)$, to determine whether answers to all queries y given by $L(N_k, A(n))$ agree with the answers given by $L(N_k, A)$. If all answers from $L(N_k, A(n))$ agree with answers from $L(N_k, A)$ then we know that N_j accepts x using oracle $L(N_k, A) \oplus A$, and hence $x \notin K(A)$. If at least one query z receives different answers then we have found either $z \in L(N_k, A(n)) - L(N_k, A)$ or $z \in L(N_k, A) - L(N_k, A(n))$.

In the first case that $z \in L(N_k, A(n)) - L(N_k, A)$, we again can use oracle Q to find an accepting computation of $N_k^{A(n)}(z)$. Then, in this computation, there must be at least one query made by N_k is in $A(n+1) - A(n)$ (otherwise, z would be in $L(N_k, A)$). We use oracle A to find this string and hence the set $A(n+1)$. Then we answer $x \in K(A)$ iff $x \in K(A(n+1))$.

In the second case that $z \in L(N_k, A) - L(N_k, A(n))$, we use B_k to find an accepting computation of $N_k^A(z)$. (Strictly speaking, if $L(N_k, A) = \{z \mid \exists w\, \sigma(A; z, w)\}$ and $B_k = \{\langle z, u\rangle \mid \exists v\, \sigma(A; z, uv)\}$ then we can find, using B_k as the oracle, a string w such that $\sigma(A; z, w)$. We call w as the "accepting computation.") Again, one of the queries made in this computation must be in $A(n+1) - A(n)$. We use oracle A to find this string and we are done. ∎

Remark. In the above algorithm, the oracle B_k cannot be replaced by $L(N_k, A)$, because in the last case, we need to find an accepting computation of $N_k^A(z)$, and it is not known that this can be done by using oracle $L(N_k, A)$, if $L(N_k, A)$ does not have certain self-reducibility properties.

6. Open Questions

The main open question involving the relativized low and high hierarchies is, as pointed out in Section 1, whether there are oracles relative to which the hierachies have partial collapsing structures. We list some interesting ones in the following:

(1) Does there exist an oracle A such that $L_k^P(A) \neq L_{k+1}^P(A)$ for all $k \geqslant 1$, and $L_0^P(A) = L_1^P(A)$?

(2) Does there exist an oracle B such that $L_k^P(B) = L_3^P(B)$ for all $k \geqslant 3$ and $L_0^P(B) = L_2^P(B) \neq L_3^P(B)$?

(3) Does there exist an oracle C such that $H_0^P(C) = H_1^P(C)$, but $\bigcup_{i=0}^{\infty} H_i^P(C) \neq H_0^P(C)$?

In our proof of Theorem B, we constructed oracle A relative to which the polynomial-time hierarchy collapses to the kth level (and so $L_k^P(A) = H_k^P(A) = NP(A)$) and the low and high hierarchies relative to A

have exactly k levels. It is not known that the collapsing of the polynomial-time hierarchy is necessary to collapse the low and high hierarchies. In other words, the following question remains open:

(4) Does there exist an oracle D such that $\Sigma_k^P(D) \neq \Pi_k^P(D)$ for all $k \geqslant 1$, and $L_j^P(D) = L_{j+1}^P(D)$ (or, $H_j^P(D) = H_{j+1}^P(D)$) for some $j \geqslant 0$?

RECEIVED February 14, 1989; FINAL MANUSCRIPT RECEIVED September 1, 1989

REFERENCES

BAKER, T., GILL, J., AND SOLOVAY, R. (1975), Relativizations of the $P = ?\ NP$ question, *SIAM J. Comput.* **4**, 431–442.

BALCÁZAR, J., BOOK, R. V., AND SCHÖNING, U. (1986), The polynomial-time hierarchy and sparse oracles, *J. Assoc. Comput. Mach.* **33**, 603–617.

BERMAN, L., AND HARTMANIS, J. (1977), On isomorphisms and density of *NP* and other complete sets, *SIAM J. Comput.* **1**, 305–322.

FURST, M., SAXE, J., AND SIPSER, M. (1984), Parity, circuits, and the polynomial time hierarchy, *Math. Systems Theory* **17**, 13–27.

GOLDWASSER, S., AND SIPSER, M. (1986), Private coins versus public coins in interactive proof systems, *in* "Proceedings, 18th ACM Symposium on Theory of Computing," pp. 59–68.

HASTAD, J. T. (1987), "Computational Limitations for Small-Depth Circuits," Ph.D. dissertation, MIT, MIT Press, Cambridge, MA.

KARP, R., AND LIPTON, R. (1980), Some connections between nonuniform and uniform complexity classes, *in* "Proceedings, 12th ACM Symposium on Theory of Computing," pp. 302–309.

KO, K. (1989), Relativized polynomial time hierarchies having exactly k levels, *SIAM J. Comput.* **18**, 392–408.

KO, K., AND SCHÖNING, U. (1985), On circuit-size complexity and the low hierarchy in NP, *SIAM J. Comput.* **14**, 41–51.

LONG, T. (1982), Strong nondeterministic polynomial-time reducibilities, *Theoret. Comput. Sci.* **21**, 1–25.

LONG, T., AND SELMAN, A. (1986), Relativizing complexity classes with sparse oracles, *J. Assoc. Comput. Mach.* **33**, 618–627.

SCHÖNING, U. (1983), A low and a high hierarchy within NP, *J. Comput. System Sci.* **27**, 14–28.

Schöning, U. (1987), Graph isomorphism is in the low hierarchy, *in* "Proceedings, 1987 Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 247, pp. 114–124, Springer-Verlag, New York/Berlin.

SIPSER, M. (1983), Borel sets and circuit complexity, *in* "Proceedings, 15th ACM Symposium on Theory of Computing," pp. 61–69.

YAO, A. (1985), Separating the polynomial-time hierarchy by oracles, *in* "Proceedings, 26th IEEE Symposium on Foundations of Computer Science," pp. 1–10.