The Extended Low Hierarchy Is an Infinite Hierarchy [1]

Ming-Jye Sheu Timothy J. Long

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

October 20, 1992

[1]This work was supported in part by NSF Grant CCR-8909071

Abstract

Balcázar, Book, and Schöning introduced the extended low hierarchy based on the Σ-levels of the polynomial-time hierarchy as follows: for $k \geq 1$, level k of the extended low hierarchy is the set $EL_k^{P,\Sigma} = \{A \mid \Sigma_k^P(A) \subseteq \Sigma_{k-1}^P(A \oplus \mathrm{SAT})\}$. Allender and Hemachandra and Long and Sheu introduced refinements of the extended low hierarchy based on the Δ and Θ-levels, respectively, of the polynomial-time hierarchy: for $k \geq 2$, $EL_k^{P,\Delta} = \{A \mid \Delta_k^P(A) \subseteq \Delta_{k-1}^P(A \oplus \mathrm{SAT})\}$ and $EL_k^{P,\Theta} = \{A \mid \Theta_k^P(A) \subseteq \Theta_{k-1}^P(A \oplus \mathrm{SAT})\}$. In this paper we show that the extended low hierarchy is properly infinite by showing, for $k \geq 2$, that $EL_k^{P,\Sigma} \subsetneq EL_{k+1}^{P,\Theta} \subsetneq EL_{k+1}^{P,\Delta} \subsetneq EL_{k+1}^{P,\Sigma}$. Our proofs use the circuit lower bound techniques of Håstad and Ko. As corollaries to our constructions, we obtain, for $k \geq 2$, oracle sets B_k, C_k, and D_k, such that $\mathrm{PH}(B_k) = \Sigma_k^P(B_k) \supsetneq \Delta_k^P(B_k)$, $\mathrm{PH}(C_k) = \Delta_k^P(C_k) \supsetneq \Theta_k^P(C_k)$, and $\mathrm{PH}(D_k) = \Theta_k^P(D_k) \supsetneq \Sigma_{k-1}^P(D_k)$.

Key words. the extended low hierarchy, circuit lower bounds, the polynomial-time hierarchy, relativizations

AMS(MOS) subject classifications. 68Q15, 68Q05

1 Introduction

The low and high hierarchies in NP were introduced by Schöning in order to understand the internal structure of NP [Sch83]. The low hierarchy starts at P and grows "upward" towards the NP-complete sets, while the high hierarchy starts with the complete sets in NP and grows "downward" towards P. More formally, set A is in level k of the *low hierarchy* if $\Sigma_k^P(A) = \Sigma_k^P$, and A is in level k of the *high hierarchy* if $\Sigma_{k+1}^P = \Sigma_k^P(A)$. Many interesting sets have been located in these hierarchies: $\leq_T^P$-complete sets and $\leq_T^{SN}$-complete sets for NP form the first two levels of the high hierarchy, while sparse NP sets, NP sets with small circuits, and graph isomorphism have been placed in various levels of the low hierarchy.

These hierarchies were first extended to sets outside of NP by Balcázar, Book and Schöning [BBS86]. In particular, they defined the *extended low hierarchy* as follows: set A is in level k of the extended low hierarchy if $\Sigma_k^P(A) \subseteq \Sigma_{k-1}^P(A \oplus \mathrm{SAT})$. Once again, many interesting classes of sets have been located in the extended low hierarchy such as sparse sets, sets with small circuits, and general left cuts of real numbers. Recently, refinements of the extended low hierarchy based on the Δ-classes and Θ-classes of the polynomial-time hierarchy were introduced by Allender and Hemachandra [AH92] and by Long and Sheu [LS91], respectively. Many of the sets previously located in the extended low hierarchy were more carefully and precisely located in new levels of the refined extended low hierarchy.

An interesting open question is whether the low hierarchy or the extended low hierarchy consists of an infinite number of distinct levels. This question is of interest for several reasons. If these hierarchies are properly infinite, then they provide a natural formal framework for categorizing and comparing the relative "power and complexity" of sets: sets at higher levels of the low hierarchy are more "complicated" than sets at lower levels. But more importantly, for the low hierarchy, if it contains at least two distinct levels, then P $\neq$ NP. We also note that it is not even known if the low hierarchy is properly infinite under very natural assumptions, such as the polynomial-time hierarchy consisting of an

infinite number of distinct levels.

For the extended low hierarchy, the situation is somewhat different. In proving optimal lower bounds for the extended low hierarchy, Allender and Hemachandra [AH92] constructed a sparse set that is not in $EL_2^{P,\Sigma}$, level two of the extended low hierarchy based on the Σ-levels of the polynomial-time hierarchy. Since it is known that all sparse sets are in $EL_3^{P,\Theta}$, the third level of the extended low hierarchy based on the Θ-levels of the polynomial-time hierarchy, this shows that the extended low hierarchy has at least two distinct levels up through $EL_3^{P,\Theta}$. However, it was not known if the extended low hierarchy has distinct levels beyond $EL_3^{P,\Theta}$.

In this paper we show that the extended low hierarchy is in fact a properly infinite hierarchy. Specifically, we prove that

$$\forall k \geq 2, EL_k^{P,\Sigma} \subsetneq EL_{k+1}^{P,\Theta} \subsetneq EL_{k+1}^{P,\Delta} \subsetneq EL_{k+1}^{P,\Sigma},$$

where $EL_k^{P,\Sigma}$ ($EL_k^{P,\Delta}$, $EL_k^{P,\Theta}$) denotes level k of the extended low hierarchy based on the Σ-levels (Δ-levels, Θ-levels, respectively) of the polynomial-time hierarchy. This is the first example of a low hierarchy in complexity theory that has been shown to be properly infinite.

Our proofs follow the circuit lower bound techniques from Yao [Yao85], Håstad [Has87], and Ko [Ko89]. Of particular interest for us is Ko's result that there are sets relative to which the polynomial-time hierarchy collapses to any finite level:

$$\forall k \geq 1, \exists A\ [\mathrm{PH}(A) = \Sigma_k^P(A) \neq \Sigma_{k-1}^P(A)].$$

Note that A is in level $k+1$ of the extended low hierarchy based on Σ-levels of the polynomial-time hierarchy and is not likely to be in level k of the extended low hirarchy. The condition for A not being in level k of the extended low hierarchy is as follows:

$$\mathrm{PH}(A) = \Sigma_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus \mathrm{SAT}).$$

Thus, a direct modification of Ko's proofs yields that every Σ-level of the extended low hierarchy is distinct. In this paper we are able to extend Ko's collapsing results to Δ- and Θ-levels of relativized polynomial-time hieraries.

In Section 3, we extend Yao and Håstad's separation results by showing the existence of oracle sets A, B, and C such that for all $k \geq 2$, $\Delta_k^P(A) \neq \Sigma_k^P(A)$, $\Theta_k^P(B) \neq \Delta_k^P(B)$, and $\Sigma_{k-1}^P(C) \neq \Theta_k^P(C)$. In order to obtain our results, we introduce, for the first time, circuit models for the Δ and Θ-classes of relativized polynomial-time hierarchies. In Section 4, we combine the separation results of Section 3 with encoding techniques to construct oracles D_k, E_k, and F_k such that

$$\forall k \geq 1\ \exists D_k\ [\mathrm{PH}(D_k) = \Sigma_k^P(D_k) \not\subseteq \Delta_k^P(D_k \oplus \mathrm{SAT})].$$

$$\forall k \geq 2\ \exists E_k\ [\mathrm{PH}(E_k) = \Delta_k^P(E_k) \not\subseteq \Theta_k^P(E_k \oplus \mathrm{SAT})].$$

$$\forall k \geq 2\ \exists F_k\ [\mathrm{PH}(F_k) = \Theta_k^P(F_k) \not\subseteq \Sigma_{k-1}^P(F_k \oplus \mathrm{SAT})].$$

These results show that every level of the extended low hierarchy is distinct. Immediate corollaries are the existence of oracles G_k, H_k, and I_k, for all $k \geq 2$, such that $\mathrm{PH}(G_k) = \Sigma_k^P(G_k) \neq \Delta_k^P(G_k)$, $\mathrm{PH}(H_k) = \Delta_k^P(H_k) \neq \Theta_k^P(H_k)$, and that $\mathrm{PH}(I_k) = \Theta_k^P(I_k) \neq \Sigma_{k-1}^P(I_k)$. These corollaries answer open questions posed by Ko [Ko89].

2 Preliminaries

2.1 Basic definitions and notation

Σ denotes an arbitrary alphabet of size at least two. For $x \in \Sigma^*$, $|x|$ denotes the length of x, with λ denoting the string of length 0. For a set A over Σ, $\overline{A}$ denotes its complement $\Sigma^* - A$, and for a class of sets $\mathcal{C}$, co-$\mathcal{C}$ denotes the set of complements of the elements of $\mathcal{C}$. For arbitrary sets A and B, $A \oplus B = \{0x \mid x \in A\} \cup \{1x \mid x \in B\}$. Let $\| A \|$ denote the cardinality of a set A, and let $\| A^{\leq n} \|$ denote the cardinality of $A^{\leq n}$; that is, the cardinality of A restricted to strings of length at most n.

$L(M)$ denotes the set accepted by Turing machine M, and $L(M, A)$ denotes the set accepted by an oracle Turing machine M using oracle set A. The classes P and NP have their standard definitions. $\mathrm{P}(A)$ and $\mathrm{NP}(A)$ are the A-relativized versions of P and NP. For each oracle set A, we define the set $K(A)$ as follows: for a nondeterministic oracle machine M_i, a string x, and an integer j, let $\langle i, x, 1^j \rangle \in K(A)$ if and only if the nondeterministic machine M_i, using A as the oracle, accepts x in at most j steps. It is well known that, for every set A, $K(A)$ is $\leq_m^P$-complete for $\mathrm{NP}(A)$.

2.2 Polynomial-time hierarchy

The classes of the polynomial-time hierarchy are $\{\Sigma_k^P, \Pi_k^P, \Delta_k^P, \Theta_k^P \mid k \geq 0\}$, where

$$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \Theta_0^P = \mathrm{P}, \text{ and for } k \geq 0,$$
$$\Sigma_{k+1}^P = \mathrm{NP}(\Sigma_k^P),$$
$$\Pi_{k+1}^P = \text{co-}\Sigma_{k+1}^P,$$
$$\Delta_{k+1}^P = \mathrm{P}(\Sigma_k^P),$$
$$\Theta_{k+1}^P = \Theta(\Sigma_k^P).^1$$

Sets in the polynomial-time hierarchy relative to A have also been characterized in terms of polynomial bounded quantification over predicates in $\mathrm{P}(A)$ [Sto76, Wra76]. We

[1]The Θ operator is defined as follows: for any class $\mathcal{C}$, a set $L \in \Theta(\mathcal{C})$ if and only if there are a deterministic polynomial time oracle machine M and a set $A \in \mathcal{C}$ such that $L = L(M, A)$ and, on any input x, M makes at most $O(\log |x|)$ queries to A.

adopt the following definitions from [Ko89]. A predicate σ of a set variable A and a string variable x is called a polynomial-time predicate, or a P^1-predicate, if there exist an oracle machine M and a polynomial p such that p bounds the running time of M, and for all sets A and all strings x, M^A accepts x if and only if $\sigma(A;x)$ is true. Thus, a set B is in $\mathrm{P}(A)$ if and only if there exists a P^1-predicate σ such that for all x, $x \in B \Leftrightarrow \sigma(A;x)$ holds.

Define the $\Sigma_0^{P,1}$-predicates to be the P^1-predicates. For each $k \geq 1$, define a $\Sigma_k^{P,1}$-predicate to be any predicate of the form:

$$(\exists y_1, |y_1| \leq p(|x|))(\forall y_2, |y_2| \leq p(|x|)) \cdots (Q_k y_k, |y_k| \leq p(|x|))\ \sigma(A; \langle x, y_1, \cdots, y_k \rangle),$$

where σ is a P^1-predicate, p is a polynomial, and Q_k is $\exists$ if k is odd and Q_k is $\forall$ if k is even. Without loss of generality, we assume that p also bounds the running time of the oracle machine M that decides σ with oracle A. The classes of the relativized polynomial-time hierarchy ($\mathrm{PH}(A)$) can be defined as follows:

$$\begin{aligned}
&\Sigma_k^P(A) = \{L \mid (\exists \Sigma_k^{P,1}\text{-predicate } \tau)[x \in L \Leftrightarrow \tau(A;x)]\},\\
&\Pi_k^P(A) = \text{co-}\Sigma_k^P(A),\\
&\Delta_{k+1}^P(A) = \mathrm{P}(\Sigma_k^P(A)),\\
&\Theta_{k+1}^P(A) = \Theta(\Sigma_k^P(A)), \text{ for } k \geq 0.
\end{aligned}$$

It is known that for each level of the relativized polynomial-time hierarchy, there is a $\Sigma_k^P(A)$-complete set. These complete sets can be defined inductively as follows: Let $K^1(A) = K(A)$ and $K^k(A) = K(K^{k-1}(A))$ for $k > 1$. Then for each $k \geq 1$ and each set A, $K^k(A)$ is complete for $\Sigma_k^P(A)$.

2.3 Circuits

A circuit is a rooted tree, where each non-leaf node is associated with a gate. In this paper, we use four types of gates: AND, OR, MaxOdd, and XOR. For a MaxOdd gate, let its input lines from left to right be $b_0, b_1, \ldots, b_n$. A MaxOdd gate outputs 1 if and only if i is odd, where b_i is the rightmost input line with input 1. A MaxOdd gate outputs 0

if all the inputs are 0. An XOR gate outputs 1 if and only if the number of its 1-inputs is odd. The fanins of all the gates are unlimited. Each leaf of a circuit is associated with a constant 0, a constant 1, a variable x, or a negated variable $\overline{x}$. The *size* of a circuit C is defined to be the number of gates in C. Given a circuit C with only AND and OR gates, its dual circuit $\hat{C}$ is defined inductively as follows: the dual of a constant or a variable is its negation, and the dual of a circuit C which is an OR (AND) of n children $C_i, 1 \leq i \leq n$, is the AND (OR) of $\hat{C}_i$, $1 \leq i \leq n$.

A circuit computes a function on its variables in the obvious way. Each variable is represented by v_z for some string $z \in \Sigma^*$. Let V be the set of variables in a circuit C. A *restriction* ρ of a circuit C is a mapping from V to $\{0, 1, *\}$. Let $C\lceil_\rho$ denote the circuit C' obtained from C by replacing each variable x with $\rho(x) = 0$ by 0 and each y with $\rho(y) = 1$ by 1. If a restriction ρ maps no variables to $*$, we say ρ is a complete assignment for the variables of C. Also, we say a restriction ρ *completely determines* C if $C\lceil\rho$ computes a constant function 0 or 1.[2] For any set $A \subseteq \Sigma^*$, the natural assignment ρ_A on all variables v_z, $z \in \Sigma^*$ is: $\rho_A(v_z) = 1$ if $z \in A$ and $\rho_A(v_z) = 0$ if $z \notin A$. For any pair of sets B_1, B_0, with $B_1 \cap B_0 = \emptyset$, the assignment ρ_{B_1,B_0} on all variables v_z, $z \in \Sigma^*$, is $\rho_{B_1,B_0}(v_z) = 1$, if $z \in B_1$, $\rho_{B_1,B_0}(v_z) = 0$, if $z \in B_0$, and $\rho_{B_1,B_0}(v_z) = *$, if $z \notin B_1 \cup B_0$.

Note that applying a restriction ρ to a circuit C and its dual circuit $\hat{C}$ results in another pair of dual circuits. This can be expressed by the following proposition.

Proposition 2.1 *Let C be a circuit with only AND and OR gates and let $\hat{C}$ be the dual circuit of C. Let ρ be a restriction of C. Then $\hat{C}\lceil_\rho$ is the dual circuit of $C\lceil_\rho$.*

Random restrictions of circuits were first introduced by Furst, Saxe, and Sipser [FSS81]. We use two probability spaces of restrictions $R^+_{q,\mathcal{B}}$ and $R^-_{q,\mathcal{B}}$ from [Has87], as well as the restriction $g(\rho)$, where $\rho \in R^+_{q,\mathcal{B}}$ or $R^-_{q,\mathcal{B}}$. Let $\mathcal{B} = \{B_j\}_{j=1}^r$ be a partition of variables in a circuit C and let q be a value between 0 and 1. To define a random restriction ρ in $R^+_{q,\mathcal{B}}$, for each B_j, $1 \leq j \leq r$, let $s_j = *$ with probability q and let $s_j = 0$ with probability

[2] Note that a restriction that is not a complete assignment may still completely determine C.

$1-q$. Then, independently, for each variable $x \in B_j$, let $\rho(x) = s_j$ with probability q and $\rho(x) = 1$ with probability $1-q$. Next, for each $\rho \in R^+_{q,\mathcal{B}}$, define a restriction $g(\rho)$ such that for all B_j with $s_j = *$, let V_j be the set of all variables in B_j which are given value $*$ by ρ. $g(\rho)$ selects one variable y in V_j and gives value $*$ to y and value 1 to all other variables in V_j. The probability space $R^-_{q,\mathcal{B}}$ and $g(\rho)$ are defined analogously by interchanging the roles of 0 and 1.

Circuits can be used to describe the computation of oracle machines on input strings [FSS81]. The following proposition is taken from Ko [Ko89].

Proposition 2.2 *[Ko89] Let M be a deterministic oracle machine with runtime $\leq p(n)$, where p is a polynomial. Then, for each x, there is a depth-2 circuit $C = C_{M,x}$ satisfying the following properties:*

1. *C is an* OR *of* ANDs*,*
2. *the top fanin of C is $\leq 2^{p(|x|)}$ and the bottom fanin of C is $\leq p(|x|)$, and*
3. *for any set A, $C\lceil_{\rho_A} = 1$ if and only if $M^A(x)$ accepts.*

Intuitively, each AND gate of C represents one of the accepting computation paths of M on input x, where the variables of the AND gate represent the query strings and answers of that path. It is not difficult to see that the above proposition also holds if we require that the circuit C be an AND of ORs.[3]

We call a depth-$(k+1)$ circuit a Σ_k-circuit if it has alternating OR and AND gates, starting with a top OR gate. A $\Sigma_k(m)$-circuit (Figure 1) is a Σ_k-circuit that

(1) the number of gates at level 1 to level $k-1$ is bounded by 2^m, and

(2) fanins at level $k+1$ are $\leq m$.

Note that there is no restriction in fanin at level k. A depth-$(k+1)$ circuit is a Π_k-circuit ($\Pi_k(m)$-circuit) if its dual circuit is a Σ_k-circuit ($\Sigma_k(m)$-circuit). Now we can translate $\Sigma^{P,1}_k$-predicates to circuits.

[3]Since M is a deterministic oracle machine, there is a determinisitc oracle machine M' such that M' behaves exactly like M except reversing accept/reject at the end of the computation. The desired AND of ORs circuit is the dual circuit of the circuit obtained from M' by Proposition 2.2.

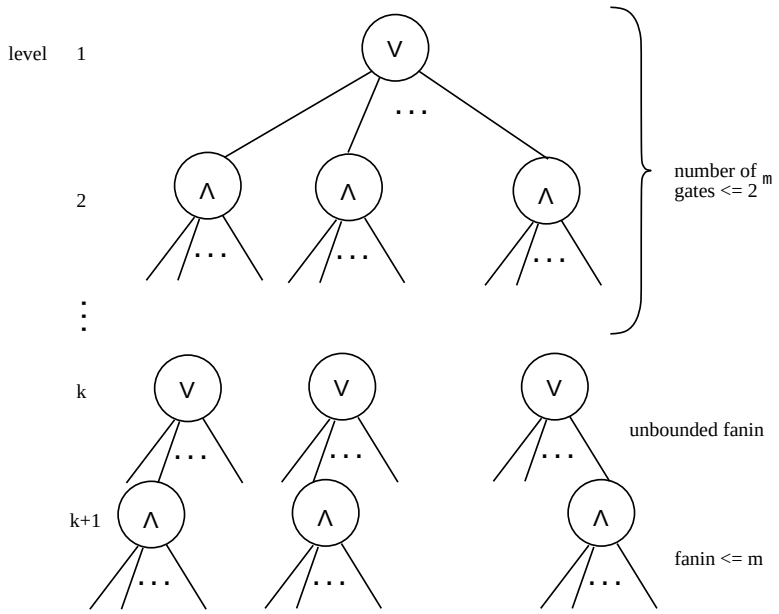

Figure 1: A $\Sigma_k(m)$-circuit

Proposition 2.3 *[Ko89] Let $k \geq 1$. For every $\Sigma_k^{P,1}$-predicate τ there is a polynomial q such that for every x, there exists a $\Sigma_k(q(|x|))$-circuit $C_{\tau,x}$, having the property that for any set A, $C_{\tau,x}\lceil_{\rho_A} = 1$ if and only if $\tau(A;x)$ is true. Also, for each variable v_z ($\overline{v_z}$) in $C_{\tau,x}$, $|z| \leq q(|x|)$.*

Sipser [Sip83] introduced a family of functions f_k^n, $n \geq 1$, $k \geq 1$, computed by special types of circuits. Håstad later introduced a modified family of f_k^n functions [Has87], and Ko, still later, introduced a further modification of the f_k^n functions in constructing oracles that separate and collapse relativized polynomial-time hierarchies [Ko89]. We will use Ko's defnition of f_k^n functions as well as introducing similar functions that will be used in our separation results in the next section. These circuits and their functions are pictured in Figures 2, 3, and 4.

Definition 2.4 *[Ko89] For each $n \geq 1$ and $k \geq 1$, a C_k^n circuit is a depth-k circuit such that*

1. *C_k^n has alternating OR and AND gates, with a top OR gate,*
2. *all the fanins of C_k^n are exactly n, except the bottom fanins which are exactly $\sqrt{n}$, and*

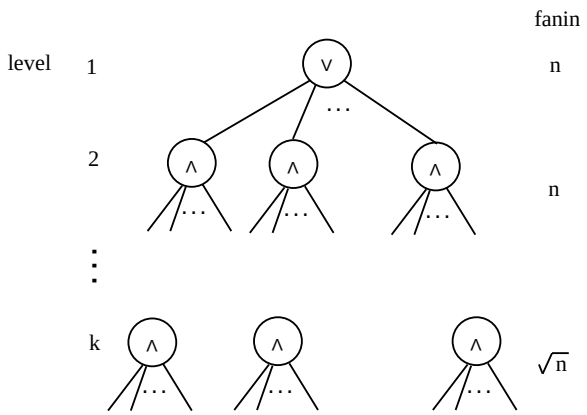

Figure 2: A C_k^n circuit

3. *each leaf of* C_k^n *is a unique positive variable.*

We let f_k^n *denote the function computed by* C_k^n.

Definition 2.5

1. *For each* $n \geq 1$ *and* $k \geq 1$, *an* H_k^n *circuit is a depth-k circuit with a top* MaxOdd *gate having fanin exactly n such that each subcircuit of the top* MaxOdd *gate is a* C_{k-1}^n *circuit computing an* f_{k-1}^n *function and that all the variables in* H_k^n *are unique. We let* h_k^n *denote the function computed by* H_k^n.

2. *For each* $n \geq 1$ *and* $k \geq 1$, *a* D_k^n *circuit is a depth-k circuit with a top* XOR *gate having fanin exactly* 2 *such that each subcircuit of the top* XOR *gate is a* C_{k-1}^n *circuit computing an* f_{k-1}^n *function and that all the variables in* D_k^n *are unique. We let* l_k^n *denote the function computed by* D_k^n.

Before ending this section, we state some important lemmas, including the switching lemma on constant-depth circuits from Håstad [Has87].

Lemma 2.6 (Switching Lemma) *[Has87] Let G be an* AND *of* ORs *with bottom fanin* $\leq t$ *and let* $\mathcal{B} = \{B_j\}$ *be a partition of the variables in G. Then, for a random restriction* ρ *from* $R_{q,\mathcal{B}}^+$, $Pr(G\lceil_{\rho g(\rho)}$ *is not equivalent to a circuit of* OR *of* ANDs *with bottom fanin* $\leq s)$ *is bounded by* α^s, *where* $\alpha < 6qt$. *The above probability also holds with* $R_{q,\mathcal{B}}^+$ *replaced*

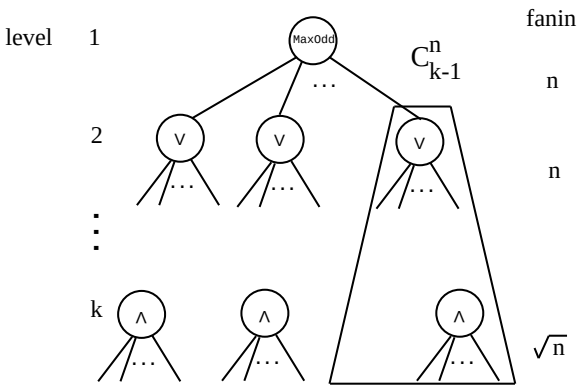

Figure 3: An H_k^n circuit

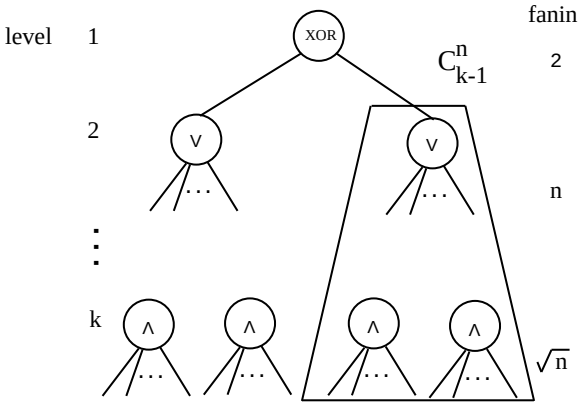

Figure 4: A D_k^n circuit

by $R^-_{q,\mathcal{B}}$, or with G being an OR *of* AND*s circuit to be converted to an* AND *of* OR*s circuits.*

The next lemma is a generalized form of the switching lemma.

Lemma 2.7 *[Has87, Ko89] Let $\{C_i\}_{i=1}^n$ be a collection depth-2* AND-of-OR *and* OR-of-AND *circuits, where the bottom fanin of each C_i is $\leq t$. Let V be the set of the variables in $\{C_i\}_{i=1}^n$ and let $\mathcal{B} = \{B_j\}$ be a partition of the variables in V. q is a number between 0 and 1 and ρ is a random restriction from $R^+_{q,\mathcal{B}}$ or $R^-_{q,\mathcal{B}}$. The probability that every $C_i\lceil_{\rho g(\rho)}$, which is an* AND-of-OR *or* OR-of-AND*, can be converted to a* OR-of-AND *or* AND-of-OR*, respectively, with bottom fanin $< s$ is greater than $2/3$, if $n \leq 1/(3\alpha^s)$, where $\alpha < 6qt$.*

The importance of the switching lemma is that a depth-$(k+1)$, $k \geq 2$, alternating OR-AND-gate circuit, with small bottom fanins, can be converted to an equivalent depth-k alternating OR-AND-gate circuit, with small bottom fanins. This "shrinking" effect is done by converting the bottom 2-level OR-of-AND (AND-of-OR) subcircuits to AND-of-OR (OR-of-AND) subcircuits and then collapsing the AND (OR) gates at level k and $k-1$ of the circuit. Our next lemma is a formal statement of this shrinking effect.

Lemma 2.8 *For each $k \geq 2$, let $\{C_i\}_{i=1}^n$ be a collection of $\Sigma_k(\gamma)$-circuits and $\Pi_k(\gamma)$-circuits. Let V be the set of the variables in $\{C_i\}_{i=1}^n$ and let $\mathcal{B} = \{B_j\}$ be a partition of the variables in V. Let q be a number between 0 and 1 and let ρ be a random restriction from $R^+_{q,\mathcal{B}}$ or $R^-_{q,\mathcal{B}}$. The probability that every $\Sigma_k(\gamma)$-circuit and every $\Pi_k(\gamma)$-circuit, after applying the random restriction ρ, can be converted to an equivalent $\Sigma_{k-1}(\gamma)$-circuit and an equivalent $\Pi_{k-1}(\gamma)$-circuit is greater than $2/3$, if $n \leq 1/(3(2\alpha)^\gamma)$ where $\alpha < 6q\gamma$.*

Lemmas 2.9 and 2.10 show the effects of a random restriction $\rho \in R^+_{q,\mathcal{B}}$ or $R^-_{q,\mathcal{B}}$ on the circuits that compute f^n_k, h^n_k, and l^n_k functions. The effect of random restrictions on the circuits computing f^n_k functions is due to Håstad [Has87]. Lemma 2.9 shows that a random restriction $\rho \in R^+_{q,\mathcal{B}}$ (or $R^-_{q,\mathcal{B}}$, depending on whether k is odd or even) transforms

a circuit computing f_k^n, (h_{k+1}^n, or l_{k+1}^n), to a circuit computing a function very close to f_{k-1}^n (h_k^n, or l_k^n). Lemma 2.10 is a stronger form of Lemma 2.9. All proofs follow easily from the proofs in Ko [Ko89].

Lemma 2.9 *For each $k \geq 2$, there exists an integer n_k such that the following hold for all $n \geq n_k$: Let C be a C_k^n circuit computing an f_k^n function (an H_{k+1}^n circuit computing an h_{k+1}^n function , or a D_{k+1}^n circuit computing an l_{k+1}^n function), let $q = n^{-1/3}$, and let ρ be a random restriction from $R_{q,\mathcal{B}}^+$ if k is even or from $R_{q,\mathcal{B}}^-$ if k is odd. The probability that $C\lceil_{\rho g(\rho)}$ contains a subcircuit computing an f_{k-1}^n function (an h_k^n function, or an l_k^n function, respectively) is greater than $2/3$.*

Lemma 2.10 *For each $k \geq 2$, let $\{C_i\}_{i=1}^t$ be t C_k^n circuits, each computing a function f_k^n, with pairwise disjoint variables. Let $q = n^{-1/3}$. If $t \leq 2^{\delta n^{1/6}}$ with $\delta < 1$, then for sufficiently large n, there exists a restriction ρ from $R_{q,\mathcal{B}}^+$ if k is even, or from $R_{q,\mathcal{B}}^-$ if k is odd, such that every $C_i\lceil_{\rho g(\rho)}$, $1 \leq i \leq t$, contains a subcircuit that computes an f_{k-1}^n function. This is also holds for H_{k+1}^n and D_{k+1}^n circuits and h_{k+1}^n and l_{k+1}^n functions.*

3 Separation results for relativized polynomial-time hierarchies

Yao and Hastad [Yao85, Has87] proved the existence of oracles, relative to which, the polynomial-time hierarchy has an infinite number of distinct levels. In this section, we further separate the Σ, Δ, and Θ-classes of relativized polynomial-time hierarchies. Our separating results show that there is an oracle A such that for all $k \geq 2$, $\Sigma_{k-1}^P(A) \subsetneq \Theta_k^P(A) \subsetneq \Delta_k^P(A) \subsetneq \Sigma_k^P(A)$. Although Yao and Hastad's results imply part of our separation results, we show different proofs here and will make use of them in the later sections.

3.1 Separating Δ-classes and Σ-classes in relativized polynomial time hierarchies

In this section, we show that there exists a recursive oracle A such that for all $k \geq 2$, $\Sigma_k^P(A) \neq \Delta_k^P(A)$. We will find, for each $k \geq 2$, a set $L_k^A \in \Sigma_k^P(A)$ such that $L_k^A \notin \Delta_k^P(A)$. By dovetailing the diagonalization for each $k > 0$, the oracle set A can be constructed so that $\Delta_k^P(A) \subsetneq \Sigma_k^P(A)$ for all $k > 0$. Note that when $k = 1$, separation of $\Sigma_1^P(A)$ and $\Delta_1^P(A)$ was already done in [BGS75].

In order to use the circuit lower bound techniques, we need to define circuits for $\Delta_k^P(A)$. Consider a deterministic polynomial time bounded oracle machine M using oracle set B. On any input x, we can computations of M on x. By Proposition 2.2, there is a two level OR of ANDs circuit $C_{M,x}$ such that $x \in L(M,B)$ if and only if $C_{M,x}\lceil_{\rho_B} = 1$. Each AND gate of $C_{M,x}$ represents an accepting computation path of T, relative to some oracle. The fanins (variables) of each AND gate are the query strings in the corresponding computation path of T. For the convenience of discussion, we will label each AND gate with $p(|x|)$ bits, where p is the polynomial bound of the running time of M on any input x. For each label $l = b_1 \cdots b_{p(|x|)}$, b_i represents the answer of the ith query string in the corresponding computation path: $b_i = 1$ if and only if the answer to the ith query string is YES. It is not difficult to see that every two labels are distinct and that for any two

labels l and l' with same prefix $b_1 \cdots b_i$, $1 \leq i < p(|x|)$, the corresponding computation paths π_l and $\pi_{l'}$ have the same query strings $q_1, \ldots, q_{i+1}$.

Now if we replace B by $K^{k-1}(A)$, then the language accepted by M is in $\Delta_k^P(A)$. As stated in Section 2, for each variable z of the form $<i, y, 1^j>$, there are a polynomial q and a $\Sigma_{k-1}(q(|z|))$-circuit $C_{K^{k-1},z}$ such that $z \in K^{k-1}(A)$ if and only if $C_{K^{k-1},z}\lceil_{\rho_A} = 1$. We then replace each positive variable v_z of $C_{M,x}$ by $C_{K^{k-1},z}$ if z is of the correct syntactic form, or by 0 if z is not of the correct syntactic form. For each negative variable v_z of $C_{M,x}$, we replace it by the dual circuit of $C_{K^{k-1},z}$ if z is of the correct syntactic form, or by 1 if z is not of the correct syntactic form. The entire circuit consists of $k+2$ levels. Now we formally define, for $k \geq 2$, $\Delta_k(m)$-circuits.

Definition 3.1 *For $k \geq 2$, C is a $\Delta_k(m)$-circuit if and only if*

1. *C has $k+2$ levels, and the top gate is an* OR *with fanin $\leq 2^m$,*
2. *the gates on the second level are* AND *gates with fanins $\leq m$,*
3. *each* AND *gate G_j at the second level has a distinct label of m bits, $b_1 \cdots b_m$, and if $b_i = 1$ then the i^{th} subcircuit from the left of G_j is a $\Sigma_{k-1}(m)$-circuit, and if $b_i = 0$ then the i^{th} subcircuit from the left of G_j is a $\Pi_{k-1}(m)$-circuit,*
4. *for any two* AND *gates G_j and G_n, labeled l_j and l_n, at the second level, if the first r bits, $r < m$, of labels l_j and l_n are the same, then the first r subcircuits from the left of G_j and G_n are the same, and*
5. *for any two* AND *gates G_j and G_n, labeled l_j and l_n, at the second level, if the first bit from the left of l_j and l_n, where l_j and l_n differ, is b_r, $r \leq m$, then the r^{th} subcircuits from the left of G_j and G_n are dual circuits.*

The following lemma states that for all $k \geq 2$ and for any arbitrary set A, if $L \in \Delta_k^P(A)$ then there is a Δ_k-circuit C such that $x \in L \Leftrightarrow C\lceil_{\rho_A} = 1$.

Lemma 3.2 *Let $k \geq 2$. For every set $L \in \Delta_k^P(A)$, there is a polynomial q such that for every input string x, there is a corresponding $\Delta_k(q(|x|))$-circuit $C_{L,x}$ such that $x \in L \Leftrightarrow C_{L,x}\lceil_{\rho_A} = 1$.*

Our goal is, for every $k \geq 2$, to find a set A such that $\Sigma_k^P(A)$ properly contains $\Delta_k^P(A)$. First, for fixed $k \geq 1$, define the language

$$L_k^A = \{x0^{kn} \mid |x| = 2n, (\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n) \cdots (Q_k y_k, |y_k| = n) \\ 0xy_1y_2 \ldots y_k \in A\},$$

where Q_k is $\exists$ if k is odd and Q_k is $\forall$ if k is even. It is clear that $L_A \in \Sigma_k^P(A)$.

Let $M_1, M_2, \cdots$ be an effective enumeration of deterministic polynomial-time bounded oracle machines with running time bounded by polynomials $p_1, p_2, \cdots$, respectively. The construction of set A is by stages. Let $t(n)$ denote the maximum length of strings in A or $\overline{A}$ decided by stage n. Let $D(n)$, the diagonalization area for stage n, be the set $\{x \in \Sigma^* \mid t(n-1)+1 \leq |x| \leq t(n)\}$. At stage n of the construction, we will find a witness 0^m, m is a multiple of $k+2$, such that, $0^m \in L_k^A \Leftrightarrow 0^m \notin L(M_n, K^{k-1}(A))$; that is, at each stage $n > 0$, we want to satisfy the following requirement R_n:

$$R_n : (\exists x = 0^m)(\exists B \subseteq D(n))[0^m \in L_k^B \Leftrightarrow 0^m \notin L(M_n, K^{k-1}(A'))],$$

where $A' = A(n-1) \cup B$ and $A(n-1)$ denotes the strings in A decided prior to stage n. Let C_0 be the obvious k-level circuit for the predicate $0^m \in L_k^B$. C_0 contains a subcircuit that computes a $f_k^{2^{m/(k+2)}}$ function. By Lemma 3.2, there exist a polynomial q and a $\Delta_k(q(m))$-circuit C_n that corresponds to the computation of the machine M_n on input 0^m with oracle $K^{k-1}(A)$. Since the length of query strings to A in the circuit C_n is bounded by $q(m)$, we let $t(n) = q(m)$. The requirement R_n can then be rewritten as

$$R_n' : (\exists x = 0^m)(\exists B \subseteq D(n))[C_0\lceil_{\rho_B} = 1 \Leftrightarrow C_n\lceil_{\rho_{A'}} = 0],$$

where $A' = A(n-1) \cup B$. Now let $s_k(v)$ be the minimum r such that a $\Delta_k(r)$-circuit can compute an f_k^v function. That requirement R_n' can be satisfied follows from the following lower bound on $s_k(v)$.

Theorem 3.3 *For all $k \geq 2$, for sufficiently large v, $s_k(v) \geq \delta v^{1/5}$, where $\delta = 1/48$.*

Proof. The proof is by induction on k. For the base case of $k = 2$, let C_2^v be a circuit computing f_2^v; that is, C_2^v is an OR of v AND's, each having fanin $= \sqrt{v}$. Let C be a $\Delta_2(r)$-circuit, with $r < \delta v^{1/5}$. We will show that C does not compute f_2^v; that is, we will find a restriction ρ such that $C_2^v\lceil_\rho \neq C\lceil_\rho$.

Consider C, a $\Delta_2(r)$-circuit. Recall that, for each AND gate g_l at second level of C, g_l is labeled with r bits, $b_{l_1}b_{l_2}\ldots b_{l_r}$, and if $b_{l_i} = 1$ then the i^{th} subcircuit from the left of g_l is a $\Sigma_1(r)$-circuit, and if $b_{l_i} = 0$ then the i^{th} subcircuit from the left of g_l is a $\Pi_1(r)$-circuit. Also, for any other AND gate $g_{l'}$ at the second level with label $b_{l'_1}b_{l'_2}\ldots b_{l'_r}$, where $b_{l_j} = b_{l'_j}$ for $1 \leq j < i$ and $b_{l_i} \neq b_{l'_i}$, the i^{th}-subcircuits of g_l and g'_l are dual circuits. Thus, if we find a restriction ρ that makes the i^{th} subcircuit of g_l output 1, then the i^{th} subcircuit of $g_{l'}$ will output 0 by Proposition 2.1. Therefore, the entire subcircuit g'_l at level 2 of C can be replaced by a constant 0. By making assignments to at most r^2 variables, we willl be able to completely determine the output of C while the output of C_2^v remais undetermined.

The process of finding an appropriate assignment of variables starts by fixing the leftmost subcircuit of each second-level AND gate of C. Note that there is a single $\Sigma_1(r)$ circuit C_1 such that the leftmost subcircuit of each second-level AND gate of C is one of C_1 or $\hat{C}_1$. If C_1 can be forced to output 1 by assigning at most r variables (recall that C_1 is an OR of ANDs, each AND with fanin at most r), then this is done. Then, all second level AND gates of C with C_1 as the leftmost subcircuit are modified by replacing C_1 with 1, and all second level AND gates of C with $\hat{C}_1$ as the leftmost subcircuit are removed from C. On the other hand, if C_1 cannot be forced to output 1, then $\hat{C}_1$ will always output 1 regardless of assignments to variables. In this case, all second-level AND gates of C with $\hat{C}_1$ as the leftmost subcircuit are modified by replacing $\hat{C}_1$ with 1, all second-level AND gates of C with C_1 as the leftmost subcircuit are removed from C, and no assignments are made. In either case, all of second-level AND gates that now remain in C have the same leftmost subcircuit, either C_1 or $\hat{C}_1$, which is now replaced by 1. This implies that there is a single $\Sigma_1(r)$ circuit C_2 such that the second subcircuit of each remaining second-level AND gate of C is one of C_2 or $\hat{C}_2$. Hence, we can repeat the same process with C_2, then

with some C_3, etc., until some C_r. At this point, at most r^2 variables will have been assigned and the output of C is completely determined. Since $r^2 \leq (\delta v^{1/5})^2 < \sqrt{v}$ for sufficiently large v, there is no bottom AND gate G of C_2^v such that $G\lceil_\rho = 1$. Also, since the top fanin of C_2^v is v, there is some bottom AND gate G' of C_2^v such that no variables of G' have been decided in the previous process; that is, we can find a restriction ρ such that

$$(\#) \qquad C\lceil_\rho \neq * \text{ and } C_2^v\lceil_\rho = *.$$

Thus, C cannot compute a function f_2^v when the size of C is less than $\delta v^{1/5}$. This establishes the base case for the inductive proof of Theorem 3.3.

Our induction hypothesis is that for some fixed $k' \geq 2$ and for sufficiently large v, $s_{k'}(v) \geq \delta v^{1/5}$; that is, for sufficiently large v, if C is a $\Delta_{k'}(r)$-circuit with $r < \delta v^{1/5}$, then C does not compute the function $f_{k'}^v$. For the induction step, consider $k = k' + 1 > 2$. Consider a C_k^v circuit that computes an f_k^v function. Let $q = v^{-1/3}$ and let $\mathcal{B} = \{B_j\}$, where B_j is the set of variables of the bottom AND gate H_j of C_k^v. By way of contradiction, assume that there exists a circuit C, that is a $\Delta_k(r)$-circuit with $r < \delta v^{1/5}$, computing the function f_k^v. We will argue that after applying a random restriction ρ from $R_{q,\mathcal{B}}^+$, if k is even, or from $R_{q,\mathcal{B}}^-$, if k is odd, to C, the resulting circuit $C\lceil_{\rho g(\rho)}$ is a $\Delta_{k-1}(r)$-circuit with $r < \delta v^{1/5}$ that computes the function f_{k-1}^v, which contradicts our induction hypothesis.

To apply the induction hypothesis, we need to verify that

(1) with a high probability, $C\lceil_{\rho g(\rho)}$ is equivalent to a $\Delta_{k-1}(r)$-circuit, and

(2) with a high probability, $C_k^v\lceil_{\rho g(\rho)}$ contains a subcircuit computing the function f_{k-1}^v.

Part (1) is proved by Lemma 2.8. To apply Lemma 2.8, let us choose $\gamma = \delta v^{1/5}$. We have $\alpha < 6q\gamma = 1/8 \cdot v^{-2/15}$. By definition, the total number of $\Sigma_k(r)$-circuits and $\Pi_k(r)$-circuits in a $\Delta_k(r)$-circuit is bounded by $r \cdot 2^r \leq 2^{2r}$. Thus, we only need to show that $2^{2r} < 2^{2\gamma} \leq 1/(3(2\alpha)^\gamma)$. We have

$$2^{2\gamma} \cdot 3(2\alpha)^\gamma = 3 \cdot (8\alpha)^\gamma \leq 3 \cdot (v^{-2/15})^\gamma \leq 1$$

for sufficiently large v. This proves part (1). Part (2) was proved in Lemma 2.9. This completes the proof of Theorem 3.3. $\square$

To satisfy requirement R'_n, notice that C_n is a $\Delta_k(q(m))$-circuit corresponding to the predicate $0^m \in L(M_n, K^{k-1}(A))$ and that C_0, corresponding to the predicate $0^m \in L_k^A$, contains a subcircuit that computes an $f_k^{2^{m/(k+2)}}$ function. Thus, we need to choose an integer m so large such that $q(m) < (1/48) \cdot 2^{m/5(k+2)}$ and that Theorem 3.3 holds for $s_k(2^{m/(k+2)})$. We now have the desired theorem.

Theorem 3.4 *There exists an oracle A such that for all $k \geq 1$, $\Delta_k^P(A) \subsetneq \Sigma_k^P(A)$.*

3.2 Separating Θ-classes and Δ-classes in relativized polynomial time hierarchies

In this section, we show that there exists a recursive oracle A such that for all $k \geq 2$, $\Theta_k^P(A) \neq \Delta_k^P(A)$. We will find, for each $k \geq 2$, a set $L_{odd,k}^A \in \Delta_k^P(A)$ and that $L_{odd,k}^A \notin \Theta_k^P(A)$. Again, by dovetailing the diagonalization for each $k \geq 2$, the oracle set A can be constructed so that $\Theta_k^P(A) \subsetneq \Delta_k^P(A)$ for all $k \geq 2$.

We will again use circuit lower bound techniques to prove this separation. Similarly to finding special circuits that represent sets in $\Delta_k^P(A)$, we need special circuits that represent sets in $\Theta_k^P(A)$. Consider a deterministic polynomial-time bounded oracle machine M using oracle set B and, on any input x, the query tree T that represents all the possible computations of M on x. Since there are at most $O(\log |x|)$ queries in any one computation path, the total number of possible paths is bounded by a polynomial $p(|x|)$. Thus, we can define $\Theta_k(m)$-circuits in a manner that is similar to the definition of $\Delta_k(m)$-circuits.

Definition 3.5 *For $k \geq 2$, C is a $\Theta_k(m)$-circuit if and only if*

1. *C has $k+2$ levels, and the top gate is an* OR *with fanin $\leq m$,*
2. *the gates on second level are* AND *gates with fanins $\leq \log m$,*

3. *each* AND *gate* G_j *at the second level has distinct label of* $n = \log m$ *bits,* $b_1 \cdots b_n$, *and if* $b_i = 1$ *then the* i^{th} *subcircuit from the left of* G_j *is a* $\Sigma_{k-1}(m)$*-circuit, and if* $b_i = 0$ *then the* i^{th} *subcircuit from the left of* G_j *is a* $\Pi_{k-1}(m)$*-circuit,*

4. *for any two* AND *gates* G_j *and* G_n, *labeled* l_j *and* l_n, *at the second level, if the first* r *bits,* $r < \log m$, *of labels* l_j *and* l_n *are the same, then the first* r *subcircuits from the left of* G_j *and* G_n *are the same, and*

5. *for any two* AND *gates* G_j *and* G_n, *labeled* l_j *and* l_n, *at the second level, if the first bit from the left of* l_j *and* l_n *where* l_j *and* l_n *differ is* b_r, $r \leq \log m$, *then the* r^{th} *subcircuits from the left of* G_j *and* G_n *are dual circuits.*

Lemma 3.6 *Let* $k \geq 2$. *For every set* $L \in \Theta_k^P(A)$, *there is a polynomial* q *such that for every input string* x, *there is a corresponding* $\Theta_k(q(|x|))$*-circuit* $C_{M,x}$ *such that* $x \in L \Leftrightarrow C_{M,x}\lceil_{\rho_A} = 1$.

To separate $\Theta_k^P(A)$ and $\Delta_k^P(A)$, for all $k \geq 2$, we define

$$L_{odd,k}^A = \{x0^{kn} \mid |x| = n, \text{ and } max\{y \mid |y| = |x|, xy0^{(k-1)n} \in L_{k-1}^A\} \text{ has odd parity } \}.$$

Recall that L_{k-1}^A, as defined in Section 3.1, is a set in $\Sigma_{k-1}^P(A)$; thus $L_{odd,k}^A$ is in $\Delta_k^P(A)$ using a binary search algorithm. The setup of the diagonalization is similar to the one in Section 3.1 with functions $t(n)$ and $D(n)$. For each stage $n > 0$ of the construction, we want to satisfy the following requirement R_n:

$$R_n : (\exists x = 0^m)(\exists B \subseteq D(n))[0^m \in L_{odd,k}^B \Leftrightarrow 0^m \notin L(M_n, K^{k-1}(A'))],$$

where $A' = A(n-1) \cup B$ and $A(n-1)$ denotes the strings in A decided prior to stage n, and M_n is the nth Θ_k^P oracle machine making at most $\log(p_n(m))$. Note that the predicate "$0^m \in L_{odd,k}^B$" can be represented by a k-level circuit with a top MaxOdd gate having fanin $2^{m/(k+1)}$. Each top level subcircuit, C_y, of MaxOdd contains a $C_{k-1}^{2^{m/(k+1)}}$ subcircuit computing an $f_{k-1}^{2^{m/(k+1)}}$ function, where y corresponds to the y in the definition of $L_{odd,k}^B$. Also, the subcircuits C_y will be arranged, in lexicographic order on y, from left to right

as the children of the top MaxOdd gate. Let C_0 be the corresponding k-level circuit for the predicate $0^m \in L^B_{odd,k}$. C_0 contains an $H_k^{2^{m/(k+1)}}$ subcircuit that computes an $h_k^{2^{m/(k+1)}}$ function.

By Lemma 3.6, there exist a polynomial q and a $\Theta_k(q(m))$-circuit C_n corresponding to the the machine M_n on input 0^m, with oracle $K^{k-1}(A)$. We let $t(n) = q(m)$. The requirement R_n can then be rewritten as

$$R'_n : (\exists x = 0^m)(\exists B \subseteq D(n))[C_0\lceil_{\rho_B} = 1 \Leftrightarrow C_n\lceil_{\rho_{A'}} = 0],$$

where $A' = A(n-1) \cup B$.

Now let $s_k(v)$ be the minimum r for a $\Theta_k(r)$-circuit to compute an h_k^v function. The following lower bound on $s_k(v)$ shows that the requirement R'_n can be satisfied.

Theorem 3.7 *For all $k \geq 2$, for sufficiently large v, $s_k(v) \geq \delta v^{1/5}$, where $\delta = 1/24$.*

Proof. Again, the proof is by induction on k. For the base case of $k = 2$, let H_2^v be an MaxOdd of v subcircuits, each computing a function f_1^v. Let C be a $\Theta_2(r)$-circuit, with $r < \delta v^{1/5}$. We need to show that C does not compute the same function as H_2^v; that is, we will find a restriction ρ such that $H_2^v\lceil_\rho \neq C\lceil_\rho$.

Consider C, the $\Theta_2(r)$-circuit. Accounting for the fact that some Σ_1 and Π_1-circuits are duals, there are at most $r < \delta v^{1/5}$ many different Σ_1-circuits and Π_1-circuits in C. (It is easy to see this by looking at the query tree of a Θ_2^P oracle machine.) During the construction of A at stage n, we will put strings into A step by step. At each step we place some strings into A in order to flip-flop the output of $H_2^v\lceil_{\rho_A}$. During the same step, at least one of the Σ_1-circuits (or Π_1-circuits) of C will be fixed to have output 1 (or 0) by assigning variables to A only. Since there are only $r < \delta v^{1/5}$ many different Σ_1-circuits and Π_1-circuits, we will eventually fix the output of C after at most $r+1 \leq \delta v^{1/5}$ flip-flops of H_2^v. We argue that, after this process, we are still able to flip-flop the output of H_2^v as we desire. Thus, we can find the needed restriction. Now we present this flip-flop process.

Process FF:

$B :=$ the variables of the leftmost OR gate of H_2^v; $\overline{B} := \emptyset$;

while $C\lceil_{\rho_B} = H_2^v\lceil_{\rho_B}$ **do**

if there is some Σ_1-circuit C_j of C with $C_j\lceil_{\rho_B} = 1$ **then**

for all Σ_1-circuits C_j of C with $C_j\lceil_{\rho_B} = 1$ **do**

Fix the output of one AND gate G of C_j to be 1 with restriction ρ_B

by assigning all negative variables of G to $\overline{B}$

end$(*$ **for** $*)$

end$(*$ **if** $*)$

if there is some Π_1-circuit C_j' of C with $C_j'\lceil_{\rho_B} = 0$ **then**

for all Π_1-circuits C_j' of C with $C_j'\lceil_{\rho_B} = 0$ **do**

Fix the output of one OR gate G' of C_j' to be 0 with restriction ρ_B

by assigning all positive variables of G' to $\overline{B}$

end$(*$ **for** $*)$

end$(*$ **if** $*)$

Find the leftmost subcircuit H_i of H_2^v such that $H_i\lceil_{\rho_B} = 0$.

Let V be the set of variables in H_i that are not in $\overline{B}$. Let $B := B \cup V$.

end$(*$ **while** $*)$

We now carefully analyze **Process FF**. Initially, B contains the variables in the leftmost OR gate of H_2^v and $\overline{B}$ is empty. We have $H_2^v\lceil_{\rho_B} = 0$. If $H_2^v\lceil_{\rho_B} \neq C\lceil_{\rho_B}$, then we are done. Otherwise, we will try to fix the output of some of the Σ_1-circuits or Π_1-circuits of C. To make sure that we only reserve polynomial many strings in the entire process, we only consider fixing the output of Σ_1-circuits to be 1 and the output of Π_1-circuits to be 0. This process is done in the following steps: for all the Σ_1-circuits C_j of C such that $C_j\lceil_{\rho_B} = 1$, fix the output of one of C_j's bottom AND gates G to be 1 with restriction ρ_B. If we assign all the negative variables in G to $\overline{B}$, then we will preserve the output of C_j in later steps. Note that the positive variables of G are already in B since G is an AND gate and $G\lceil_{\rho_B}$ outputs 1. In later steps, we will never reassign the variables in B or $\overline{B}$.

Also, for each Σ_1-circuit, there are at most r many variables that are assigned to $\overline{B}$ in this process. We carry out a similar process for the Π_1-circuits C'_j of C to obtain $C'_j\lceil_{\rho_B} = 0$.

The next step in **Process FF** is to "flip-flop" the output of H_2^v. This is done by adding the free variables of the leftmost subcircuit H_i of H_2^v such that $H_i\lceil_{\rho_B} = 0$. We will assure that there are always some free variables in H_i that are not in $\overline{B}$. This will flip-flop H_i's output between 0 and 1, and the output of the other OR gates in H_2^v remains unchanged (recall that all variables in H_2^v are unique). As output of H_2^v is flip-flopped, if the output of $C\lceil_{\rho_B}$ does not flip-flop, we are done. *Otherwise, $C\lceil_{\rho_B}$ also flip-flops its output, there must be some Σ_1-circuits (Π_1-circuits) that output 0 (1) in the previous iteration but then change their outputs to 1 (0).* This is because Σ_1-circuits outputting 1 and Π_1-circuits outputting 0 are already fixed. In this case, we repeat the same process. Note that at each iteration of the process, at least one Σ_1- or Π_1-circuit is fixed.

Now we argue that the variables reserved in $\overline{B}$ will not make any H_i's of H_2^v permanently output 0; that is, flip-flopping the output of H_2^v is always possible. Observe the following two facts.

(1) There are at most $r < \delta v^{1/5}$ Σ_1-circuits and Π_1-circuits in C. Thus, after at most $r+1$ iterations, the while-loop in **Process FF** terminates.

(2) There are v subcircuits of the top MaxOdd gate of H_2^v and each subcircuit is an OR of $\sqrt{v}$ inputs.

Since we assign at most r variables to $\overline{B}$ for each Σ_1-circuit and Π_1-circuit, there are, by fact (1), at most $r(r+1) \leq (\delta v^{1/5})^2 < \sqrt{v}$ variables reserved to $\overline{B}$ during **Process FF**. Thus, none of the H_i's of H_2^v will permanently output 0 based on assignments to $\overline{B}$. Finally, since there are at most $r+1 \leq v^{1/5}$ iterations of the while-loop and since H_2^v can flip-flop at least v times, we then conclude that there is a pair of finite sets B and $\overline{B}$ that

$$(*) \qquad C\lceil_{\rho_{B,\overline{B}}} \neq * \text{ and } H_2^v\lceil_{\rho_{B,\overline{B}}} = *.$$

This completes the base case of $k = 2$.

Our induction hypothesis is that for some fixed $k' \geq 2$ and for sufficiently large v, $s_{k'}(v) \geq \delta v^{1/5}$; that is, for sufficiently large v, if C is a $\Theta_{k'}(r)$-circuit with $r < \delta v^{1/5}$, then

C does not compute the function $h^v_{k'}$. For the induction step, consider $k = k' + 1 > 2$. Consider an H^v_k circuit that computes an h^v_k function. Let $q = v^{-1/3}$ and let $\mathcal{B} = \{B_j\}$, where B_j is the set of variables of the bottom AND gate G_j of H^v_k. By way of contradiction, assume that there exists a circuit C, that is a $\Theta_k(r)$-circuit with $r < \delta v^{1/5}$, computing the function h^v_k. We will argue that after applying a random restriction from $R^+_{q,\mathcal{B}}$, if k is odd or from $R^-_{q,\mathcal{B}}$, if k is even, to C, the resulting circuit $C\lceil_{\rho g(\rho)}$ is a $\Theta_{k-1}(r)$-circuit with $r < \delta v^{1/5}$ that computes the function h^v_{k-1}, which contradicts our induction hypothesis.

To apply the induction hypothesis, we need to verify that

(1) with a high probability, $C\lceil_{\rho g(\rho)}$ is equivalent to a $\Theta_{k-1}(r)$-circuit, and

(2) with a high probability, $H^v_k\lceil_{\rho g(\rho)}$ contains a subcircuit computing the function h^v_{k-1}.

Again, part (1) is proved by the Lemma 2.8. To apply Lemma 2.8, we choose $\gamma = \delta v^{1/5}$. We have $\alpha < 6q\gamma = 1/4 \cdot v^{-2/15}$. The total number of $\Sigma_{k-1}(r)$-circuits and $\Pi_{k-1}(r)$-circuits of C is bounded by $r \cdot \log r \leq 2^r < 2^\gamma$. To apply Lemma 2.8, we need to show that $2^\gamma \leq 1/(3(2\alpha)^\gamma)$. We have

$$2^\gamma \cdot 3(2\alpha)^\gamma = 3(4\alpha)^\gamma \leq 3(v^{-2/15})^\gamma \leq 1,$$

for sufficiently large v. This proves part (1). Part (2) follows from Lemma 2.9. This completes the proof of Theorem 3.7. □

To satisfy requirement R'_n, we need to choose an integer m so large that Theorem 3.7 holds for $s_k(2^{m/(k+1)})$ and such that $q(m) < 1/24 \cdot 2^{m/5(k+1)}$.

Theorem 3.8 *There exists an oracle A such that for all $k \geq 2$, $\Theta^P_k(A) \subsetneq \Delta^P_k(A)$.*

3.3 Separating Σ-classes and Θ-classes in relativized polynomial time hierarchies

In this section, we show that there exists a recursive oracle A such that for all $k \geq 2$, $\Sigma^P_{k-1}(A) \neq \Theta^P_k(A)$. We will find, for each $k \geq 2$, a set $L^A_{xor,k} \in \Theta^P_k(A)$ such that $L^A_{xor,k} \notin$

$\Sigma_{k-1}^P(A)$. Once again, by dovetailing the diagonalization for each $k \geq 2$, the oracle set A can be constructed so that $\Sigma_{k-1}^P(A) \subsetneq \Theta_k^P(A)$ for all $k \geq 2$.

To separate $\Sigma_{k-1}^P(A)$ and $\Theta_k^P(A)$, for all $k \geq 2$, we define

$$L_{xor,k}^A = \{x0^{kn} \mid |x| = n, \text{ and exactly one of } 0^n x 0^{(k-1)n} \text{ or } 1^n x 0^{(k-1)n} \text{ is in } L_{k-1}^A\},$$

where L_{k-1}^A is a set in $\Sigma_{k-1}^P(A)$, as defined in Section 3.1. Thus $L_{xor,k}^A$ is in $\Theta_k^P(A)$.

The setup of the diagonaization is similar to the one in Section 3.1. For each stage $n > 0$, we want to satisfy the following requirement R_n:

$$R_n : (\exists x = 0^m)(\exists B \subseteq D(n))[0^m \in L_{xor,k}^B \Leftrightarrow \sigma_n(A; 0^m) \text{ is false }],$$

where σ_n is the nth $\Sigma_{k-1}^{P,1}$-predicate. Note that the predicate "$0^m \in L_{xor,k}^B$?" can be represented by a k-level circuit with a top XOR gate having fanin of 2. Each subcircuit of the XOR gate contains a $C_{k-1}^{2^{m/(k+1)}}$ subcircuit computing an $f_{k-1}^{2^{m/(k+1)}}$ function. Let C_0 be the corresponding k-level circuit for the predicate $0^m \in L_{xor,k}^B$. C_0 contains a $D_k^{2^{m/(k+1)}}$ subcircuit that computes an $l_k^{2^{m/(k+1)}}$ function.

By Lemma 2.3, there exist a polynomial q and a $\Sigma_{k-1}(q(m))$-circuit C_n corresponding the predicate $\sigma_n(A; 0^m)$. Again, let $t(n) = q(m)$. The requirement R_n can then be rewritten as

$$R_n' : (\exists x = 0^m)(\exists B \subseteq D(n))[C_0\lceil_{\rho_B} = 1 \Leftrightarrow C_n\lceil_{\rho_{A'}} = 0],$$

where $A' = A(n-1) \cup B$ and $A(n-1)$ is the set of strings in A decided prior to stage n. Now let $s_k(v)$ be the minimum r such that a $\Sigma_{k-1}(r)$-circuit can compute an l_k^v function. The following lower bound on $s_k(v)$ shows that the requirement R_n' can be satisfied.

Theorem 3.9 *For all $k \geq 2$, for sufficiently large v, $s_k(v) \geq \delta v^{1/3}$, where $\delta = 1/12$.*

Proof. Again, the proof is by induction on k. For the base case of $k = 2$, let D_2^v be an XOR of 2 subcircuits D_1 and D_2, each computing an f_1^v function. Let C be a $\Sigma_1(r)$-circuit, where $r < \delta v^{1/3}$. We will find a restriction ρ such that $C\lceil_\rho \neq *$ and $D_2^v\lceil_\rho = *$.

Let v_1 be the leftmost variable of D_1. Initially, let $B = \overline{B} = \emptyset$. Let $\rho_\emptyset$ denote the restriction that assigns 0 to all the variables, let $\rho_{\{v_1\}}$ denote the restriction that assigns 1 to v_1 and 0 to all the other variables, and let $\rho_{\emptyset,\Sigma^*-\{v_1\}}$ denote the restriction that assigns $*$ to v_1 and assigns 0 to all the other variables. There are three cases:

Case 1: $C\lceil_{\rho_\emptyset} = 1$

In this case, there must be an AND gate G of C with only negative variables. Assign all the variables in G to $\overline{B}$. The number of variables being assigned is bounded by $r < \delta v^{1/3} < \sqrt{v}$. Thus, neither D_1 nor D_2 will be forced to output a constant by $\rho_{B,\overline{B}}$ and $D_2^v\lceil_{\rho_{B,\overline{B}}} = *$, where $B = \emptyset$.

Case 2: $C\lceil_{\rho_\emptyset} = 0$ and $C\lceil_{\rho_{\{v_1\}}} = 0$

In this case, changing the assignment of v_1 from 0 to 1 does not change the output of $C\lceil_{\rho_\emptyset}$. Thus, $C\lceil_{\rho_{\emptyset,\Sigma^*-\{v_1\}}} = 0$. $D_2^v\lceil_{\rho_{\emptyset,\Sigma^*-\{v_1\}}} = *$ since the output of $D_2^v\lceil_{\rho,\Sigma^*}$ can be set to 1 by assigning 1 to variable v_1 or left at 0 by assigning 0 to v_1.

Case 3: $C\lceil_{\rho_\emptyset} = 0$ and $C\lceil_{\rho_{\{v_1\}}} = 1$

In this case, there is an AND gate G of C that outputs 1 by $\rho_{\{v_1\}}$. Since v_1 is the only variable that is assinged 1 by $\rho_{\{v_1\}}$, all the other variables in G must be negative variables. Assign all negative variables to $\overline{B}$ and assign v_1 to B. Since the number of negative variables is bounded by $r < \delta v^{1/3} < \sqrt{v}$, and since only one positive variable has been assigned, it is easy to see that $C\lceil_{\rho_{B,\overline{B}}} = 1$ and $D_2^v\lceil_{\rho_{B,\overline{B}}} = *$.

Our induction hypothesis is that for some fixed $k' \geq 2$ and for sufficiently large v, $s_{k'}(v) \geq \delta v^{1/3}$; that is, for sufficiently large v, if C is a $\Sigma_{k'-1}(r)$-circuit with $r < \delta v^{1/3}$, then C does not comupte the function $l_{k'}^v$. For the induction step, consider $k = k' + 1 > 2$. Consider a D_k^v circuit that computes an l_k^v function. Let $q = v^{-1/3}$ and let $\mathcal{B} = \{B_j\}$, where B_j is the set of variables of the bottom AND gate G_j of D_k^v. By way of contradiction, assume that there exists a circuit C, that is a $\Sigma_{k-1}(r)$-circuit with $r < \delta v^{1/3}$, computing the function l_k^v. We will argue that after applying a random restriction from $R_{q,\mathcal{B}}^+$, if k is

odd, or from $R^-_{q,\mathcal{B}}$, if k is even, to C, the resulting circuit $C\lceil_{\rho g(\rho)}$ is a $\Sigma_{k-2}(r)$-circuit with $r < \delta v^{1/3}$ that computes the function l^v_{k-1}, which contradicts our induction hypothesis.

For induction step, let C be a $\Sigma_{k-1}(r)$-circuit and let D^v_k be the circuit computing an l^v_k function. To apply the induction hypothesis, we need to verify that

(1) with a high probability, $C\lceil_{\rho_{\rho,g(\rho)}}$ is equivalent to a $\Sigma_{k-2}(r)$-circuit, and

(2) with a high probability, $D^v_k\lceil_{\rho g(\rho)}$ contains a subcircuit computing the function l^v_{k-1}.

Part (1) is proved by Lemma 2.7. To apply the Lemma 2.7, we choose $\gamma = \delta v^{1/3}$ We have $\alpha < 6q\gamma = 1/2$. The total number of depth-2 subcircuits at the bottom level of C is bounded by its size $2^r < 2^\gamma$. We need to show that $2^\gamma \leq 1/(3 \cdot \alpha^\gamma)$. We have

$$2^\gamma \cdot 3 \cdot \alpha^\gamma = 3(2\alpha)^{\delta v^{1/3}} \leq 1,$$

for sufficiently large v. Part (2) follows from Lemma 2.9. This completes the proof of Theorem 3.9. □

To satisfy requirement R'_n, we need to choose an integer m so large that Theorem 3.9 holds for $s_k(2^{m/(k+1)})$ and such that $q(m) < (1/12) \cdot 2^{m/3(k+1)}$.

Theorem 3.10 *There exists an oracle A such that for all $k \geq 2$, $\Sigma^P_{k-1}(A) \subsetneq \Theta^P_k(A)$.*

The constructions of Theorems 3.3, 3.7, 3.9 can all be dovetailed to obtain the following:

Theorem 3.11 *For all $k \geq 2$, there exists an oracle A such that $\Sigma^P_{k-1}(A) \subsetneq \Theta^P_k(A) \subsetneq \Delta^P_k(A) \subsetneq \Sigma^P_k(A)$.*

4 The extended low hierarchy has an infinite number of distinct levels

An interesting question is whether the low hierarchy or the extended low hierarchy consists of an infinite number of distinct levels. The existence of infinite number of distinct levels in the low hierarchy not only implies that P $\neq$ NP, but also provides a natural complexity categorization of sets in NP: sets at the higher levels of the low hierarchy are more "complicated" than sets at the lower levels. It is interesting to note that there exists an oracle set relative to which the low hierarchy is a properly infinite hierarchy, and that there also exists an oracle set relative to which the low hierarchy has a finite number of distinct levels [Ko91].

The situation is different for the extended low hierarchy. In proving optimal lower bounds for the extended low hierarchy, Allender and Hemachandra [AH92] constructed a sparse set that is not in $EL_2^{P,\Sigma}$, level 2 of the extended low hierarchy based on the Σ-levels of the polynomial-time hierarchy. It is known that sparse sets are in $EL_3^{P,\Theta}$, level 3 of the extended low hierarchy based on the Θ-levels of the polynomial-time hierarchy. This shows that the extended low hierarchy has at least two distinct levels.

In this section, we show that the extended low hierarchy actually consists of an infinite number of distinct levels. Our theorems are motivated by the observation that there are hard sets in the extended low hierarchy. As noted in [BBS86], any PSPACE complete set, for example, is in the extended low hierarchy. Sets, like hard sets, that can collapse the polynomial-time hierarchy to a fixed level, but no lower, are candidates for sets separating various levels of the extended low hierarchy.

To better understand why this happens, first consider the definition of the extended low hierarchy.

Definition 4.1 *[BBS86, AH92, LS91]*

1. *For each* $k \geq 1$, $EL_k^{P,\Sigma} = \{A \mid \Sigma_k^P(A) \subseteq \Sigma_{k-1}^P(A \oplus \mathrm{SAT})\}$.
2. *For each* $k \geq 2$, $EL_k^{P,\Delta} = \{A \mid \Delta_k^P(A) \subseteq \Delta_{k-1}^P(A \oplus \mathrm{SAT})\}$.
3. *For each* $k \geq 2$, $EL_k^{P,\Theta} = \{A \mid \Theta_k^P(A) \subseteq \Theta_{k-1}^P(A \oplus \mathrm{SAT})\}$.

Proposition 4.2 shows the basic relationships between the levels of the extended low hierarchy.

Proposition 4.2 *[AH92, LS91]*

1. *For each* $k \geq 1$, $EL_k^{P,\Sigma} \subseteq EL_{k+1}^{P,\Theta} \subseteq EL_{k+1}^{P,\Delta} \subseteq EL_{k+1}^{P,\Sigma}$.
2. $EL_1^{P,\Sigma} = EL_2^{P,\Theta} = EL_2^{P,\Delta}$.

The following observation is very useful: for all $k \geq 1$ and for all sets A,

(1) $\Sigma_k^P(A) \subseteq \Sigma_k^P(A \oplus \mathrm{SAT}) \subseteq \Sigma_{k+1}^P(A)$,
(2) $\Delta_k^P(A) \subseteq \Delta_k^P(A \oplus \mathrm{SAT}) \subseteq \Delta_{k+1}^P(A)$,
(3) $\Theta_{k+1}^P(A) \subseteq \Theta_{k+1}^P(A \oplus \mathrm{SAT}) \subseteq \Theta_{k+2}^P(A)$.

Thus, if a set A is so powerful that $\Sigma_{k+1}^P(A)$ collapses to $\Sigma_k^P(A)$, then A, by definition, is in the extended low hierarchy. This also holds for Δ-levels and Θ-levels of the polynomial-time hierarchy as well. We express this property in the following proposition.

Proposition 4.3

1. $\forall k \geq 0, \forall A\ [\Sigma_{k+1}^P(A) \subseteq \Sigma_k^P(A) \Rightarrow A \in EL_{k+1}^{P,\Sigma}]$.
2. $\forall k \geq 1, \forall A\ [\Delta_{k+1}^P(A) \subseteq \Delta_k^P(A) \Rightarrow A \in EL_{k+1}^{P,\Delta}]$.
3. $\forall k \geq 1, \forall A\ [\Theta_{k+1}^P(A) \subseteq \Theta_k^P(A) \Rightarrow A \in EL_{k+1}^{P,\Theta}]$.

The proofs of our main theorems, separating the levels of the extended low hierarchy, are motivated by Ko's recent result that, for each $k \geq 1$, there exists an oracle set relative to which the polynomial-time hierarchy has exactly k levels [Ko89]:

$$\forall k \geq 1, \exists A\ [\mathrm{PH}(A) = \Sigma_k^P(A) \supsetneq \Sigma_{k-1}^P(A)].$$

Note that this result strongly suggests that the Σ-levels of the extended low hierarchy are all distinct. Specifically, if $\mathrm{PH}(A) = \Sigma_k^P(A) \not\subseteq \Sigma_k^P(A \oplus \mathrm{SAT})$, then $A \in EL_{k+1}^{P,\Sigma} - EL_k^{P,\Sigma}$. It is a simple matter to modify Ko's result to show that the Σ-levels of the extended low hierarchy are all distinct. We significantly extend Ko's work by showing that each of the levels of the extended low hierarchy are distinct; that is,

$$\forall k \geq 2, EL_k^{P,\Sigma} \subsetneq EL_{k+1}^{P,\Theta} \subsetneq EL_{k+1}^{P,\Delta} \subsetneq EL_{k+1}^{P,\Sigma}.$$

4.1 Separating $EL_k^{P,\Sigma}$ and $EL_k^{P,\Delta}$

In this section, we will separate $EL_k^{P,\Sigma}$ and $EL_k^{P,\Delta}$ for each $k \geq 2$; that is, for arbitrary $k \geq 2$, we will construct a set A such that $A \in EL_k^{P,\Sigma}$ and $A \notin EL_k^{P,\Delta}$. From Proposition 4.3, we know that if $\Sigma_k^P(A) \subseteq \Sigma_{k-1}^P(A)$, then A is in $EL_k^{P,\Sigma}$. To separate $EL_k^{P,\Sigma}$ and $EL_k^{P,\Delta}$, we need to show that $\Delta_k^P(A) \not\subseteq \Delta_{k-1}^P(A \oplus \mathrm{SAT})$. Since $\Sigma_{k-1}^P(A) \subseteq \Delta_k^P(A)$, we will construct a set A such that $\mathrm{PH}(A)$ collapses to $\Sigma_{k-1}^P(A)$ and $\Sigma_{k-1}^P(A) \not\subseteq \Delta_{k-1}^P(A \oplus \mathrm{SAT})$. As a result, $\Delta_k^P(A) \not\subseteq \Delta_{k-1}^P(A \oplus \mathrm{SAT})$. For the case of $k = 2$, it is known that there exists an oracle A such that $\mathrm{PH}(A) = \mathrm{NP}(A) \neq \mathrm{P}(A)$ [BGS75]. A direct modification yields that $\mathrm{PH}(A) = \mathrm{NP}(A) \not\subseteq \mathrm{P}(A \oplus \mathrm{SAT})$. The case of $k \geq 3$ is handled in the following theorem.

Theorem 4.4 *For every recursive set B, and for each $k \geq 2$, there exists a recursive set A such that* $\mathrm{PH}(A) = \Sigma_k^P(A) \not\subseteq \Delta_k^P(A \oplus B)$.

Proof. Our proof will again use circuit lower bound techniques. For arbitrary $k \geq 2$, we need to collapse $\mathrm{PH}(A)$ to $\Sigma_k^P(A)$ and at the same time separate $\Sigma_k^P(A)$ and $\Delta_k^P(A \oplus B)$. To collapse PH to $\Sigma_k^P(A)$, we will encode the $\Sigma_{k+1}^P(A)$ complete set $K^{k+1}(A)$ into a set in $\Sigma_k^P(A)$. To separate $\Delta_k^P(A \oplus B)$ and $\Sigma_k^P(A)$, we will use the set L_k^A defined in the previous section:

$$L_k^A = \{x0^{kn} \mid |x| = 2n, (\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n) \cdots (Q_k y_k, |y_k| = n) \\ 0xy_1y_2 \ldots y_k \in A\}.$$

For each $k \geq 1$, define the predicate

$$\tau_k(A; x) \equiv (\exists y_1, |y_1| = n) \cdots (Q_k y_k, |y_k| = n)\ 1xy_1y_2 \ldots y_k \in A,$$

where $|x| = 2n$. Thus, for each x of length $2n$, the value of $\tau_k(A; x)$ only depends on strings in A of length $(k+2)n + 1$. Note that $\tau_k(A; x)$ is false if $|x|$ is odd. It is easy to verify that $\tau_k(A; x)$ is a $\Sigma_k^{P,1}$-predicate. Also, it is clear that, for any pair of strings x and y, whether $x \in L_k^A$ and $\tau_k(A; y)$ are true depends on the membership in A of different strings.

The construction of A will make sure that for all strings $x \in \Sigma^*$, $x \in K^{k+1}(A) \Leftrightarrow \tau_k(A; x0^{|x|})$. Thus, $\mathrm{PH}(A) = \Sigma^P_{k+1}(A) = \Sigma^P_k(A)$. We will also construct A so that $L^A_k \notin \Delta^P_k(A \oplus B)$. More specifically, A will be constructed by stages. Let $M_1, M_2, \cdots$ be an effective enumeration of deterministic polynomial time-bounded oracle machines. The set up of $t(n)$ and $D(n)$ is similar to that in the previous section. Let m be a multiple of $k+2$. At stage n of the construction, we will find an assignment of strings to A and $\overline{A}$ so that $0^m \in L^A_k$ if and only if $0^m \notin L(M_n, K^{k-1}(A \oplus B))$. In stage n, we also "$\Sigma^P_k(A)$-encode" into A strings $x \in K^{k+1}(A)$, $t(n-1)+1 \leq (k+2)\cdot|x|+1 \leq t(n)$, such that $x \in K^{k+1}(A) \Leftrightarrow \tau_k(A; x0^{|x|})$. The encoding is divided into two steps: before and after finding the witness 0^m separating $\Sigma^P_k(A)$ and $\Delta^P_k(A \oplus B)$. In the first step, we "$\Sigma^P_k(A)$-encode" into A all the strings $x \in K^{k+1}(A)$, where $t(n-1)+1 \leq (k+2)\cdot|x|+1 < m$. In the second step, we "$\Sigma^P_k(A)$-encode" into A all the strings $x \in K^{k+1}(A)$, where $m \leq (k+2)\cdot|x|+1 \leq t(n)$.

Let C_x be the circuit corresponding to the predicate $\tau_k(A; x0^{|x|})$ and let C_0 be the circuit corresponding to the predicate $0^m \in L^A_k$. Also, by Lemma 3.2, there is a polynomial q and a corresponding $\Delta_k(q(m))$-circuit C_n such that $C_n\lceil_{\rho_{A\oplus B}} = 1$ if and only if $0^m \in L(M_n, K^{k-1}(A \oplus B))$. We let $t(n) = q(m)$. Note that C_n can be simplified to another $\Delta_k(q(m))$-circuit C'_n by the following replacements:

(1) all the variables v_z in the leaves of C_n having the form $z = 1z'$ are replaced by 1 if $z' \in B$ and by 0 if $z' \notin B$, and

(2) all the variables v_z of the form $z = 0z'$ are replaced by $v_{z'}$.

Thus, for any set A, and a fixed set B, $C_n\lceil_{\rho_{A\oplus B}} = C'_n\lceil_{\rho_A}$ and $C'_n\lceil_{\rho_A} = 1$ if and only if $0^m \in L(M_n, K^{k-1}(A \oplus B))$.

The requirement for stage n can be described as the following

$$
\begin{aligned}
R_n : (\exists B_0, B_1 \subseteq D(n))[&B_0 \cap B_1 = \emptyset,\ C'_n\lceil_{\rho_{B_1,B_0}} \neq *, \text{ and for all } x,\\
&t(n-1) < (k+2)|x|+1 < m, C_x\lceil_{\rho_{B_1,B_0}} = 1 \Leftrightarrow x \in K^{k+1}(A(n-1) \cup B_1),\\
&\text{and } C_0\lceil_{\rho_{B_1,B_0}} = C_x\lceil_{\rho_{B_1,B_0}} = *, \text{ for all } x, m \leq (k+2)|x|+1 \leq t(n)],
\end{aligned}
$$

where ρ_{B_1,B_0} is the restriction that v_z is 1 if $z \in B_1$ and v_z is 0 if $z \in B_0$. Construction of A at stage n is done by the following steps.

Step 1. For all the strings x, $t(n-1) < (k+2)|x| + 1 < m$, assign strings to $A(n)$ and $\overline{A(n)}$ such that $\tau_k(A; x0^{|x|})$ is true if and only if $x \in K^{k+1}(A)$.

Step 2. Replace all the variables in C'_n that are in $A(n)$ by 1 and replace those in $\overline{A(n)}$ by 0. The resulting circuit, let's still call it C'_n, remains a $\Delta_k(q(m))$-circuit.

Step 3. Find a pair of finite sets B_1 and B_0 that satisfy the requirement R_n. Assign variables in B_1 to $A(n)$ and assign variables in B_0 to $\overline{A(n)}$.

Step 4. For all the strings x, $m \leq (k+2)|x| + 1 \leq t(n)$, assign remaining free strings to $A(n)$ and $\overline{A(n)}$ such that $\tau_k(A; x0^{|x|})$ is true iff $x \in K^{k+1}(A)$. If there are still remaining free strings of length between $t(n-1)+1$ and $t(n)$, assign them to $\overline{A(n)}$. Go to the next stage.

The key step is Step 3. The following lemma guarantees that the requirement R_n can be satisfied at Step 3.

Lemma 4.5 *For all $k \geq 2$, let $\{C_i\}_{i=1}^t$ be t circuits, each computing a function f_k^v, with pairwise disjoint variables. Let C be a $\Delta_k(r)$-circuit. If $t \leq 2^{\delta v^{1/6}}$ and $r \leq \delta v^{1/5}$, with $\delta = 1/48$, then for sufficiently large v, there exists a restriction ρ such that $C\lceil_\rho \neq *$ and $C_i\lceil_\rho = *$ for all $i = 1, \cdots, t$.*

Proof. The proof is by induction on k. For the case of $k = 2$, the proof is similar to the proof of Theorem 3.3. Recall that in the proof of Theorem 3.3, we found a restriction ρ such that $C\lceil_\rho \neq *$ and $C_2^v\lceil_\rho = *$, where C is an $\Delta_2(r)$-circuit with $r < \delta v^{1/5}$ and C_2^v computes an f_2^v function. (See (#) in the proof of Theorem 3.3.) The number of strings that we added to the oracle set was at most $r^2 \leq (\delta v^{1/5})^2 < \sqrt{v}$. Here, every C_i, $1 \leq i \leq t$, is an OR of v AND gates, each of fanin $\sqrt{v}$. Therefore, the same restriction ρ satisfies that $C_i\lceil_\rho = *$, for all i, $1 \leq i \leq t$.

For the induction step, consider any $k > 2$. As in the proof of Theorem 3.3, proceeding by contradiction, we need to verify that (1) with a high probability, $C\lceil_{\rho g(\rho)}$ is equivalent to a $\Delta_{k-1}(r)$-circuit, and (2) with a high probability, $C_i\lceil_{\rho g(\rho)}$ contains a subcircuit computing the function f^v_{k-1} for all $i = 1, \cdots, t$. Part (1) is identical to that in the proof of Theorem 3.3. Part (2) follows from Lemma 2.10. □

It is easy to see that C_0 contains a $C_k^{2^{m/(k+2)}}$ subcircuit computing an $f_k^{2^{m/(k+2)}}$ function. Also, for all x, $m \leq (k+2)|x| + 1 \leq t(n)$, C_x contains a subcircuit computing $f_k^{2^{m/(k+2)}}$. There are at most $2^{q(m)}$ many such C_x. To satisfy requirement R_n, we need to choose an integer m so large that $q(m) \leq (1/48)2^{m/6(k+2)}$. Thus, there exists an oracle set A such that $\Sigma^P_{k+1}(A) = \Sigma^P_k(A)$ and $\Sigma^P_k(A) \not\subseteq \Delta^P_k(A \oplus B)$, for each $k \geq 2$. This completes the proof of Theorem 4.4.

Corollary 4.6 *For each $k \geq 2$, $EL_k^{P,\Delta} \subsetneq EL_k^{P,\Sigma}$.*

Proof. As a point of interset, recall that $EL_2^{P,\Delta} = EL_1^{P,\Sigma}$ by Proposition 4.2. For $k = 2$, a direct modification of Baker, Gill, and Solovay's proof [BGS75] yields the existence of an oracle A such that $\mathrm{PH}(A) = \mathrm{NP}(A) \neq \mathrm{P}(A \oplus \mathrm{SAT})$. For $k > 2$, just let B be SAT in Theorem 4.4. □

Corollary 4.7 *For each $k \geq 1$, there exists an oracle set A such that $\Delta^P_k(A) \subsetneq \Sigma^P_k(A) = \mathrm{PH}(A) = \mathrm{PSPACE}(A)$.*

Proof. Simply replace the complete set $K^{k+1}(A)$ by a complete set for $\mathrm{PSPACE}(A)$ and let $B = \emptyset$ in the proof of Theorem 4.4. □

Ko also constructed an oracle set relative to which the polynomial-time hierarchy has a finite number of levels and is properly contained in PSPACE [Ko89]. By combining his proof techniques with those in Theorem 4.4, we have the following corollary:

Corollary 4.8 *For each $k \geq 1$, there exists an oralce set A such that $\Delta_k^P(A) \subsetneq \Sigma_k^P(A) =$ PH$(A) \subsetneq$ PSPACE(A).*

4.2 Separating $EL_k^{P,\Delta}$ and $EL_k^{P,\Theta}$

In this section, we will separate $EL_k^{P,\Delta}$ and $EL_k^{P,\Theta}$ for each $k \geq 3$. The set up is similar to that in Section 4.1: for arbitrary $k \geq 2$, we will construct a set A such that PH$(A) = \Delta_k^P(A) \not\subseteq \Theta_k^P(A \oplus \text{SAT})$. As a result, $A \in EL_{k+1}^{P,\Delta}$ and $A \notin EL_{k+1}^{P,\Theta}$.

Theorem 4.9 *For every recursive set B, and for each $k \geq 2$, there exists an oracle set A such that* PH$(A) = \Delta_k^P(A) \not\subseteq \Theta_k^P(A \oplus B)$

Proof. To collapse PH to $\Delta_k^P(A)$, we will encode the $\Sigma_k^P(A)$ complete set $K^k(A)$ into a set in $\Delta_k^P(A)$. To separate $\Delta_k^P(A)$ and $\Theta_k^P(A \oplus B)$, we will use the set $L_{odd,k}^A \in \Delta_k^P(A)$ defined in the previous section:

$$L_{odd,k}^A = \{x0^{kn} \mid |x| = n, \text{ and } max\{y \mid |y| = |x|, xy0^{(k-1)n} \in L_{k-1}^A\} \text{ has odd parity } \}.$$

Now define the following set:

$$\hat{L}_{odd,k}^A = \{x \mid max\{y \mid |y| = |x| \text{ and } \tau_{k-1}(A; xy) \text{ is true } \} \text{ has odd parity}\},$$

where τ_{k-1} is the predicate from Section 4.1. Notice that for each x of length n, whether $x \in \hat{L}_{odd,k}^A$ only depends on membership of strings in A of length $(k+1)n+1$.

It is easy to verify that $\hat{L}_{odd,k}^A$ is in $\Delta_k^P(A)$: just use binary search querying a set in $\Sigma_{k-1}^P(A)$. Also, it is clear that, for any pair of strings x and y, whether $x \in L_{odd,k}^A$ and $y \in \hat{L}_{odd,k}^A$ are true depends on the membership of different strings in A.

The rest of the proof is very similar to that in Section 4.1. A is constructed by stages. The set up of m, $t(n)$, and $D(n)$ is similar to that in Section 3.2. At stage n of the construction, we will find an assignment of strings to A and $\overline{A}$ so that $0^m \in L_{odd,k}^A$ if and only if $0^m \notin L(M_n, K^{k-1}(A \oplus B))$, where M_n is the nth machine in an enumeration of $\Theta_k^P(A \oplus B)$. In stage n, we "$\Delta_k^P(A)$-encode" into A any strings $x \in K^k(A)$, $t(n-1)+1 \leq (k+1) \cdot |x| + 1 \leq t(n)$, such that $x \in K^k(A) \Leftrightarrow x \in \hat{L}_{odd,k}^A$.

Again, let C_x be the circuit corresponding to the predicate $x \in \hat{L}^A_{odd,k}$ and let C_0 be the circuit corresponding to the predicate $0^m \in L^A_{odd,k}$. Also, by Lemma 3.6, there is a polynomial q and a corresponding $\Theta_k(q(m))$-circuit C_n such that $C_n\lceil_{\rho_{A\oplus B}} = 1$ if and only if $0^m \in L(M_n, K^{k-1}(A \oplus B))$. Let $t(n) = q(m)$. Simiar to Section 4.1, C_n can be simplified to a $\Theta_k(q(m))$-circuit C'_n such that $C_n\lceil_{\rho_{A\oplus B}} = C'_n\lceil_{\rho_A}$ for arbitrary set A and a fixed set B. The requirement of stage n can be described as the following

$$\begin{aligned} R_n : (\exists B_0, B_1 \subseteq D(n))[&B_0 \cap B_1 = \emptyset,\ C'_n\lceil_{\rho_{B_1,B_0}} \neq *, \text{ and for all } x, \\ &t(n-1) < (k+1)|x| + 1 < m, C_x\lceil_{\rho_{B_1,B_0}} = 1 \Leftrightarrow x \in K^k(A(n-1) \cup B_1), \\ &\text{and } C_0\lceil_{\rho_{B_1,B_0}} = C_x\lceil_{\rho_{B_1,B_0}} = *, \text{ for all } x, m \leq (k+1)|x| + 1 \leq t(n)], \end{aligned}$$

where ρ_{B_1,B_0} is the restriction that v_z is 1 if $z \in B_1$ and v_z is 0 if $z \in B_0$. Steps in construction of A at stage n are essentially the same as that in Section 4.1. The following lemma ensures that the requirement R_n can be satisfied.

Lemma 4.10 *For all $k \geq 2$, let $\{C_i\}_{i=1}^t$ be t circuits, each computing an h^v_k function, with pairwise disjoint variables. Let C be a $\Theta_k(r)$-circuit. If $t \leq 2^{\delta v^{1/6}}$ and $r \leq \delta v^{1/5}$, with $\delta = 1/24$, then for sufficiently large n, there exists a restriction ρ such that $C\lceil_\rho \neq *$ and $C_i\lceil_\rho = *$ for all $i = 1, \cdots, t$.*

Proof. Again, the proof is by induction on k. The base case of $k = 2$ is proved similarly to the proof of Theorem 3.7. Recall that in **Process FF**, we flip-flopped the output of an H^v_k circuit computing an h^v_k function by adding to the oracle set the variables in the leftmost undecided OR gate of H^v_k. We then find a restriction ρ such that $C\lceil_\rho \neq *$ and $H^v_2\lceil_\rho = *$ (see $(*)$ in the proof of Theorem 3.7.) Here we have t many H^v_k circuits. We can easily modify **Process FF** such that at each step of flip-flop, we add to the oracle set the variables of the leftmost undecided OR gate of C_i, for all $1 \leq i \leq t$. It is not difficult to see that, by a similar argument, we can find a restriction ρ such that $C\lceil_\rho \neq *$, and $C_i\lceil_\rho = *$, for all $1 \leq i \leq t$.

For the induction step, consider any $k > 2$. As in the proof of Theorem 3.7, proceeding by contradiction, we need to verify that (1) with a high probability, $C\lceil_{\rho g(\rho)}$ is equivalent

to a $\Theta_{k-1}(r)$-circuit, and (2) with a high probability, $C_i\lceil_{\rho g(\rho)}$ contains a subcircuit computing the function h^v_{k-1} for all $i = 1, \cdots, t$. Part (1) is identical to that in the proof of Theorem 3.7 and part (2) is proved by Lemma 2.10. □

Note that C_0 contains an $H_k^{2^{m/(k+1)}}$ subcircuit computing a function $h_k^{2^{m/(k+1)}}$. Also, for all x, $m \leq (k+1)|x| + 1 \leq t(n)$, C_x contains a subcircuit computing $h_k^{2^{m/(k+1)}}$. There are at most $2^{q(m)}$ many such C_x. To satisfy requirement R_n, we need to choose an integer m so large that $q(m) \leq (1/24)2^{m/6(k+1)}$. Thus, there exists an oracle set A such that $\Sigma_k^P(A) = \Delta_k^P(A)$ and $\Delta_k^P(A) \not\subseteq \Theta_k^P(A \oplus B)$, for each $k \geq 2$. This completes the proof of Theorem 4.9.

Corollary 4.11 *For each $k \geq 3$, $EL_k^{P,\Theta} \subsetneq EL_k^{P,\Delta}$.*

Proof. Simply replace B by SAT in the proof of Theorem 4.9. □

Corollary 4.12 *For each $k \geq 2$, there exists an oracle set A such that $\Theta_k^P(A) \subsetneq \Delta_k^P(A) =$ PH$(A) =$ PSPACE(A).*

Proof. Simply replace $K^k(A)$ by a complete set for PSPACE(A) and replace B by the empty set in the proof of Theorem 4.9. □

Corollary 4.13 *For each $k \geq 2$, there exists an oralce set A such that $\Theta_k^P(A) \subsetneq \Delta_k^P(A) =$ PH$(A) \subsetneq$ PSPACE(A).*

4.3 Separating $EL_k^{P,\Theta}$ and $EL_{k-1}^{P,\Sigma}$

In this section, we will find a recursive set A that is in $EL_k^{P,\Theta}$ but not in $EL_{k-1}^{P,\Sigma}$ for each $k \geq 3$. The set up is similar to that in Section 4.1 and 4.2: for each $k \geq 2$, we construct a set A such that PH$(A) = \Theta_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus$ SAT$)$. Thus, A is in $EL_{k+1}^{P,\Theta}$ and not in $EL_k^{P,\Sigma}$. We first prove the following theorem.

Theorem 4.14 *For every recursive set B, for each $k \geq 2$, there exists a recursive set A such that* $\mathrm{PH}(A) = \Theta_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus B)$.

Proof. Recall that $L_{xor,k}^A = \{x0^{kn} \mid |x| = n$, and exactly one of $0^n x 0^{(k-1)n}$ or $1^n x 0^{(k-1)n}$ is in $L_{k-1}^A\}$. Define the following language:

$$\hat{L}_{xor,k}^A = \{x \mid \text{exactly one of } \tau_{k-1}(A; 0^{|x|}x) \text{ or } \tau_{k-1}(A; 1^{|x|}x) \text{ is true } \},$$

where τ_k is defined in the previous section. It is easy to see that both $L_{xor,k}^A$ and $\hat{L}_{xor,k}^A$ are in $\Theta_k^P(A)$. A will be constructed so that $L_{xor,k}^A \notin \Sigma_{k-1}^P(A \oplus B)$ and so that for all x, $x \in K^k(A) \Leftrightarrow x \in \hat{L}_{xor,k}^A$. It will follow that $\mathrm{PH}(A) = \Theta_k^P(A)$ and $\Theta_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus B)$.

The set up of construction of A, as well as m, $t(n)$, and $D(n)$ are the same as in Section 4.1 and 4.2. At stage n of the construction, we will ensure that for all x, $t(n-1) < (k+1)|x| + 1 \leq t(n)$, $x \in K^k(A) \Leftrightarrow x \in \hat{L}_{xor,k}^A$. Let C_0 be the circuit corresponding to the predicate $0^m \in L_{xor,k}^A$ and for all x, $t(n-1) < (k+1)|x| + 1 \leq t(n)$, let C_x represent the circuit corresponding to the predicate $x \in \hat{L}_{xor,k}^A$. Let $\sigma_1, \sigma_2, \cdots$ be an enumeration of $\Sigma_{k-1}^{P,1}$-predicates. By Lemma 2.3, there exist a polynomial q and a $\Sigma_{k-1}(q(m))$-circuit C_n corresponding to the nth predicate σ_n. We let $t(n) = q(m)$. Using the same replacement technique as in the Section 4.1, C_n can be simplified to a Σ_{k-1}-circuit C_n' such that $C_n\lceil_{\rho_{A\oplus B}} = C_n'\lceil_{\rho_A}$ for arbitrary set A and a fixed set B. The requirement R_n of stage n of the construction is:

$$\begin{aligned} R_n : (\exists B_0, B_1 \subseteq D(n))[&B_0 \cap B_1 = \emptyset,\ C_n'\lceil_{\rho_{B_1,B_0}} \neq *, \text{ for all } x, \\ &t(n-1) < (k+1)|x| + 1 < m, C_x\lceil_{\rho_{B_1,B_0}} = 1 \Leftrightarrow x \in K^k(A(n-1) \cup B_0), \\ &\text{and } C_0\lceil_{\rho_{B_1,B_0}} = C_x\lceil_{\rho_{B_1,B_0}} = *, \text{ for all } x, m \leq (k+1)|x| + 1 \leq t(n)], \end{aligned}$$

where ρ_{B_1,B_0} is the restriction that v_z is 1 if $z \in B_1$ and v_z is 0 if $z \in B_0$. The steps in the construction of A at stage n are essentially the same as that in Section 4.1. We need the following lemma to ensure that R_n will can be satisfied at stage n.

Lemma 4.15 *For all $k \geq 2$, let $\{C_i\}_{i=1}^t$ be t circuits, each computing an l_k^v function, with pairwise disjoint variables. Let C be a $\Sigma_{k-1}(r)$-circuit. If $t \leq 2^{\delta v^{1/6}}$ and $r \leq \delta v^{1/3}$, with*

$\delta = 1/12$, then for sufficiently large n, there exists a restriction ρ such that $C\lceil_\rho \neq *$ and $C_i\lceil_\rho = *$ for all $i = 1, \cdots, t$.

Proof. The proof is by induction on k. To prove the base case of $k = 2$, let us consider circuits C and C_i for all $i = 1, \cdots, t$. C is an OR of AND's with bottom fanin $\leq r$. C_i is an XOR of two subcircuits, each computing an f_1^v function. Note that all the leaves of all C_i's are unique positive variables. Let v_i be the leftmost variable in C_i, $1 \leq i \leq t$, and let $V = \{v_i \mid 1 \leq i \leq t\}$.

Let F denote the variables that are assigned 1 and let $\overline{F}$ denote the variables that are assigned 0. Initially, both F and $\overline{F}$ are empty. We will first present the algorithm for constructing F and $\overline{F}$ and then show that the restriction $\rho_{F,\overline{F}}$ is the desired restriction. Recall that $\rho_\emptyset$ is the restriction that assigns 0 to all variables.

Process F:

$F = \overline{F} = \emptyset$.

if $C\lceil_{\rho_\emptyset} = 1$ **then**

Choose one AND gate of C with all negative variables; Assign all the variables to $\overline{F}$

else

if there is an $S \subseteq V$ such that $C\lceil_{\rho_S} = 1$ **then**

Select such an S;

Choose one AND gate D of C such that $D\lceil_{\rho_S} = 1$;

Assign all negative variables in D to $\overline{F}$ and assign all positive variables in D to F

else

$F = \emptyset, \overline{F} = \Sigma^* - V$.

end(* **if** *)

end(* **if** *)

We now argue the correctness of **Process F**. Consider the first case when $F = \emptyset$ and $C\lceil_{\rho_\emptyset} = 1$. Since C is a circuit of an OR of AND gates, there must exist an AND gate D in C such that D contains only negative variables. Put all the variables in D into $\overline{F}$ to preserve the the output of C. We have $C\lceil_{\rho_{F,\overline{F}}} = 1$. Note that the number of variables in D is $\leq r \leq \delta v^{1/3} < \sqrt{v}$, thus $C_i\lceil_{\rho_{F,\overline{F}}} = *$, for all $i = 1, \ldots, t$. In this case, $\rho_{F,\overline{F}}$ is the desired restriction.

On the other hand, if $C\lceil_{\rho_\emptyset} = 0$, then we choose some j many different v_i's and put these variables into S. Suppose, for all the different choices of S, $C\lceil_{\rho_S}$ remains 0. Then we know that for all 2^t possible S, $C\lceil_{\rho_S} = 0$, and, for each C_i, $C_i\lceil_{\rho_S} = 1$ if and only if $v_i \in S$. This implies that we can arbitrarily force any combination of C_i's to output 1 or 0, by choice of S, without changing the output of C. Thus, $C\lceil_{\rho_{\emptyset,\Sigma^*-V}} = 0$ and $C_i\lceil_{\rho_{\emptyset,\Sigma^*-V}} = *$, for all $1 \leq i \leq t$. Let $F = \emptyset$ and $\overline{F} = \Sigma^* - V$ and $\rho_{F,\overline{F}}$ is the desired restriction.

The last case is that there is an S such that $C\lceil_{\rho_S} = 1$. In this case, fix an AND gate D of C such that $D\lceil_{\rho_S} = 1$. We reserve the output of C by assigning the negative variables in D to $\overline{F}$ and the positive variables in D to F. Note that each positive variable in D is in S since D is an AND gate and $D\lceil_{\rho_S} = 1$. Also important is that for any restriction ρ_S, there is only one positive variable in C_i that is assigned 1 by ρ_S. The number of variables in $\overline{F}$ is bounded by $r \leq \delta v^{1/3} < \sqrt{v}$. Thus, $C\lceil_{\rho_{F,\overline{F}}} = 1$ and $C_i\lceil_{\rho_{F,\overline{F}}} = *$ for all $1 \leq i \leq t$. In summary, there is a pair of finite sets F and $\overline{F}$ such that $F \cap \overline{F} = \emptyset, C\lceil_{\rho_{F,\overline{F}}} \neq *$, and $C_i\lceil_{\rho_{F,\overline{F}}} = *$ for all i, $1 \leq i \leq t$.

Finally, the induction step is proved by Lemma 2.7 and Lemma 2.10. □

Note that C_0 contains a subcircuit computing an $l_k^{2^{m/(k+1)}}$ function. Also, for all x, $m \leq (k+1)|x| + 1 \leq t(n)$, C_x contains a subcircuit computing an $l_k^{2^{m/(k+1)}}$ function. To satisfy requirement R_n, we only need to choose an integer m so large that $t(n) \leq (1/12)2^{m/6(k+1)}$. We then conclude that there exists an oracle set A such that $\Sigma_k^P(A) = \Theta_k^P(A)$ and $\Theta_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus B)$, for each $k \geq 2$. This completes the proof of Theorem 4.14.

Corollary 4.16 *For each $k \geq 3$, $EL_{k-1}^{P,\Sigma} \subsetneq EL_k^{P,\Theta}$.*

Proof. Simply replace B by SAT in the proof of Theorem 4.14. $\square$

Corollary 4.17 *For each $k \geq 2$, there exists an oracle set A such that $\Sigma_{k-1}^P(A) \subsetneq \Theta_k^P(A) =$ PH$(A) =$ PSPACE(A).*

Proof. Simply replace $K^k(A)$ by a complete set for PSPACE(A) and replace B by the empty set in the proof of Theorem 4.14. $\square$

Corollary 4.18 *For each $k \geq 2$, there exists an oralce set A such that $\Sigma_{k-1}^P(A) \subsetneq \Theta_k^P(A) =$ PH$(A) \subsetneq$ PSPACE(A).*

Acknowledgment: The authors are grateful to the anonymous referees for their comments.

References

[AH92] E. Allender and L. Hemachandra. Lower bounds for the low hierarchy. *J. ACM*, 39(1):234–251, 1992.

[BBS86] J. Balcázar, R. Book, and U. Schöning. Sparse sets, lowness, and highness. *SIAM J. Comput.*, 15:739–747, 1986.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–441, Dec. 1975.

[FSS81] M. Furst, J. Saxe, and M. Sipser. Pairty, circuits, and the polynomial-time hierarchy. In *Proc. 22th Annual IEEE Symposium on Foundations of Computer Science*, pages 260–270, 1981.

[Has87] J. D. Hastad. *Computational limitations for small-depth circuits.* PhD thesis, Massachusetts Institute of Technology, 1987.

[Ko89] K. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM J. Comput.*, 18(2):392–408, April 1989.

[Ko91] K. Ko. Separating the low and high hierarchies by oracles. *Information and Computation*, 90(2):156–177, Feburary 1991.

[LS91] T. Long and M. Sheu. A refinement of the low and high hierarchies. Technical Report OSU-CISRC-2/91-TR6, The Ohio State University, 1991.

[Sch83] U. Schöning. A low and a high hierarchy within NP. *J. Comput. System Sci.*, 27:14–28, 1983.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 61–69, 1983.

[Sto76] L. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3:1–22, 1976.

[Wra76] C. Wrathall. Complete sets and the polynomial hierarchy. *Theor. Comput. Sci*, 3:23–33, 1976.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 1–10, 1985.