UP and the Low and High Hierarchies: A Relativized Separation [1]

Ming-Jye Sheu and Timothy J. Long

sheu-m@cis.ohio-state.edu, long@cis.ohio-state.edu
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

January 10, 1993

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

Abstract

The low and high hierarchies within NP were introduced by Schöning in order to classify sets in NP. It is not known whether the low and high hierarchies include all sets in NP. In this paper, using the circuit lower bound techniques of Håstad and Ko, we construct an oracle set relative to which UP contains a language that is not in any level of the low hierarchy and such that no language in UP is in any level of the high hierarchy. Thus, in order to prove that UP contains languages that are in the high hierarchy or that UP is contained in the low hierarchy, one needs to use non-relativized proof techniques. Since it is known that UP^A is low for PP^A for all sets A, our result also shows that the interaction between UP and PP is crucial for the lowness of UP for PP; changing the base class to any level of the polynomial-time hierarchy destroys the lowness of UP.

1 Introduction

The low and high hierarchies in NP were first introduced by Schöning in order to analyze the internal structure of NP [Sch83]. A set $A \in$ NP is said to be in level k of the low hierarchy if $\Sigma_k^P(A) = \Sigma_k^P$; and a set $B \in$ NP is said to be in level k of the high hierarchy if $\Sigma_{k+1}^P = \Sigma_k^P(B)$. Intuitively, when a set A is in level k of the low hierarchy, then the information content of A, with respect to the $\Sigma_k^P()$ operator, is so low that $\Sigma_k^P(A)$ has no more power than Σ_k^P. On the other hand, when a set B is in level k of the high hierarchy, then the information content of B, with respect to the $\Sigma_k^P()$ operator, is so high that $\Sigma_k^P(B)$ includes all of Σ_{k+1}^P. The low and high hierarchies provide a structure for classifying NP sets A based on the first level, of either hierarchy, to which A belongs.[1] There are numerous examples of sets in the low hierarchy including sparse sets in NP, NP sets with polynomial-time probabilistic algorithms, NP sets whose complements are in AM (Arthur-Merlin games [Bab85]), and NP sets with small size circuits. On the other hand, sets in the high hierarchy are complete sets for NP with respect to various polynomial-time reducibilities. For example, level 0 of the high hierarchy is the class of $\leq_T^P$-complete sets for NP and level 1 of the high hierarchy is the class of $\leq_T^{SN}$-complete sets for NP [Sch83].

The low and high hierarchies demonstrate an interesting structure within NP: the low hierarchy starts at P and grows "upward" toward the complete sets for NP, while the high hierarchy starts with complete sets for NP and grows "downward" toward P. One of the important open questions about this structure is what are the sets in NP that are in neither the low hierarchy nor the high hierarchy. In another words, what are the sets in NP that fall into the "gap" between the low and high hierarchies? The existence of a gap between the low and high hierarchy is strongly connected to the structure of the polynomial-time hierarchy. In [Sch83], Schöning showed that there exist sets between the low and high hierarchies if and only if the polynomial time hierarchy is infinite. However, it is not clear what kinds of NP sets these might be.

[1] As stated later, this does not classify all NP sets if the polynomial-time hierarchy is infinite.

In this paper we investigate whether the class UP includes sets that fall in the gap between the low and high hierarchies for NP. Recall that UP is the class of sets that are recognized by nondeterministic polynomial-time Turing machines that, for all inputs, have either zero or one accepting computations. Not only is UP an important class in complexity theory, but it also has strong connections to fundamental theoretical issues in cryptography [GS88].

Because the high hierarchy for NP consists of sets that are complete for NP with respect to various reducibilities, it seems very unlikely that any sets in UP would also belong to the high hierarchy. In fact, there is an oracle relative to which UP is not in level 1 of the high hierarchy. In [Bei92a], Beigel constructed an oracle set A such that $\Delta_2^{P,A} \not\subseteq \mathrm{PP}^A$. If there is a set $L \in \mathrm{UP}^A$ that is $\leq_T^{SN}$-complete for NP^A, then, $\Sigma_2^{P,A} \subseteq \mathrm{NP}^L \subseteq \mathrm{PP}^L \subseteq \mathrm{PP}^{\mathrm{UP}^A} = \mathrm{PP}^A$. This contradicts the fact that $\Delta_2^{P,A} \not\subseteq \mathrm{PP}^A$. Note that $\mathrm{PP}^{\mathrm{UP}^A} = \mathrm{PP}^A$ follows from the fact that UP^A is low for PP^A for all sets A. This is just a relativization of the recent result by Köbler, Schöning, Toda, and Torán that UP is low for PP[KSTT92]. We extend this level 1 result by constructing an oracle set A relative to which no languages in UP are in any level of the high hierarchy.

On the other hand, considering the fact that UP is low for PP, it is not entirely unreasonable to think that UP might belong to the low hierarchy for NP; that is, that each set in UP is low for some level of the polynomial-time hierarchy. To the contrary, we construct an oracle set A such that there is a language in UP^A that is not in the low hierarchy for NP^A.

Some observations can be made from our results. First, relative to the constructed oracle set A, no language in UP^A is in any level of the high hierarchy; that is, UP^A and the high hierarchy are completely disjoint. On the other hand, since $\mathrm{P} \subseteq \mathrm{UP}$ and since P is level 0 of the low hierarchy, we cannot make UP^A completely disjoint from the low hierarchy. Instead, we prove the existence of a language in UP^A that is not in any level of the low hierarchy. Finally, since UP^A is not low for NP^A, or for any level of the polynomial-time hierarchy relative to A, and since UP^B is low for PP^B for all sets B, this shows that the lowness of UP for PP depends critically on the interaction between the base class PP and the oracle class UP; replacing the base class with any level of the polynomial-time hierarchy destroys the lowness of UP.

Our construction of an oracle set A relative to which UP is not in any level of the low and high hierarchies follows the circuit lower bound techniques from Yao [Yao85], Håstad [Has87], and Ko [Ko89]. In the case of the high hierarchy, we use an interesting result from Hartmanis and Hemachandra [HH91]. They show that if a nondeterministic polynomial time oracle machine N, relative to every oracle set and on every input string, has at most one accepting computation, then N behaves like a deterministic polynomial time oracle machine with SAT added to the oracle set. Using this property, together with the circuit lower bound techniques, we show that

1. $\exists A\ \forall k \geq 0\ [\mathrm{UP}(A) \cap H_k^{P,\Sigma}(A) = \emptyset]$.

In constructing an oracle set A relative to which UP is not in any level of the low hierarchy, we show that Håstad's switching lemma also holds for a new type of random restriction: restrictions that satisfy the "U property", where a restriction satisfies the "U property" if it assigns at most one variable the value of 1 in each block of a partition of the variables. Using such restrictions, we are able to show that

2. $\exists A\ \exists U_A \in \mathrm{UP}(A) \forall k \geq 0\ [U_A \notin L_k^{P,\Sigma}(A)]$.

Some of the immediate corollaries include:

3. $\exists B\ [\mathrm{BPP}^{\mathrm{UP}(B)} \not\subseteq \mathrm{BPP}^B]$: UP is not low for BPP.

4. $\exists C\ [\#\mathrm{P}^{\mathrm{UP}(C)} \not\subseteq \#\mathrm{P}^C]$: UP is not low for #P.

5. $\exists D\ [\mathrm{UP}(D) \not\subseteq \mathrm{BPP}(D)]$: UP is not contained in BPP.

6. $\exists E\ [\mathrm{UP}(E) \not\subseteq \mathrm{P}/poly(E)]$: UP does not have polynomial size circuits.

Corollary 4 is particularly interesting since UP is known to be low for GapP, the function class obtained by closing #P under subtraction. Our results indicate that UP is unlikely to be low for #P. The fact that UP is low for GapP but is unlikely to be low for #P demonstrates a distinction between GapP and #P. Corollary 4 also shows that #P in not closed under subtraction relative to A. This observation is due to Beigel [Bei92b].

Note that there are oracles relative to which UP is in the low hierarchy while the polynomial-time hierarchy contains an infinite number of distinct levels. In [BI87], Blum and Impagiazzo constructed an oracle relative to which P $=$ NP $\cap$ co-NP $=$ UP and, at the same time, the polynomial-time hierarchy is infinite. Note that the constructed oracle collapses the first two levels of the low hierarchy. It would be interesting to know if the the same oracle does not collapse the entire low hierarchy. It remains an open question that if there exists an oracle relative to which UP is in the high hierarchy and yet the polynomial-time hierarchy is infinite.

This paper is organized as follows. In Section 3, we define a new type of random restrictions and show that Håstad's Switching Lemma still holds for this new type of random restriction. In Section 4, we construct an oracle set relative to which UP contains a set that is not in any level of the low hierarchy. In Section 5, we construct an oracle set relative to which UP and the high hierarchy are disjoint. The constructions in Section 4 and 5 can be combined to yield a oracle set relative to which UP is not in low or high hierarchies. Finally, in Section 6, we combine our proof with the proof in Ko [Ko91] and show that there is an oracle A relative to which the low and high hierarchies have infinitely many distinct levels and UP contains a language outside the low hierarchy and contains no languages in the high hierarchy.

2 Definitions

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. Let $<,>$ be a standard one-to-one pairing function. 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 class of complements of the elements of $\mathcal{C}$. $L(M)$ 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. P(A) and NP(A) are the A-relativized versions of P and NP.

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

$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \mathrm{P}$, and for $k \geq 0$,

$\Sigma_{k+1}^P = \mathrm{NP}(\Sigma_k^P)$,

$\Pi_{k+1}^P = \text{co-}\Sigma_{k+1}^P$,

$\Delta_{k+1}^P = \mathrm{P}(\Sigma_k^P)$.

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

Now let a $\Sigma_0^{P,1}$-predicate be a P^1-predicate. For all $k \geq 1$, define a $\Sigma_k^{P,1}$-predicate to be a predicate of the form:

$$\tau(A; x) \equiv (\exists_p y_1)(\forall_p y_2) \cdots (Q_k y_k)\ \sigma(A; \langle x, y_1, \cdots, y_k \rangle),$$

where σ is a P^1-predicate and Q_k is $\exists_p$ if k is odd and Q_k is $\forall_p$ if k is even.[2] The classes of the relativized polynomial-time hierarchy (PH(A)) can be defined as follows:

[2] $\exists_p y$ is defined as $\exists y, |y| \leq p(|x|)$ for some polynomial p. $\forall_p y$ has the similar definition.

$\Sigma_k^P(A) = \{L \mid (\exists \Sigma_k^{P,1}\text{-predicate } \tau)[x \in L \Leftrightarrow \tau(A;x)]\}$,

$\Pi_k^P(A) = \text{co-}\Sigma_k^P(A)$, for $k \geq 0$.

It is well know that the polynomial-time hierarchy can also be defined as the relativized polynomial-time hierarchy relative to the empty set and that the oracle machine and the quantifier characterizations of the polynomial-time hierarchy are equivalent [BDG88]. For any set A, let $K(A)$ be the standard many-one complete set for NP(A). The many-one complete set for Σ_k^P, where $k \geq 1$, is defined inductively as follows: $K^1(A) = K(A)$, and $K^{k+1}(A) = K(K^k(A))$.

Based on the idea from recursion theory, Schöning [Sch83] intrudoced the low and high hierarchies within NP. Observe the following containments:

$$\Sigma_k^P \subseteq \Sigma_k^P(A) \subseteq \Sigma_{k+1}^P$$

where $A \in$ NP and $k \geq 1$. If $A \in$ P, then the left containment becomes equality and if A is an NP-complete set, then the right containment becomes equality. This motivates the definition of the low and high hierarchies.

Definition 2.1 *[Sch83]*

1. *The low hierarchy in* NP *is* $\{L_k^{P,\Sigma} \mid k \geq 0\}$*, where* $L_k^{P,\Sigma} = \{A \in \text{NP} \mid \Sigma_k^P(A) \subseteq \Sigma_k^P\}$.
2. *The high hierarchy in* NP *is* $\{H_k^{P,\Sigma} \mid k \geq 0\}$*, where* $H_k^{P,\Sigma} = \{A \in \text{NP} \mid \Sigma_{k+1}^P \subseteq \Sigma_k^P(A)\}$.

It is known that $L_k^{P,\Sigma} \subseteq L_{k+1}^{P,\Sigma}$ and $H_k^{P,\Sigma} \subseteq H_{k+1}^{P,\Sigma}$ for all $k \geq 0$.

UP is the class of sets that are recognizable by nondeterministic polynomial-time Turing machines that have, on all inputs, either zero or one accepting computation. More formally, let $ACC_M(x)$ denote the number of accepting paths of nondeterministic polynomial-time Turing machine M on input x. The class UP is defined as follows.

Definition 2.2 *A set* L *is in* UP *if and only if there is a nondeterministic polynomial-time Turing machine* M_L *such that for all* x*,*

$$x \in L \Rightarrow ACC_{M_L}(x) = 1, \textit{ and}$$
$$x \notin L \Rightarrow ACC_{M_L}(x) = 0.$$

The class FewP is the class of sets that are recognized by nondeterministic polynomial-time Turing machines that have at most polynomially many accepting computations.

Definition 2.3 *A set L is in* FewP *if and only if there is a nondeterministic polynomial-time Turing machine M_L and a polynomial q such that for all x,*

$$x \in L \Rightarrow 0 < ACC_{M_L}(x) \leq q(|x|), \textit{ and}$$
$$x \notin L \Rightarrow ACC_{M_L}(x) = 0.$$

It is clear that UP $\subseteq$ FewP $\subseteq$ NP.

In [FFK91] the class UP is generalized to the class SPP as follows

Definition 2.4 *A set L is in* SPP *if and only if there is a nondeterministic polynomial-time Turing machine M_L such that for all x,*

$$x \in L \Rightarrow ACC_{M_L}(x) - REJ_{M_L}(x) = 1, \textit{ and}$$
$$x \notin L \Rightarrow ACC_{M_L}(x) = REJ_{M_L}(x),$$

where $REJ_{M_L}(x)$ is the number of rejecting computation paths of M_L on input x.

Clearly, UP $\subseteq$ SPP.

3 Circuits and the switching lemma

A circuit is a rooted tree, where each non-leaf node is labeled with a gate (AND or OR), and has an unlimited, but finite number of children. Each leaf is labeled with a constant 0, a constant 1, a variable x, or a negated variable $\overline{x}$. The dual circuit $\hat{C}$ of a circuit C is defined inductively as follows: the dual of a constant or a variable is its negation, the dual of a circuit C which is an OR (or, AND) of n children $C_i, 1 \leq i \leq n$, is the AND (or, OR) of $\hat{C}_i$, $1 \leq i \leq n$.

A circuit computes a function of its variables. 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 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 an *assignment* of C. Also, we say a restriction ρ *completely determines* C if $C\lceil\rho$ computes a constant function 0 or 1. 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 a pair of sets B and B', where $B \cap B' = \emptyset$, let $\rho_{B,B'}$ be a restriction that assigns 1 to variables in B, 0 to variables in B', and * to all other variables. The composition of two restrictions $\rho_1\rho_2(x)$ for any variable x is defined as $\rho_2(\rho_1(x))$.

Random restrictions of circuits were first introduced by Furst, Saxe and Sipser [FSS84]. Here we use restrictions that are motivated by, but not identical to, the restrictions used by Håstad [Has87]. We define two probability spaces of restrictions $\hat{R}^+_{q,\mathcal{B}}$ and $\hat{R}^-_{q,\mathcal{B}}$. We also define a function g' that maps a random restriction to a restriction. Let $\mathcal{B} = \{B_j\}_{j=1}^r$ be a partition of the variables in a circuit C and let q be a value between 0 and 1. $\hat{R}^+_{q,\mathcal{B}}$, $\hat{R}^-_{q,\mathcal{B}}$, and g' are defined in such a way that a restriction $\rho g'(\rho)$ (composition of ρ and $g'(\rho)$) has the following property: at most one variable is assigned * or 0 in each block B_j, if ρ is a random restriction from $\hat{R}^+_{q,\mathcal{B}}$, and at most one variable is assigned * or 1 in each block B_j, if ρ is a random restriction from $\hat{R}^-_{q,\mathcal{B}}$. We call this property the *U property*.

To define a random restriction ρ in $\hat{R}^+_{q,\mathcal{B}}$, for each variable $x \in B_j$, $1 \leq j \leq r$, let $\rho(x) = *$ with probability q and let $\rho(x) = 1$ with probability $1 - q$. The probability space $\hat{R}^-_{q,\mathcal{B}}$ is defined analogously by changing the 1 to 0. Next, for each $\rho \in \hat{R}^+_{q,\mathcal{B}}$, define a restriction $g'(\rho)$ such that for each B_j, $1 \leq j \leq r$, let $s_j = *$ with probability q and let $s_j = 0$ with probability $1-q$. Then, 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 s_j to y and value 1 to all other variables in V_j. For ρ from $\hat{R}^-_{q,\mathcal{B}}$, $g'(\rho)$ is defined analogously by interchanging the roles of 0 and 1.

Circuits can be used to describe the computation of oracle machines on input strings. The following proposition states this fact.

Proposition 3.1 *[FSS81, Ko89] Let M be an 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* AND*s,*
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.

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 is a Σ_k-circuit such that

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

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

A depth-$(k+1)$ circuit is a Π_k-circuit ($\Pi_k(m)$-circuit) if it is the dual circuit of a Σ_k-circuit ($\Sigma_k(m)$-circuit). Now we can translate $\Sigma_k^{P,1}$-predicates to circuits.

Proposition 3.2 *[Ko89, SL91] 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 *in* $C_{\tau,x}$*,* $|z| \leq q(|x|)$*.*

Ko [Ko91] introduced a family of functions f_k^n, $n \geq 1, k \geq 1$, computed by special types of circuits.

Definition 3.3 *[Ko91] 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*
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*.*

The switching lemma from Håstad [Has87] has been an important tool in proofs constructing oracle sets that separate the polynomial-time hierarchy from PSPACE and that separate every level of the polynomial-time hierarchy [Has87, Ko89]. The following lemma states that the switching lemma also holds for random restrictions with the *U property.*

Lemma 3.4 (Switching Lemma) *Let* G *be an* AND *of* OR*s with bottom fanin* $\leq t$*, let* $\mathcal{B} = \{B_j\}$ *be a partition of the variables in* G*, and let* q *be a number between 0 and 1. Then, for a random restriction* ρ *from* $\hat{R}^+_{q,\mathcal{B}}$*,* $Pr(G\lceil_{\rho g'(\rho)}$ *is not equivalent to a circuit consisting of an* OR *of* AND*s with bottom fanin* $\leq s)$ *is bounded by* α^s*, where* $\alpha < 6qt$*. The above probability also holds with* $\hat{R}^+_{q,\mathcal{B}}$ *replaced by* $\hat{R}^-_{q,\mathcal{B}}$*, or with* G *being an* OR *of* AND*s circuit to be converted to an* AND *of* OR*s circuit.*

Proof. See Appendix. □

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 fanin. 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 states this shrinking effect.

Lemma 3.5 *For arbitrary $k \geq 2$, let C be a $\Sigma_k(r)$-circuit (or a $\Pi_k(r)$-circuit), where $r \leq t$. Let V be the set of the variables in C and let $\mathcal{B} = \{B_j\}$ be a partition of the variables in V. Let $q = 1/(12 \cdot t)$ and let ρ be a random restriction from $\hat{R}^+_{q,\mathcal{B}}$ or $\hat{R}^-_{q,\mathcal{B}}$. The probability that C, after applying the random restriction $\rho g'(\rho)$, can be converted to an equivalent $\Sigma_{k-1}(r)$-circuit (or a $\Pi_{k-1}(r)$-circuit, respectively) is greater than $2/3$.*

Proof. See appendix. □

Lemma 3.6 shows that a random restriction with the *U property* transforms a circuit computing f_k^n to a circuit that contains a subcircuit computing an f_{k-1}^n function.

Lemma 3.6 *For each $k \geq 2$, there exists an integer n_k such that the following holds for all $n \geq n_k$: Let C be a C_k^n circuit computing an f_k^n function, let $q = n^{-1/3}$, and let ρ be a random restriction from $\hat{R}^+_{q,\mathcal{B}}$ if k is even or from $\hat{R}^-_{q,\mathcal{B}}$ if k is odd. The probability that $C\lceil_{\rho g'(\rho)}$ contains a subcircuit computing an f_{k-1}^n function is greater than $2/3$.*

Proof. See appendix. □

4 UP is not in the Low Hierarchy

In this section, we will construct an oracle set A relative to which UP contains a language that is not in any level of the low hierarchy in NP; that is, we show that there exist an oracle set A and a set $U_A \in \mathrm{UP}(A)$ such that, for each $k \geq 1$, $\Sigma_{k-1}^P(U_A) \not\subseteq \Pi_k^P(A)$. This implies that $\Sigma_k^P(U_A) \not\subseteq \Sigma_k^P(A)$ and thus $\Sigma_k^P(U_A \oplus A) \not\subseteq \Sigma_k^P(A)$. Therefore, U_A is not in level k of the relativized low hierarchy for all $k \geq 1$. Note that if U_A is not in level k, $k \geq 1$, of the relativized low hierarchy, then U_A cannot be in level 0 of the relativized low hierarchy.

Let $e(0) = 1$ and $e(n+1) = 2^{2^{e(n)}}$ for all $n > 0$. Then, for arbitrary set A, U_A is defined in such a way that the strings in U_A are of the format $0^k10^nw0^j$. For a string x with this particular format, the membership of x in U_A depends on membership of strings in A of length $e(\langle 4k, m\rangle)$, for some m. Also, in this format, $j = e(\langle 4k, m\rangle)$ MOD $k+1$, $|0^nw| = kn$, and $|x| = e(\langle 4k, m\rangle) - n$. The definition of U_A is as follows:

$x \in U_A \Rightarrow x$ is of the form $0^k10^nw0^j$, $k \geq 1$, $0 \leq j < k+1$, $|w| = (k-1)n$,
$|x| = e(\langle 4k, m\rangle) - n$ for some m, and there exists exactly one y,
$|y| = n$, such that $xy \in A$,

$x \notin U_A \Rightarrow x$ is not of the form $0^k10^nw0^j$, $k \geq 1, 0 \leq j < k+1$, $|w| = (k-1)n$,
$|x| = e(\langle 4k, m\rangle) - n$ for some m, or x is of this form and for all y,
$|y| = n$, $xy \notin A$.

Note that for every x, if x is of the specific form in the definition of U_A, then x's membership in U_A only depends on strings with prefix 0^k1 in A of length $e(\langle 4k, m\rangle)$ for some m. Now for any fixed $k \geq 1$, when k is odd define

$$I_k^A = \{0^n \mid (\exists m)(\exists j, 0 \leq j < k+1)[(k+1)(n+1)+j = e(\langle 4k, m\rangle)] \text{ and }$$
$$(\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(\forall y_{k-1}, |y_{k-1}| = n)0^k10^ny_1y_2\ldots y_{k-1}0^j \in U_A\},$$

and when k is even, define

$$I_k^A = \{0^n \mid (\exists m)(\exists j, 0 \leq j < k+1)[(k+1)(n+1)+j = e(\langle 4k, m\rangle)] \text{ and }$$
$$(\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(\exists y_{k-1}, |y_{k-1}| = n)0^k10^ny_1y_2\ldots y_{k-1}0^j \notin U_A\}.$$

It is clear that $I_k^A \in \Sigma_{k-1}^P(U_A)$ for $k \geq 1$.

Our goal is to construct an oracle set A so that $U_A \in \mathrm{UP}(A)$ and $I_k^A \notin \Pi_k^P(A)$ for all $k \geq 1$. This shows that, for all $k \geq 1$, $U_A \notin L_k^{P,\Sigma}$ relative to A and thus $\mathrm{UP}(A)$ contains a language that is not in any level of the low hierarchy relative to A.

The construction of set A is by stages. Let $\sigma_1^k, \sigma_2^k, \cdots$ be an effective enumeration of $\Sigma_k^{P,1}$-predicates. We divide each goal $I_k^A \notin \Pi_k^P(A)$ into an infinite number of requirements:

$$R_{k,i} : (\exists n)[0^n \in I_k^A \Leftrightarrow \sigma_i^k(A; 0^n) \text{ is true }] \text{ and } U_A \in \mathrm{UP}(A).$$

If requirements $R_{k,i}$ are satisfied for all i, then $I_k^A \notin \Pi_k^P(A)$. Also, since $I_k^A \in \Sigma_{k-1}^P(U_A)$ and $U_A \in \mathrm{UP}(A)$, we have $I_k^A \in \Sigma_{k-1}^P(\mathrm{UP}(A))$. Thus, $U_A \notin L_k^{P,\Sigma}(A)$.

For arbitrary fixed k, $k \geq 1$, let n and m be a pair of numbers such that $(\exists j, 0 \leq j < k+1)[(k+1)(n+1)+j = e(\langle 4k, m\rangle)]$. Let C_f be the following circuit: C_f is a k-level circuit having alternating OR and AND gates with a top OR gate. The fanin at each level is 2^n and the variables at the bottom level of C_f are the variables of the form $0^k10^n y_1 y_2 \ldots y_{k-1} 0^j y$, where $|y_1| = |y_2| = \cdots = |y_{k-1}| = |y| = n$. Also, if k is odd then the variables in C_f are all positive and if k is even then the variables in C_f are all negative. Note that if k is odd, then C_f contains a $C_k^{2^n}$ subcircuit computing an $f_k^{2^n}$ function. If k is even, then C_f contains a subcircuit with the same structure as a $C_k^{2^n}$-circuit except that all variables are negated. In order to ensure that U_A is in $\mathrm{UP}(A)$, we need to place an appropriate restriction on the assignment of variables in C_f. Let circuit C_k^v compute an f_k^v function. Consider the following restriction on assignments of variables appearing in C_k^v:

> **The U condition for** C_k^v: let $G_1, G_2, \ldots$ be the bottom OR gates of C_k^v if k is odd or the bottom AND gates of C_k^v if k is even. Let $V_1, V_2, \ldots$ be the set of variables in $G_1, G_2, \ldots$, respectively. If k is odd, then the assignment of variables in V_i, for all i, must have at most one variable assigned a 1. If k is even, then the assignment of variables in V_i, for all i, must have at most one variable assigned a 0.

Note that, if k is odd, the bottom OR gates of C_f consist of unique positive variables, and C_f contains a $C_k^{2^n}$ subcircuit. Thus, for an assignment of variables satisfying the U condition for C_f, C_f correctly computes the predicate "$0^n \in I_k^A$" and $U_A \in \mathrm{UP}(A)$. When k is even, the bottom AND gates of C_f consist of unique negative variables. For each negative variable $\bar{x}_j$ in C_f, we replace it by a positive variable y_j. Let this new circuit be C'_f. C'_f then contains a $C_k^{2^n}$ subcircuit computing an $f_k^{2^n}$ function. It is easy to see that an assignment that satisfies the U condition for C'_f assigns value 1 to at most one variable x_j and assigns 0 to all other variables x_l. Therefore, for any assignment satisfying the U condition for C'_f, C_f correctly computes the predicate "$0^n \in I_k^A$" and $U_A \in \mathrm{UP}(A)$.

Let C_i be the $\Pi_k(p_i(n))$-circuit corresponding to the negation of the predicate $\sigma_i^k(A;0^n)$. Our goal is to find an assignment of strings to A and $\overline{A}$ corresponding to the U condition for C_f, if k is odd, or the U condition for C'_f, if k is even, such that C_f and C_i output different values under this assignment. The requirement $R_{k,i}$ can be rewritten as

$$R'_{k,i} : (\exists x = 0^n)[C_f\lceil_{\rho_A} = 1 \Leftrightarrow C_i\lceil_{\rho_A} = 0], \text{ where } \rho_A \text{ satisfies}$$
$$\text{the U condition for } C_f \text{ if } k \text{ is odd or for } C'_0 \text{ if } k \text{ is even.}$$

For each $k \geq 1$, let $s_k(v)$ be the minimum r such that some $\Pi_k(r)$-circuit can compute an f_k^v function when the assignment of variables in the circuit C_k^v for f_k^v satisfies the U condition for C_k^v. More formally,

$$s_k(v) = min\{r \mid (\exists\ D,\ \text{a } \Pi_k(r)\text{−circuit})(\forall\sigma)[\sigma \text{ satisfies the U condition for } C_k^v \Rightarrow D\lceil_\sigma = C_k^v\lceil_\sigma]\}$$

The following lower bound on $s_k(v)$ implies that the requirement $R'_{k,i}$ can be satisfied.

Theorem 4.1 *For all $k \geq 1$, for sufficiently large v, $s_k(v) \geq \delta v^{1/3}$, where $\delta = 1/12$.*

Proof. The proof is by induction on k. For the base of $k = 1$, let C_1^v be a circuit computing an f_1^v function. C_1^v is an OR gate with $\sqrt{v}$ fanin. Let C be a $\Pi_1(r)$-circuit with $r < \delta v^{1/3}$. We will find a restriction ρ such that ρ satisfies the U condition for C_1^v and such that $C_1^v\lceil_\rho \neq C\lceil_\rho$.

C is an AND of ORs with bottom fanins $\leq r$. Let $B = \emptyset$. There are two cases to consider.

Case 1: $C\lceil_{\rho_B}= 0$.

In this case, there is an OR gate G in C that contains only positive variables. Assign these variables to $\overline{B}$ to preserve the output of C. Choose one free variable in C_1^v and assign it to B and assign all other free variables in C_1^v to $\overline{B}$.

Case 2: $C\lceil_{\rho_B}= 1$

In this case assign all free variables to $\overline{B}$.

Now we argue that ρ_B is the desired restriction. Consider case 1. Since every OR gate of C has at most r variables and $r < \delta v^{1/3} < \sqrt{v}$, there is a free variable in C_1^v that has not been assigned to $\overline{B}$. Thus, $C\lceil_{\rho_B}= 0$ and $C_1^v\lceil_{\rho_B}= 1$. It is clear that we only assign one variable in C_1^v to B. Therefore the restriction ρ_B satisfies the U condition for C_1^v.

For case 2, all the variables are assigned to $\overline{B}$ and we have $C\lceil_{\rho_B}= 1$ and $C_1^v\lceil_{\rho_B}= 0$. It is also easy to see that the restriction ρ_B satisfies the U condition for C_1^v.

Our inductive hypothesis is that for some fixed $k' \geq 1$, if C is a $\Pi_{k'}(r)$-circuit with $r < \delta v^{1/3}$, then there is an assignment σ of variables in the circuit $C_{k'}^v$ computing $f_{k'}^v$ such that σ satisfies the U condition for $C_{k'}^v$ and such that $C_{k'}^v\lceil_\sigma \neq C\lceil_\sigma$.

For the induction step, let $k = k' + 1 \geq 2$. Consider a C_k^v circuit that computes f_k^v. Let $q = v^{-1/3}$ and let $\mathcal{B} = \{B_j\}$, where B_j is the set of variables of the bottom OR gate G_j of C_k^v if k is odd, or is the set of variables of the bottom AND gate G_j of C_k^v if k is even.

Let C be a $\Pi_k(r)$-circuit with $r < \delta v^{1/3}$. In order to apply the inductive hypothesis, we need to show that after applying a random restriction ρ from $\hat{R}_{q,\mathcal{B}}^+$ if k is even or from $\hat{R}_{q,\mathcal{B}}^-$ if k is odd to C_k^v and C,

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

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

Part (1) is proved by Lemma 3.6 and part (2) is proved by Lemma 3.5.

Now by inductive hypothesis, there is an assignment σ such that $C_k^v\lceil_{\rho g'(\rho)\sigma} \neq C\lceil_{\rho g'(\rho)\sigma}$. It remains to show that the assignment $\rho g'(\rho)\sigma$ satisfies the U condition for C_k^v.

Consider the case when k is even. Recall that ρ is a restriction from $\hat{R}^+_{q,\mathcal{B}}$. For each j, the restriction $\rho g'(\rho)$ chooses at most one variable from B_j and assigns 0 or $*$ to it and assigns 1 to all other variables in B_j. If the variable in B_j was assigned $*$ by $\rho g'(\rho)$, then it will be assigned either 1 or 0 by σ. In either case, there is at most one variable in B_j being assigned 0 in the assignment $\rho g'(\rho)\sigma$. Thus, the assignment $\rho g'(\rho)\sigma$ satisfies the U condition for C_k^v.

In the case that k is odd, recall that ρ is a restriction from $\hat{R}^-_{q,\mathcal{B}}$. For each j, the restriction $\rho g'(\rho)$ chooses at most one variable from B_j and assigns 1 or $*$ to it and assigns 0 to all other variables in B_j. If the variable in B_j was assigned $*$ by $\rho g'(\rho)$, then it will be assigned either 1 or 0 by σ. In either case, there is at most one variable in B_j being assigned 1 in the assignment $\rho g'(\rho)\sigma$. Thus, the assignment $\rho g'(\rho)\sigma$ satisfies the U condition for C_k^v. This completes the proof of Theorem 4.1. □

Now we can start our construction of A. Initially, let $A(1) = \overline{A}(1) = \emptyset$. In each stage α, we will define sets $A(\alpha+1)$ and $\overline{A}(\alpha+1)$ such that $A(\alpha+1) \cap \overline{A}(\alpha+1) = \emptyset$ and that $A(\alpha+1)$ is an extension of $A(\alpha)$ and $\overline{A}(\alpha+1)$ is an extension of $\overline{A}(\alpha)$.

At stage α, $\alpha \neq \langle 4k, m\rangle$, we do nothing and let $A(\alpha+1) = A(\alpha)$ and $\overline{A}(\alpha+1) = \overline{A}(\alpha)$. Go to the next stage.

At stage α, $\alpha = \langle 4k, m\rangle$, we will try to satisfy $R'_{k,i}$ for the least i such that $R'_{k,i}$ is not yet satisfied. For each $k \geq 0$, we will ensure our diagonalization area for $R'_{k,i}$ to be $D_\alpha = \{x \mid e(\alpha) \leq |x| < 2^{e(\alpha)}\}$, where $\alpha = \langle 4k, m\rangle$ for some m (Figure 1). Thus, the diagonalization areas for $R'_{k,i}$ and $R'_{k',j}$ will be disjoint for $k \neq k'$ or $i \neq j$.

Let $n = \lfloor e(\alpha)/(k+1)\rfloor - 1$ and $j = e(\alpha) - (k+1)(n+1)$. If n is such that either Theorem 4.1 does not hold for $s_k(2^n)$ or $p_i(n) \geq (1/12) \cdot 2^{n/3}$, then let $A(\alpha+1) = A(\alpha)$ and $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup D_\alpha$. Go to the next stage.

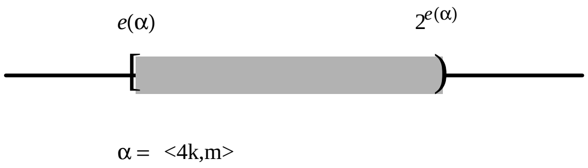


Figure 1: A diagonalization region for $I_k^A \notin \Pi_k^P(A)$

Otherwise, let C_f be the circuit corresponding to the predicate $0^n \in I_k^A$ and let C_i be the circuit corresponding to the negation of the predicate $\sigma_i^k(A;0^n)$. For all the variables v_y in C_i such that (1) if $y \in A(\alpha) \cup \overline{A}(\alpha)$, then replace v_y with 1 if $y \in A(\alpha)$ and with 0 if $y \in \overline{A}(\alpha)$, and (2) if $y \notin A(\alpha) \cup \overline{A}(\alpha)$ and $|y| < e(\alpha)$, then replace v_y with 0. Extend $\overline{A}(\alpha)$ to include strings y's that were assigned 0 in case (2). If the remaining circuit is a constant function, then we are done. It is easy to find an extension A' and $\overline{A'}$ of $A(\alpha)$ and $\overline{A}(\alpha)$ such that $C_f\lceil_{\rho_{A',\overline{A'}}} \neq C_i\lceil_{\rho A',\overline{A'}}$. We simply let $A(\alpha+1) = A'$ and $\overline{A}(\alpha+1) = \overline{A'}$. Cancel i and go to the next stage.

Otherwise, the remaining circuit is still a $\Pi_k(p_i(n))$-circuit having variables of length l, where $e(\alpha) \le l < (1/12)\cdot 2^{n/3} < 2^{e(\alpha)}$. There are two cases to consider. In the case that k is odd, C_f contains a $C_k^{2^n}$ subcircuit computing an $f_k^{2^n}$ function. By Theorem 4.1, there is a restriction $\rho_{B,B'}$ that satisfies the U condition for C_f and that $C_f\lceil_{\rho_{B,B'}} \neq C_i\lceil_{\rho_{B,B'}}$. Let $A(\alpha+1) = A(\alpha) \cup B$ and $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup B'$. Cancel i and go to the next stage.

In the case that k is even, we need to replace all of the negative variables $\bar{x}_j$ in C_f with positive variables y_j. If $\bar{x}_j$ also appears in C_i, replace it there with y_j, and if x_j appears in C_i, replace it there with $\bar{y}_j$. The resulting circuit C_0' contains a $C_k^{2^n}$ subcircuit computing an $f_k^{2^n}$ function. By Theorem 4.1, there is a restriction $\rho_{B,B'}$ that satisfies the U condition for C_0' and such that $C_0'\lceil_{\rho_{B,B'}} \neq C_i\lceil_{\rho_{B,B'}}$. We let $A(\alpha+1) = A(\alpha) \cup \{y \mid y$ is not a replaced variable and $y \in B\} \cup \{x_j \mid y_j(=\bar{x}_j) \in B'\}$ and let $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup \{y \mid y$ is not a replaced variable and $y \in B'\} \cup \{x_j \mid y_j(=\bar{x}_j) \in B\}$. Cancel i and go to the next stage. This completes the description of the stages.

Now let $A = \cup_{\alpha=1}^{\infty} A(\alpha)$. We claim that all the requirements $R_{k,i}$ will eventually be satisfied. To see this, assume that i is the least i such that $R_{k,i}$ is not yet satisfied. Assume that each of $R_{k,i'}$, $i' < i$, is satisfied by stage $\alpha = \langle 4k, m \rangle$, for some m. Then $R_{k,i}$ will be satisfied in the first stage $\alpha' = \langle 4k, m' \rangle$ such that $m' > m$, Theorem 4.1 holds for $s_k(2^n)$, and $p_i(n) < (1/12) \cdot 2^{n/3}$, where $n = \lfloor e(\alpha')/(k+1) \rfloor$.

Theorem 4.2 $\exists A\ \forall k \geq 1\ [\Sigma_k^P(\mathrm{UP}(A)) \not\subseteq \Sigma_k^P(A)]$.

Corollary 4.3 *There exists an oracle set A and a set $U_A \in \mathrm{UP}(A)$ such that for all $k \geq 0$, $U_A \notin L_k^{P,\Sigma}(A)$.*

Corollary 4.4 *There exists an oracle set A and a set $L \in \mathrm{FewP}(A)$ such that for all $k \geq 0$, $L \notin L_k^{P,\Sigma}(A)$.*

An interesting consequence of Corollary 4.3 is that there is an oracle set relative to which UP is not contained in the following classes: APT, R, BPP, P/*poly*.

Corollary 4.5 *There exists an oracle set A relative to which* UP *is not contained in the following classes:* APT, R, BPP, P/*poly*.

Proof. It is known that NP sets that are also in APT, R, BPP, or P/*poly* are in the low hierarchy [KS85, Sch86]. These results can be relativized to all oracles. Since UP $\subseteq$ NP, the corollary follows immediately from Corollary 4.3. $\square$

Corollary 4.6 *There exists an oracle set relative to which* UP *and* FewP *are not low for* BPP.

Proof. From Corollary 4.5, there is an oracle set A such that $\mathrm{UP}^A \not\subseteq \mathrm{BPP}^A$. If UP^A is low for BPP^A, then $\mathrm{UP}^A \subseteq \mathrm{BPP}^A$, which is a contradiction. Thus, there is an oracle set relative to which UP is not low for BPP. The case for FewP is similar. $\square$

Finally, we show that there is an oracle relative to which UP is not low for #P.

Theorem 4.7 *There exists an oracle set A such that* UP(A) *is not low for* #P(A).

Proof. Consider an oracle set A such that $\mathrm{UP}(A) \not\subseteq \mathrm{NP}(A) \cap \text{co-NP}(A)$ [BGS75]. Consequently, $\mathrm{NP}(A)(\mathrm{UP}(A)) \not\subseteq \mathrm{NP}(A)$. Let a language L in $\mathrm{NP}(\mathrm{UP}(A))$ be the witness that $\mathrm{NP}(\mathrm{UP}(A)) \not\subseteq \mathrm{NP}(A)$. Then L is accepted by a nondeterministic polynomial-time oracle machine M_L that uses an oracle set X in $\mathrm{UP}(A)$. Since $L \notin \mathrm{NP}(A)$, then for all the nondeterministic polynomial time oracle machines M_i using A as the oracle, there is a string $x_i \in \Sigma^*$ such that $x_i \in L \Leftrightarrow x_i \notin L(M_i, A)$. Let f_L be the function whose value on any x is the number of accepting paths of M_L using X as the oracle and let f_i be the function whose value on any x is the number of accepting paths of M_i using A as the oracle. Clearly, $f_L \in \#\mathrm{P}(\mathrm{UP}(A))$ and $f_i \in \#\mathrm{P}(A)$. We have

$$f_L(x_i) > 0 \Leftrightarrow x_i \in L \Leftrightarrow x_i \notin L(M_i, A) \Leftrightarrow f_i(x_i) = 0.$$

Thus, $f_L \notin \#\mathrm{P}(A)$. $\square$

A direct corollary from Theorem 4.7 is that there is an oracle relative to which SPP is not low for #P.

Corollary 4.8 *There exists an oracle set A such that* $\mathrm{SPP}(A)$ *is not low for* $\#\mathrm{P}(A)$.

Corollary 4.8 implies an interesting observation by Beigel [Bei92b]. It is known that SPP is low for GapP, which is the function class obtained by closing #P under subtraction.[3] The fact that SPP might not be low for #P demonstrates a distinction between GapP and #P. The oracle A in Corollary 4.8 also shows that #P is not closed under subtraction relative to A.

[3] More formally, a function f is in GapP if and only if there are two #P functions g_1 and g_2 such that $f = g_1 - g_2$.

5 UP is not in the High Hierarchy

In this section we construct an oracle set A relative to which, for all $k \geq 1$, $\Sigma_k^P(A) \not\subseteq \Sigma_{k-1}^P(\mathrm{UP}(A))$. Our diagonalization enumerates all sets in $\mathrm{UP}(A)$. As a result, there is an oracle set A relative to which UP is completely disjoint from every level of the high hierarchy. Our proof is based on an interesting observation from Hartmanis and Hemachandra [HH91]: if a nondeterministic polynomial time oracle machine N is a *robustly categorical* machine (machines that, for every oracle, have at most one accepting computation for each input), then N behaves like a P oracle machine with SAT added to the oracle set. We will make use of this property to separate $\Sigma_k^P(A)$ from $\Sigma_{k-1}^P(\mathrm{UP}(A))$, for all $k \geq 1$.

For arbitrary set A, define sets $\tilde{I}_k^A$ for each $k \geq 1$ as follows:

$$\tilde{I}_k^A = \{1^n \mid (\exists m)(\exists j, 0 \leq j < k+1)[(k+1)n + j = e(\langle 4k{+}1, m\rangle) \text{ and } (\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(Q_k y_k, |y_k| = n)1^n y_1 y_2 \ldots y_k 1^j \in A]\}$$

where Q_k is $\exists$ if k is odd and Q_k is $\forall$ if k is even. Clearly $\tilde{I}_k^A \in \Sigma_k^P(A)$. Also, for every string $x = 1^n$, the membership of x in $\tilde{I}_k^A$ only depends on the membership in A of strings with prefix 1 and of length $\langle 4k{+}1, m\rangle$ for some k and m. Our goal is to construct an oracle set A such that $\tilde{I}_k^A \notin \Sigma_{k-1}^P(\mathrm{UP}(A))$, for all $k \geq 1$, implying that no $\mathrm{UP}(A)$ languages are in any level of the high hierarchy. The construction of A will depend on two crucial technical lemmas. The first one is a modified version of Yao and Håstad's separation of the Σ-levels of PH [Yao85, Has87] and it shows that there is an oracle A such that $\tilde{I}_k^A \notin \Sigma_{k-1}^P(A \oplus \mathrm{SAT})$, for all $k \geq 1$. The proof follows trivially from the proofs in [Yao85, Has87] and is omitted.

Lemma 5.1 *For arbitrary but fixed $k \geq 1$, let σ be a $\Sigma_{k-1}^{P,1}$-predicate with polynomial bound p. There is a constant n_{k-1} such that if $n \geq n_{k-1}$ and $p(n) < \delta \cdot 2^{n/5}$, where $\delta = 1/48$, then there exists an oracle set A such that $1^n \in \tilde{I}_k^A \Leftrightarrow \sigma(A \oplus \mathrm{SAT}; 1^n)$ is false.*

The second technical lemma is based on ideas from Hartmanis and Hemachandra [HH91].

Lemma 5.2 *Let NM be an arbitrary nondeterministic polynomial time oracle machine whose running time is bounded by $p(|n|)$ on each input of length n. Let $U = \{x \mid \forall B\ [NM^B(x)$ has at most 1 accepting path $]\}$. Then, there exists a deterministic polynomial time oracle machine M, depending only on NM, such that for all B and for all $x \in U$, $NM^B(x)$ accepts if and only if $M^{\mathrm{SAT}\oplus B}(x)$ accepts. Furthermore, there exists a polynomial q, depending on p, such that the running time of $M^{\mathrm{SAT}\oplus B}$ is bounded by $q(|x|)$.*

Proof. First note that $\overline{U}$ is in NP and thus U is in co-NP. We will give a deterministic polynomial-time oracle machine M, depending only on NM, such that for all oracle sets B and all $x \in U$, $M^{\mathrm{SAT}\oplus B}$ accepts if only if if $NM^B(x)$ accepts.

Define L as follows:

> $L = \{x\#c(Y)\#c(N) \mid$ there is an accepting computation of NM on x such that for each query string w in the accepting path: (a) if $w \in Y \cup N$ then w is answered YES if and only if $w \in Y$; (b) if w is not in $Y \cup N$, w can be answered either way. $\}$

Clearly, $L \in$ NP. Following is the oracle machine M:

Algorithm M

if $x \notin U$ **then**
  reject
else
  Let $Y = N = \emptyset$, and let $j = 0$.
  repeat
    if $x\#c(Y)\#c(N) \notin L$ **then**
      reject
    else
      Let S_j be the set of query strings in the unique accepting computation of NM on x witnessed by $x\#c(Y)\#c(N) \in L$;

Using B, check if the query answers to strings in S_j are correct;

if all the answers are correct **then**

accept

else

$Y = Y \cup \{y \in S_j \mid y \in B\}$, $N = N \cup \{y \in S_j \mid y \notin B\}$, and $j = j+1$.

endif

endif

until accept **or** reject

endif

Claim 1: Algorithm **M** terminates in polynomial time.

Proof of Claim 1. If the input string x is not in U, then **M** terminates after one query to $\overline{U} \in$ NP. Otherwise, let input string x be in U and let $S_0, S_1, \ldots, S_k$ be the query sets from the algorithm **M** and let W_0, $W_1, \ldots$, W_k be the queries with wrong answers in $S_0, S_1, \ldots, S_k$, respectively. Since for all possible B and for all $x \in U$, $NM^B(x)$ has no more than one accepting path, there must be at least one string $w \in S_j \cap S_k$, for all $j < k$, and w is answered differently in S_j and S_k. Otherwise, $NM^B(x)$ has more than one accepting path. Note that all of the incorrectly-answered strings in S_j, $j < k$, are corrected in S_i, for all $i > j$. Thus, $W_j \cap W_i = \emptyset$ for all $i \neq j$. Also, for all $j < k$, the string $w \in S_j \cap S_k$ that answered differently must be answered incorrectly in S_j and answered correctly in S_k; that is, $w \in W_j \cap S_k - W_k$, for all $j < k$. Thus, we have $\| S_k \| \geq k$.

Recall that p is the polynomial that bounds the running time of NM on any input x. There are at most $p(|x|)$ strings in each S_j, $0 \leq j \leq k$. Thus, $k \leq p(|x|)$ and algorithm **M** runs in $q(|x|) = p'(p(|x|))$ steps for some fixed polynomial p'.

Claim 2: For all sets B and for all $x \in U$, $MN^B(x)$ accepts if and only if algorithm $\mathbf{M}^{\mathrm{SAT} \oplus B}$ accepts x.

Proof of Claim 2. Let S_k be the last query set obtained in the loop. If all the queries in S_k are answered correctly, then we have found an accepting path of $NM^B(x)$ and algorithm $\mathrm{M}^{\mathrm{SAT}\oplus B}$ accepts x. If some answers in S_k are not correct, then the else clause extends Y and N relative to B. Since S_k is the last constructed query set, $x\#c(Y)\#c(N)$ is not in L, $NM^B(x)$ does not accept, and $\mathrm{M}^{\mathrm{SAT}\oplus B}$ correctly rejects x. This completes the proof of Lemma 5.2. $\square$

Now we are ready to construct the oracle set A. The construction is by stages. To diagonalize out of $\Sigma_{k-1}^P(\mathrm{UP}(A))$, we enumerate $\Sigma_{k-1}^P(\mathrm{NP}(A))$. For machine $NM_i(A)$, in the enumeration of oracle machines from $NP(A)$, not acting like a $\mathrm{UP}(A)$ machine, we can skip $NM_i(A)$ without diagonalization. For machine $NM_i(A)$ acting like a $\mathrm{UP}(A)$ machine, NM_i is "categorical" and is replaced by $\mathrm{P}^{\mathrm{SAT}\oplus A}$ to reduce the diagonalization to the case of $\Sigma_k^P(A) \not\subseteq \Sigma_{k-1}^P(A \oplus \mathrm{SAT})$. More formally, let $M_1, M_2, \ldots$ be an effective enumeration of deterministic polynomial time oracle machines and let $N_1, N_2, \ldots$ be an effective enumeration of nondeterministic polynomial time oracle machines. Let polynomial $p_i(n) = n^i + i$ bound the running time of M_i and N_i on any input string of length n. For arbitrary but fixed $k \geq 1$, and for all $l \geq 1$, define the predicate $\tau_l^k(A; x)$ as follows

$$\tau_l^k(A;x) \equiv (\exists y_1, |y_1| \leq p_i(|x|))(\forall y_2, |y_2| \leq p_i(|x|)) \cdots$$
$$(Q_{k-1} y_{k-1}, |y_{k-1}| \leq p_i(|x|))\ [M_i^{N_j^A}(xy_1y_2 \ldots y_{k-1}) \text{ accepts}]$$

where Q_{k-1} is $\exists$ if k is even and Q_{k-1} is $\forall$ if k is odd, and $l = \langle i, j\rangle$. Note that $\tau_1^k, \tau_2^k \cdots$ is an effective enumeration of $\Sigma_{k-1}^P(\mathrm{NP}(A))$.

Our goal of $\Sigma_k^P(A) \not\subseteq \Sigma_{k-1}^P(\mathrm{UP}(A))$, for all $k \geq 1$, is divided into an infinite number of requirements:

$\tilde{R}_{k,l}$: (1) $l = \langle i, j\rangle$ and there exist an n and A such that, there is a string of the format $1^n y_1 y_2 \ldots y_{k-1}$, where $|y_j| \leq p_i(n)$ for all j, $1 \leq j \leq k-1$, and M_i on input $1^n y_1 y_2 \ldots yk - 1$ makes a query "$y \in L(N_j, A)$" and $N_j^A(y)$ has more than one accepting computation, or

(2) $l = \langle i, j\rangle$ and there exist an n and A such that $1^n \in \tilde{I}_k^A \Leftrightarrow \tau_l^k(A; 1^n)$ is false.

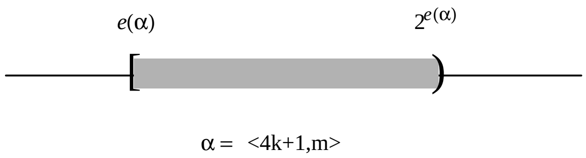


Figure 2: A diagonalization region for $\tilde{I}_k^A \notin \Sigma_{k-1}^P(\mathrm{UP}(A))$

Initially, let $A(1) = \overline{A}(1) = \emptyset$. In each stage α, we will define sets $A(\alpha+1)$ and $\overline{A}(\alpha+1)$ such that $A(\alpha+1) \cap \overline{A}(\alpha+1) = \emptyset$ and such that $A(\alpha+1)$ is an extension of $A(\alpha)$ and $\overline{A}(\alpha+1)$ is an extension of $\overline{A}(\alpha)$. For each $k \geq 0$, we will ensure that the diagonalization area for the requirement $\tilde{R}_{k,l}$ is $D_\alpha = \{x \mid e(\alpha) \leq |x| < 2^{e(\alpha)}\}$, where $\alpha = \langle 4k+1, m\rangle$ for some m. Thus, the diagonalization areas for $\tilde{R}_{k,l}$ and $\tilde{R}_{k',j}$ will be disjoint for $k \neq k'$ or $l \neq j$.

At stage α, $\alpha \neq \langle 4k+1, m\rangle$, we do nothing and let $A(\alpha+1) = A(\alpha)$ and $\overline{A}(\alpha+1) = \overline{A}(\alpha)$. Go to the next stage.

At stage α, $\alpha = \langle 4k+1, m\rangle$, we will try to satisfy the requirement $\tilde{R}_{k,l}$, where $l = \langle i, j\rangle$ is the least l for which $\tilde{R}_{k,l}$ is not yet satisfied. Let $n = \lfloor e(\alpha)/(k+1)\rfloor$ and let $p_{M_i}(n) = p_i(n + (k-1) \cdot p_i(n))$ be the running time of M_i on inputs of the form $xy_1y_2\cdots y_{k-1}$, where $|x| = n$ and $|y_r| \leq p_i(n)$ for all $1 \leq r < k$. Let $p_l(n) = p_{M_i}(n) \cdot q(p_{M_i}(n))$, where q is the polynomial from Lemma 5.2. Note that $p_j(p_{M_i}(n))$, the maximum length of strings queried, is less than $p_l(n)$ by the definition of q in Lemma 5.2. Thus, whether $\tau_l^k(A; 1^n)$ is true depends only on the membership of strings in A of length $\leq p_l(n)$. There are three cases.

Case 1: If $p_l(n) \geq (1/48) \cdot 2^{n/5}$ or $n < n_{k-1}$, where n_{k-1} is the constant from Lemma 5.1, then let $A(\alpha+1) = A(\alpha)$ and $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup D_\alpha$. Go to the next stage.

Case 2: Otherwise, try to find a valid extension A' of $A(\alpha)$ and $\overline{A'}$ of $\overline{A}(\alpha)$ such that, in the computation of $\tau_l^k(A'; 1^n)$, $l = \langle i, j\rangle$, M_i makes a query "$y \in L(N_j, A')$?" and $N_j^{A'}(y)$ has more than one accepting computation. If there are such A' and $\overline{A'}$, then let $A(\alpha+1) = A'$ and $\overline{A}(\alpha+1) = \overline{A'}$. Since $L(N_j, A')$ is not in $\mathrm{UP}(A')$, we can cancel l and go to the next stage.

Case 3: If for all valid extensions A' of $A(\alpha)$ and $\overline{A'}$ of $\overline{A}(\alpha)$, M_i, in the computation of $\tau_l^k(A';1^n)$, does not make a query "$y \in L(N_j, A')$?" where $N_j^{A'}(y)$ has more than one accepting computation, then by Lemma 5.2 there is a deterministic polynomial time algorithm, with oracle SAT $\oplus A'$, that computes whether $w \in L(N_j, A')$ for every query w made by M_i. Thus, there is a deterministic oracle machine M^* with oracle SAT $\oplus A'$ such that on inputs $z = 1^n y_1 y_2 \ldots y_{k-1}$, $|y_r| \leq p_i(n)$ for all r, $1 \leq r < k$, $M^{*(\mathrm{SAT} \oplus A')}$ accepts z if and only if $M_i^{N_j^{A'}}$ accepts z. Note that the running time of $M^{*(\mathrm{SAT} \oplus A')}$ on input z is bounded by $p_{M_i}(n) \cdot q(p_{M_i}(n)) = p_l(n)$, since there are at most $p_{M_i}(n)$ queries, each of length at most $p_{M_i}(n)$. Also note that $\tau_l^k(A';1^n)$ can be computed by $\Sigma_{k-1}^P(\mathrm{SAT} \oplus A')$ for any valid extension A' of $A(\alpha)$. Let C_l be the circuit corresponding to the predicate $\tau_l^k(A';1^n)$. C_l is a $\Sigma_{k-1}(p_l(n))$-circuit. For all the variables v_y in C_l: (1) if $y \in A(\alpha) \cup \overline{A}(\alpha)$, then replace v_y with 1 if $y \in A(\alpha)$ and with 0 if $y \in \overline{A}(\alpha)$, and (2) if $y \notin A(\alpha) \cup \overline{A}(\alpha)$ and $|y| < e(\alpha)$, then replace v_y with 0. Extend $\overline{A}(\alpha)$ to include strings y's that were assigned 0 in case (2). If the remaining circuit is a constant function, then we are done. It is easy to find an extension A' and $\overline{A'}$ of $A(\alpha)$ and $\overline{A}(\alpha)$ such that $1^n \in \tilde{I}_k^{A'} \Leftrightarrow C_l\lceil_{\rho_{A',\overline{A'}}}$ outputs 0. We simply let $A(\alpha+1) = A'$ and $\overline{A}(\alpha+1) = \overline{A'}$. Cancel l and go to the next stage.

Otherwise, the remaining circuit is still a $\Sigma_{k-1}(p_l(n))$-circuit having variables of length s, where $e(\alpha) \leq s < (1/48) \cdot 2^{n/5} < 2^{e(\alpha)}$. By Lemma 5.1, there is a valid extension A' and $\overline{A'}$ of $A(\alpha)$ and $\overline{A}(\alpha)$ such that $1^n \in \tilde{I}_k^{A'} \Leftrightarrow C_l\lceil_{\rho_{A,\overline{A'}}}$ outputs 0. Let $A(\alpha+1) = A'$ and $\overline{A}(\alpha+1) = \overline{A'}$. Cancel l and go to the next stage.

Let $A = \cup_{\alpha=1}^{\infty} A(\alpha)$. We claim that all the requirements $\tilde{R}_{k,l}$ will eventually be satisfied. Assume that $l = \langle i,j \rangle$ is the least l such that $\tilde{R}_{k,l}$ is not yet satisfied. Assume that each of $\tilde{R}_{k,l'}$, $l' < l$, is satisfied by stage $\alpha = \langle 4k+1, m \rangle$. Then $\tilde{R}_{k,l}$ will be satisfied in the first stage $\alpha' = \langle 4k+1, m' \rangle$ such that $m' > m$, $p_l(n) < 2^{e(\alpha')}$, $n \geq n_{k-1}$, and $p_{M_i}(n) \cdot q(p_{M_i}(n)) < (1/48) \cdot 2^{n/5}$, where $n = \lfloor e(\alpha')/(k+1) \rfloor$.

Theorem 5.3 $\exists A\ \forall k \geq 1\ [\Sigma_k^P(A) \not\subseteq \Sigma_{k-1}^P(\mathrm{UP}(A))]$.

Corollary 5.4 $\exists A\ \forall k \geq 0\ [\mathrm{UP}(A) \cap H_k^{P,\Sigma}(A) = \emptyset]$.

Corollary 5.5 $\exists A\ \exists U_A \in \mathrm{UP}(A)\ \forall k \geq 0\ [U_A \notin L_k^{P,\Sigma}(A)$ *and* $\mathrm{UP}(A) \cap H_k^{P,\Sigma}(A)]$.

Proof. The only thing we need to take care of in combining the diagonalizations in Theorem 4.2 and in Theorem 5.3 is to make sure that $U_A \in \mathrm{UP}(A)$. In case 2, strings in diagonalization region $D_\alpha = \{x \mid e(\alpha) \leq |x| < 2^{e(\alpha)}, \alpha = \langle 4k+1, m\rangle$ for some $k, m\}$ may be assigned arbitrarily to force an NP machine to be non-categorical. In Case 3, strings in the same type of diagonalization region may be assigned arbitrarily to diagonalize against a circuit. In both cases, assignments to strings in D_α have no effect on U_A since $U_A \subseteq \{x||x| = e(\alpha), \alpha = \langle 4k, m\rangle$ for some $k, m\}$. $\square$

6 Relativized Low and High Hierarchies with Infinitely Many Levels

Corollary 5.5 shows that there exists an oracle set A such that UP(A) contains a set that is not in any level of the low hierarchy relative to A and such that UP(A) contains no languages in any level of the high hierarchy relative to A. However, it is not clear whether this holds for an oracle relative to which the low and high hierarchies have an infinite number of distinct levels. Note that it is not known that if the polynomial-time hierarchy has an infinite number of distinct levels, then the low and high hierarchies also have an infinite number of distinct levels.

In [Ko91], Ko constructed an oracle set relative to which the low and high hierarchies have an infinite number of distinct levels. In this section, we combine Ko's construction with the constructions in Theorem 4.2 and Theorem 5.3 and show that there exists an oracle set relative to which UP contains a language outside the low hierarchy and contains no languages in the high hierarchy and, at the same time, the low and high hierarchies have an infinite number of distinct levels.

Our goal is to construct an oracle set A such that there is a set $U_A \in \mathrm{UP}(A)$ and such that for all $k \geq 0$, there are sets $B_k(A)$ and $\tilde{B}_k(A)$ in NP(A) and that

(a) $\Sigma_k^P(U_A) \not\subseteq \Sigma_k^P(A)$,

(b) $\Sigma_{k+1}^P(A) \not\subseteq \Sigma_k^P(\mathrm{UP}(A))$,

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

(d) $\Sigma_k^P(B_k(A) \oplus A) \not\subseteq \Sigma_k^P(A)$,

(e) $\Sigma_{k+2}^P(A) = \Sigma_{k+1}^P(\tilde{B}_k(A) \oplus A)$, and

(f) $\Sigma_{k+1}^P(A) \not\subseteq \Sigma_k^P(\tilde{B}_k(A) \oplus A)$.

Condition (a) ensures that UP(A) contains a set that is not in the relativized low hierarchy. Condition (b) ensures that UP(A) and the relativized high hierarchy are disjoint. Conditions (c) and (d) ensure that the relativized low hierarchy has an infinite number of distinct levels, and conditions (e) and (f) ensure that the relativized high hierarchy has an infinite number of distinct levels. To satisfy condition (a), we will use the set $I_k^A \in \Sigma_{k-1}^P(U_A)$ from Theorem 4.2 and make sure that $I_k^A \in \Sigma_{k-1}^P(U_A)$ and that $I_k^A \notin \Pi_k^P(A)$, which implies that $\Sigma_k^P(U_A) \not\subseteq \Sigma_k^P(A)$. To satisfy condition (b), we will use the set $\tilde{I}_{k+1}^A \in \Sigma_{k+1}^P(A)$ from Theorem 5.3 and make sure that $\tilde{I}_{k+1}^A \notin \Sigma_k^P(\mathrm{UP}(A))$. To satisfy conditions $(c)-(f)$, we will use the following sets from Ko [Ko91]. For a fixed $k \geq 0$, define

$$B_k(A) = \{x \mid |x| = e(\langle 4k+2, n\rangle) \text{ for some } n \text{ and } (\exists y, |x| = |y|)[0xy \in A]\},$$
$$\tilde{B}_k(A) = \{x \mid 2^{e(\langle 4k+3,n\rangle)} \leq |x| < e(\langle 4k+3, n+1\rangle) \text{ for some } n \text{ and } x \in K(A)]\}.$$

For all $k \geq 0$, when k is even define

$$J_k^A = \{0^n \mid (\exists m)(\exists j, 0 \leq j < k)[kn+j = e(\langle 4k+2, m\rangle)] \text{ and }$$
$$(\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(\forall y_k, |y_k| = n) y_1 y_2 \ldots y_k 0^j \in B_k(A)\},$$

and when k is odd, define

$$J_k^A = \{0^n \mid (\exists m)(\exists j, 0 \leq j < k)[kn+j = e(\langle 4k+2, m\rangle)] \text{ and }$$
$$(\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(\exists y_k, |y_k| = n) y_1 y_2 \ldots y_k 0^j \notin B_k(A)\}.$$

Note that whether $0^n \in J_k^A$ depends only on membership of strings in A of length $2 \cdot e(\langle 4k+2, m\rangle) + 1$. For any $k \geq 0$, define

$$E_k^A = \{0^n \mid (\exists y_1, |y_1| = n)(\forall y_2, |y_2| = n)\cdots(Q_{k+1} y_{k+1}, |y_{k+1}| = n)$$
$$0 y_1 y_2 \ldots y_{k+1} \in A\},$$

where Q_{k+1} is $\forall$ if k is odd and Q_{k+1} is $\exists$ if k is even. It is clear that $B_k(A)$ and $\tilde{B}_k(A)$ are in $\mathrm{NP}(A)$, J_k^A is in $\Sigma_k^P(B_k(A))$, and E_k^A is in $\Sigma_{k+1}^P(A)$. The set J_k^A is used to ensure that $B_k(A)$ is not in $L_k^{P,\Sigma}(A)$ and the set E_k^A is used to ensure that $\tilde{B}_k(A)$ is not in $H_k^{P,\Sigma}(A)$. We rewrite our goal, for each $k \geq 0$, as the following

(a') $U_A \in \mathrm{UP}(A)$ and $I_k^A \notin \Pi_k^P(A)$,

(b') $\tilde{I}_{k+1}^A \notin \Sigma_k^P(\mathrm{UP}(A))$,

(c') $\Sigma_{k+1}^P(B_k(A) \oplus A) = \Sigma_{k+1}^P(A)$,

(d') $J_k^A \notin \Sigma_k^P(A)$,

(e') $\Sigma_{k+2}^P(A) = \Sigma_{k+1}^P(\tilde{B}_k(A) \oplus A)$, and

(f') $E_k^A \notin \Sigma_k^P(\tilde{B}_k(A) \oplus A)$.

Conditions (a') and (b') ensure that $\Sigma_{k+1}^P(A) \not\subseteq \Sigma_k^P(\mathrm{UP}(A)) \not\subseteq \Sigma_k^P(A)$. Conditions ($c'$) and ($d'$) ensure that $L_k^{P,\Sigma}(A) \subsetneq L_{k+1}^{P,\Sigma}(A)$ and condtions (e') and (f') ensure that $H_k^{P,\Sigma}(A) \subsetneq H_{k+1}^{P,\Sigma}(A)$.

The construction of A is by stages. For each $k \geq 0$, there are four major stages for our conditions. Stages $\alpha = \langle 4k, m\rangle$, for all $m \geq 0$, are designated to satisfy condition (a'); that is, to make sure U_A is not in $L_k^{P,\Sigma}$. Stages $\alpha = \langle 4k+1, m\rangle$, for all $m \geq 0$, are designated to satisfy condition (b'); that is, to make sure $\mathrm{UP}(A)$ and $H_k^{P,\Sigma}(A)$ are disjoint. Stages $\alpha = \langle 4k+2, m\rangle$, for all $m \geq 0$, are designated to satisfy conditions (c') and (d'); that is, to make sure that $B_k(A) \in L_{k+1}^{P,\Sigma}(A) - L_k^{P,\Sigma}(A)$. Finally, stages $\alpha = \langle 4k+3, m\rangle$, for all $m \geq 0$, are designated to satisfy conditions (e') and (f'); that is, to make sure that $\tilde{B}_k(A) \in H_{k+1}^{P,\Sigma}(A) - H_k^{P,\Sigma}(A)$. At the end of the construction, we will ensure that all the conditions will be satisfied. Note that to satisfy conditions (c') and (e'), we need to "Σ_{k+1}^P-encode" the complete set for $\Sigma_{k+1}^P(B_k(A) \oplus A)$ into A and the complete set for $\Sigma_{k+2}^P(A)$ into $\tilde{B}_k(A) \oplus A$. There are potential conflicts between these encoding requirements and diagonalization requirements. However, by the definitions of sets $B_k(A)$ and $\tilde{B}_k(A)$, we only need to encode "small" intervals of strings into A. This can be seen by the following facts from Ko [Ko91] (Fact 3.2 and Fact 4.2).

Fact 1: For arbitrary set A and a fixed k, $k \geq 0$, there exists a set $D_k(A) \in \Sigma_{k+1}^P(A)$ such that if $2^{e(\langle 4k+2,m\rangle)} \leq |x| < e(\langle 4k+2, m+1\rangle)$ for some m, then $x \in K^{k+1}(B_k(A) \oplus A) \Leftrightarrow x \in D_k(A)$.

Fact 2: For arbitrary set A and a fixed k, $k \geq 0$, there exists a set $\tilde{D}_k(A) \in \Sigma_{k+1}^P(\tilde{B}_k(A) \oplus A)$ such that if $2^{e(\langle 4k+3,m\rangle)} \leq |x| < e(\langle 4k+3, m+1\rangle)$ for some m, then $x \in K^{k+1}(K(A)) \Leftrightarrow x \in \tilde{D}_k(A)$.

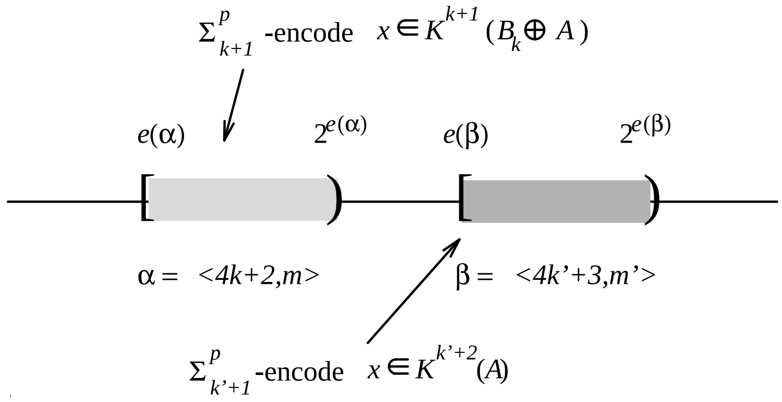


Figure 3: Pairwise Disjoint Encoding Areas

For each $\alpha \geq 0$, define the interval $D_\alpha = \{x \mid e(\alpha) \leq |x| < 2^{e(\alpha)}\}$. Fact 1 says that for some x, if x is not in the interval D_α, where $\alpha = \langle 4k+2, m\rangle$ for some m, then deciding $x \in K^{k+1}(B_k(A) \oplus A)$ can be simulated by a predicate in $\Sigma^P_{k+1}(A)$. Similarly, Fact 2 says that for some x if x is not in the interval D_α, where $\alpha = \langle 4k+3, m\rangle$ for some m, then deciding $x \in K^{k+1}(K(A)) = K^{k+2}(A)$ can be simulated by a predicate in $\Sigma^P_{k+1}(\tilde{B}^A_k \oplus A)$. Thus, for a fixed k, we only need to "Σ^P_{k+1}-encode" strings in D_α, where $\alpha = \langle 4k+2, m\rangle$ for all m, to satisfy condition (c') and only need to "Σ^P_{k+1}-encode" strings in D_α, where $\alpha = \langle 4k+3, m\rangle$ for all m, to satisfy condition (e'). Also note that the encoding areas D_α are pairwise disjoint where $\alpha = \langle 4k+2, m\rangle$ or $\alpha = \langle 4k+3, m\rangle$ for all k and for all m. See Figure 3.

Recall from Section 4 that for all $k \geq 0$, $\sigma^k_1, \sigma^k_2, \ldots$ is an effective enumeration of $\Sigma^{P,1}_k$-predicates and, from Section 5, that $\tau^k_1, \tau^k_2, \ldots$ is an effective enumeration of predicates in $\Sigma^P_{k-1}(\mathrm{NP}(A))$. We divide condition (a') into an infinite number of requirements $R_{1,k,i}$, $i \geq 0$,

$$R_{1,k,i} : (\exists n)[0^n \in I^A_k \Leftrightarrow \sigma^k_i(A; 0^n) \text{ is true and } U_A \in \mathrm{UP}(A)].$$

We divide condition (b') into an infinite number of requirements $R_{2,k,l}$, $l \geq 0$,

> $R_{2,k,l}$: (1) $l = \langle i, j\rangle$ and there exist an n and A such that, M_i, in the computation of $\tau^k_l(A; 1^n)$, makes a query "$y \in L(N_j, A)$" and $N^A_j(y)$ has more than one accepting computation, or

(2) $l = \langle i, j \rangle$ and there exist an n and A such that $1^n \in \tilde{I}_k^A \Leftrightarrow \tau_l^k(A; 1^n)$ is false.

We divide condition (c') into an infinite number of requirements $R_{3,k,t}$, $t \geq 0$.

$R_{3,k,t}$: If $e(\langle 4k+2, m\rangle) \leq t < 2^{e(\langle 4k+2,m\rangle)}$ for some m, we let

$$R_{3,k,t} : (\forall x, |x| = t)[x \in K^{k+1}(B_k(A) \oplus A) \Leftrightarrow \pi_k(A; x)],$$

where $\pi_k(A; x)$ is the following $\Sigma_{k+1}^{P,1}$-predicate

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

If $2^{e(\langle 4k+2,m\rangle)} \leq t < e(\langle 4k+2, m+1\rangle)$ for some m, we let

$$R_{3,k,t} : (\forall x, |x| = t)[x \in K^{k+1}(B_k(A) \oplus A) \Leftrightarrow x \in D_k(A)],$$

where $D_k(A)$ is from Fact 1.

Note that for all t, where $2^{e(\langle 4k+2,m\rangle)} \leq t < e(\langle 4k+2, m+1\rangle)$, requirement $R_{3,k,t}$ is already satisfied by Fact 1. We divide condition (d') into an infinite number of requirements $R_{4,k,i}$, $i \geq 0$,

$R_{4,k,i} : (\exists n)[0^n \in J_k^A \Leftrightarrow \sigma_i^k(A; 0^n)$ is false$]$.

We divide condition (e') into an infinite number of requirements $R_{5,k,t}$, $t \geq 0$.

$R_{5,k,t}$: If $e(\langle 4k+3, m\rangle) \leq t < 2^{e(\langle 4k+3,m\rangle)}$ for some m, we let

$$R_{5,k,t} : (\forall x, |x| = t)[x \in K^{k+1}(K(A)) \Leftrightarrow \pi_k(A; x)],$$

where $\pi_k(A; x)$ is the $\Sigma_{k+1}^{P,1}$-predicate defined in the requirement $R_{3,k,t}$:

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

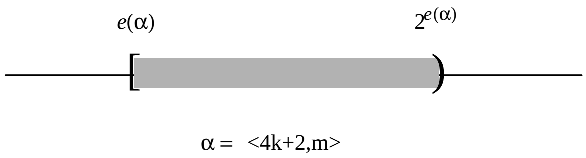


Figure 4: A diagonalization and encoding area for $B_k(A) \in L_{k+1}^{P,\Sigma}(A) - L_k^{P,\Sigma}(A)$

$$10^k 1 x u_1 u_2 \cdots u_{k+1} \in A.$$

If $2^{e(\langle 4k+3,m\rangle)} \le t < e(\langle 4k+3, m+1\rangle)$ for some m, we let

$$R_{5,k,t} : (\forall x, |x| = t)[x \in K^{k+1}(K(A)) \Leftrightarrow \tilde{D}_k(A)],$$

where $\tilde{D}_k(A)$ is from Fact 2.

Note that for all t, where $2^{e(\langle 4k+3,m\rangle)} \le t < e(\langle 4k+3, m+1\rangle)$, requirement $R_{5,k,t}$ is already satisfied by Fact 2. Finally, we divide condition (f') into an infinite number of requirements $R_{6,k,i}$, $i \ge 0$,

$$R_{6,k,i} : (\exists n)[0^n \in E_k^A \Leftrightarrow \sigma_i^k(\tilde{B}_k(A) \oplus A; 0^n) \text{ is false}].$$

The construction of A is by stages. At stages $\alpha = \langle 4k, m\rangle$, for all $m \ge 0$, we will satisfy requirements $R_{1,k,i}$ for all $i \ge 1$. At stages $\alpha = \langle 4k+1, m\rangle$, for all $m \ge 0$, we will satisfy requirements $R_{2,k,l}$ for all $l \ge 1$. At stages $\alpha = \langle 4k+2, m\rangle$, for all $m \ge 0$, we will satisfy requirements $R_{3,k,t}$ for all $t \ge 0$ and requirements $R_{4,k,i}$ for all $i \ge 1$. Finally, at stages $\alpha = \langle 4k+3, m\rangle$, for all $m \ge 0$, we will satisfy requirements $R_{5,k,t}$ for all $t \ge 0$ and requirements $R_{6,k,i}$ for all $i \ge 1$.

Initially, let $A(1) = \overline{A}(1) = \emptyset$. In each stage α, we will define sets $A(\alpha+1)$ and $\overline{A}(\alpha+1)$ such that $A(\alpha+1) \cap \overline{A}(\alpha+1) = \emptyset$ and that $A(\alpha+1)$ is an extension of $A(\alpha)$ and $\overline{A}(\alpha+1)$ is an extension of $\overline{A}(\alpha)$. The diagonalization and encoding area for stage α will be D_α.

Stage α, $\alpha = \langle 4k, m\rangle$**:** At this stage, we try to satisfy $R_{1,k,i}$ for the least i for which $R_{1,k,i}$ is not yet satisfied. We use exactly the same construction as in stage $\alpha, \alpha = \langle 4k, m\rangle$, in Theorem 4.2. If we have satisfied the requirement $R_{1,k,i}$, then extend the

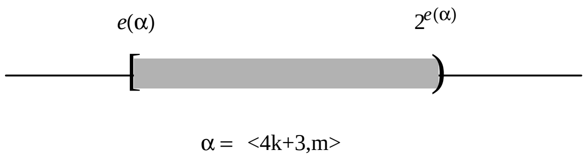


Figure 5: A diagonalization and encoding area for $\tilde{B}_k(A) \in H_{k+1}^{P,\Sigma}(A) - H_k^{P,\Sigma}(A)$

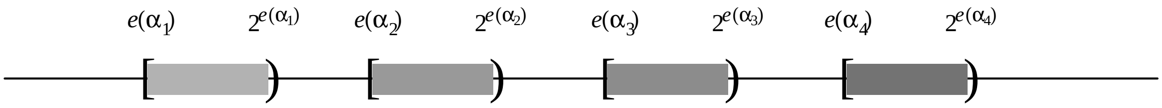
sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed, cancel i, and go to the next stage. Otherwise, let $A(\alpha+1) = A(\alpha)$, $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup D_\alpha$, and go to the next stage.

Stage α, $\alpha = \langle 4k+1, m\rangle$: At this stage, we try to satisfy $R_{2,k,l}$ for the least l for which $R_{2,k,l}$ is not yet satisfied. We use exactly the same construction as in stage $\alpha, \alpha = \langle 4k+1, m\rangle$, in Theorem 5.3. If we have satisfied the requirement $R_{2,k,l}$, then extend the sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed, cancel l, and go to the next stage. Otherwise, let $A(\alpha+1) = A(\alpha)$, $\overline{A}(\alpha+1) = \overline{A}(\alpha) \cup D_\alpha$, and go to the next stage.

Stage α, $\alpha = \langle 4k+2, m\rangle$: At this stage, we will satisfy requirements $R_{3,k,t}$ for all t, where $e(\alpha) \le t < 2^{e(\alpha)}$. At the same time, we try to satisfy $R_{4,k,i}$ for the least i for which $R_{4,k,i}$ is not yet satisfied. We use exactly the same construction as in Ko's proof of Theorem 3.1 in [Ko91]. If we have satisfied the requirement $R_{4,k,i}$, then extend the sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed, cancel i, and go to the next stage. Otherwise, only extend the sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed for requirements $R_{3,k,t}$ for all t, where $e(\alpha) \le t < 2^{e(\alpha)}$. Go to the next stage.

Stage α, $\alpha = \langle 4k+3, m\rangle$: At this stage, we will satisfy requirements $R_{5,k,t}$ for all t, where $e(\alpha) \le t < 2^{e(\alpha)}$. At the same time, we try to satisfy $R_{6,k,i}$ for the least i for which $R_{6,k,i}$ is not yet satisfied. We use exactly the same construction as in Ko's proof of Theorem 4.1 in [Ko91]. If we have satisfied the requirement $R_{6,k,i}$, then extend the sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed, cancel i, and go to the next stage. Otherwise, only extend the sets $A(\alpha)$ and $\overline{A}(\alpha)$ as needed for requirements $R_{5,k,t}$ for all t, where $e(\alpha) \le t < 2^{e(\alpha)}$. Go to the next stage.

We now carefully analyze the potential conflicts between different stages. The diagonalization area for stage α is D_α. Thus, the diagonalization areas for all the requirements $R_{1,k,i}$, $R_{2,k',l}$, $R_{4,j,s}$, and $R_{6,j',u}$ are pairwise disjoint. The requirements $R_{3,k,t}$ $R_{5,k',r}$ can



$\alpha_1 = \langle 4k, m\rangle$ diagonalization region for $I_k^A \notin \Sigma_k^p(A)$

$\alpha_2 = \langle 4k+1, m\rangle$ diagonalization region for $\tilde{I}_k^A \notin \Sigma_{k-1}^p(\mathrm{UP}(A))$

$\alpha_3 = \langle 4k+2, m\rangle$ diagonalization and encoding region for $B_k(A) \in L_{k+1}^{P,\Sigma}(A) - L_k^{P,\Sigma}(A)$

$\alpha_4 = \langle 4k+3, m\rangle$ diagonalization and encoding region for $\tilde{B}_k(A) \in H_{k+1}^{P,\Sigma}(A) - H_k^{P,\Sigma}(A)$

Figure 6: Summary of requirements

be satisfied without interference with each other is guaranteed by Fact 1 and Fact 2. It is also clear that the diagonalization at stages α, where $\alpha = \langle 4k, m\rangle$ or $\alpha = \langle 4k+1, m\rangle$, will not interfere with the encoding at these stages. Thus, there is no conflicts between different stages. The only possible conflicts occur between encoding and diagonalization at stages $\alpha = \langle 4k+2, m\rangle$ or $\alpha = \langle 4k+3, m\rangle$. See Figures 4 and 5. Ko's ingenious proof techniques avoid all potential conflicts and simultaneously satisfy requirements $R_{3,k,t}$ and $R_{4,k,i}$ at stages α, where $\alpha = \langle 4k+2, m\rangle$ for all m, and requirements $R_{5,k,t}$ and $R_{6,k,i}$ at stages α, where $\alpha = \langle 4k+3, m\rangle$ for all m. This completes our construction of A. Figure 6 is a summary of all the requirements.

Let $A = \bigcup_{\alpha=0}^{\infty} A(\alpha)$. It is easy to see that eventually all of the requirements are satisfied.

Theorem 6.1 *There exists an oracle set A and a set $U_A \in \mathrm{UP}(A)$ such that for all $k \geq 0$, $U_A \notin L_k^{P,\Sigma}(A)$, $\mathrm{UP}(A) \cap H_k^{P,\Sigma}(A) = \emptyset$, $L_k^{P,\Sigma}(A) \subsetneq L_{k+1}^{P,\Sigma}(A)$, and $H_k^{P,\Sigma}(A) \subsetneq H_{k+1}^{P,\Sigma}(A)$.*

Acknowledgement

The authors would like to thank J. Ramachandran for valuable discussions.

References

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th Annual ACM Symp. Theory of Computing*, pages 421–429, 1885.

[BDG88] J. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, Berlin, 1988.

[Bei92a] R. Beigel. Perceptrons, pp, and the polynomial-time hierarchy. In *Proc. Seventh Annual IEEE Structure in Complexity Theory Conference*, pages 14–19, 1992.

[Bei92b] R. Beigel. Personal correspondence, 1992.

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

[BI87] M. Blum and R. Impagliazzo. Generic oracles andd oracle classes. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987.

[FFK91] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. In *Proc. Structure in Complexity Theory Sixth Annual Conference*, pages 30–42. IEEE Computer Society Press, 1991.

[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.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Pairty, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 11(2), April 1988.

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

[HH91] J. Hartmanis and L. Hemachandra. One-way functions and non-isomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[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.

[KS85] K. Ko and U. Schöning. On circuit-size and the low hierarchy in NP. *SIAM J. Comput.*, 14(1):41–51, 1985.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2), 1992.

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

[Sch86] U. Schöning. *Complexity and Structure.* Lecture Notes in Computer Science Vol. 211. Springer-Verlag, 1986.

[SL91] M. Sheu and T. Long. The extended low hierarchy is an infinite hierarchy. Technical Report OSU-CISRC-8/91-TR22, The Ohio State University, 1991.

[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.

Appendix

Lemma 3.4 (Switching Lemma) *Let G be an* AND *of* OR*s with bottom fanin $\leq t$, let $\mathcal{B} = \{B_j\}$ be a partition of the variables in G, and let q be a number between 0 and 1. Then, for a random restriction ρ from $\hat{R}^+_{q,\mathcal{B}}$, $Pr(G\lceil_{\rho g'(\rho)}$ is not equivalent to a circuit of* OR *of* AND*s with bottom fanin $\leq s$) is bounded by α^s, where $\alpha < 6qt$. The above probability also holds with $\hat{R}^+_{q,\mathcal{B}}$ replaced by $\hat{R}^-_{q,\mathcal{B}}$, or with G being an* OR *of* AND*s circuit to be converted to an* AND *of* OR*s circuit.*

Proof. The proof follows the original proof of the switching lemma in [Has87]. We will prove a stronger form of this lemma by conditioning the probability of not being able to convert G upon an arbitrary function being forced to 1 by $\rho g'(\rho)$. Let $AND(G\lceil_{\rho g'(\rho)}) > s$ denote the event that the circuit G, after applying a random restriction $\lceil_{\rho g'(\rho)}$, where ρ is from $\hat{R}^+_{q,\mathcal{B}}$, cannot be converted to a circuit of OR of ANDs with fanin $\leq s$. We will prove that the probability of this event is bounded by α^s, where $\alpha = \frac{4q}{2^{1/t}-1} < 6qt$.

Lemma A.1 *Let $G = \wedge_{i=1}^{w} G_i$, where G_i are* OR*'s with fanin $\leq t$. Let F be an arbitrary function and let ρ be a restriction from $\hat{R}^+_{q,\mathcal{B}}$. Then*

$$Pr(AND(G\lceil_{\rho g'(\rho)}) \geq s \mid F\lceil_{\rho g'(\rho)} \equiv 1) \leq \alpha^s$$

where $\alpha = \frac{4q}{2^{1/t}-1}$.

Proof. The proof is by induction on w. The lemma is trivially true for the base case of $w = 0$. For the induction step, let us consider the left most bottom OR gate G_1. There are two cases: G_1 is either forced to be 1 by $\rho g'(\rho)$ or it is not. Thus

$$
\begin{aligned}
&Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1) \leq \\
&\qquad max[Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \equiv 1), \qquad (1)\\
&\qquad\quad Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1)] \qquad (2)
\end{aligned}
$$

Considering term (1):

$$Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid (F \wedge G_1)\lceil_{\rho g'(\rho)} \equiv 1)$$
$$= Pr(AND(\wedge_{i=1}^{w} G_i\lceil_{\rho g'(\rho)}) > s \mid (F \wedge G_1)\lceil_{\rho g'(\rho)} \equiv 1)$$
$$= Pr(AND(\wedge_{i=2}^{w} G_i\lceil_{\rho g'(\rho)}) > s \mid (F \wedge G_1)\lceil_{\rho g'(\rho)} \equiv 1) \leq \alpha^s$$

by the inductive hypothesis.

Now we only have to consider term (2): $Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1)$. We first describe a strategy for estimating this probability. Recall that ρ is a random restriction from $\hat{R}^+_{q,\mathcal{B}}$. Let $\mathcal{B} = \{B_i\}_{i=1}^m$ be the partition of the variables in G. We need to consider those restrictions $\rho g'(\rho)$ that do not force G_1 to 1. If G_1 is not forced to a constant 0 by $\rho g'(\rho)$, then $\rho g'(\rho)$ must assign * to some of the variables in G_1. Assume y variables in G_1 are assigned * by $\rho g'(\rho)$. These variables will remain in $G\lceil_{\rho g'(\rho)}$. In order to apply the inductive hypothesis, we will assign values 1 or 0 to these y variables. In each case, G_1 will not have any remaining variables. We then convert the remaining circuit $\wedge_{i=2}^{w} G_i$, after applying a random restriction $\rho g'(\rho)$, to an OR of AND's with bottom fanin $\leq s - y$. We can use the inductive hypothesis to estimate a bound on the probability that this conversion fails.

More formally, let T denote the set of variables in G_1 and let $B_1, \ldots, B_r$ be the blocks of variables that contains variables in T. We say that a block B is *exposed* by $\rho g'(\rho)$ if there is a variable $x \in B$ such that $x \in T$ and $\rho g'(\rho)(x) = *$. For $Y \subseteq \{B_1, \ldots, B_r\}$, let $Exp_{\rho g'(\rho)}(Y)$ denote the event that the blocks in Y are exactly the blocks exposed by the restriction $\rho g'(\rho)$. Let $[r]$ denote the set $\{B_1, \ldots, B_r\}$. Term (2) can be estimated as follows:

$$Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1) \leq$$
$$\sum_{Y \subset [r], Y \neq \emptyset} [Pr(Exp_{\rho g'(\rho)}(Y) \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1) \times \tag{2a}$$
$$Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1 \wedge Exp_{\rho g'(\rho)}(Y))] \tag{2b}$$

Note that Y is not an empty set in the above formula. If $Y = \emptyset$, then all the variables in G_1 must be assigned either 0 or 1 by $\rho g'(\rho)$. Under the condition that $G_1\lceil_{\rho g'(\rho)} \not\equiv 1$,

$G_1\lceil_{\rho g'(\rho)}$ must be a constant 0 when $Y = \emptyset$. In this case, the whole $G\lceil_{\rho g'(\rho)}$ circuit is a constant 0. The probability that it cannot be converted is equal to 0.

Now let us estimate $(2a)$. We first estimate this probability assuming that Y consists of only a single block B_i.

Lemma A.2 $Pr(Exp_{\rho g'(\rho)}(B_i) \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1) \leq 2q$.

Proof. Let $K_1 = \{\delta \mid \delta$ is a restriction of the form $\rho g'(\rho)$, $\rho \in \hat{R}^+_{q,\mathcal{B}}$, $Exp_\delta(B_i)$ is true, $F\lceil_{\rho g'(\rho)} \equiv 1$, and $G_1\lceil_\delta \not\equiv 1\}$ and let $K_2 = \{\delta \mid \delta$ is a restriction of the form $\rho g'(\rho)$, $\rho \in \hat{R}^+_{q,\mathcal{B}}$, $Exp_\delta(B_i)$ is false, $F\lceil_{\rho g'(\rho)} \equiv 1$, and $G_1\lceil_\delta \not\equiv 1\}$. The probability $Pr(Exp_{\rho g'(\rho)}(B_i) \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1)$ can be rewritten as the following:

$$\frac{\sum_{\delta \in K_1} Pr(\delta)}{\sum_{\delta \in K_1 \cup K_2} Pr(\delta)} \tag{3}$$

To estimate (3), we will define a function H that maps a restriction in K_1 to a restriction in K_2. First, let $P_i = \{x_j \mid x_j \in B_i \wedge x_j \in G_1\}$ and let $N_i = \{x_j \mid x_j \in B_i \wedge \bar{x}_j \in G_1\}$. Now for each $\delta = \rho g'(\rho) \in K_1$, let $H(\delta) = \bar{\delta} = \bar{\rho} g'(\bar{\rho})$ such that

1. $\bar{\rho}(x_j) = 1$ for $x_j \in N_i$.
2. $\bar{\rho}(x_j) = \rho(x_j)$ for $x_j \notin N_i$.
3. $g'(\bar{\rho})$ chooses $s_i = 0$ for B_i.

We now argue that if δ is a restriction from K_1, then $\bar{\delta} = H(\delta)$ is a restriction in K_2. Let $\delta = \rho g'(\rho)$ be a restriction from K_1. There are three different cases to consider.

Case 1 $|P_i| \geq 2$ **:** This is impossible. Note that δ is a restriction from K_1. If there is more than 1 positive variable from G_1 in B_i, then at least one of them will be assigned 1 by δ, which contradicts the fact that $G_1\lceil_\delta \not\equiv 1$.

Case 2 $|P_i| = 1$ **:** In this case, the only variable x_j in P_i must be assigned * by δ. Let $\bar{\delta} = H(\delta)$. Then the only difference between δ and $H(\delta)$ is the assignment to x_j. It is clear that changing the assignment of x_j from $*$ to 0 does not force G_1 to 1 if δ

does not force G_1 to 1. Also, since changing a * to non-* will not change the output of a function that is already outputting a constant value, $F\lceil_{\bar{\delta}}$ remains 1 if $F\lceil_{\delta}$ is 1. Thus, $\delta \in K_1$ implies $H(\delta) \in K_2$.

Case 3 $|P_i| = 0$ **:** In this case, the variable x_j that is assigned $*$ by δ is in N_i. In this case, the only difference between δ and $H(\delta)$ is the assignment to x_j. Using an argument similar to that for Case 2, we can conclude that $\delta \in K_1$ implies $H(\delta) \in K_2$.

Now to estimate the value of (3), we will compare the probabilities of the restrictions in K_1 and K_2. Note that if δ_1 and δ_2 are both in K_1 and are identical restrictions except for the assignments to variables in N_i, then $H(\delta_1) = H(\delta_2)$. Thus, there are $2^{|N_i|}$ different δ's in K_1 that map to the same restriction $H(\delta)$ in K_2. We have

$$\frac{\sum_{\delta \in K_1} Pr(\delta)}{\sum_{\delta \in K_1} Pr(H(\delta))} \leq \frac{q}{1-q} \sum_{l=0}^{|N_i|} \binom{|N_i|}{l} \frac{q^l (1-q)^{|N_i|-l}}{(1-q)^{|N_i|}} = \frac{q}{1-q}(1-q)^{-|N_i|}$$

The first factor $q/(1-q)$ comes from the fact that g' of δ chooses $s_i = *$ for block B_i and g' of $H(\delta)$ chooses $s_i = 0$ for block B_i. Note that in the case of $|P_i| = 0$, the subscript l of the sum should start with 1. Since we are looking for the upper bound of this ratio, l starting from 0 is acceptable. Also, since $|N_i| \leq t$ and $q < \frac{1}{6t}$, we have $(1-q)^{-|N_i|} < 2$. Thus, the above ratio is bounded by $\dfrac{2q}{1-q}$.

Now the probability (3) is

$$\leq \frac{\sum_{\delta \in K_1} Pr(\delta)}{\sum_{\delta \in K_1} Pr(\delta) + \sum_{\delta \in K_1} Pr(H(\delta))} \leq \frac{\frac{2q}{1-q}}{\frac{2q}{1-q} + 1} \leq 2q$$

Note that the first inequality is true because $H(\delta)$ is in K_2 and the fact that we are only looking for an upper bound for (3). This completes the proof of Lemma A.2. $\Box$

Lemma A.3 $Pr(Exp_{\rho g'(\rho)}(Y) \mid F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1) \leq (2q)^{|Y|}$.

Proof. This is proved in the same way as Lemma A.2. Note that for each block B_i in Y, B_i can contain at most one positive variable in G_1. We can define the function H such that H makes the same changes in δ as in the proof of Lemma A.2 on all the blocks simultaneously. $\square$

We have a bound of $(2q)^{|Y|}$ for the term $(2a)$. We now need to estimate the term $(2b)$. Consider a restriction $\rho g'(\rho)$ such that $F\lceil_{\rho g'(\rho)} \equiv 1 \wedge G_1\lceil_{\rho g'(\rho)} \not\equiv 1 \wedge Exp_{\rho g'(\rho)}(Y)$. It is not difficult to see that $\rho g'(\rho)$ assigns 0 or 1 to the variables of G_1 that are not in the exposed blocks. The condition $G_1\lceil_{\rho g'(\rho)} \not\equiv 1$ can be removed by incorporating into $F\lceil_{\rho g'(\rho)} \equiv 1$ (a) assignments to variables in G_1 not in exposed blocks, and (b) conditions on assignements to variables in the exposed blocks. The conditions in (b) have to guarantee that no restrictions $\rho g'(\rho)$ result in $G_1\lceil_{\rho g'(\rho)} \equiv 1$. This is necessary since there may exist a restriction $\rho g'(\rho)$ such that $\rho g'(\rho)$ exposes Y and such that $F\lceil_{\rho g'(\rho)} \equiv 1$, but $G_1\lceil_{\rho g'(\rho)} \equiv 1$.

Now the term (2b) can be rewritten as the following

$$Pr(AND(G\lceil_{\rho g'(\rho)}) > s \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1 \wedge Exp_{\rho g'(\rho)}(Y)),$$

where $\hat{F}$ is the new boolean formula that combines F and requirements of (a) and (b) mentioned above.

To get rid of the condition $Exp_{\rho g'(\rho)}(Y)$, the strategy is to remove all the variables that are both in Y and G. The probability of not being able to convert the remaining circuit G' will be estimated by the inductive hypothesis. Consider a restriction $\rho g'(\rho)$ such that $\hat{F}\lceil_{\rho g'(\rho)} \equiv 1$ and $Exp_{\rho g'(\rho)}(Y)$ are true. The circuit $G_1\lceil_{\rho g'(\rho)}$ contains only $|Y|$ different variables from the blocks of Y. This is because, for each block of Y, the restriction $g'(\rho)$ keeps only one variable that was assigned * by ρ and assigns 1 to the other variables in the block. If we can remove these $|Y|$ variables that are assigned *'s in $G\lceil_{\rho g'(\rho)}$, then the remaining circuit G' does not have any variables that are also in the blocks of Y. In particular, G_1 has no variables remaining in G'. Once there are no shared variables in G' and Y, the event that G' cannot be converted and the event $Exp_{\rho g'(\rho)}(Y)$ are independent. We have

$$\begin{aligned} & Pr(AND(G') > s - |Y| \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1 \wedge Exp_{\rho g'(\rho)}(Y)) \\ = & \frac{Pr(AND(G') > s - |Y| \wedge Exp_{\rho g'(\rho)}(Y) \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1)}{Pr(Exp_{\rho g'(\rho)}(Y) \mid \hat{F}\lceil_{\rho} \equiv 1)} \end{aligned}$$

$$= \frac{Pr(AND(G') > s - |Y| \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1) \times Pr(Exp_{\rho g'(\rho)}(Y) \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1)}{Pr(Exp_{\rho g'(\rho)}(Y) \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1)}$$

$$= Pr(AND(G') > s - |Y| \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1).$$

It is clear that we can apply the inductive hypothesis to estimate $Pr(AND(G') > s - |Y| \mid \hat{F}\lceil_{\rho g'(\rho)} \equiv 1)$ since the circuit G' has at most $w-1$ OR gates (all the variables in G_1 have been replaced by constants).

To remove the variables in Y from $G\lceil_{\rho g'(\rho)}$ that are assigned *'s after the restriction $\rho g'(\rho)$, we will try all the possible assignments for these variables. Let $G' = G\lceil_{\rho g'(\rho)}\lceil_{\sigma}$, where σ is one of the possible assignments. There are $2^{|Y|}$ possible G'. Note that if $G\lceil_{\rho g'(\rho)}$ cannot be converted to an OR of AND's with bottom fanin $\leq s$ then there is at least one of the G's cannot be converted to an OR of AND's with bottom fanin $\leq s - |Y|$. By inductive hypothesis, the probability that a G' cannot be converted to an OR of AND's with bottom fanin $\leq s - |Y|$ is bounded by $\alpha^{s-|Y|}$. Thus, the probability that $G\lceil_{\rho g'(\rho)}$ cannot be converted to an OR of AND's with bottom fanin $\leq s$ is bounded by $2^{|Y|}\alpha^{s-|Y|}$, which is the bound for $(2b)$.

Finally, the term (2) is

$$\sum_{Y \subset [r], Y \neq \emptyset} (2q)^{|Y|} 2^{|Y|} \alpha^{s-|Y|} = \alpha^s \sum_{i=1}^{r} \binom{r}{i} (\frac{4q}{\alpha})^i \leq \alpha^s((1 + \frac{4q}{\alpha})^r - 1) = \alpha^s(2^{r/t} - 1) \leq \alpha^s$$

The last inequality is by the fact that $r \leq t$ and thus $2^{r/t} \leq 2$. This completes the induction step and the proof of Lemma A.1. $\square$

To convert a circuit G of OR of AND's, let us consider its negation $\bar{G}$, which is an AND or OR's. By using Lemma A.1, we can convert $\bar{G}$ to an equivalent OR of AND's, whose negation is equivalent to G.

It is also straightforward to replace the distribution $\hat{R}^{+}_{q,\mathcal{B}}$ with $\hat{R}^{-}_{q,\mathcal{B}}$ and obtain the same probability using a similar proof. This concludes the proof of Lemma 3.4. $\square$

Lemma 3.5 *For arbitrary $k \geq 2$, let C be a $\Sigma_k(r)$-circuit (or a $\Pi_k(r)$-circuit), where $r = t$. Let V be the set of the variables in C and let $\mathcal{B} = \{B_j\}$ be a partition of the variables in V. Let $q = 1/(12 \cdot t)$ and let ρ be a random restriction from $\hat{R}^{+}_{q,\mathcal{B}}$ or $\hat{R}^{-}_{q,\mathcal{B}}$. The probability that C, after applying the random restriction $\rho g'(\rho)$, can be converted to an equivalent $\Sigma_{k-1}(r)$-circuit (or a $\Pi_{k-1}(r)$-circuit, respectively) is greater than $2/3$.*

Proof. By definition of $\Sigma_k(r)$-circuits, there are at most $2^r = 2^t$ many bottom depth-2 subcircuits in a $\Sigma_k(r)$-circuit. Each of these subcircuits has bottom fanin $\leq r = t$. To apply the switching lemma, let $s = t$. The probability that not every bottom level AND-of-OR (OR-of-AND) circuit, after applying the random restriction $\rho g'(\rho)$, can be converted to an OR-of-AND (AND-of-OR) circuit is bounded above by $2^t \alpha^s = (2\alpha)^s$. The probability that every circuit can be converted is at least $1-(2\alpha)^s$. Since $\alpha < 6qt = 1/2$, we have $1-(2\alpha)^s \geq 2/3$ for sufficiently large s.

After the conversion, we can collapse level k and $k-1$ of C. Therefore, the new circuits have only k levels. Also, since all the gates at level 1 to level $k-2$ are not effected by the restriction $\rho g'(\rho)$, the number of gates at levels 1 to $k-2$ is less than 2^r. Thus, circuit $C\lceil_{\rho g'(\rho)}$ is a $\Sigma_{k-1}(r)$-circuit. The proof for $\Pi_k(r)$-circuits is similar. □

Lemma 3.6 shows that a random restriction with the *U property* transforms a circuit computing f_k^n to a circuit that contains a subcircuit computing an f_{k-1}^n function.

Lemma 3.6 *For each $k \geq 2$, there exists an integer n_k such that the following holds for all $n \geq n_k$: Let C be a C_k^n circuit computing an f_k^n function, let $q = n^{-1/3}$, and let ρ be a random restriction from $\hat{R}^{+}_{q,\mathcal{B}}$ if k is even or from $\hat{R}^{-}_{q,\mathcal{B}}$ if k is odd. The probability that $C\lceil_{\rho g'(\rho)}$ contains a subcircuit computing an f_{k-1}^n function is greater than $2/3$.*

Proof. The proof is bascially the same as the proof of Lemma 6.8 of [Has87]. Even though the f_k^n functions we use here is slightly different from the f_k^n functions in [Has87] and the definition of $\rho g'(\rho)$ is different from the original $\rho g(\rho)$ in the proof of Lemma 6.8

of [Has87] , it is easy to see that we obtain the same probability for $C_k^n\lceil_{\rho g'(\rho)}$ having a subcircuit computing an f_{k-1}^n function. $\square$