Electronic Colloquium on Computational Complexity, Report No. 26 (1998)

Gaps in Bounded Query Hierarchies

Richard Beigel*
Lehigh University

May 4, 1998

Abstract

Prior results show that most bounded query hierarchies cannot contain finite gaps. For example, it is known that

$$\mathrm{P}^{\mathrm{SAT}}_{(m+1)\text{-tt}} = \mathrm{P}^{\mathrm{SAT}}_{m\text{-tt}} \Rightarrow \mathrm{P}^{\mathrm{SAT}}_{\mathrm{btt}} = \mathrm{P}^{\mathrm{SAT}}_{m\text{-tt}}$$

and for all sets A

- $\mathrm{FP}^{A}_{(m+1)\text{-tt}} = \mathrm{FP}^{A}_{m\text{-tt}} \Rightarrow \mathrm{FP}^{A}_{\mathrm{btt}} = \mathrm{FP}^{A}_{m\text{-tt}}$
- $\mathrm{P}^{A}_{(m+1)\text{-T}} = \mathrm{P}^{A}_{m\text{-T}} \Rightarrow \mathrm{P}^{A}_{\mathrm{bT}} = \mathrm{P}^{A}_{m\text{-T}}$
- $\mathrm{FP}^{A}_{(m+1)\text{-T}} = \mathrm{FP}^{A}_{m\text{-T}} \Rightarrow \mathrm{FP}^{A}_{\mathrm{bT}} = \mathrm{FP}^{A}_{m\text{-T}}$

where $\mathrm{P}^{A}_{m\text{-tt}}$ is the set of languages computable by polynomial-time Turing machines that make m nonadaptive queries to A; $\mathrm{P}^{A}_{\mathrm{btt}} = \bigcup_m \mathrm{P}^{A}_{m\text{-tt}}$; $\mathrm{P}^{A}_{m\text{-T}}$ and $\mathrm{P}^{A}_{\mathrm{bT}}$ are the analogous adaptive queries classes; and $\mathrm{FP}^{A}_{m\text{-tt}}$, $\mathrm{FP}^{A}_{\mathrm{btt}}$, $\mathrm{FP}^{A}_{m\text{-T}}$, and $\mathrm{FP}^{A}_{\mathrm{bT}}$ in turn are the analogous function classes.

It was widely expected that these general results would extend to the remaining case — languages computed with nonadaptive queries — yet results remained elusive. The best known was that

$$\mathrm{P}^{A}_{2m\text{-tt}} = \mathrm{P}^{A}_{m\text{-tt}} \Rightarrow \mathrm{P}^{A}_{\mathrm{btt}} = \mathrm{P}^{A}_{m\text{-tt}}.$$

We disprove the conjecture. In fact,

$$\mathrm{P}^{A}_{\lfloor\frac{4}{3}m\rfloor\text{-tt}} = \mathrm{P}^{A}_{m\text{-tt}} \not\Rightarrow \mathrm{P}^{A}_{(\lfloor\frac{4}{3}m\rfloor+1)\text{-tt}} = \mathrm{P}^{A}_{\lfloor\frac{4}{3}m\rfloor\text{-tt}}.$$

Thus there is a $\mathrm{P}^{A}_{m\text{-tt}}$ hierarchy that contains a finite gap.

We also make progress on the 3-tt vs. 2-tt case:

$$\mathrm{P}^{A}_{3\text{-tt}} = \mathrm{P}^{A}_{2\text{-tt}} \Rightarrow \mathrm{P}^{A}_{\mathrm{btt}} \subseteq \mathrm{P}^{A}_{2\text{-tt}}/\mathrm{poly}.$$

*Research performed at the University of Maryland while on sabbatical from Yale University. Address: Elect Eng Comp Science, 19 Memorial Dr W Ste 2, Bethlehem PA 18015-3084, USA. Supported in part by the National Science Foundation under grants CCR-8958528, CCR-9700417, and CCR-9796317 and by NASA under grant NAG 52895. Email: `beigel@eecs.lehigh.edu`. Web: `http://www.eecs.lehigh.edu/~beigel/`

ISSN 1433-8092

1. Introduction

Gaps have been studied in many kinds of computational hierarchies, and many different behaviors have been found, such as

- arbitrarily large gaps,
- only small gaps,
- no gaps at all, or
- no gaps unless the hierarchy collapses.

Time, space, and other Blum complexity measures The linear speedup and tape compression theorems [11] say that there are no complexity classes strictly between $\mathrm{DTIME}(t(n))$ and $\mathrm{DTIME}(2t(n))$ or between $\mathrm{DSPACE}(s(n))$ and $\mathrm{DSPACE}(2s(n))$. In fact, every Blum complexity measure contains arbitrarily large gaps [4]. The time- and space-hierarchy theorems [11] show that only small gaps are possible if you restrict to constructible complexity bounds.

Polynomial Hierarchy [6] The polynomial hierarchy has what is called the "upward collapse" property: if it contains a gap at level m then it collapses to level m.

- $\Sigma^p_{m+1} = \Sigma^p_m \Rightarrow \mathrm{PH} = \Sigma^p_m$
- $\Pi^p_{m+1} = \Pi^p_m \Rightarrow \mathrm{PH} = \Pi^p_m$
- $\Delta^p_{m+1} = \Delta^p_m \Rightarrow \mathrm{PH} = \Delta^p_m$

Boolean Hierarchy [5] $\mathrm{BH}(0) = \mathrm{P}$, $\mathrm{BH}(m+1) = \{A - B : A \in \mathrm{NP}, B \in \mathrm{BH}(m)\}$, and $\mathrm{BH} = \bigcup_m \mathrm{BH}(m)$. The Boolean hierarchy has the upward collapse property: if it contains a gap at level m then it collapses to level m.

- $\mathrm{BH}(m+1) = \mathrm{BH}(m) \Rightarrow \mathrm{BH} = \mathrm{BH}(m)$

Arithmetical Hierarchy The arithmetical hierarchy is the recursion theoretic analogue of the more modern polynomial hierarchy. All levels of the arithmetical hierarchy are distinct [17, 18], *i.e.*, it contains no gaps.

- $\Delta_1 \subset \Sigma_1 \subset \Delta_2 \subset \Sigma_2 \subset \cdots$

Bounded Query Hierarchies [1] Let A be a language. $\mathrm{FP}^A_{m\text{-T}}$ is the class of functions computed by polynomial-time oracle Turing machines that make at most m queries to A on each input. $\mathrm{FP}^A_{m\text{-tt}}$ is the class of functions computed by polynomial-time oracle Turing machines that make at most m nonadaptive queries to A on each input (that is, all m queries are made in parallel). $\mathrm{FP}^A_{\mathrm{bT}} = \bigcup_m \mathrm{FP}^A_{m\text{-T}}$ and $\mathrm{FP}^A_{\mathrm{btt}} = \bigcup_m \mathrm{FP}^A_{m\text{-tt}}$. For each reduction r, P^A_r is the class of all languages whose characteristic function is contained in FP^A_r.

Three of the four kinds of bounded query hierarchies are known to have the upward collapse property (like the polynomial and Boolean hierarchies): if any of them contains a gap at level m then it collapses to level m.

- $\mathrm{FP}^A_{(m+1)\text{-tt}} = \mathrm{FP}^A_{m\text{-tt}} \Rightarrow \mathrm{FP}^A_{\mathrm{btt}} = \mathrm{FP}^A_{m\text{-tt}}$
- $\mathrm{FP}^A_{(m+1)\text{-T}} = \mathrm{FP}^A_{m\text{-T}} \Rightarrow \mathrm{FP}^A_{\mathrm{bT}} = \mathrm{FP}^A_{m\text{-T}}$
- $\mathrm{P}^A_{(m+1)\text{-T}} = \mathrm{P}^A_{m\text{-T}} \Rightarrow \mathrm{P}^A_{\mathrm{bT}} = \mathrm{P}^A_{m\text{-T}}$

Many important problems have been classified using bounded query classes [15, 9, 14, 8, 19, 20]. In this paper we investigate the hierarchy of languages decided by a polynomial-time Turing machine that makes a bounded number of parallel queries to a fixed language A.

In light of the three results listed above, one might expect that

$$\mathrm{P}^A_{(m+1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}. \tag{1}$$

One of the seminal works on bounded queries [1] gives a simple divide-and-conquer argument from which one can easily deduce a weak form of (1):

$$\mathrm{P}^A_{2m\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}. \tag{2}$$

(2) can be understood informally as follows: Consider an unknown assignment α to a set of Boolean variables. Suppose that we are given a black box that takes a $2m$-ary Boolean formula $f(x_1,\ldots,x_{2m})$ and produces an m-ary Boolean formula $g(y_1,\ldots,y_m)$ with the guarantee that

$$f(\alpha(x_1),\ldots,\alpha(x_{2m})) = g(\alpha(y_1),\ldots,\alpha(y_m)).$$

We can then use this black box in a polynomial-time algorithm that takes a Boolean formula of *arbitrary* arity $F(x_1,\ldots,x_m)$ and produces an m-ary Boolean formula $G(y_1,\ldots,y_m)$ with the guarantee that

$$F(\alpha(x_1),\ldots,\alpha(x_m)) = G(\alpha(y_1),\ldots,\alpha(y_m)).$$

(In relation to (2), the assignment α is the characteristic function of A, the x_i's and y_i's are the queries in the reductions, and f, g, F, and G are the truth-table evaluators.)

Lozano [16] and Gasarch [10] both conjectured Equation 1. Chang [7] conjectures it now for the special case of $A \in \mathrm{NP}$.

In this paper, we disprove (1), showing in fact that

$$\mathrm{P}^A_{\lfloor\frac{4}{3}m\rfloor\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \not\Rightarrow \mathrm{P}^A_{(\lfloor\frac{4}{3}m\rfloor+1)\text{-tt}} = \mathrm{P}^A_{\lfloor\frac{4}{3}m\rfloor\text{-tt}}. \tag{3}$$

In other words, we have constructed an A for which the $\mathrm{P}^A_{\mathrm{btt}}$ hierarchy does not have the upward collapse property: it has a gap at level m but it does not collapse to level m.

(3) can be understood informally as follows: Consider an unknown assignment α to a set of Boolean variables. Suppose that we are given a black box that takes a $4k$-ary Boolean formula $f(x_1,\ldots,x_{4k})$ and produces a $3k$-ary Boolean formula $g(y_1,\ldots,y_{3k})$ with the guarantee that

$$f(\alpha(x_1),\ldots,\alpha(x_{4k})) = g(\alpha(y_1),\ldots,\alpha(y_{3k})).$$

Even using such a black box, there is no algorithm that takes a $(4k+1)$-ary Boolean formula $F(x_1,\ldots,x_{4k+1})$ and produces a $3k$-ary Boolean formula $G(y_1,\ldots,y_{3k})$ with the guarantee that

$$F(\alpha(x_1),\ldots,\alpha(x_{4k+1})) = G(\alpha(y_1),\ldots,\alpha(y_{3k})).$$

In fact, we prove something stronger: there is no such algorithm even if g and F are fixed to be the majority functions on $3k$ variables and on $4k+1$ variables respectively (f and G are still unrestricted).

In proving (3), we use some basic facts about representing Boolean functions as polynomials over $Z/2$, but we also develop new techniques to prove nonrepresentability of finite functions. We hope that, once these new techniques are better understood, they might be useful in proving lower bounds on circuit complexity or Boolean-formula complexity.

The positive result (2) and the negative result (3) do not match. It is an open problem to improve either of them. The simplest open question along these lines is

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{2\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{2\text{-tt}} \tag{4}$$

In this paper we also give a partial answer to that question:

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{2\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{2\text{-tt}}/\mathrm{poly}. \tag{5}$$

The proof uses a delicate hard/easy argument (cf. [2, 12]) to exploit very convenient representations of 2-ary Boolean functions. It appears very difficult to generalize even to 3-ary Boolean functions.

Question (4) can be understood informally as follows: Consider an unknown assignment α to a set of Boolean variables. Suppose that we are given a black box that takes a 3-ary Boolean formula $f(x_1,x_2,x_3)$ and produces a 2-ary Boolean formula $g(y_1,y_2)$ with the guarantee that

$$f(\alpha(x_1),\alpha(x_2),\alpha(x_3)) = g(\alpha(y_1),\alpha(y_2)).$$

Is there an algorithm using such a black box that takes a 4-ary Boolean formula $F(x_1,x_2,x_3,x_4)$ and produces a 2-ary Boolean formula $G(y_1,y_2)$ with the guarantee that

$$F(\alpha(x_1),\alpha(x_2),\alpha(x_3),\alpha(x_4)) = G(\alpha(y_1),\alpha(y_2))?$$

If we fix the predicate g (but not its inputs y_1,y_2), then the answer is yes, *i.e.*,

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{2\text{-fixed}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{2\text{-fixed}}.$$

This is one of the ideas behind our proof of (5), which can be interpreted as saying that the answer to our question is yes if we give our algorithms access to a relatively small amount of information about the set A. However, the general case where g is allowed to depend on x_1,x_2,x_3 and no information is given about A is tantalizingly open.

2. Definitions

Throughout let $\circ$ denote a binary operation on $\{0,1\}$. The symbols $\wedge$, $\vee$, and $\oplus$ denote logical AND, OR, and XOR, respectively. Let f,g,h denote Boolean formulas. Let x,y,x_i,y_i denote strings for all integer subscripts i.

Many versions of m-truth-table reducibility can be defined by specifying the complexity of the reduction and the permitted set of truth tables.

Definition 1. Let F be a set of Boolean functions; C be a class of functions; $\circ$ be $\vee$, $\wedge$, or $\oplus$; and A and L be languages.

- $L \le_F^C A$ if there exist functions $t : \Sigma^* \to F$ and $q_1, \ldots, q_m : \Sigma^* \to \Sigma^*$ such that $t, q_1, \ldots, q_m \in C$ and for all x
$$L(x) = t(x)(A(q_1(x)), \ldots, A(q_m(x))).$$

- $C_F^A = \{L : L \le_F^C A\}$.

- If f is a Boolean function, $\le_f^C$ denotes $\le_{\{f\}}^C$ and C_f^A denotes $C_{\{f\}}^A$.

- $\mathrm{P}_F^A/\mathrm{poly}$ denotes $(\mathrm{P}/\mathrm{poly})_F^A$.

- m-tt is the set of m-ary Boolean functions.

- d-$\circ$NF is the set of Boolean functions that can be written in the form $(x_{11} \otimes \cdots \otimes x_{1i_1}) \circ \cdots \circ (x_{k1} \otimes \cdots \otimes x_{ki_k})$, where $\otimes$ is some associative Boolean operation that distributes over $\circ$ and $i_1, \ldots, i_k \le d$. We say that functions in d-$\circ$NF have *degree d* over $\circ$. (This generalizes the notions of d-CNF and d-DNF.)

Some other kinds of reductions are defined by limiting the truth table's dependence on x.

Definition 2. Let C be a class of functions.

- $L \le_{\mathrm{btt}}^C A$ if there exists m such that $L \le_{m\text{-tt}}^C A$. (The truth table may depend on the input, but its arity m may not.)

- $L \le_{m\text{-fixed}}^C A$ if there exists $f : \{0,1\}^m \to \{0,1\}$ such that $L \le_f^C A$. (The truth table may not depend on the input at all.)

3. $2m$-tt vs. m-tt

The following is an easy modification of a theorem in [1].

Theorem 3.

i. $\mathrm{P}_{2m\text{-tt}}^A = \mathrm{P}_{m\text{-tt}}^A \Rightarrow \mathrm{P}_{\mathrm{btt}}^A = \mathrm{P}_{m\text{-tt}}^A$

ii. $\mathrm{P}_{2m\text{-tt}}^A \subseteq \mathrm{P}_{m\text{-tt}}^A/\mathrm{poly} \Rightarrow \mathrm{P}_{\mathrm{btt}}^A/\mathrm{poly} \subseteq \mathrm{P}_{m\text{-tt}}^A/\mathrm{poly}$

Proof: i. Assume $\mathrm{P}_{2m\text{-tt}}^A = \mathrm{P}_{m\text{-tt}}^A$. We prove by induction on n that, for all $n \ge m$, $\mathrm{P}_{(n+1)\text{-tt}}^A \subseteq \mathrm{P}_{m\text{-tt}}^A$. This is clear for $n = m$. Let $n \ge m$ and assume that $\mathrm{P}_{n\text{-tt}}^A = \mathrm{P}_{m\text{-tt}}^A$. We will show that $\mathrm{P}_{(n+1)\text{-tt}}^A = \mathrm{P}_{m\text{-tt}}^A$.

Let $B \in \mathrm{P}_{(n+1)\text{-tt}}^A$. Then there is a polynomial-time computable function t from $\Sigma^* \to 2^{2^{n+1}} \times (\Sigma^*)^m$ such that for all x
$$B(x) = f(A(q_1), \ldots, A(q_{n+1}))$$
where $(f, q_1, \ldots, q_{n+1}) = t(x)$. Note that
$$f(A(q_1), \ldots, A(q_{n+1})) = A(q_{n+1}) \wedge f(A(q_1), \ldots, A(q_n), 1) \ \vee\ \overline{A}(q_{n+1}) \wedge f(A(q_1), \ldots, A(q_n), 0).$$
Since $f(A(q_1), \ldots, A(q_n), 1)$ can be computed with n parallel queries to A, it can be computed with m parallel queries to A by the inductive hypothesis. Therefore $A(q_{n+1}) \wedge f(A(q_1), \ldots, A(q_n), 1)$ can

be computed with $m+1$ parallel queries to A; since $m+1 \le 2m$, it can be computed with m parallel queries to A by assumption. Similarly $\overline{A}(q_{n+1}) \wedge f(A(q_1),\ldots,A(q_n),0)$ can be computed with m parallel queries to A. Thus $f(A(q_1),\ldots,A(q_{n+1}))$ can be computed with $2m$ parallel queries to A, so it can be computed with m parallel queries to A by assumption. Thus $B \in \mathrm{P}^A_{m\text{-tt}}$.

ii. Similar. ∎

Note: all of the positive results (gap implies collapse) in this paper hold also in the presence of polynomial advice.

4. $\oplus$-Degree

A few facts about the degree of functions over $\oplus$ will be needed in order to apply our key lemma (next section) and prove our main result. Let $\oplus$-deg(f) denote the least d such that f has degree d over $\oplus$.

Definition 4.

- $\mathrm{OR}_n(b_1,\ldots,b_n) = b_1 \vee \cdots \vee b_n$
- $\mathrm{AND}_n(b_1,\ldots,b_n) = b_1 \wedge \cdots \wedge b_n$
- $n\text{-maj}(b_1,\ldots,b_n) = \begin{cases} 1 & \text{if } b_1+\cdots+b_n > \frac{1}{2}n \\ 0 & \text{otherwise} \end{cases}$
- A *subfunction* of a (formal) Boolean function is obtained by substituting 0 or 1 for some of its variables.

Fact 5. *Let f be a Boolean function of n variables $x_1,\ldots,x_n$.*

i. *f has a unique representation as a polynomial p_f in $x_1,\ldots,x_n$ over $Z/2$. Furthermore, the degree of p_f is $\oplus$-deg(f).*

ii. $\oplus$-deg$(f) = \oplus$-deg$(\neg f)$.

iii. *If $g(x_1,\ldots,x_i,\ldots,x_n) = f(x_1,\ldots,\neg x_i,\ldots,x_n)$ then $\oplus$-deg$(f) = \oplus$-deg(g).*

iv. $\oplus$-deg$(\mathrm{AND}_n) = n$

v. $\oplus$-deg$(\mathrm{OR}_n) = n$

vi. *In the representation of n-maj as a polynomial over $Z/2$ each term has degree $\lfloor n/2 \rfloor + 1$ or greater.*

Proof:

i. Every Boolean function on k variables can be written as an OR of ANDs of literals in such a way that at most one of the ANDs is true for each input. In particular, it can be written as an XOR of ANDs of literals. Let $d = \oplus$-deg(f). Given a d-$\oplus$NF representation of f, rewrite each term in as an XOR of ANDs of at most k literals. Replace AND by multiplication and $\neg x$ by $1+x$. Expand by the distributive law, to obtain p_f, whose degree is at most d. The mapping $f \to p_f$ described above is 1–1. Because there are 2^{2^n} polynomials in $x_1,\ldots,x_n$ over $Z/2$ and 2^{2^n} Boolean functions on $x_1,\ldots,x_n$, the mapping must be onto as well. Therefore the representation p_f is unique.

ii. $p_{\neg f} = 1 + p_f$ so $\oplus$-deg$(\neg f) = \oplus$-deg(f).

iii. $p_g(x_1,\ldots,x_n) = p_f(x_1,\ldots,1+x_i,\ldots,x_n)$, so $\oplus$-deg$(g) \le \oplus$-deg(f), and $p_f(x_1,\ldots,x_n) = p_g(x_1,\ldots,1+x_i,\ldots,x_n)$, so $\oplus$-deg$(f) \le \oplus$-deg(g).

iv. Let $f = x_1 \wedge \cdots \wedge x_n$. Then $p_f = x_1 \cdots x_n$, so $\oplus$-deg$(f) = n$.

v. This follows from parts ii–iv.

vi. Let the lowest-degree term in the polynomial representation of n-maj over $Z/2$ have degree d. If we assign 1 to all variables in that term and 0 to all other variables, then the polynomial must evaluate to 1. Then n-maj is 1 under this assignment, so $d > n/2$. ∎

Lemma 6.

i. If h is a nonconstant Boolean function such that $\mathrm{OR}_k \Rightarrow h$ *then* $\oplus$-deg$(h) \ge k$.

ii. If h is a nonconstant Boolean function such that $\neg\mathrm{AND}_k \Rightarrow h$ *then* $\oplus$-deg$(h) \ge k$.

iii. Let $a > k/2$. If h_1 is an a-ary subfunction of k-maj *then one of the following is true:*

- $(\forall$ *nonconstant* $h_2)[$ *if* $h_1 \Rightarrow h_2$ *then* $\oplus$-deg$(h_2) > a/2]$ *or*
- $(\forall$ *nonconstant* $h_2)[$ *if* $\neg h_1 \Rightarrow h_2$ *then* $\oplus$-deg$(h_2) > a/2]$

iv. If $\neg k$-maj$(x_1,\ldots,x_k) \Rightarrow h_1(x_1,\ldots,x_{k-1}) \oplus h_2(x_1,\ldots,x_k)$ *where h_2 depends (semantically) on x_k, then* $\oplus$-deg$(h_2) > k/2$.

Proof:

i. Assume that $\mathrm{OR}_k(x_1,,\ldots,x_k) \Rightarrow h(x_1,\ldots,x_m)$ and h is nonconstant. Without loss of generality, $m \ge k$ (otherwise allow dummy arguments to h). Since h is nonconstant, there exist $a_1,\ldots,a_m$ such that $h(a_1,\ldots,a_m) = 0$. Since $\mathrm{OR}_k(x_1,\ldots,x_k \Rightarrow h(x_1,\ldots,x_m)$ we must have $a_1 = \cdots = a_k = 0$. Let $h'(x_1,\ldots,x_k) = h(x_1,\ldots,x_k,a_{k+1},\ldots,a_m)$. Then $\mathrm{OR}_k = h'$, so $\oplus$-deg$(h') = k$. Therefore $\oplus$-deg$(h) \ge k$.

ii. Negate all variables, which does not effect $\oplus$-deg$()$, and apply part (i).

iii. Because $a > k/2$, h_1 is a nonconstant subfunction of k-maj. Therefore, either $\mathrm{OR}_{\lfloor a/2 \rfloor + 1}$ or $\mathrm{AND}_{\lfloor a/2 \rfloor + 1}$ is a subfunction of h_1. Let h be the corresponding subfunction of h_2 (obtained by setting the same variables to 0 or 1 in h_2 as in h_1).

In the fi rst case, if $h_1 \Rightarrow h_2$ then $\mathrm{OR}_{\lfloor a/2 \rfloor + 1} \Rightarrow h$, so $\oplus$-deg$(h_2) \ge \oplus$-deg$(h) > a/2$ by part (i). In the second case, if $\neg h_1 \Rightarrow h_2$ then $\neg\mathrm{AND}_{\lfloor a/2 \rfloor + 1} \Rightarrow h$, so $\oplus$-deg$(h_2) \ge \oplus$-deg$(h) > a/2$ by part (ii).

iv. Assume that $\neg k$-maj $\Rightarrow h_1 \oplus h_2$. Equivalently, we have k-maj $\vee (h_1 \oplus h_2)$. That, in turn, is equivalent to k-maj $\oplus h_1 \oplus h_2 \oplus (k$-maj $\wedge (h_1 \oplus h_2))$. Call that formula ϕ.

Assume, for the sake of contradiction, that $\oplus$-deg$(h_2) \le k/2$. Then h_2 contributes a term t involving x_k and having degree $k/2$ or less. However, h_1 contributes no terms involving x_k, and the rest of the formula contributes only terms of degree greater than $k/2$, because all terms in k-maj have degree greater than $k/2$. Thus the term t is not canceled out from ϕ, so ϕ cannot be constant. But ϕ is identically equal to 1. This contradiction proves that $\oplus$-deg$(h_2) > k/2$. ∎

5. Key Lemma

In this section we present a lemma that is the key to our main result. The lemma is proved in Appendix 1. This lemma will be applied in the next section, with $m = 3k$, $n = 4k$, $g = 3k$-maj, and $F = (4k+1)$-maj. We have stated the lemma in terms of m, n, g, and F, so that it will be clear which properties of the majority function are used in the proof, and also so that it will be clear where $3k$ and $4k$ come from. This may be helpful to anyone trying to understand the proof of the key lemma or trying to prove a stronger gap result.

Notation 7.

- Let X be an infinite set.
- Let $\prec$ be a well-founded partial order on X. That is, $\prec$ has no infinite descending chains.
- We extend the definition $\prec$ to subsets of X as follows: $U \prec \{v_1, \ldots, v_k\}$ if
  - $U \neq V$ and
  - there exists a partition $U_1, \ldots, U_k$ of U such that $(\forall i)\,[U_i = \{v_i\}$ or $(\forall u \in u_i)[u \prec v_i]]$.
- Let Pred_X^n denote the set of formal n-ary Boolean predicates over X. That is, Pred_X^n is the set of formulas $h(x_1, \ldots, x_n)$ where h is a Boolean formula and $x_1, \ldots, x_n \in X$.
- We write $h(u_1, \ldots, u_j) \prec (v_1, \ldots, v_k)$ and $h_1(u_1, \ldots, u_j) \prec h_2(v_1, \ldots, v_k)$ if $\{u_1, \ldots, u_j\} \prec \{v_1, \ldots, v_k\}$.
- A partial function α' *extends* a partial function α (denoted $\alpha' \sqsupseteq \alpha$) if $\mathrm{dom}(\alpha') \supseteq \mathrm{dom}(\alpha)$ and $(\forall x \in \mathrm{dom}(\alpha))[\alpha'(x) = \alpha(x)]$.
- When we write "extend α to satisfy some condition" we mean "find a total assignment $\alpha' \sqsupseteq \alpha$ such that α' satisfies that condition, and then let $\alpha = \alpha'$."
- When we write "extend α on Y to satisfy some condition" we mean "find a partial assignment $\alpha' \sqsupseteq \alpha$ such that $Y \subseteq \mathrm{dom}(\alpha')$ and α' satisfies that condition, and then let $\alpha = \alpha'$."
- If Φ and Γ are formal Boolean predicates over a set X and α is a partial function from X to $\{0,1\}$, we write $\Phi =_\alpha \Gamma$ if and only if Φ and Γ take the same value under every assignment that extends α, *i.e.*,

  $$f(x_1, \ldots, x_n) =_\alpha g(y_1, \ldots, y_n) \text{ iff } (\forall \alpha' \sqsupseteq \alpha)[f(\alpha'(x_1), \ldots, \alpha'(x_n)) = g(\alpha'(y_1), \ldots, \alpha'(y_n))].$$

Lemma 8.

- *Let m and n be natural numbers such that $m < n < 2m$.*
- *Let g be a function: $\{0,1\}^m \to \{0,1\}$ such that*
  - *g is monotone*
  - *g is not a function of $m-1$ variables or fewer,*
  - *if g' is a $(2m-n)$-ary subfunction of g then one of the following is true:*
    - $(\forall$ *nonconstant* $h)[$ *if* $g' \Rightarrow h$ *then* $\oplus\text{-deg}(h) > n-m]$ *or*
    - $(\forall$ *nonconstant* $h)[$ *if* $\neg g' \Rightarrow h$ *then* $\oplus\text{-deg}(h) > n-m]$
  - *if $\neg g(x_1,\ldots,x_m) \Rightarrow h_1(x_1,\ldots,x_{m-1}) \oplus h_2(x_1,\ldots,x_m)$ where h_2 depends (semantically) on x_m, then $\oplus\text{-deg}(h_2) > n-m$.*
- *Let F be a function $\{0,1\}^{n+1} \to \{0,1\}$ such that*
  - *$F \neq h_1 \oplus h_2$ for any Boolean functions h_1 and h_2 such that h_1 is a function of n variables or fewer and $\oplus\text{-deg}(h_2) \leq n-m$.*
  - *F has no minterm or maxterm of size $n-m$ or less,*
  - *If F' is an $(m+1)$-ary subfunction of F, then $\oplus\text{-deg}(F) > n-m$.*
- *Let r be a function: $\mathrm{Pred}_X^n \cup X^n \to X^m$ such that*
  - $(\forall Q)[Q \prec r(Q)]$
  - $u \neq v \Rightarrow [r(u)] \cap [r(v)] = \emptyset$,

  where $[(x_1,\ldots,x_m)]$ denotes $\{x_1,\ldots,x_m\}$.
- *Let R be a function: $X^{n+1} \to \mathrm{Pred}_X^n$.*

Then there exists a total function $\alpha : X \to \{0,1\}$ such that

(1) $(\forall Q \in \mathrm{Pred}_X^n)[Q =_\alpha g(r(Q))]$, *and*

(2) $(\exists \vec{x} \in X^{n+1})[F(\vec{x}) =_\alpha 1 - R(\vec{x})]$,

6. $4k$-tt vs. $3k$-tt

In this section we prove that $\mathrm{P}^A_{4k\text{-tt}} = \mathrm{P}^A_{3k\text{-tt}} \not\Rightarrow \mathrm{P}^A_{(4k+1)\text{-tt}} = \mathrm{P}^A_{4\text{-tt}}$.

Theorem 9. *Let $n \leq \frac{4}{3}m$. There exists a set A such that*

$$\mathrm{P}^A_{(n+1)\text{-maj}} \not\subseteq \mathrm{P}^A_{n\text{-tt}} = \mathrm{P}^A_{m\text{-maj}}$$

Proof: Definitions:

- Fix a string alphabet Σ and a tupling function $\langle\rangle$ from $(\Sigma^*)^{<\infty}$ to Σ^*. Our only requirement on $\langle\rangle$ is that its result always be longer than each of its arguments.
- For strings x and y, we say $x \prec y$ iff $|x| < |y|$.

- For a formal Boolean predicate $h(x_1,\ldots,x_n)$, let

  $$r(h(x_1,\ldots,x_n)) = (\langle x_1,\ldots,x_n,\hat{h},1\rangle,\ldots,\langle x_1,\ldots,x_n,\hat{h},m\rangle),$$

  where $\hat{h}$ is a nonempty string encoding h.

- Let $r((x_1,\ldots,x_n)) = (\langle x_1,\ldots,x_n,\Lambda,1\rangle,\ldots,\langle x_1,\ldots,x_n,\Lambda,m\rangle)$.

In order to make $\mathrm{P}^A_{m\text{-maj}} = \mathrm{P}^A_{n\text{-tt}}$ we will ensure that

$$m\text{-maj}(A(\langle x_1,\ldots,x_n,\hat{h},1\rangle),\ldots,A(\langle x_1,\ldots,x_n,\hat{h},m\rangle)) = h(A(x_1),\ldots,A(x_n)).$$

There is some flexibility in the coding. It will permit us to diagonalize.

We will think of the oracle A as a partial function from Σ^* to $\{0,1\}$. We construct A via the initial segment method. Initially, A is everywhere undefined. Let $M_1,M_2,\ldots$ be an enumeration of oracle Turing machines such that M_i makes at most n parallel oracle queries on every input. At Stage i, we will extend A in order to defeat M_i.

Stage i: Let $X = \Sigma^* - \mathrm{dom}(A)$. M_i computes a mapping from $(\Sigma^*)^{n+1} \to \mathrm{Pred}^n_{\Sigma^*}$. Restrict the domain of that mapping to X^{n+1}; in the predicates output by the mapping, substitute $A(z)$ for any $z \in \mathrm{dom}(A)$. Let R denote the resulting mapping from $X^{n+1} \to \mathrm{Pred}^n_X$. Let $f = (n+1)$-maj. Let $g = m$-maj. By Lemma 6, f and g satisfy the conditions of Lemma 8. It is clear that X, $\prec$, r, and R satisfy those conditions as well. Therefore there exists a total function $\alpha : X \to \{0,1\}$ such that

- $(\forall Q \in \mathrm{Pred}^n_X)[Q =_\alpha m\text{-maj}(r(Q))]$, and
- $(\exists \vec{x} \in X^{n+1})[(n+1)\text{-maj}(\vec{x}) =_\alpha 1 - R(\vec{x})]$

Let $\vec{x}$ be as promised and let ℓ be the length of the longest string in $\vec{x}$ or $R(\vec{x})$. Extend A by letting

$$A(x) = \begin{cases} A(x) & \text{if } x \in \mathrm{dom}(A) \\ \alpha(x) & \text{if } x \in \mathrm{dom}(\alpha) \text{ and } |x| \le \ell \\ \text{undefined} & \text{otherwise} \end{cases}$$

This completes Stage i. ▮

Corollary 10. *Let $k \ge 1$. There exists a set A such that*

$$\mathrm{P}^A_{\lfloor\frac{2}{3}k\rfloor\text{-fixed}} \subset \mathrm{P}^A_{k\text{-fixed}} = \mathrm{P}^A_{k\text{-tt}} = \mathrm{P}^A_{\lfloor\frac{4}{3}k\rfloor\text{-fixed}} = \mathrm{P}^A_{\lfloor\frac{4}{3}k\rfloor\text{-tt}} \subset \mathrm{P}^A_{(\lfloor\frac{4}{3}k\rfloor+1)\text{-fixed}} = \mathrm{P}^A_{(\lfloor\frac{4}{3}k\rfloor+1)\text{-tt}} = \mathrm{PSPACE}^A$$

Proof: The middle equalities and inequalities follow from Theorem 9. The first inequality follows from Theorem 3. For the final equality, it is necessary to modify the proof of Theorem 9 to code arbitrary PSPACE^A predicates into the majority function on blocks of size $\lfloor\frac{4}{3}k\rfloor + 1$. The additional coding does not cause any new difficulty in the diagonalization. ▮

Note 1: Because we did not clock the Turing machine M_i, the predicate $(n+1)\text{-maj}(A(x_1),\ldots,A(x_{n+1}))$ is not n-truth-table reducible to A by any Turing machine. In fact, $(n+1)\text{-maj}(A(x_1),\ldots,A(x_{n+1}))$ is not even n-weak-truth-table reducible to A (see [18] for recursion-theoretic definitions). $Q_{\|}(k,A)$ is the set of languages k-weak-truth-table reducible to A (see [2] for definitions of bounded query classes in recursion theory). If we modify the coding above by allowing h to be an index for a partial recursive function from $\{0,1\}^n$ to $\{0,1\}$ then the coding above makes $Q_{\|}(n,A) = Q_{\|}(m,A)$. Thus, if $n \leq \frac{4}{3}m$, then we have a set A such that

$$Q_{\|}(m,A) = Q_{\|}(n,A) \subset Q_{\|}(n+1,A).$$

Note 2: If, on the other hand, we clock the Turing machine M_i, then we can make the set A be recursive.

Note 3: We do not think that our gaps are the largest possible. In fact, we conjecture that $\mathrm{P}^A_{(2m-1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \not\Rightarrow \mathrm{P}^A_{2m\text{-tt}} = \mathrm{P}^A_{(2m-1)\text{-tt}}$. If you prove an analogue to our key lemma and can apply it to some $2m$-ary F and m-ary g, then you will have proved our conjecture.

Note 4: If we hope to obtain a $2m-1 : m$ gap, it will, however, be necessary to code using some function other than majority:

Theorem 11. *For all languages A and natural numbers k,*

$$\mathrm{P}^A_{(3k+2)\text{-tt}} \subseteq \mathrm{P}^A_{(2k+1)\text{-maj}} \Rightarrow \mathrm{P}^A_{\text{btt}} \subseteq \mathrm{P}^A_{(2k+1)\text{-maj}}.$$

The proof is given in Appendix 2.

Note 5: Instead of $m\text{-maj}(x_1,\ldots,x_m)$ we could use any unweighted threshold function with threshold between $m/3$ and $2m/3$. Unfortunately this does not seem to help us to obtain a larger gap. In order to improve on this technique it would seem necessary either to use an asymmetric function in place of m-maj or else to improve on inductive case 2.3 of the key lemma, which necessitates that g be monotone and satisfy the "subfunction" condition.

7. 3-tt vs. 2-tt

Lemma 12. *Let $\circ$ be an associative Boolean operation.*

$$\mathrm{P}^A_{(m+1)\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{P}^A_{1\text{-}\circ\text{NF}} \subseteq \mathrm{P}^A_{m\text{-tt}}.$$

Proof: This is a simple induction. ∎

Theorem 13. $\mathrm{P}^A_{3\text{-tt}} \subseteq \mathrm{P}^A_{2\text{-fixed}} \Rightarrow \mathrm{P}^A_{\text{btt}} \subseteq \mathrm{P}^A_{2\text{-fixed}}$.

Proof: By assumption, $P^A_{2\text{-tt}} \subseteq P^A_f$ where f is a 2-ary Boolean formula. Without loss of generality, $f(a,b) = u \circ v$ where $\circ$ is $\wedge$, $\vee$, or $\oplus$ and u and v are literals (a, $\overline{a}$, b, $\overline{b}$, 0 or 1). Let $L \in P^A_{m\text{-tt}} \subseteq P^A_{m\text{-}\circ\text{NF}}$, so

$$\begin{aligned} L(x) &= A((g_{11}(x) \otimes \cdots \otimes g_{1i_1}(x)) \circ \cdots \circ (g_{k1}(x) \otimes \cdots \otimes g_{ki_k}(x))) \\ &= A((g'_{11}(x) \circ g'_{12}(x)) \circ \cdots \circ (g'_{k1}(x) \circ g'_{k2}(x))) \qquad \text{by Lemma 12 and the assumption } P^A_{2\text{-tt}} \subseteq P^A_f \\ &= A(g''_1(x) \circ g''_2(x)) \qquad \text{for the same reason.} \end{aligned}$$

Therefore $L \in P^A_{2\text{-tt}} \subseteq P^A_{3\text{-tt}} \subseteq P^A_{2\text{-fixed}}$ by assumption. ∎

Theorem 14. $P^A_{3\text{-tt}} \subseteq P^A_{2\text{-tt}}/\text{poly} \Rightarrow P^A_{\text{btt}}/\text{poly} \subseteq P^A_{2\text{-tt}}/\text{poly}$.

Proof: Assume that $P^A_{3\text{-tt}} \subseteq P^A_{2\text{-tt}}/\text{poly}$. We will show that $P^A_{4\text{-tt}}/\text{poly} \subseteq P^A_{2\text{-tt}}/\text{poly}$. The conclusion then follows from Theorem 3(ii).

For each input length n, we will construct polynomial-size advice. Let p be a large integer that we will specify later. Let $V = \Sigma^{\leq n^p}$. Let $L = V \cup \{\overline{v} : v \in V\}$, the corresponding set of literals. Henceforth we will consider only literals in L. S will denote a subset of $L \times L$.

The language

$$\{\langle x,y,z\rangle : A((x \wedge y) \oplus z)\},$$

is in $P^A_{3\text{-tt}}$ and thus, by assumption, is contained in $P^A_{2\text{-tt}}/\text{poly}$. Therefore there exist polynomial computable functions $t : (\Sigma^*)^3 \to 2^{2^2}$ and $q_1, q_2 : (\Sigma^*)^3 \to \Sigma^*$ such that

$$A((x \wedge y) \oplus z) = A(t(x,y,z)(q_1(x,y,z), q_2(x,y,z))).$$

For each x,y,z, let $t(x,y,z)(a,b) = u \circ v$ where $\circ$ is $\wedge$, $\vee$, or $\oplus$, and u and v are literals (a, $\overline{a}$, b, $\overline{b}$, 0, or 1). We write $(x \wedge y) \oplus z \to u \circ v$, and take four cases. (In what follows, the quantifier $(\geq p\ z \in S)$ means "for at least p of all $z \in S$", *i.e.*, $(\geq p\ z \in S)[Q(z)]$ means that $|S \cap \{z : Q(z)\}|/|S| \geq p$.)

Case 1: $(\forall S)(\exists z)(\geq \frac{1}{7}\ (x,y) \in S)[(x \wedge y) \oplus z \to u \oplus v]$**.** By a greedy algorithm construct a set ADVICE consisting of $O(n^p)$ literals such that

$$(\forall (x,y) \in L^2)(\exists z \in \text{ADVICE})[(x \wedge y) \oplus z \to u \oplus v]$$

Thus given any pair of literals x,y we can find in the set ADVICE a literal z such that $A((x \wedge y) \oplus z) = A(u \oplus v)$ for some u and v, so

$$A(x \wedge y) = A(z \oplus u \oplus v).$$

Now, suppose we want to evaluate a 4-place predicate whose variables are of length n or less. Write the predicate in 4-$\oplus$NF. By the Lemma, each AND in that formula can be replaced by a formula in 2 variables. Rewrite each of those formulas in 2-$\oplus$NF. Thus the original 4-place predicate is now expressed in 2-$\oplus$NF. By the equation above, we can replace each $\wedge$ of two literals by the $\oplus$ of three literals. Thus the formula becomes an $\oplus$ of literals. By Lemma 12, it can be converted to a function of two literals, and so we are done.

Case 2: $(\forall S)(\exists z)(\geq \frac{1}{7}\ (x,y) \in S)[(A(z) = 0)$ **and** $((x \wedge y) \oplus z \to u \vee v)]$**.** Construct advice, as in Case 1, so we get $A(x \wedge y) = A(u \vee v)$. Continue as in Case 1, but use $\vee$NF instead of $\oplus$NF.

Case 3: $(\forall S)(\exists z)(\geq \frac{1}{7}\ (x,y) \in S)[(A(z) = 1)$ **and** $((x \wedge y) \oplus z \to u \wedge v)]$**.** Construct advice, as in Case 1, so we get $A(x \wedge y) = A(\overline{u} \vee \overline{v})$. Continue as in Case 2.

Case 4: $(\exists S)(\forall z)(> \frac{4}{7}\ (x,y) \in S)\left[(x \wedge y) \oplus z \to \begin{cases} u \vee v & \text{if } A(z) = 1 \\ u \wedge v & \text{if } A(z) = 0 \end{cases}\right]$. By sampling once from S, we can compute $A(z)$ correctly with probability greater than $\frac{4}{7}$. Amplify probabilities by majority voting, obtaining a probabilistic algorithm that samples $O(n^p)$ times from S and computes $A(z)$ with probability greater than $1 - 1/2^{n^p+1}$. Thus for some fixed set of $O(n^p)$ samples, we compute $A(z)$ correctly for all z. Let ADVICE be that set of samples. Now, we can evaluate any 4-ary predicate without querying A at all.

The case analysis is complete. It remains to specify p. Each time we transform formulas by using the function $t(x,y,z)$ or by applying Lemma 12 we increase the length of strings by a polynomial amount. Because we are dealing with formulas on only 4 variables, there is a constant bound on the number of transformations performed, no matter which case holds. Therefore there is some polynomial bound n^p on the length of queries to A in the final 2-ary formula we obtain. ∎

8. Functions vs. Languages

In contrast to our results for languages, the following is well known:

Fact 15. $\mathrm{FP}^A_{(m+1)\text{-tt}} \subseteq \mathrm{FP}^A_{m\text{-tt}} \Rightarrow \mathrm{FP}^A_{\mathrm{btt}} \subseteq \mathrm{FP}^A_{m\text{-tt}}$

Thus output length affects translation of equality in bounded query classes. Is there something special about decision problems or do other output lengths prevent equality from translating upward? We show that there is in fact something special about decision problems: even $\log_2 3$ bits of output are enough to make equality translate upward.

Definition 16. Let $k \geq 2$.

- $\mathrm{F}_k\mathrm{P}^A_{m\text{-tt}}$ is the set of functions from Σ^* to $\{0,\ldots,k-1\}$ computable by polynomial-time Turing machines that make m parallel queries to A.
- $\mathrm{F}_k\mathrm{P}^A_{\mathrm{btt}} = \bigcup_m \mathrm{F}_k\mathrm{P}^A_{m\text{-tt}}$

Note that $\mathrm{F}_2\mathrm{P}^A_{m\text{-tt}} = \mathrm{P}^A_{m\text{-tt}}$, etc.

Theorem 17. $\mathrm{F}_3\mathrm{P}^A_{(m+1)\text{-tt}} \subseteq \mathrm{F}_3\mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{F}_3\mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{F}_3\mathrm{P}^A_{m\text{-tt}}$.

Proof: Part (i). Define $A(x) = 2$ if $x \in A$, 1 if $x \notin A$. Let $G \in \mathrm{F}_3\mathrm{P}^A_{k\text{-tt}}$. Then there exist polynomial-time computable functions $t : \Sigma^* \to 3^{\{1,2\}^m}$ and $q_1,\ldots,q_m : \Sigma^* \to \Sigma^*$ such that for all x

$$G(x) = t(x)(A(q_1(x)),\ldots,A(q_k(x))).$$

Let $g() = t(x)()$, and write $g()$ as a polynomial over $Z/3$, *i.e.*,

$$g(x_1,\ldots,x_m) = \sum_i c_i \prod_j x_{ij}$$

where $c_1, \ldots, c_s \in Z/3$. Thus we have

$$G(x) = \sum_i c_i \prod_j A(q_{ij}(x))$$

where each q_{ij} is polynomial-time computable. A fortiori we have

$$\Gamma(x) = f(x) + \sum_i c_i \prod_j A(q_{ij}(x))$$

where $f \in \mathrm{F_3P}^A_{m\text{-tt}}$ (take $f(x)$ identically equal to 0). We complete the proof by showing that every function in the form above actually belongs to $\mathrm{F_3P}^A_{m\text{-tt}}$. It is enough to prove this for functions of the form

$$f(x) + c \prod_{1 \le j \le d} A(q_j(x))$$

because then our assertion follows by induction. We now prove the weaker assertion by induction on d. If $d = 0$, then the assertion is trivial. If $d \geq 1$ then we have

$$f(x) + c \prod_{1 \le j \le d} A(q_j(x)) = A(q_j(x)) \left[f(x)/A(q_j(x)) + c \prod_{1 \le j \le d-1} A(q_j(x)) \right].$$

$f(x)/A(q_j(x))$ is in $\mathrm{F_3P}^A_{(m+1)\text{-tt}} \subseteq \mathrm{F_3P}^A_{m\text{-tt}}$ by assumption. Thus, the bracketed expression is in $\mathrm{F_3P}^A_{m\text{-tt}}$ by the inductive hypothesis. Therefore $f(x) + c\prod_{1 \le j \le d} A(q_j(x))$ is in $\mathrm{F_3P}^A_{(m+1)\text{-tt}} \subseteq \mathrm{F_3P}^A_{m\text{-tt}}$ by assumption.

Part (ii) is similar. ∎

9. Open Questions

We proved an upper bound on the size of finite gaps in bounded query hierarchies:

$$\mathrm{P}^A_{2m\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}.$$

Can this $2m$ upper bound be improved, *i.e.*,

$$\mathrm{P}^A_{(2m-1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}$$

Does a smaller gap imply a collapse higher up, *i.e.*,

$$\mathrm{P}^A_{(m+1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \stackrel{?}{\Rightarrow} (\exists j)[\mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{j\text{-tt}}]$$

We also proved a lower bound:

$$\mathrm{P}^A_{\lfloor \frac{4}{3}m \rfloor\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \not\Rightarrow \mathrm{P}^A_{(\lfloor \frac{4}{3}m \rfloor+1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}}.$$

Can this $\lfloor \frac{4}{3}m \rfloor$ lower bound be improved, *i.e.*,

$$\mathrm{P}^A_{(\lfloor \frac{4}{3}m \rfloor-1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\lfloor \frac{4}{3}m \rfloor\text{-tt}} = \mathrm{P}^A_{m\text{-tt}}$$

In general we would like to know, when does

$$\mathrm{P}^A_{h\text{-tt}} = \mathrm{P}^A_{i\text{-tt}} \Rightarrow \mathrm{P}^A_{j\text{-tt}} = \mathrm{P}^A_{k\text{-tt}}?$$

The simplest case that is open is

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{2\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{2\text{-tt}}$$

In the polynomial-size circuits model, we know the answer to this particular question:

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{2\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{2\text{-tt}}/\mathrm{poly}.$$

Can this nonuniform upper bound be extended to larger numbers of queries? For example,

$$\mathrm{P}^A_{(m+1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{m\text{-tt}}/\mathrm{poly}.$$

Or, can the $\lfloor \frac{4}{3}m \rfloor$ lower bound be extended to nonuniform computation. In general we would like to know, when does

$$\mathrm{P}^A_{h\text{-tt}} = \mathrm{P}^A_{i\text{-tt}} \Rightarrow \mathrm{P}^A_{j\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}}/\mathrm{poly}?$$

We have shown that certain pairs of gaps imply a collapse:

$$\left(\mathrm{P}^A_{(d+1)\text{-tt}} = \mathrm{P}^A_{d\text{-tt}} \text{ and } \mathrm{P}^A_{(m+d)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}}\right) \Rightarrow \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}.$$

It would be interesting to know what combinations of gaps and separations are possible. For example, does there exist A such that

$$\mathrm{P}^A_{3\text{-tt}} = \mathrm{P}^A_{4\text{-tt}} \subset \mathrm{P}^A_{5\text{-tt}} = \mathrm{P}^A_{6\text{-tt}} \subset \mathrm{P}^A_{7\text{-tt}} \cdots ?$$

(See [3] for an oracle that makes Kintala and Fischer's β hierarchy [13] behave in that way.)

What about Chang's conjecture?

$$(\forall A \in \mathrm{NP})[\mathrm{P}^A_{(m+1)\text{-tt}} = \mathrm{P}^A_{m\text{-tt}} \stackrel{?}{\Rightarrow} \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{m\text{-tt}}]$$

Acknowledgments

We are grateful to Frank Stephan for pointing out that $\mathrm{P}^A_{2\text{-tt}} = \mathrm{P}^A_{1\text{-tt}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} = \mathrm{P}^A_{1\text{-tt}}$, to Bin Fu and Richard Chang for helping us to debug early versions of our proofs and for other helpful discussions, and to Bill Gasarch for helpful suggestions and continual encouragement.

References

[1] R. Beigel. *Query-Limited Reducibilities*. PhD thesis, Stanford University, 1987. Available as Report No. STAN-CS-88-1221.

[2] R. Beigel, W. I. Gasarch, J. T. Gill, and J. C. Owings. Terse, superterse, and verbose sets. *Inf. & Comp.*, 103:68–85, 1993.

[3] R. Beigel and J. Goldsmith. Downward separation fails catastrophically for limited nondeterminism classes. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory*, pages 134–138, 1994.

[4] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM*, 14, 1967.

[5] J. Cai, T. Gundermann, J. Hartmanis, L. A. Hemachandra, V. Sewelson, K. W. Wagner, and G. Wechsung. The Boolean hierarchy I: structural properties. *SICOMP*, 17(6):1232–1252, Dec. 1988.

[6] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[7] R. Chang, 1997. Personal communication.

[8] R. Chang, W. I. Gasarch, and C. Lund. On bounded queries and approximation. *SICOMP*, 26(1):188–209, February 1997.

[9] W. Gasarch, M. W. Krentel, and K. Rappoport. OptP-completeness as the normal behavior of NP-complete problems. *MST*, 28:487–514, 1995.

[10] W. I. Gasarch, 1996. Personal communication.

[11] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *T. AMS*, 117:285–306, 1965.

[12] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SICOMP*, 17(6):1263–1282, Dec. 1988.

[13] C. M. R. Kintala and P. C. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SICOMP*, 9(1):46–53, Feb. 1980.

[14] M. Krentel. Generalizations of OptP to the polynomial hierarchy. *Theoretical Computer Science*, 97:183–198, 1992.

[15] M. W. Krentel. The complexity of optimization problems. *JCSS*, 36(3):490–509, 1988.

[16] A. Lozano. On bounded queries to arbitrary sets, 1991. Unpublished Manuscript.

[17] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.

[18] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1987.

[19] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23:325–356, 1986.

[20] K. W. Wagner. More complicated questions about maxima and minima and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.

Appendix 1: Proof of key lemma

Definitions and notation concerning coding blocks:

- An m-tuple $r(Q)$ is called a *coding block*.
- A single element of a coding block is called a *coding variable*.
- If x is a coding variable, then the coding block that contains x is called $\vec{b}(x)$.
- Let $[\vec{b}]$ denote the set of variables in the coding block $\vec{b}$.
- All noncoding variables are called *diagonalization variables*.

Definitions and notation concerning legal assignments:

- An assignment α is called *legal* if it satisfies condition (1).
- A partial assignment α is called *legal* if there exists a legal assignment α' that extends α.
- By "legally extend α to satisfy some condition" we mean "find a total assignment $\alpha' \sqsupseteq \alpha$ such that α' is legal and α' satisfies that condition, and then let $\alpha = \alpha'$"
- By "legally extend α on Y to satisfy some condition" we mean "find a total assignment $\alpha' \sqsupseteq \alpha$ such that α' is legal, $Y \subseteq \mathrm{dom}(\alpha')$, and α' satisfies that condition, and then let $\alpha = \alpha'$"
- Let α be a legal partial assignment. We say that two formal Boolean predicates $Q_1, Q_2 \in \mathrm{Pred}_X^n$ are *congruent modulo* α (written $Q_1 \sim_\alpha Q_2$) if $Q_1 =_{\alpha'} Q_2$ for every **legal** assignment α' that extends α.
- We say that Q_1 is *congruent* to Q_2 (written $Q_1 \sim Q_2$) if $Q_1 =_\alpha Q_2$ for every legal assignment α.

Construction:

- Let $d_1, \ldots, d_{n+1}$ be any $n+1$ distinct diagonalization variables, let $\vec{d} = (d_1, \ldots, d_{n+1})$, and let $D = \{d_1, \ldots, d_{n+1}\}$.
- Let $R(d_1, \ldots, d_{n+1}) = \Gamma(x_1, \ldots, x_n)$ where $\Gamma \in 2^{2^n}$ and $x_1, \ldots, x_n \in X$.
- Let Γ' be a formal Boolean predicate such that
  - (a) $\Gamma' \sim \Gamma$,
  - (b) $\Gamma' = h_k(\cdots h_1(H, z^1_{m+1}, \ldots, z^1_n) \cdots, z^k_{m+1}, \ldots, z^k_n)$ where
    - ∗ H is a formal n-ary Boolean predicate over X
    - ∗ each h_i is an $(n-m+1)$-ary Boolean predicate
    - ∗ each $z^j_i \in X$
  - (c) H is minimal (under the partial order $\prec$) among all choices of Γ' that satisfy (a) and (b).

Since $\Gamma' \sim \Gamma$, it is sufficient to prove that $F(d_1, \ldots, d_{n+1}) \not\sim \Gamma'$. We prove that by induction on k.

For the base case we take $k = 0$, so $\Gamma' = H$. Let Z be the set of variables in H. We consider several subcases:

Base case 1: m elements of Z form a coding block. Let $\vec{b} = (b_1,\ldots,b_m)$ be that coding block. Because of the minimality of H, H is not congruent to any function of $g(\vec{b})$ and the variables in $Z - [\vec{b}]$. Therefore, there exists a legal partial assignment $\alpha : Z - [\vec{b}] \to \{0,1\}$ such that $H \neq_\alpha 0$, $H \neq_\alpha 1$, $H \neq_\alpha g(\vec{b})$ and $H \neq_\alpha \neg g(\vec{b})$. Let h be a Boolean predicate such that $h(\vec{b}) =_\alpha H$. Then $(\exists v_1, v_2 \in \{0,1\}^m)[h(v_1) \neq h(v_2)$ and $g(v_1) = g(v_2)]$. Let $c = g(v_1) = g(v_2)$.

If possible, extend α to all of X in a legal way so that $g(\vec{b}) =_\alpha c$. Then we have $F(d_1,\ldots,d_{n+1}) =_\alpha a$ for some $a \in \{0,1\}$. Now we can modify α so that $(\alpha(b_1),\ldots,\alpha(b_m)) = v_1$ or $(\alpha(b_1),\ldots,\alpha(b_m)) = v_2$, as we wish; both modifications result in legal assignments and keep $F(d_1,\ldots,d_{n+1}) =_\alpha a$. Because $h(v_1) \neq h(v_2)$, one of those modifications makes $H =_\alpha 1 - a$.

If no such extension exists, it must be the case that $g(\vec{b}) \sim_\alpha 1 - c$. Extend α to $[\vec{b}]$ in any way such that $g(\vec{b}) = 1 - c$. Now we have $H =_\alpha a$ for some $a \in \{0,1\}$. Then, since F has no minterm or maxterm of size $n - m$ or less, we can extend alpha to D so that $F(d_1,\ldots,d_{n+1}) =_\alpha 1 - a$.

Base case 2: At most $m-1$ elements of Z belong to any single coding block. For each coding block $\vec{b}$, define α on $[\vec{b}] \cap Z$ in such a way that the value of $g(\vec{b})$ is not determined; this is possible because g is not a function of $m-1$ variables or fewer. Define α arbitrarily on diagonalization variables in $Z - \{d_1,\ldots,d_{n+1}\}$. The partial assignment α defined in this way is clearly legal. In addition, $H =_\alpha h(d_1,\ldots,d_{n+1})$ where h is a function of n variables or fewer. By the first condition on F, F is not a function of n variables or fewer; therefore we can extend α to the rest of Z without forcing a value for $F(d_1,\ldots,d_{n+1})$. Then we have $H =_\alpha c$ for some $c \in \{0,1\}$; define α on $\{d_1,\ldots,d_{n+1}\}$ so that $F(d_1,\ldots,d_{n+1}) =_\alpha 1 - c$.

That completes the base case of the induction. Now assume $k \geq 1$. We write $h \equiv h'$ if the Boolean functions h and h' are identically equal. Every $(n-m+1)$-ary Boolean function h satisfies exactly one of the following three conditions:

- $h(z_m,\ldots,z_n) \equiv z_m$ or $h(z_m,\ldots,z_n) \equiv \neg z_m$
- $(\exists a_{m+1},\ldots,a_n \in \{0,1\})[h(0,a_{m+1},\ldots,a_n) = h(1,a_{m+1},\ldots,a_n)]$
- there is a nonconstant $(n-m)$-ary Boolean function $\hat{h}$ such that $h(z_m,\ldots,z_n) \equiv z_m \oplus \hat{h}(z_{m+1},\ldots,z_n)$

We take cases depending on the functions $h_1,\ldots,h_k$.

Inductive case 1: $(\exists \ell \leq k)[h_\ell(z_m,\ldots,z_n) \equiv z_m$ or $h_\ell(z_m,\ldots,z_n) \equiv \neg z_m]$. Then we are done by induction on k.

Inductive case 2: $(\exists \ell \leq k)(\exists a_{m+1},\ldots,a_n \in \{0,1\})[h_\ell(0,a_{m+1},\ldots,a_n) = h_\ell(1,a_{m+1},\ldots,a_n)]$. Choose the largest such ℓ. Set $(\alpha(z^\ell_{m+1}),\ldots,\alpha(z^\ell_n)) = (a_{m+1},\ldots,a_n)$.

$$\Gamma' \sim_\alpha \hat{h}_{\ell+1}(z^{\ell+1}_{m+1},\ldots,z^{\ell+1}_n) \oplus \cdots \oplus \hat{h}_m(z^k_{m+1},\ldots,z^k_n)$$

for some nonconstant $(n-m)$-ary Boolean functions $\hat{h}_{\ell+1},\ldots,\hat{h}_m$ (the constant $h_\ell(0,a_{m+1},\ldots,a_n)$ is absorbed into $\hat{h}_{\ell+1}$).

Choose a formal $(n-m)$-$\oplus$NF predicate $\Gamma'' \sim_\alpha \Gamma'$ such that the number of variables in Γ'' is minimum. If $\Gamma'' \equiv c$ for some $c \in \{0,1\}$, then define α on $\{d_1,\ldots,d_{n+1}\}$ so that $F(d_1,\ldots,d_{n+1}) =_\alpha 1 - c$. Otherwise, let

- $W = \{z_i^\ell : m+1 \le i \le n\} = \mathrm{dom}(\alpha)$.
- Z be the set of variables in Γ''.
- $z = \max(Z)$
- $\vec{b} = \vec{b}(z)$

We consider subcases:

Inductive case 2.1: $z \in D$**.** Then $F(d_1,\ldots,d_{n+1}) \sim_\alpha F'(d_1,\ldots,d_{n+1})$ where F' is a $|D-W|$-ary subfunction of F. Since $|D-W| \ge n+1-(n-m) = m+1$, $F' \notin (n-m)$-$\oplus$NF. But $\Gamma'' \in (n-m)$-$\oplus$NF, so $\Gamma'' \not\equiv_\alpha F'(d_1,\ldots,d_{n+1})$. Therefore we can define α on $D-W$ so that $\Gamma'' =_\alpha 1-F'(d_1,\ldots,d_{n+1})$. Since α has been defined on only $n-m$ elements of $X-D$, α can be extended to a legal assignment.

Inductive case 2.2: z **is not a coding variable and** $z \notin D$**.** We assert that α can be legally extended to $D \cup Z - \{z\}$ in such a way that $\Gamma'' \sim_\alpha z$ or $\Gamma'' \sim_\alpha \neg z$. For the sake of contradiction, suppose not. Then, for every legal extension α' of α to $D \cup Z - \{z\}$, we have $\Gamma' \sim_{\alpha'} 0$ or $\Gamma' \sim_{\alpha'} 1$. Substitute an arbitrary value for z in the formal Boolean predicate Γ'' to obtain a congruent formal Boolean predicate modulo α, on one variable fewer. This contradicts the minimality of Γ''.

So, extend α legally to $D \cup Z - \{z\}$ in such a way that $\Gamma'' \sim_\alpha z$ or $\Gamma'' \sim_\alpha \neg z$. Now we have $F(d_1,\ldots,d_{n+1}) =_\alpha c$ for some $c \in \{0,1\}$, and we can define $\alpha(z)$ so that $\Gamma'' =_\alpha 1-c$.

Inductive case 2.3: z **is a coding variable.** Let $\vec{b} = \vec{b}(z)$, and let $\{b_1,\ldots,b_p\} = \left[\vec{b}\right] - W$. Then $p \ge m-(n-m) = 2m-n$, and $g(\vec{b}) =_\alpha g'(b_1,\ldots,b_p)$ where g' is a p-ary subfunction of g. By assumption

$$(\exists a \in \{0,1\})(\forall \text{ nonconstant } h)[\text{ if } g' \oplus a \Rightarrow h \text{ then } \oplus\text{-deg}(h) > n-m].$$

Choose a accordingly. We consider sub-subcases:

Inductive case 2.3.1: for every legal extension α' **of** α **to the variables less than** $\min\left(\left[\vec{b}\right]\right)$**, we have** $g(\vec{b}) \sim_{\alpha'} 1-a \Rightarrow ((\Gamma'' =_{\alpha'} 0)$ **or** $(\Gamma'' =_{\alpha'} 1))$**.** Obtain Γ''' by substituting the value a for z in the formula Γ''. By the minimality of Γ'', we have $\Gamma''' \not\sim_\alpha \Gamma''$. Legally extend α on the variables less than or equal to z so that $\Gamma''' =_\alpha 1-\Gamma''$. Then we must have $g(\vec{b}) =_\alpha a$. Let $c \in \{0,1\}$ such that $F(d_1,\ldots,d_{n+1}) =_\alpha c$. If $\Gamma'' =_\alpha 1-c$ then we are done. Otherwise modify α by letting $\alpha(z) = a$. The resulting α is legal because g is monotone. Now we have $\Gamma'' =_\alpha 1-c$.

Inductive case 2.3.2: it is possible to legally extend α **to the variables less than** $\min\left(\left[\vec{b}\right]\right)$ **so that** $g'(b_1,\ldots,b_p) \sim_\alpha 1-a$**,** $(\Gamma'' \neq_\alpha 0)$**, and** $(\Gamma'' \neq_\alpha 1)$**.** Extend α accordingly. Then $F(d_1,\ldots,d_{n+1}) =_\alpha c$ for some $c \in \{0,1\}$.

We would like to find a legal extension $\alpha' \sqsupseteq \alpha$ such that $\Gamma'' =_{\alpha'} 1-c$. Suppose, for the sake of contradiction, that we cannot. Then $\Gamma'' =_{\alpha'} c$ for all assignments α' such that $\alpha' \sqsupseteq \alpha$ and $g'(b_1,\ldots,b_p) =_{\alpha'} 1-a$. Let $h(b_1,\ldots,b_p) =_\alpha \Gamma''$. Then $g' \oplus a \Rightarrow h \oplus (1-c)$. Since g' is a p-ary subfunction of g and $p \ge 2m-n$, $\oplus$-deg$(h) > n-m$, a contradiction. Thus the desired extension exists, and we have $\Gamma'' =_{\alpha'} 1-F(d_1,\ldots,d_{n+1})$.

Inductive case 3: for all $\ell \le k$ there is a nonconstant $(n-m)$-ary Boolean function $\hat{h}_\ell$ such that $h_\ell(z_1,\ldots,z_{n-m+1}) \equiv z_1 \oplus \hat{h}_\ell(z_2,\ldots,z_{k+1})$**.** Then

$$\Gamma' \equiv H \oplus \hat{H}_1 \oplus \cdots \oplus \hat{H}_k$$

where H is an n-ary formal Boolean predicate and $\hat{H}_1,\ldots,\hat{H}_k$ are nonconstant $(n-m)$-ary formal Boolean predicates.

Choose a formal Boolean predicate $\Gamma'' \sim_\alpha \Gamma'$ having the form given above such that the number of variables in Γ'' is minimum. If $\Gamma'' \equiv c$ for some $c \in \{0,1\}$, then define α on $\{d_1,\ldots,d_{n+1}\}$ so that $F(d_1,\ldots,d_{n+1}) =_\alpha 1-c$. Otherwise, let

- $Z =$ the set of variables in Γ'',
- $Z_H =$ the set of variables in H,
- $z = \max(Z)$.

We consider several subcases.

Inductive case 3.1: $z \in D$ Then $\Gamma'' = h_1(d_1,\ldots,d_{n+1}) \oplus h_2(d_1,\ldots,d_{n+1})$ where h_1 is an n-ary Boolean function and $\oplus$-deg$(h_2) \le n-m$. Since $F \ne h_1 \oplus h_2$, we can define α on D so that $\Gamma'' =_\alpha 1-F(d_1,\ldots,d_{n+1})$.

Inductive case 3.2: z **is not a coding variable and** $z \notin D$**.** The proof for this case is the same as in inductive case 2.2.

Inductive case 3.3: z **is a coding variable.** Let $\vec{b} = (b_1,\ldots,b_m) = \vec{b}(z)$.

We consider sub-subcases:

Inductive case 3.3.1: $\left[\vec{b}\right] \subseteq Z_H$**.** By the minimality of H, H is not congruent to any formal $(n-m)$-$\oplus$NF predicate $\oplus$ any formal Boolean predicate that depends only on $g(\vec{b})$ and the variables in $Z_H - \left[\vec{b}\right]$. Therefore it is possible to legally extend α to $Z_H - \left[\vec{b}\right]$ so that $H =_\alpha h(\vec{b})$ where h is not equal to any $(n-m)$-$\oplus$NF predicate $\oplus$ any of the following: 0, 1, $g(\vec{b})$, or $\neg g(\vec{b})$. Extend α accordingly. Extend α in some legal way to the variables less than $\min\left(\left[\vec{b}\right]\right)$. Now we have

- $F(d_1,\ldots,d_{n+1}) =_\alpha a$ for some $a \in \{0,1\}$,
- $g(\vec{b}) \sim_\alpha c$ for some $c \in \{0,1\}$, and
- $\Gamma'' =_\alpha h'(\vec{b})$ where h' is not equal to 0, 1, g, or $\neg g$.

Because of the condition above on Γ'' we can extend α to $\left[\vec{b}\right]$ so that $g(\vec{b}) =_\alpha c$ and $\Gamma'' =_\alpha 1-a$. By our choice of c, this extension is legal.

Inductive case 3.3.2: $\left[\vec{b}\right] \not\subseteq Z$ Since g is not a function of $m-1$ variables or fewer, there is a partial assignment β to $\left[\vec{b}\right] \cap Z$ such that $g(\vec{b}) \ne_\beta 0$ and $g(\vec{b}) \ne_\beta 1$. Obtain Γ''' by substituting $\beta(x)$ for x in Γ'' for each x in $\left[\vec{b}\right] \cap Z$. Γ''' contains fewer variables than Γ'' because $z \in \left[\vec{b}\right] \cap Z$. By the minimality of Γ'', $\Gamma''' \not\sim_\alpha \Gamma''$. Legally extend α to the variables less than or equal to z so that $\Gamma''' =_\alpha 1-\Gamma''$. Let $c \in \{0,1\}$ such that $F(d_1,\ldots,d_{n+1}) =_\alpha c$. If $\Gamma'' =_\alpha 1-c$ then we are done. Otherwise modify α by letting $\alpha(x) = \beta(x)$ for all $x \in \left[\vec{b}\right] \cap Z$, and then re-defining α on $\left[\vec{b}\right] \cap Z$ so that α is legal. This is possible because $g(\vec{b}) \ne_\beta 0$ and $g(\vec{b}) \ne_\beta 1$. Now we have $\Gamma'' =_\alpha 1-c$.

Inductive case 3.3.3: $\left[\vec{b}\right] \cap Z - Z_H \neq \emptyset$. We consider sub-sub-subcases:

Inductive case 3.3.3.1: every legal extension α' of α to the variables less than $\min\left(\left[\vec{b}\right]\right)$ makes $g(\vec{b}) \sim_{\alpha'} 1$, $\Gamma'' =_{\alpha'} 0$ or $\Gamma'' =_{\alpha'} 1$. Choose $(a_1,\ldots,a_m) \in \{0,1\}^m$ such that $g(a_1,\ldots,a_m) = 1$. Obtain Γ''' by substituting $a_1,\ldots,a_m$ for $b_1,\ldots,b_m$ respectively in Γ''. Γ''' has fewer variables than Γ'' because $\left[\vec{b}\right] \cap Z \neq \emptyset$. By the minimality of Γ'', $\Gamma''' \not\sim_{\alpha} \Gamma''$. Legally extend α'' to the variables less than or equal to z so that $\Gamma''' =_{\alpha} 1 - \Gamma''$. Then we must have $g(\vec{b}) =_{\alpha} 1$. Let $c \in \{0,1\}$ such that $F(d_1,\ldots,d_{n+1}) =_{\alpha} c$. If $\Gamma'' =_{\alpha} 1-c$ then we are done. Otherwise modify α by letting $(\alpha(b_1),\ldots,\alpha(b_m)) = (a_1,\ldots,a_m)$. The resulting α is legal because $g(a_1,\ldots,a_m) = 1$. Now we have $\Gamma'' =_{\alpha} 1-c$.

Inductive case 3.3.3.2: it is possible to legally extend α to the variables less than $\min\left(\left[\vec{b}\right]\right)$ so that $g(\vec{b}) \sim_{\alpha'} 0$, $\Gamma'' \neq_{\alpha'} 0$ and $\Gamma'' \neq_{\alpha'} 1$. Extend α accordingly. Then $F(d_1,\ldots,d_{n+1}) =_{\alpha} a$ for some $a \in \{0,1\}$, and $\Gamma'' =_{\alpha} h(\vec{b})$ for some m-ary Boolean predicate h. Because $\left[\vec{b}\right] \cap Z - Z_H \neq \emptyset$, $h = h_1 \oplus h_2$ where h_1 is an $(m-1)$-ary predicate and $h_2 \in (n-m)$-$\oplus$NF.

We would like to find a legal extension $\alpha' \sqsupseteq \alpha$ such that $\Gamma'' =_{\alpha'} 1-a$. Suppose, for the sake of contradiction, that we cannot. Then $h(\vec{b}) =_{\alpha'} a$ for all assignments α' such that $\alpha' \sqsupseteq \alpha$ and $g(\vec{b}) =_{\alpha'} 0$. Thus $\neg g \Rightarrow (h \oplus (1-a)))$. Therefore $\oplus$-deg$(h_2) > n-m$, a contradiction. Thus the desired extension exists, and we have $\Gamma'' =_{\alpha'} 1 - F(d_1,\ldots,d_{n+1})$. ▮

Appendix 2: Miscellaneous

Definition 18. If A is a set and f a Boolean formula, we define $A(f)$ recursively:

- $A(g \circ h) = A(g) \circ A(h)$, for all $g, h, \circ$
- $A(\neg g) = \neg A(g)$, for all g
- $A(x) = \chi_A(x)$, for all x

Definition 19. $L \leq^C_{d\text{-degree}} A$ if there exists $\circ$ such that $L \leq^C_{d\text{-}\circ\mathrm{NF}} A$. (The truth table may depend on the input, but the operation $\circ$ may not.)

Lemma 20.

i. $\mathrm{P}^A_{(d+1)\text{-tt}} \subseteq \mathrm{P}^A_{d\text{-}\circ\mathrm{NF}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{d\text{-}\circ\mathrm{NF}}$

ii. $\mathrm{P}^A_{(d+1)\text{-tt}} \subseteq \mathrm{P}^A_{d\text{-degree}} \Rightarrow \mathrm{P}^A_{\mathrm{btt}} \subseteq \mathrm{P}^A_{d\text{-degree}}$

Proof: Part (i). Assume every language in $\mathrm{P}^A_{(d+1)\text{-tt}}$ is also reducible to A via degree-d formulas over $\circ$. Consider any language L such that $L \leq^{\mathrm{P}}_{m\text{-tt}} A$. On input x, we can compute in polynomial time a Boolean formula f and m strings $q_1,\ldots,q_m$ such that $L(x) = A(f(q_1,\ldots,q_m))$. Write f in the form $(q_{11} \otimes \cdots \otimes q_{1j_1}) \circ \cdots \circ (q_{k1} \otimes \cdots \otimes q_{kj_k})$, where each $j_i \leq m$ and $\otimes$ is an associative Boolean operation ($\otimes = \vee$ if $\circ = \wedge$, and $\otimes = \wedge$ otherwise). Because $\otimes$ is associative, we can, by Lemma 12 rewrite $A(q_{i1} \otimes \cdots \otimes q_{im})$ as $A(f_i)$ where f_i has degree d over $\circ$. Thus

$$L(x) = A(f) = A(f_1 \circ \cdots \circ f_k).$$

The predicate $f_1 \circ \cdots \circ f_k$ has degree d over $\circ$.

Part (ii) is immediate from part (i). ∎

Lemma 21. *Let $m, d \geq 0$.*

i. $\mathrm{P}^A_{(m+d)\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}} \Rightarrow \mathrm{P}^A_{d\text{-degree}} \subseteq \mathrm{P}^A_{m\text{-tt}}$

ii. $\mathrm{P}^A_{(m+d)\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}} \cap \mathrm{P}^A_{d\text{-degree}} \Rightarrow \mathrm{P}^A_{\text{btt}} \subseteq \mathrm{P}^A_{m\text{-tt}} \cap \mathrm{P}^A_{d\text{-degree}}$

Proof: Part (i). Assume $\mathrm{P}^A_{(m+d)\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}}$. Let $L \leq^{\mathrm{P}}_{d\text{-degree}} A$. Then, on input x, we can compute in polynomial time a Boolean formula f and strings $q_1, \ldots, q_k$ such that $L(x) = A(f(q_1, \ldots, q_k))$ where f has degree d over some associative Boolean operation $\circ$. Let $f = f_1 \circ \cdots \circ f_s$ where each f_i involves at most d variables. Because $m \geq 0$, f_1 involves at most $m + d$ variables. Since $\mathrm{P}^A_{(m+d)\text{-tt}} \subseteq \mathrm{P}^A_{m\text{-tt}}$, f_1 can be rewritten as a predicate on m variables. The predicate $f_1 \circ f_2$ involves at most $m + d$ variables so it can be rewritten as a predicate on m variables. We continue in this way, until we have rewritten $f_1 \circ \cdots \circ f_s$ as a predicate on m variables.

Part (ii). Let $L \leq^{\mathrm{P}}_{\text{btt}} A$. By Lemma 20(ii), $L \leq^{\mathrm{P}}_{d\text{-degree}} A$. By (i), $L \leq^{\mathrm{P}}_{m\text{-tt}} A$. ∎

Theorem 11 *For all languages A and natural numbers m,*

$$\mathrm{P}^A_{(3m+2)\text{-tt}} \subseteq \mathrm{P}^A_{(2m+1)\text{-maj}} \Rightarrow \mathrm{P}^A_{\text{btt}} \subseteq \mathrm{P}^A_{(2m+1)\text{-maj}}.$$

Proof: Because $(2m+1)$-maj $\in (m+1)$-$\vee$NF, the conclusions follow from Lemma 21(ii,iv). ∎

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'