PP is Closed Under Truth-Table Reductions

Lance Fortnow*
University of Chicago

Nick Reingold†
Yale University

Abstract

Beigel, Reingold and Spielman [2] showed that PP is closed under intersection and a variety of special cases of polynomial-time truth-table closure. We extend the techniques in [2] to show PP is closed under general polynomial-time truth-table reductions. We also show that PP is closed under constant-round truth-table reductions.

1 Introduction

In the seminal paper on probabilistic computation, Gill [5] defined the class PP, the class of problems decidable by a probabilistic polynomial-time Turing machine that need only accept a string with probability at least one-half. Gill left open the question as to whether PP is closed under intersection.

Recently Beigel, Reingold and Spielman [2] showed that in fact PP is closed under intersection. They also showed PP is closed under a variety of other reductions including polynomial-time conjunctive and disjunctive reductions, bounded-depth Boolean formula reductions, $O(\log n)$ Turing reductions, threshold reductions, symmetric reductions, and multilinear reductions. However they left open the question as to whether PP is closed under general truth-table reductions.

In this paper, we extend the techniques of [2] to show that PP is closed under general polynomial-time truth-table reductions. This implies PP is closed under all the reductions listed above. We also show that PP is closed under constant-round truth-table reductions.

2 Notation and Definitions

Uppercase letters such as X and Y will denote variables ranging over inputs to a Turing machine, and lowercase letters such as x and z will denote variables ranging over the reals. We will use $|x|$ to mean the absolute value of the real number x. To avoid confusion, we will write $length(X)$ for the length of the input X. All logarithms are base two logarithms.

For a polynomial p we will use the following notation:

- $deg(p)$ is the degree of p,
- $\mathcal{M}(p)$ is the largest absolute value of any coefficient of p,
- $\mathcal{T}(p)$ is the number of distinct monomials in p.

Following [2], the definition of PP we will use is different from, but equivalent to, the one used by Gill. We will also make use of the "Gap" notation of Fenner, Fortnow, and Kurtz [4].

Definition 2.1 *For a nondeterministic Turing machine N and input X, let $Gap(N, X)$ denote the number of accepting paths of N on input X minus the number of rejecting paths of N on input X.*

*Supported by NSF grant CCR-9009936.
†Supported in part by NSF grants CCR-8808949 and CCR-8958528. Currently at AT&T Bell Laboratories.

Definition 2.2 *A function $f(X)$ is a* Gap-P function *if there is a polynomial-time bounded nondeterministic Turing machine M such that for all inputs X, $f(X) = Gap(M, X)$.*

Definition 2.3 *$L \in PP$ if and only if there is a Gap-P function f such that*

$$\begin{array}{rcl} X \in L & \Longrightarrow & f(X) > 0, \\ X \notin L & \Longrightarrow & f(X) < 0. \end{array} \tag{1}$$

If χ_L is the characteristic function of L then (1) is equivalent to

$$\chi_L(X) = \frac{1 + sign\,(f(X))}{2}.$$

Since we will be dealing with rational functions it is convenient to make the following definition.

Definition 2.4 *A* Gap-Q function *is the quotient of two Gap-P functions, where the denominator never vanishes.*

Since $p(X)q(X)$ has the same sign as $p(X)/q(X)$ we have that

Fact 2.5 *$L \in PP$ if and only if there is a Gap-Q function g such that*

$$\begin{array}{rcl} X \in L & \Longrightarrow & g(X) > 0, \\ X \notin L & \Longrightarrow & g(X) < 0. \end{array}$$

One of the fundamental techniques for proving closure properties for counting classes such as PP is that of using certain polynomials or rational functions to combine Turing machine computations. The definitions and lemma that follow are from [2]. Similar forms for polynomials have appeared elsewhere, *e.g.* [4].

Definition 2.6 *A sequence of polynomials $\{p_n(x_1, \ldots, x_k)\}_{n \geq 0}$ is* $s(n)$-uniform *if each coefficient of each p_n is an integer and $s(n)$ is a bound on the time needed to compute the degree of p_n or to compute the coefficient of any monomial in p_n. We allow k to depend on n.*

Definition 2.7 *The* degree *of a rational function is the maximum of the degrees of its numerator and denominator. A sequence $\{r_n(x_1, \ldots, x_k)\}$ of rational functions is* $s(n)$-uniform *if both the sequence of numerators and the sequence of denominators are $s(n)$-uniform.*

Lemma 2.8 *Let $N_1, \ldots, N_k$ be polynomial-time bounded nondeterministic Turing machines. Let the sequence $\{r_n(x_1, \ldots, x_k)\}$ be an $s(n)$-uniform sequence of rational functions, where the degree of r_n is d_n, and both the numerator and denominator of r_n have integer coefficients bounded in absolute value by M_n. If k, $s(n)$, d_n, and $\log M_n$ are all bounded by some polynomial in n then there exists a nondeterministic Turing machine N that runs in polynomial time such that $Gap(N, X)$ and $r_n(y_1, \ldots, y_k)$ have the same sign for all X whenever the latter is defined, where $n = length\,(X)$ and y_i is $Gap(N_i, X)$.*

3 Gap-P and Gap-Q Functions

We now state some closure properties for the class of Gap-P functions. Proofs can be found in [4].

Lemma 3.1 *Let f be a Gap-P function and q be a polynomial. Then the following functions are all in Gap-P.*

(a) $-f$.

(b) $\sum f(\langle X, Y \rangle)$,

where the sum is over all Y such that $length\,(Y) \leq q(length\,(X))$.

(c) $\prod f(\langle X, z\rangle)$,

where the product is over all z *with* $0 \leq z \leq q(length\,(X))$.

Gap-P functions are closed under (uniform) exponential-size sums by Lemma 3.1 (b). We would not expect the same to be true of Gap-Q functions, however, since the sum of exponentially many rational functions can have exponential degree. In the following lemma we show that certain exponential sums of Gap-Q functions are still Gap-Q functions, namely exponential sums given by a uniform sequence of polynomials.

Lemma 3.2 *Suppose that* $\{P_n\}_{n=0}^{\infty}$ *is an* $s(n)$*-uniform sequence of polynomials, where* P_n *has* k_n *variables,* $deg(P) = d_n$*, and* $\mathcal{M}(P) = M_n$*, such that* $s(n)$*,* d_n*,* k_n*, and* $\log M_n$ *are all polynomial in* n*. Suppose also that for* $1 \leq i \leq k_n$*,* $r(i, X)$ *is a Gap-Q function. Then the function* $g(X) = P_n(r(1, X), \ldots, r(k_n, X))$*, where* $n = length\,(X)$*, is a Gap-Q function.*

Proof Let $r(i, X) = u(i, X)/v(i, X)$ where $u(i, X)$ and $v(i, X)$ are Gap-P functions. Define

$$\begin{aligned} t(X) &= \prod_{i=1}^{k_n} v(i, X), \\ h(i, X) &= u(i, X)(t(X)/v(i, X)). \end{aligned}$$

Note that $t(X)$ and $h(i, X)$ are Gap-P functions.

Let $P_n(x_1, \ldots, x_{k_n}) = \sum_{\vec{\alpha}} c_{\vec{\alpha}} \prod_i x_i^{\alpha_i}$, where the sum is over all sequences of non-negative integers, $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_{k_n}\rangle$, with $\sigma(\vec{\alpha}) = \sum_i \alpha_i \leq d_n$. Then,

$$\begin{aligned} P_n\left(\frac{u(1, X)}{v(1, X)}, \ldots, \frac{u(k_n, X)}{v(k_n, X)}\right) &= P_n\left(\frac{h(1, X)}{t(X)}, \ldots, \frac{h(k_n, X)}{t(X)}\right) \\ &= \sum_{\vec{\alpha}} c_{\vec{\alpha}} \prod_i \left(\frac{h(i, X)}{t(X)}\right)^{\alpha_i} \\ &= \sum_{\vec{\alpha}} c_{\vec{\alpha}} \frac{\prod_i h^{\alpha_i}(i, X)}{t^{\sigma(\vec{\alpha})}(X)} \\ &= \left(\frac{1}{t^{d_n}(X)}\right) \sum_{\vec{\alpha}} c_{\vec{\alpha}} t^{d_n - \sigma(\vec{\alpha})}(X) \prod_i h^{\alpha_i}(i, X). \end{aligned}$$

Clearly $t^{d_n}(X)$ is a Gap-P function. Also, the sum is a Gap-P function, by Lemma 3.1. ∎

We will need one more fact about Gap-P functions. The following lemma on the "arithmetization" of Gap-P functions follows from the work on straight line programs of Babai and Fortnow [1] and arithmetic circuits of Venkateswaran [6]. For completeness, we give a direct proof.

Lemma 3.3 *If* f *is a Gap-P function then there exists a polynomial* $s(n)$ *and an* $s(n)$*-uniform sequence of polynomials* P_n*, such that* P_n *has* n *variables, both* $deg(P_n)$ *and* $\log \mathcal{M}(P_n)$ *are polynomials in* n*, and for all inputs* X*,*

$$f(X) = P_n(x_1, \ldots, x_n),$$

where $n = length\,(X)$ *and* x_i *is the* i*th bit of* X*.*

Proof Let N be a nondeterministic polynomial-time machine. By the proof of Cook's Theorem [3] there is a polynomial-time computatable function $g(1^n)$ that will produce a 3CNF formula $\varphi(X, Y)$ with $Y = (y_1, \ldots, y_m)$ such that for every $X \in \{0, 1\}^n$, the number of accepting paths of $N(X)$ is equal to the number of Y that make $\varphi(X, Y)$ true.

We now arithmetize the formula φ in a standard way: Replace each positive literal z with the variable z, each literal $\overline{z}$ with the expression $1-z$. For each clause of literals $z_1 \vee z_2 \vee z_3$ replace with the expression $z_1+z_2+z_3-z_1z_2-z_1z_3-z_2z_3+z_1z_2z_3$. Let $\tilde{\varphi}$ be the product of these clauses. Note that $\tilde{\varphi}$ is exactly zero or one when φ is false or true respectively. We define the arithmetic function $f_1(X)$ as

$$f_1(X) = \sum_{Y \in \{0,1\}^m} \tilde{\varphi}(X,Y).$$

Note that for every $X \in \{0,1\}^n$ we have that $f_1(X)$ is equal to the number of accepting paths of $N(X)$.

We can define a similar function $f_2(X)$ that is equal to the number of rejecting paths of $N(X)$ for every $X \in \{0,1\}^n$. Note that $f(X) = f_1(X) - f_2(X)$ has the desired properties for Lemma 3.3. ∎

4 BRS Rational Approximations

We will also need the rational approximations to the *sign* function constructed in [2]. Let $h(k)$ be the least odd integer greater than $\frac{1}{2}\log(2k+1)$. We define

$$\begin{aligned} P_n(x) &= (x-1)\prod_{i=1}^{n}(x-2^i)^2, \\ S_n^{(k)}(x) &= \frac{(P_n(-x))^{h(k)} - (P_n(x))^{h(k)}}{(P_n(-x))^{h(k)} + (P_n(x))^{h(k)}}, \end{aligned}$$

Lemma 4.1 *[2]*

(a) $deg(S_n^{(k)}) = O(n \log k)$.

(b) $\mathcal{M}(S_n^{(k)}) = 2^{O\left(n^2 \log k\right)}$.

(c) If $1 \le x \le 2^n$ *then* $1 \le S_n^{(k)}(x) < 1 + 1/k$.

(d) If $-2^n \le x \le -1$ *then* $-1 - 1/k < S_n^{(k)}(x) \le -1$.

Notice that given n and k it takes time $O\left(n^2 k \log k\right)$ to compute any coefficient of $S_n^{(k)}$. Thus the $S_n^{(k)}$ are $O\left(n^2 k \log k\right)$-uniform in the sense of Definition 2.7.

5 Truth-Table Reductions

Definition 5.1 *A language A is* polynomial-time truth-table reducible *to a language B, written $A \le_{tt}^{p} B$, if there exists a polynomial-time computable function g mapping an input X to a polynomial number of inputs $Y_1, \ldots, Y_k$ and a polynomial-time computable function f (called the* truth-table predicate*), which depends on X and which maps $\{0,1\}^k$ to $\{0,1\}$ such that $X \in A$ if and only if $f(\chi_B(Y_1), \ldots, \chi_B(Y_k)) = 1$, where $\chi_B(X)$ is the characteristic function for the language B.*

In this section we will prove that PP is closed under polynomial-time truth-table reductions. This closure property subsumes all the closure properties of [2]. We start with a few lemmas.

Lemma 5.2 *If $0 < b < a$ then* $(1+1/a)^b < \dfrac{a}{a-b}$.

Proof First of all, we have

$$b \ln(1 + 1/a) = b \ln\left(\frac{a+1}{a}\right) = b \int_a^{a+1} \frac{dt}{t} < b\left(\frac{1}{a}\right).$$

The bound on the integral follows from the fact that the integrand is bounded above by $1/a$ and the interval of integration has length 1. Similarly, we have

$$\frac{b}{a} < \int_{a-b}^{a} \frac{dt}{t} = \ln\left(\frac{a}{a-b}\right).$$

Combining the two inequalities and raising e to both sides gives the desired result. ∎

In our proof of closure under truth-table reductions, we will need to replace the *sign* function by rational approximations. We will use the next lemma to show that our approximations are suffiently close.

Lemma 5.3 *Suppose $P(x_1, \ldots, x_k)$ is a polynomial of degree d in k variables. Let $M = \mathcal{M}(P)$ and $t = \mathcal{T}(P)$. For any $\epsilon > 0$ let h be an integer that is at least $(d/2)(1 + (tM/\epsilon))$. If for all i, $1 \leq |x_i| \leq 2^l$ then*

$$\left| P\left(\frac{1+sign(x_1)}{2}, \ldots, \frac{1+sign(x_k)}{2}\right) - P\left(\frac{1+S_l^{(h)}(x_1)}{2}, \ldots, \frac{1+S_l^{(h)}(x_k)}{2}\right)\right| < \epsilon.$$

Proof Let $b_i = (1 + sign(x_i))/2$ and $c_i = (1 + S_l^{(h)}(x_i))/2$. Let $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_k \rangle$ be a sequence of non-negative integers with $\sum_i \alpha_i \leq d$. We define

$$\rho(\vec{\alpha}) = \left| \prod_i b_i^{\alpha_i} - \prod_i c_i^{\alpha_i} \right|.$$

Since P has t monomials and $\mathcal{M}(P) = M$,

$$|P(b_1, \ldots, b_k) - P(c_1, \ldots, c_k)| \leq tM \max_{\vec{\alpha}}\{\rho(\vec{\alpha})\}.$$

We complete the proof by showing that, for each $\vec{\alpha}$, $tM\rho(\vec{\alpha}) < \epsilon$. Fix $\vec{\alpha}$. For each j Lemma 4.1 and the assumption that $1 \leq |x_i| \leq 2^l$ imply that if $b_j = 0$ then $-1/2h < c_j \leq 0$, and if $b_j = 1$ then $1 \leq c_j < 1 + 1/2h$. We consider two cases:

Case I: For some j, $\alpha_j > 0$ and $x_j < 0$.

In this case $b_j = 0$. Therefore $\prod_i b_i^{\alpha_i} = 0$ and $-1/2h < c_j \leq 0$. Thus,

$$\begin{aligned}
tM\rho(\vec{\alpha}) &= tM \prod_i |c_i| \\
&< tM\left(\frac{(1+1/2h)^{d-1}}{2h}\right) \\
&< tM\left(\frac{(1+1/2h)^{d}}{2h}\right) \\
&< tM\left(\frac{1}{2h-d}\right) \qquad \text{(by Lemma 5.2)} \\
&\leq \frac{tM}{dtM(1/\epsilon)} \\
&\leq \epsilon.
\end{aligned}$$

Case II: For all j, $\alpha_j > 0$ implies $x_j > 0$.

In this case $\prod_i b_i^{\alpha_i} = 1$. Therefore,

$$\begin{aligned} tM\rho(\vec{\alpha}) &= tM\left|\prod_i c_i^{\alpha_i} - 1\right| \\ &< tM\left((1+1/2h)^d - 1\right) \\ &< tM\left(\frac{2h}{2h-d} - 1\right) \qquad \text{(by Lemma 5.2)} \\ &= tM\left(\frac{d}{2h-d}\right) \\ &\leq \frac{tMd}{dtM(1/\epsilon)} \\ &= \epsilon. \end{aligned}$$

∎

We are now ready to prove the main theorem of the section.

Theorem 5.4 *PP is closed under polynomial-time truth-table reductions.*

Proof Suppose that $B \in PP$ and $A \leq_{tt}^p B$. Let N be a polynomial-time bounded nondeterministic Turing machine, where for all inputs X

$$\chi_B(X) = \frac{1 + sign\,(Gap(N, X))}{2}. \tag{2}$$

Futhermore, let the polynomial $T(n)$ be an upper bound on the running time of N on inputs of length at most n.

Let X be an input of length n. Denote the queries made to B on input X by $Y_1, \ldots, Y_k$, the query answers by $z_1, \ldots, z_k$, where $k = n^{O(1)}$, and the truth-table predicate by f. Thus,

$$\begin{aligned} X \in A &\implies f\left(\chi_B(Y_1), \ldots, \chi_B(Y_k)\right) = 1, \\ X \notin A &\implies f\left(\chi_B(Y_1), \ldots, \chi_B(Y_k)\right) = 0. \end{aligned} \tag{3}$$

Since f maps $\{0,1\}^k$ to $\{0,1\}$, f has a unique multilinear extension $P(x_1, \ldots, x_k)$ that maps R^k to R. We can exhibit P explicitly as follows. For $y \in \{0,1\}$, define

$$h(x, y) = \begin{cases} x & y = 1, \\ 1 - x & y = 0. \end{cases}$$

Then

$$P(x_1, \ldots, x_k) = \sum_{\vec{y} \in \{0,1\}^k} f(y_1, \ldots, y_k) \prod_{i=1}^{k} h(x_i, y_i).$$

Notice that $deg(P) \leq k$, $\mathcal{M}(P) \leq 2^k$ (since each summand above contributes -1, 0, or 1 to the final coefficient of any monomial), and $\mathcal{T}(P) \leq 2^k$ (since P is multilinear).

Substituting (2) into (3) and replacing f by P gives

$$X \in A \implies P\left(\frac{1 + sign\,(Gap(N, Y_1))}{2}, \ldots, \frac{1 + sign\,(Gap(N, Y_k))}{2}\right) = 1,$$

$$X \notin A \implies P\left(\frac{1 + sign\,(Gap(N, Y_1))}{2}, \ldots, \frac{1 + sign\,(Gap(N, Y_k))}{2}\right) = 0.$$

We wish to replace *sign* by a sufficiently good approximation $S_l^{(h)}$ to *sign* and show that the resulting function is a Gap-Q function. Let

$$\begin{aligned} l &= \max_i T(length\,(Y_i)), \qquad \text{and} \\ h &= \left\lceil (k/2)(1+2^{2k+1}) \right\rceil . \end{aligned}$$

Since the length of each Y_i is bounded by a polynomial in n, both l and $\log h$ are bounded by some polynomial in n. By Lemma 2.8, $S_l^{(h)}(Gap(N,Y_i))$ is a Gap-Q functions for $1 \le i \le k$. Define

$$\begin{aligned} Q(Y_1,\ldots,Y_k) &= P\left(\frac{1+sign\,(Gap(N,Y_1))}{2},\ldots,\frac{1+sign\,(Gap(N,Y_k))}{2}\right) \\ \hat{Q}(Y_1,\ldots,Y_k) &= P\left(\frac{1+S_l^{(h)}(Gap(N,Y_1))}{2},\ldots,\frac{1+S_l^{(h)}(Gap(N,Y_k))}{2}\right) \end{aligned}$$

Lemma 5.3 shows that Q and $\hat{Q}$ differ by less than $1/2$. Also, Lemma 3.2 shows that $\hat{Q}$ is a Gap-Q function. Finally, let $G(X) = 2\hat{Q}(Y_1,\ldots,Y_k) - 1$. The function G is a Gap-Q function and

$$X \in A \Rightarrow Q(Y_1,\ldots,Y_k) = 1 \Rightarrow \hat{Q}(Y_1,\ldots,Y_k) > \frac{1}{2} \Rightarrow G(X) > 0,$$

$$X \notin A \Rightarrow Q(Y_1,\ldots,Y_k) = 0 \Rightarrow \hat{Q}(Y_1,\ldots,Y_k) < \frac{1}{2} \Rightarrow G(X) < 0. \qquad \blacksquare$$

6 Constant Round Truth-Table Reductions

Definition 6.1 *A* k-round polynomial-time truth-table reduction *is a polynomial-time Turing reduction in which the queries are divided into* k rounds *such that the queries made on each round depend only on the results of queries made on* previous *rounds (and on the input string). Let* m_r *be the number of queries made on round* r*. Each* m_r *is bounded by* $p(n)$ *where* p *is a polynomial and* n *is the length of* X*. We denote the queries made on round* r *by* $Y_1^{(r)},\ldots,Y_{m_r}^{(r)}$ *and the results of the queries by* $z_1^{(r)},\ldots,z_{m_r}^{(r)}$*. We may assume that each* $Y_i^{(r)}$ *has length exactly* $p(n)$*. We write* f_1 *for the polynomial-time computable function that produces the first-round queries, so that* $Y_i^{(1)} = f_1(X,i)$*. We write* f_r *for the polynomial-time computable function that produces the queries made on the* r*th round, so that* $Y_i^{(r)} = f_r(X, z_1^{(1)},\ldots,z_1^{(r-1)},\ldots,z_m^{(r-1)},i)$*. We write* f_{k+1} *for the polynomial-time computable boolean function that takes as input* X *and the results of all queries made and returns the final result of the reduction.*

When $k = 1$ the definition coincides with the definition of polynomial-time truth-table reduction given in Section 5. If language A is reducible to language B by a k-round polynomial-time truth-table reduction we write $A \le^p_{k\text{-}round} B$.

In this section will prove that PP is closed under k-round truth-table reductions.

Theorem 6.2 *PP is closed under k-round polynomial-time truth-table reductions.*

Proof We will prove this by induction on k. The case $k = 1$ is Theorem 5.4. Suppose that k is greater than 1, $A \le^p_{k\text{-}round} B$, and $B \in PP$. Without loss of generality, we may assume that B is complete for PP under polynomial-time m-reductions (since queries to a PP set may be replaced by queries to a complete set). On input X let the second-round queries be $Y_1^{(2)},\ldots,Y_{m_2}^{(2)}$. We will construct polynomial-time computable

functions g_i such that $\chi_B(Y_i^{(2)}) = \chi_B(g_i(X))$. Thus we can replace the second-round queries with first-round queries. This converts the k-round reduction to a $(k-1)$-round reduction, which yields the induction step.

Let $h(Z)$ be a Gap-P function such that $\chi_B(Z) = (1 + sign\,(h(Z)))/2$. By Lemma 3.3 there is a uniform sequence of polynomials P_k, with $deg(P_k)$ and $\log \mathcal{M}(P_k)$ polynomial in $k = length\,(Z)$, that takes as inputs the bits of Z and computes $h(Z)$. Let $Y_i(X, j)$ be the jth bit of $Y_i^{(2)}$. Then

$$h(Y_i^{(2)}) = P_k\left(Y_i(X,1), Y_i(X,2), \ldots, Y_i(X,p(n))\right).$$

Notice that the language $L = \{\, \langle X, j\rangle : Y_i(X,j) = 1 \,\}$ is polynomial-time truth-table reducible to B: To determine whether $\langle X, j\rangle$ is in L, use f_1, f_2, and the first-round queries of the reduction from A to B to compute $Y_i^{(2)}$ and accept if and only if the jth bit is 1. By Theorem 5.4 $L \in PP$, and so there is a Gap-P function $r_i(X, j)$ such that

$$Y_i(X,j) = \frac{1 + sign\,(r_i(X,j))}{2}.$$

Consider the two functions

$$\begin{aligned} h(Y_i^{(2)}) &= P_k\left(Y_i(X,1), Y_i(X,2), \ldots, Y_i(X,p(n))\right) \\ &= P_k\left(\frac{1+sign\,(r_i(X,1))}{2}, \ldots, \frac{1+sign\,(r_i(X,p(n)))}{2}\right) \qquad \text{and,} \\ H_i(X) &= P_k\left(\frac{1+S_l^{(h)}(r_i(X,1))}{2}, \ldots, \frac{1+S_l^{(h)}(r_i(X,p(n)))}{2}\right). \end{aligned}$$

We must choose h and l large enough that $S_l^{(h)}(r_i(X,j))$ approximates $sign\,(r_i(X,j))$ sufficiently well that $h(Y_i^{(2)})$ and $H_i(X)$ differ by less than $1/2$. We must also choose h and l small enough that $H_i(X)$ is a Gap-Q function.

By Lemma 5.3 a sufficiently good approximation can be obtained by choosing h and l so that $|r_i(X,j)| \leq 2^l$ and $h \geq (deg(P_k)/2)(1 + 2\mathcal{T}(P_k)\mathcal{M}(P_k))$. Since $deg(P_k)$ and the number of variables in P_k are both polynomial in n, $\log \mathcal{T}(P_k)$ is also polynomial in n. Furthermore, $\log |r_i(X,j)|$ is polynomial in n. Therefore we may choose h and l so that $\log h$ and l are both polynomial in n. This will ensure, by Lemma 3.2, that $H_i(X)$ is a Gap-Q function. So, there exist Gap-P functions $u_i(X)$, $v_i(X)$ such that $2H_i(X) - 1 = u_i(X)/v_i(X)$. Therefore, $h(Y_i^{(2)}) = 1$ if and only if $u_i(X)v_i(X) > 0$.

Consider the language $C_i = \{\, X : u_i(X)v_i(X) > 0 \,\}$. Since u_i and v_i are Gap-P functions, $C_i \in PP$. Since we have assumed that B is m-complete for PP, $C_i \leq_k^P B$. Therefore there is a polynomial-time computable function g_i such that $X \in C_i$ if and only if $g_i(X) \in B$. This is the desired polynomial-time computable function, since

$$\begin{aligned} g_i(X) \in B &\iff X \in C_i \\ &\iff u_i(X)v_i(X) > 0 \\ &\iff 2H_i(X) - 1 > 0 \\ &\iff P_k\left(\frac{1+sign\,(r_i(X,1))}{2}, \ldots, \frac{1+sign\,(r_i(X,p(n)))}{2}\right) > 0 \\ &\iff h(Y_i^{(2)}) = 1 \\ &\iff Y_i^{(2)} \in B. \end{aligned}$$

Thus each second-round query $Y_i^{(2)}$ can be replaced by a first round query $g_i(X)$. ∎

References

[1] L. Babai and L. Fortnow. Arithmetization: A new method in structural complexity theory. *Computational Complexity*, 1(1):41–67, 1991.

[2] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *Journal of Computing and System Sciences*, 50, 1995, to appear.

[3] S. Cook. The Complexity of Theorem Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. Association for Computing Machinery, 1971.

[4] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computing and System Sciences*, 48:116-148, 1994.

[5] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6:675–695, 1977.

[6] H. Venkateswaran. Circuit Definitions of Nondeterministic Complexity Classes. *SIAM Journal on Computing*, 21:655–670, 1992.