PP is Closed Under Intersection*

Richard Beigel† Yale University
Nick Reingold† Yale University
Daniel Spielman‡ Yale College

Abstract

In his seminal paper on probabilistic Turing machines, Gill [13] asked whether the class PP is closed under intersection and union. We give a positive answer to this question. We also show that PP is closed under a variety of polynomial-time truth-table reductions. Consequences in complexity theory include the definite collapse and (assuming P $\neq$ PP) separation of certain query hierarchies over PP.

Similar techniques allow us to combine several threshold gates into a single threshold gate. Consequences in the study of circuits include the simulation of circuits with a small number of threshold gates by circuits having only a single threshold gate at the root (perceptrons), and a lower bound on the number of threshold gates needed to compute the parity function.

1. Introduction

The class PP was defined in 1972 by John Gill [13, 14] and independently by Janos Simon [26] in 1974. PP is the class of languages accepted by a polynomial-time bounded nondeterministic Turing machine that accepts when more than half of its paths are accepting and rejects when more than half of its paths are rejecting (this definition is from [31], and is slightly different from, but equivalent to, the usual definition; see Section 2). Gill noted that PP is closed under complementation, but stated that it was not known if PP is closed under intersection and union.

Since Gill's paper, PP and related counting classes have been studied extensively by numerous researchers [2, 8, 16, 19, 25, 28, 29, 30, 31], though few closure properties have been shown for the class. In 1985 Russo [25] showed that the symmetric difference of two sets in PP is also in PP, and in 1991 Beigel, Hemachandra,

*The authors may be reached by writing to Department of Computer Science, P.O. Box 2158, New Haven, CT 06520-2158, or by sending electronic mail to lastname-firstname@cs.yale.edu.

†Supported in part by NSF grants CCR-8808949 and CCR-8958528.
‡Supported in part by NSF grant CCR-8958528 under an REU supplement.

and Wechsung [8] showed that PP is closed under polynomial-time parity reductions. Gill's question remained open, however, and it was widely conjectured that PP was not closed under intersection or union.

We prove that PP is in fact closed under intersection and union and even under polynomial-time conjunctive and disjunctive reductions. Consequently, PP is closed under polynomial-time truth-table reductions in which the truth table predicate is computed by a bounded-depth Boolean formula, and hence under polynomial-time Turing reductions that make $O(\log n)$ queries. That is, $\mathrm{P}^{\mathrm{PP}}_{O(\log n)\text{-T}} = \mathrm{PP}$. Relative to oracles, this collapse cannot be extended to a larger number of queries. For *functions* computed with a bounded number of queries the behavior is quite different: $\mathrm{PF}^{\mathrm{PP}}_{(k+1)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{k\text{-T}}$ for any oracle X unless $\mathrm{P} = \mathrm{PP}$.

Our strongest closure property is that PP is closed under polynomial-time truth-table reductions in which the truth predicate is computed by an explicitly produced multilinear polynomial (this includes all symmetric functions as a special case). The techniques presented here have been extended by Fortnow and Reingold [11] to show that PP is closed under general polynomial-time truth-table reductions.

The technique for combining PP machines can also be applied to threshold gates with polynomial-sized weights. For example, we show how to compute the AND of k threshold gates as the threshold of ANDs. We also show that any constant depth circuit with AND, OR, NOT, and threshold gates can be simulated by a circuit with a single threshold gate (at the root) with depth greater by a constant and only a limited increase in size. If the original circuit has size $2^{\mathrm{polylog}\, n}$ and only $O(\log\log n)$ threshold gates then the new circuit still has size $2^{\mathrm{polylog}\, n}$.

As an application, we prove that no constant depth circuit with $o(\log n)$ threshold gates, $2^{n^{o(1)}}$ AND, OR, and NOT gates (in arbitrary positions), and $2^{n^{o(1)}}$ wires can compute parity. This is the first natural example of a function that is known to require more than a constant number of threshold gates in such a circuit. Previous lower bounds had been obtained for circuits consisting entirely of threshold gates. Hajnal *et al.* [17] have shown that inner product mod 2 cannot be computed by any polynomial size depth-2 circuit of threshold gates. Paturi and Saks [23] have shown that a depth-2 circuit of threshold gates which computes the parity on n inputs requires $\Omega(n/\log^2 n)$ threshold gates. Siu, *et al.* [27] have shown that a depth-$(d+1)$ circuit of threshold gates which computes the parity on n inputs requires $\Omega(dn^{1/d}/\log^2 n)$ threshold gates. Recently, Beigel [6], extending our techniques, has shown that that no constant depth circuit with $n^{o(1)}$ threshold gates, $2^{n^{o(1)}}$ AND, OR, and NOT gates, and $2^{n^{o(1)}}$ wires can compute parity.

The remainder of the paper is organized as follows. In Section 2 we define $\mathrm{PrTIME}(t(n))$ and PP, and we show how to combine nondeterministic Turing machines according to a sequence of rational functions. In Section 3 we construct the rational functions appropriate for our closure properties. The closure properties are proved in Section 4. In Section 5 we show how the techniques of Section 2 can be modified to apply to threshold circuits, and in Section 6 we apply these techniques to obtain the parity lower bound mentioned above. Finally, in Section 7 we consider query hierarchies over PP.

Notation

Throughout this paper we will use X (rather than the customary x) to denote an input to a Turing machine. We will use x to denote a variable ranging over the reals. We will use $|x|$ to mean the absolute value of the real number x. To avoid confusion, we will never use $|X|$ to denote the length of the input X. All logarithms are base two logarithms.

2. Building Turing machines from rational functions

Beigel and Gill [7] and Gundermann, Nasser, and Wechsung [16] have used polynomials to prove closure properties of various counting classes. In this section we extend the techniques of [7]: where they used a single polynomial, we use a *sequence* of *rational* functions. These new twists appear to be crucial to obtaining our closure properties for PP.

Fenner, *et al.* [10] provide a convenient notation for studying counting classes like PP.

Definition 1. [10] For a nondeterministic Turing machine N and input X, let $\mathrm{Gap}(N, X)$ denote the number of accepting paths of N on input X minus the number of rejecting paths of N on input X.

Definition 2. A language L is in $\mathrm{PrTIME}(t(n))$ if there exists a $t(n)$-time bounded nondeterministic Turing machine N such that for all inputs X,

$$\begin{aligned} X \in L &\Rightarrow \mathrm{Gap}(N, X) > 0, \\ X \notin L &\Rightarrow \mathrm{Gap}(N, X) < 0. \end{aligned}$$

Gill [13] shows that $\mathrm{NTIME}(t(n)) \subseteq \mathrm{PrTIME}(t(n))$.

It should be noted that in our definition of $\mathrm{PrTIME}(t(n))$ all accepting (rejecting) paths are counted equally, regardless of length, as in [31]. Other definitions of $\mathrm{PrTIME}(t(n))$ either insist that all paths have the same length [3] or weight the paths according to length [13]. For time-constructible $t(n)$ these definitions are equivalent.

Definition 3. $\mathrm{PP} = \mathrm{PrTIME}(n^{O(1)})$.

For any nondeterministic Turing machine N, let the complement machine, denoted $\overline{N}$, be the machine which runs N and then rejects if N accepts and accepts if N rejects. Clearly, $\mathrm{Gap}(\overline{N}, X) = -\mathrm{Gap}(N, X)$. Let N_1 and N_2 be two nondeterministic Turing machines. Consider the Turing machine N_+ which nondeterministically chooses to run either N_1 or N_2. It is easy to see that for all inputs X, $\mathrm{Gap}(N_+, X) = \mathrm{Gap}(N_1, X) + \mathrm{Gap}(N_2, X)$. Let N_* be a nondeterministic Turing machine which runs as follows. First, run N_1; if N_1 accepts then run N_2, otherwise run $\overline{N_2}$. It is not hard to verify that for all X, $\mathrm{Gap}(N_*, X) = \mathrm{Gap}(N_1, X)\mathrm{Gap}(N_2, X)$. Combining these observations, we obtain Lemma 5, below.

Definition 4. A sequence of polynomials $\{p_n(x_1, \ldots, x_k)\}$ 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.

Lemma 5. *Let $N_1, \ldots, N_k$ be $t(n)$-time bounded nondeterministic Turing machines. Let $\{p_n(x_1, \ldots, x_k)\}$ be an $s(n)$-uniform sequence of polynomials. Suppose p_n has degree d_n and each coefficient of p_n is bounded in absolute value by M_n. Then there exists a nondeterministic Turing machine N that runs in time $O\left(\log\left(M_n\binom{d_n+k}{k}\right)\right) + d_n t(n) + 2s(n)$ such that for all X, $\mathrm{Gap}(N, X) = p_n(y_1, \ldots, y_k)$, where n is the length of X and y_i is $\mathrm{Gap}(N_i, X)$.*

Proof: N first nondeterministically chooses a monomial of p_n, say $\pm c{x_1}^{\alpha_1} \cdots x_k^{\alpha_k}$, with $c > 0$. Since p_n has at most $\binom{d_n+k}{k}$ monomials, this requires time $O\left(\log\binom{d_n+k}{k}\right) + s(n)$. Then N nondeterministically chooses one of c branches, computes the product as described above, and complements if necessary. This takes an additional time $O(\log(M_n)) + d_n t(n) + s(n)$. Figure 1 shows the computation tree for machine N. ∎

Note that in the definition of $\mathrm{PrTIME}(t(n))$ only the *sign* of the Gap is essential. Although we cannot directly apply Lemma 5 to a rational function (since we cannot divide), we can build a Turing machine such that the sign of the Gap is given by (the sign of) a rational function. We define the *degree* of a rational function to be 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 6. *Let $N_1, \ldots, N_k$ be $t(n)$-time bounded nondeterministic Turing machines. Let $\{r_n(x_1, \ldots, x_k)\}$ 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. Then there exists a nondeterministic Turing machine N that runs in time $O\left(\log\left(M_n\binom{d_n+k}{k}\right)\right) + 2d_n t(n) + 2s(n)$ such that $\mathrm{Gap}(N, X)$ and $r_n(y_1, \ldots, y_k)$ have the same sign for all X where the latter is defined, where n is the length of X and y_i is $\mathrm{Gap}(N_i, X)$.*

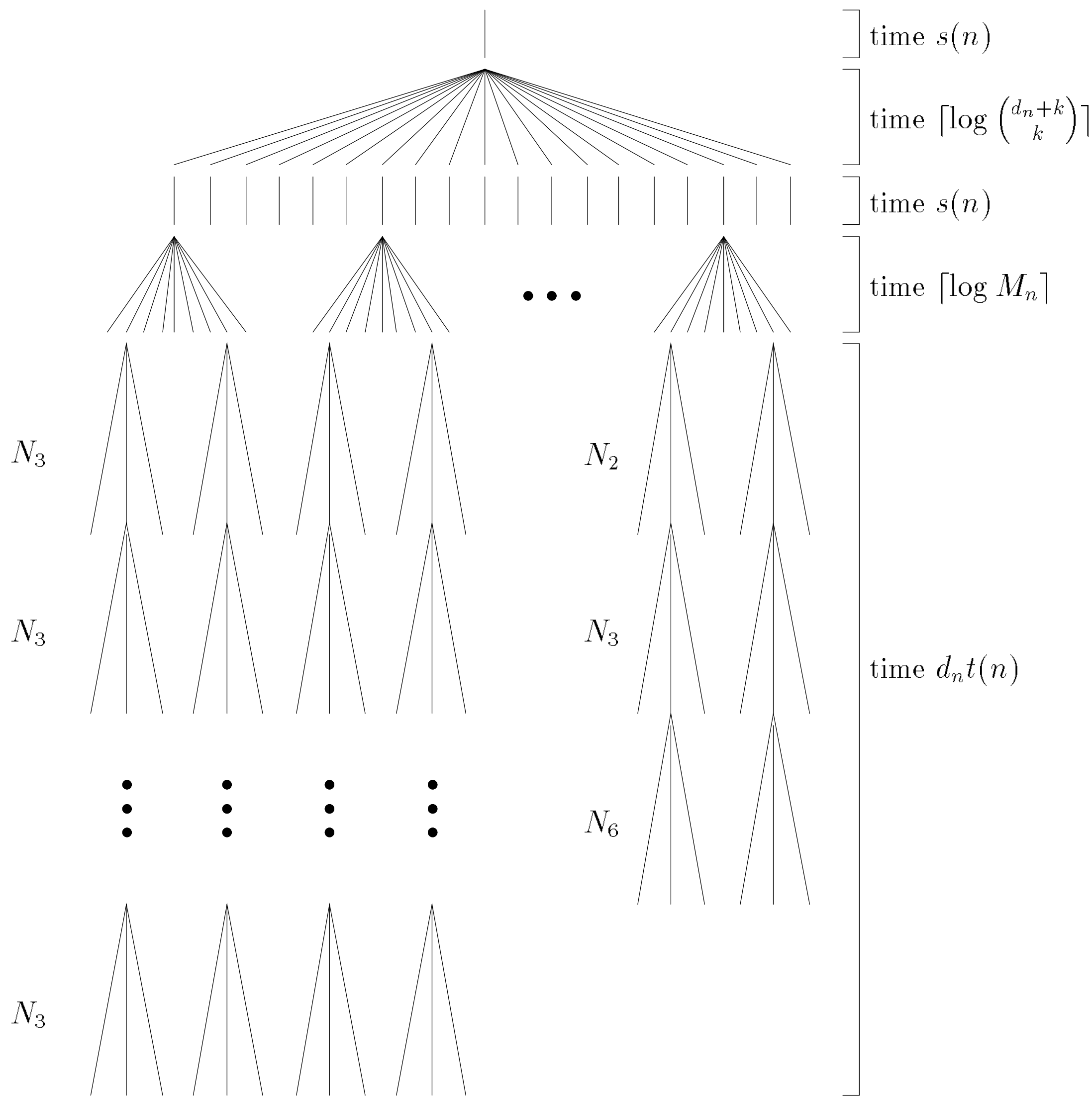

Figure 1: The computation tree for a nondeterministic Turing machine N, with $\mathrm{Gap}(N, X) = p_n(\mathrm{Gap}(N_1, X), \ldots, \mathrm{Gap}(N_k, X))$. In the first stage, N computes the degree of p_n. In the second stage, N nondeterministically chooses a monomial. In the third stage, N computes the coefficient on the monomial, say $\pm c$, for $c > 0$. In the fourth stage, N makes a c-way branch. In the last stage, N runs the machines corresponding to the monomial, complementing if necessary. In this figure, the leftmost monomial is $x_3^{d_n}$, and the rightmost monomial is $x_2 x_3 x_6$.

Proof: If $r_n = a_n/b_n$, where a_n and b_n are polynomials with integral coefficients, then apply Lemma 5 with $p_n = a_n * b_n$. ∎

3. Rational Approximation

In this section we define the rational functions, $A_n(x, y)$, which are the key to proving that PP is closed under intersection. The key property of $A_n(x, y)$ is that for $1 \leq |x|, |y| \leq 2^n$, $A_n(x, y)$ is positive if and only if both x and y are positive. In his seminal paper on rational approximation, Newman [22] constructed rational functions that closely approximate $|x|$ for $-1 \leq x \leq 1$. Our A_n's will be defined in terms of similar rational functions, $S_n(x)$, that closely approximate $sign(x)$ for $1 \leq |x| \leq 2^n$. (The function $sign(x)$ is 1 if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$.)

Define

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

The lemma below shows that $S_n(x)$ is a good approximation to $sign(x)$ for appropriate values of x. In fact, the approximation is much better than indicated here, but these estimates will suffice.

Lemma 7. *For $n \geq 1$,*

(i) If $1 \leq x \leq 2^n$ then $0 \leq 4P_n(x) < -P_n(-x)$.

(ii) If $1 \leq x \leq 2^n$ then $1 \leq S_n(x) < 5/3$.

(iii) If $-2^n \leq x \leq -1$ then $-5/3 < S_n(x) \leq -1$.

Proof:

(i) Since $x \geq 1$, we have that $P_n(x) \geq 0$. Clearly, $x - 1 < x + 1$, and $(x-2^i)^2 < (-x-2^i)^2$ for $i = 1, \ldots, n$. Also, if $2^{k-1} \leq x < 2^k$ then $4(x-2^k)^2 \leq 4 \cdot 2^{2k-2} = 2^{2k} < (-x-2^k)^2$. Together these imply that $4P_n(x) < -P_n(-x)$.

(ii) If $P_n(x) = 0$ then $S_n(x) = 1$. If $P_n(x) \neq 0$, then we can write

$$S_n(x) = 1 + \frac{2}{\left(\frac{-P_n(-x)}{P_n(x)}\right) - 1}.$$

Simple algebra and part (i) yield the desired result.

(iii) This follows from (ii) and the fact that $S_n(x)$ is an *odd* function, *i.e.*, $S_n(-x) = -S_n(x)$.

∎

Lemma 8. *Let x and y satisfy $1 \leq |x|, |y| \leq 2^n$.*

(i) The degree of A_n is $O(n)$.

(ii) Each coefficient of A_n has absolute value $2^{O(n^2)}$.

(iii) If $x > 0$ and $y > 0$ then $A_n(x,y) > 0$.

(iv) If $x < 0$ or $y < 0$ then $A_n(x,y) < 0$.

Proof: The assertions about the degree and size of the coefficients are easily verified. The last two assertions follow immediately from Lemma 7(ii–iii). ∎

We now consider analogous rational functions in many variables. Let $h(k)$ be the least odd integer greater than or equal to $\log(2k+1)$. We define

$$\begin{aligned} S_n^{(k)}(x) &= \frac{(P_n(-x))^{h(k)} - (P_n(x))^{h(k)}}{(P_n(-x))^{h(k)} + (P_n(x))^{h(k)}}, \\ A_n^{(k)}(x_1, \ldots, x_k) &= S_n^{(k)}(x_1) + \cdots + S_n^{(k)}(x_k) - (k-1). \end{aligned}$$

Lemma 9. *(i) The degree of $S_n^{(k)}$ is $O(n \log k)$.*

(ii) Each coefficient of $S_n^{(k)}$ has absolute value bounded by $2^{O(n^2 \log k)}$.

(iii) If $1 \leq x \leq 2^n$ then $1 \leq S_n^{(k)}(x) < 1 + 1/k$.

(iv) If $-2^n \leq x \leq -1$ then $-1 - 1/k < S_n^{(k)}(x) \leq -1$.

Proof: The bounds on the degree and coefficient sizes are easily verified. Suppose that $1 \leq x \leq 2^n$. By Lemma 7(i), $0 \leq (2k+1)\,(P_n(x))^{h(k)} < -\,(P_n(-x))^{h(k)}$. We now proceed as in the proof of Lemma 7(ii). Part (iv) follows from (iii) and the fact that $S_n^{(k)}(x)$ is an odd function. ∎

Lemma 10. *Let x_i satisfy $1 \leq |x_i| \leq 2^n$, for $i = 1, \ldots, k$.*

(i) The degree of $A_n^{(k)}$ is $O(nk \log k)$.

(ii) Each coefficient of $A_n^{(k)}$ has absolute value bounded by $2^{O(n^2 k \log k)}$.

(iii) If every $x_i > 0$ then $A_n^{(k)}(x_1, \ldots, x_k) > 0$.

(iv) If some $x_i < 0$ then $A_n^{(k)}(x_1, \ldots, x_k) < 0$.

Proof: The first two assertions are easily verified. The last two follow from Lemma 9 just as Lemma 8(iii–iv) follow from Lemma 7. ∎

4. Closure Properties for PP

It has been shown that PP is closed under complementation [13], under symmetric difference [25], and under polynomial-time parity reductions [8]. In this section we use the rational functions built in the previous section to prove closure under intersection and several special cases of closure under polynomial-time truth-table reductions. (The techniques of this section have been extended by Fortnow and Reingold [11], who show that PP is closed under general polynomial-time truth-table reductions.)

Theorem 11. *The intersection of finitely many* $\mathrm{PrTIME}(t(n))$ *languages is in* $\mathrm{PrTIME}(t(n)^2)$.

Proof: Let $L_1, \ldots, L_k$ be languages in $\mathrm{PrTIME}(t(n))$ and let $N_1, \ldots, N_k$ be nondeterministic Turing machines such that for $i = 1, \ldots, k$

$$\begin{aligned} X \in L_i &\Rightarrow \mathrm{Gap}(N_i, X) > 0, \\ X \notin L_i &\Rightarrow \mathrm{Gap}(N_i, X) < 0. \end{aligned}$$

Also let $r_n(x_1, \ldots, x_k) = A^{(k)}_{t(n)}(x_1, \ldots, x_k)$. By Lemma 10 the degree of r_n is $O(t(n) k \log k)$, and every coefficient of r_n is bounded in absolute value by $2^{O(t(n)^2 k \log k)}$. By Lemma 6 there is a nondeterministic Turing machine N whose running time is $O(t(n)^2)$, and such that $\mathrm{Gap}(N, X)$ and $r_n(y_1, \ldots, y_k)$ have the same sign, where n is the length of X, and, for $i = 1, \ldots, k$, $y_i = \mathrm{Gap}(N_i, X)$. Since $1 \leq |y_i| \leq 2^{t(n)}$, Lemma 10 yields that

$$\begin{aligned} X \in \bigcap_{i=1}^{k} L_i &\Rightarrow \text{every } y_i > 0 \Rightarrow r_n(y_1, \ldots, y_k) > 0 \Rightarrow \mathrm{Gap}(N, X) > 0, \\ X \notin \bigcap_{i=1}^{k} L_i &\Rightarrow \text{some } y_i < 0 \Rightarrow r_n(y_1, \ldots, y_k) < 0 \Rightarrow \mathrm{Gap}(N, X) < 0, \end{aligned}$$

so that $\bigcap_{i=1}^{k} L_i \in \mathrm{PrTIME}(t(n)^2)$. ∎

Definition 12. A polynomial-time *conjunctive reduction* is a polynomial-time truth-table reduction in which the truth-table predicate is a polynomial size conjunction (AND). A polynomial-time *disjunctive reduction* is defined similarly.

Torán [30] asked whether PP is closed under conjunctive reductions, and he noted that an affirmative answer would imply closure under $O(\log n)$-Turing reductions (defined below).

Theorem 13. *The class* PP *is closed under polynomial-time conjunctive reductions and disjunctive reductions.*

Proof: The proof is similar to the proof of Theorem 13. For polynomial-time conjunctive reductions we use the rational functions $A_n^{(k)}(x_1, \ldots, x_k)$, as before, although k now depends on n, with $k = n^{O(1)}$. The details are as in the previous proof. Closure under polynomial-time disjunctive reductions is obtained by using the rational functions $-A_n^{(k)}(-x_1, \ldots, -x_k)$. ∎

Definition 14. A polynomial-time *bounded-depth Boolean formula reduction* is a polynomial-time truth-table reduction in which the truth-table predicate is computed by a bounded-depth Boolean formula that is explicitly produced by the reduction before any queries are made.

Theorem 15. PP *is closed under polynomial-time bounded-depth Boolean formula reductions.*

Proof: An easy induction on the depth of the formula. ∎

Definition 16. A polynomial-time $f(n)$*-Turing reduction* is a polynomial-time Turing reduction that makes at most $f(n)$ queries. Let $\mathrm{P}^{A}_{f(n)\text{-T}}$ denote the class of languages polynomial-time $f(n)$-Turing reducible to a set A.

Theorem 17. PP *is closed under polynomial-time* $O(\log n)$*-Turing reductions. That is,* $\mathrm{P}^{\mathrm{PP}}_{O(\log n)\text{-T}} = \mathrm{PP}$.

Proof: Every polynomial-time $O(\log n)$-Turing reduction can be converted to a polynomial-time depth-2 Boolean formula reduction (write the reduction as a CNF or DNF formula over the query answers). ∎

It was previously known that $\mathrm{P}^{\mathrm{NP}}_{O(\log n)\text{-T}} \subseteq \mathrm{PP}$ [8, 29], and that $\mathrm{P}^{\mathrm{C_=P}}_{O(\log n)\text{-T}} \subseteq \mathrm{PP}$ [16]. (A language L is in $\mathrm{C_=P}$ if there is a nondeterministic Turing machine N such that for all inputs X, $X \in L$ if and only if $\mathrm{Gap}(N, X) = 0$.)

Definition 18. A polynomial-time *threshold reduction* is a polynomial-time truth-table reduction in which the truth-table predicate is true if and only if at least half of its inputs are true.

Definition 19. A polynomial-time *symmetric reduction* is a truth-table reduction in which the truth-table predicate is a symmetric function, i.e., a function that depends only on the number of inputs that are true.

Theorem 20. PP *is closed under polynomial-time threshold reductions.*

Proof: Define

$$T_n^{(k)}(x_1,\ldots,x_k) \;=\; 2S_n^{(2k)}(x_1)+\cdots+2S_n^{(2k)}(x_k)+1.$$

Assume that $1 \le |x_i| \le 2^n$ for $1 \le i \le k$. Then $T_n^{(k)}(x_1,\ldots,x_k)$ is a rational function that is positive if at least half of the x_i's are positive, and negative otherwise. The degree of $T_n^{(k)}$ is $O(nk\log k)$, and the absolute value of each of its coefficients is bounded by $2^{O(n^2 k \log k)}$. The result now follows from Lemma 6. ∎

Since every symmetric function can be computed as a threshold of thresholds [17], this immediately implies

Corollary 21. PP *is closed under polynomial-time symmetric reductions.*

Lemma 22. *Suppose $P(x_1,\ldots,x_k)$ is a multilinear polynomial in k variables with integer coefficients bounded in absolute value by M. Then there exists a rational function, $U_n(x_1,\ldots,x_k)$, of degree $O(kn(k+\log M))$, with coefficients bounded in absolute value by $2^{O\left(kn^2(k+\log M)\right)}$, such that for $1 \le |x_i| \le 2^n$,*

$$|P(sign\,(x_1),\ldots,sign\,(x_k)) - U_n(x_1,\ldots,x_k)| < \frac{1}{2}.$$

Proof: Let $U_n(x_1,\ldots,x_k)$ be $P(y_1,\ldots,y_k)$, where $y_i = S_n^{(h)}(x_i)$ and $h = k(1+2^{k+1}M)$. The verification of the bounds on the degree and coefficients is straightforward.

The error between corresponding monomials of $P(sign\,(x_1),\ldots,sign\,(x_k))$ and $P(y_1,\ldots,y_k)$ is at most

$$M\left|sign\,(x_1)\cdots sign\,(x_k) - S_n^{(h)}(x_1)\cdots S_n^{(h)}(x_k)\right|,$$

which, by Lemma 9, is at most $M\left((1+1/h)^k-1\right)$. Since $P(x_1,\ldots,x_k)$ has at most 2^k monomials the total error is at most $2^kM\left((1+1/h)^k-1\right)$.

Let $\exp(x)=e^x$. We will make use of the inequality $\exp(1/(1+y)) < 1+1/y$, which can be derived from the well known fact that $1+x<\exp(x)$ for $x \ne 0$ by substituting $x=-1/(y+1)$. The total error is therefore

$$\begin{aligned}
|P(sign\,(x_1),\ldots,sign\,(x_k)) - U_n(x_1,\ldots,x_k)| &\le 2^kM\left((1+1/h)^k-1\right)\\
&\le 2^kM\left(\exp(k/h)-1\right)\\
&= 2^kM\left(\exp\left(\frac{1}{1+2^{k+1}M}\right)-1\right)\\
&< 2^kM\left(1+\frac{1}{2^{k+1}M}-1\right)\\
&= \frac{1}{2}.
\end{aligned}$$

∎

Although seemingly stronger results may be stated, note that every polynomial over variables in $\{-1, 1\}$ is equal to a multilinear polynomial over those variables.

Definition 23. A polynomial-time *multilinear reduction* is a polynomial-time truth-table reduction in which the truth-table predicate is computed by a multilinear polynomial that is explicitly produced by the reduction before any queries are made.

By Lemma 22 we now have

Theorem 24. PP *is closed under multilinear reductions.*

We note that all previous proofs of containment in PP hinge on the existence of certain polynomials having fixed degree, in fact 2 or less in each variable. In our proofs we implicitly use a sequence of polynomials: For non-zero integral values of x and y within the $2^{n+1} \times 2^{n+1}$ square centered at the origin, the rational function $A_n(x, y)$ of Section 3 is positive if and only if x and y are both positive. By clearing denominators, we can construct a polynomial of degree $4n + 1$ which takes on the correct sign for such x and y.

If there were a single polynomial $P(x, y)$ with integral coefficients such that for all non-zero integers x, y, $P(x, y) > 0$ if and only if both x and y are positive, then a simpler proof of Theorem 13 could have been given. We would have no need for the rational approximations to the *sign* function, and no need for Lemma 5. Minsky and Papert [21] have shown, however, that no such $P(x, y)$ can exist. Careful analysis of their proof shows that a polynomial in two variables of degree d with each non-zero coefficient having absolute value between 1 and M cannot take on the correct sign in a $2^{n+1} \times 2^{n+1}$ square centered at the origin when $n \geq \log M + d \log d$.

The approach taken in this paper — building such a sequence of polynomials by using an approximation to the *sign* function — cannot be carried out using polynomial approximations to *sign*. By a simple application of Markov's inequality (see [24]), no sequence of polynomials $P_n(x)$, where the degree of P_n is polynomial in n, can satisfy

$$\begin{aligned} 1 \leq x \leq 2^n &\implies 1 \leq P_n(x) < 1 + \epsilon, \\ -2^n \leq x \leq -1 &\implies -1 - \epsilon < P_n(x) \leq -1, \end{aligned}$$

for a fixed $\epsilon < 1$.

Our rational function $A_n(x, y)$ has a special form, namely $Q(x) + Q(y) - 1$, where Q is a rational function. It is possible to prove [9] that any degree n polynomial $A(x, y)$ of this form satisfying the conclusions of Lemma 8 can be directly converted to a rational function S_n of degree $O(n)$ which satisfies the conditions of Lemma 7(ii–iii). That is, S_n is a good approximation to the *sign* function.

We speculate that the need for polynomials whose degree varies with the numerical range of the inputs and the need to consider rational approximations are the main reasons why the questions answered in this paper remained open until now.

5. Threshold Gates and Perceptrons

In the next two sections we will consider circuits containing threshold gates. For convenience of exposition, we will use -1 to represent *false* and 1 to represent *true* in our circuits — a slight departure from standard practice. By a *threshold gate* we mean a boolean function of n inputs $x_1, \ldots, x_n$, with weights $w_1, \ldots, w_n \in \{-1, 1\}$ and threshold t, which outputs true if $\sum_{i=1}^{n} x_i w_i > t$, and outputs false otherwise. [1] If we call a number i "accepting" if $x_i w_i = 1$ and "rejecting" if $x_i w_i = -1$, then a threshold gate with $t = 0$ acts like a PP computation, in that the gate outputs true if more i's are accepting than rejecting, and outputs false if more i's are rejecting than accepting. For this reason, our techniques for manipulating PP computations have analogues for threshold gates.

By a *perceptron* we mean a circuit with a single threshold gate at the output and with constant-depth Boolean circuits as inputs to the threshold gate. All gates have unbounded fanin. We define the *size* of a perceptron to be the number of wires in the circuit. The depth of a perceptron is defined as for general circuits. The *top fanin* of a perceptron is the fanin of the threshold gate.

The following two lemmas are analogous to the lemmas of Section 2.

Lemma 25. *For $1 \leq i \leq k$, $1 \leq j \leq f$, let $C_{i,j}$ be an AND-OR circuit of size s and depth $D-1$. Let $c_{i,j}$ denote the output of circuit $C_{i,j}$, and let $s_i = \sum_{1 \leq j \leq f} c_{i,j}$. Let $p(x_1, \ldots, x_k)$ be a polynomial of degree d, whose coefficients are integers bounded in absolute value by M. Then there exists a perceptron whose inputs are the union of the inputs to all the $C_{i,j}$, with top fanin at most $M f^d \binom{d+k}{k}$, size at most $(M + (d+1)2^{d-1}) f^d \binom{d+k}{k} + kfs$, and depth $D+2$ that outputs true if and only if $p(s_1, \ldots, s_k)$ is positive.*

Proof: Expand $p(s_1, \ldots, s_k)$ using the distributive law to obtain a sum of monomials over the $c_{i,j}$'s. There are at most $f^d \binom{d+k}{k}$ monomials, each of degree

[1] The usual definition of threshold gate has no restriction on the size of the weights; though in the definition of TC^0 (see [17]), all weights are assumed to be polynomial sized. If some input to a threshold gate had an arbitrary integral weight w, it could be replaced by $|w|$ copies of the input, each of weight 1 if w is positive or -1 if w is negative. Therefore, the techniques presented in this paper will work for polynomial bounded weights as well.

at most d. The value of a monomial with coefficient 1 is the XOR of the various $c_{i,j}$'s occurring in it.

We now build a circuit which determines whether $p(s_1, \ldots, s_k) > 0$. The circuit consists of a threshold of XOR's. We use one XOR for each monomial occurring in the expansion of p. Each XOR has the correct number of wires (weighted 1 or -1) leading to the threshold gate so that the total weight of the XOR is the coefficient of the corresponding monomial. The inputs to the XOR's are the AND-OR circuits computing the appropriate $c_{i,j}$'s. The fanin of the threshold gate is at most $Mf^d\binom{d+k}{k}$.

Each XOR can be computed by a depth-2 Boolean formula over the $c_{i,j}$'s in disjunctive normal form having size $(d+1)2^{d-1}$. If we replace each XOR by the appropriate OR of ANDs then the resulting circuit is the desired depth-$(D+2)$ perceptron. The top fanin is at most $Mf^d\binom{d+k}{k}$, as indicated above. The AND-OR circuits computing the $c_{i,j}$'s contribute kfs to the size of the perceptron. Each AND-OR circuit for an XOR contributes at most $M + (d+1)2^{d-1}$ wires. Thus the total size is at most $(M + (d+1)2^{d-1})f^d\binom{d+k}{k} + kfs$. ▮

Note. It is also possible to build a depth-$(D+1)$ perceptron which output true if and only if $p(s_1, \ldots, s_k) > 0$. Assume that true is represented by 1 and false by 0. In this case the $c_{i,j}$'s are 0 or 1, so in the expansion of p we must replace each $c_{i,j}$ by $2c_{i,j} - 1$. Call the resulting polynomial q. The degree of q is the same as the degree of p, and the coefficients of q have absolute value at most $M2^d$. Since a product of the $c_{i,j}$'s can now be computed as the AND of the $c_{i,j}$'s, the resulting perceptron has depth $D+1$. The top fanin is at most $M2^d f^d\binom{d+k}{k}$, the fanin of the AND gates is d, and the size of the perceptron is $(d+1)M2^d f^d\binom{d+k}{k} + kfs$.

Lemma 26. *For $1 \le i \le k$, $1 \le j \le f$, let $C_{i,j}$ be an AND-OR circuit having size s and depth $D-1$. Let $c_{i,j}$ denote the output of circuit $C_{i,j}$. Let $s_i = \sum_{1\le j\le f} c_{i,j}$. Let $r(x_1, \ldots, x_k)$ be a rational function of degree d, whose coefficients are integers bounded in absolute value by M. Then there exists a perceptron whose inputs are the union of the inputs to all the $C_{i,j}$, with top fanin at most $M^2 f^d\binom{d+k}{k}^2$, size at most $kfs + (M^2\binom{d+k}{k} + (d+1)2^{d-1})f^d\binom{d+k}{k}$, and depth $D+2$ such that when $r(s_1, \ldots, s_k)$ is defined the perceptron outputs true if and only if $r(s_1, \ldots, s_k)$ is positive.*

Proof: This follows from the above lemma just as Lemma 6 follows from Lemma 5. ▮

6. Applications to Threshold Circuits

By a *threshold circuit* we mean a circuit with any number of threshold, AND, OR, and NOT gates. In this section we prove some simulation results for such

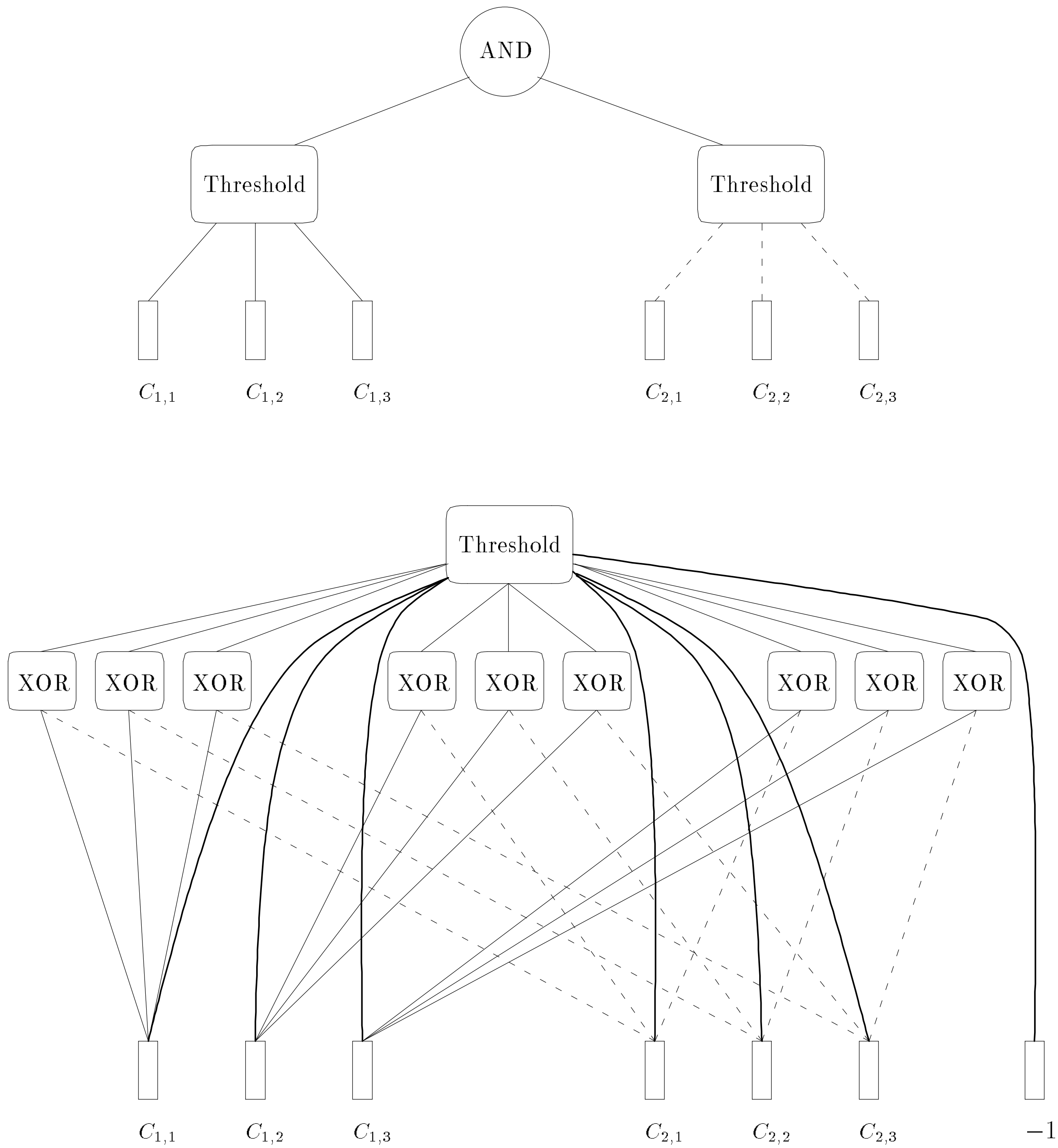

Figure 2: The AND of two thresholds as the threshold of XORs. In the bottom circuit, we have used the polynomial $x_1x_2 + 2(x_1 + x_2) - 2$, which works for this small case. The heavier lines (which go directly from the $c_{i,j}$'s or -1 to the threshold gate) represent a *pair* of wires.

circuits. We first apply the techniques of the previous section to show that the AND of k perceptrons can be computed by a single perceptron of slightly larger size and depth. Next, we show how to simulate a circuit containing many threshold gates by a perceptron of only slightly higher depth.

Siu, *et al.* [27] have shown that a depth-$(d+1)$ circuit consisting entirely of threshold gates which computes parity must have $\Omega(dn^{1/d}/\log^2 n)$ threshold gates. Combining our simulation result with Fred Green's lower bound on the fanin of a perceptron which computes parity, we show that a constant depth circuit having size $2^{n^{o(1)}}$ and only $o(\log n)$ threshold gates cannot compute parity — answering a question that arose during discussions with Russell Impagliazzo. (Recently, Beigel [6] has extended our techniques to show that no constant depth circuit of size $2^{n^{o(1)}}$ and even $n^{o(1)}$ threshold gates can compute parity.)

Theorem 27. *Consider k perceptrons having top fanin f, size s, and depth D. The AND of these k perceptrons can be computed via a perceptron having top fanin $2^{O\left(k\log k(\log f)^2\right)}$, size $2^{O\left(k\log k(\log f)^2\right)} + kfs$, and depth $D+2$.*

Proof: This follows from Lemma 26 and Lemma 10 with $n = \lceil \log f \rceil$. ▮

Corollary 28. *Consider k perceptrons having top fanin f, size s, and depth D. The OR of these k perceptrons can be computed via a perceptron having top fanin $2^{O\left(k\log k(\log f)^2\right)}$, size $2^{O\left(k\log k(\log f)^2\right)} + kfs$, and depth $D+2$.*

Lemma 29. *Consider any threshold circuit C having size s, depth D, and only k threshold gates. There is a perceptron having top fanin $2^{O\left(2^k k^3(\log k)^2(\log s)^4\right)}$, size $2^{O\left(2^k k^3(\log k)^2(\log s)^4\right)}$, and depth $D+4$ which computes the same function as C.*

Proof: Number C's threshold gates $1,\ldots,k$. Let $C(b_1,\ldots,b_k)$ be the result of replacing the ith threshold gate of C by the bit b_i for every i and then evaluating the resulting threshold-free circuit (see Figure 3). Let $A(b_1,\ldots,b_k)$ be a circuit that verifies that the result of the ith threshold gate in C is b_i for every i, using the parameters to A as the output values for any threshold gates below gate i (see Figure 5(a)). The output of C can be computed by taking the OR over all 2^k sequences $b_1,\ldots,b_k$ of the AND of $A(b_1,\ldots,b_k)$ and $C(b_1,\ldots,b_k)$ (see Figure 4).

Negations can be pushed to the leaves, so $A(b_1,\ldots,b_k)$ can be evaluated as the AND of k perceptrons. Since each of these perceptrons has fanin bounded by s, Theorem 27 implies that $A(b_1,\ldots,b_k)$ can be computed by a perceptron having top fanin $2^{O\left(k\log k(\log s)^2\right)}$, size $2^{O\left(k\log k(\log s)^2\right)} + ks^2$, and depth $D+2$ (see Figure 5(b)).

The AND of $C(b_1,\ldots,b_k)$ and the perceptron which computes $A(b_1,\ldots,b_k)$ can be computed by a perceptron having top fanin $2^{O\left(k\log k(\log s)^2\right)}$, size $2^{O\left(k\log k(\log s)^2\right)} + (k+1)s^2$, and depth $D+2$, since $C(b_1,\ldots,b_k)$ does not involve any threshold

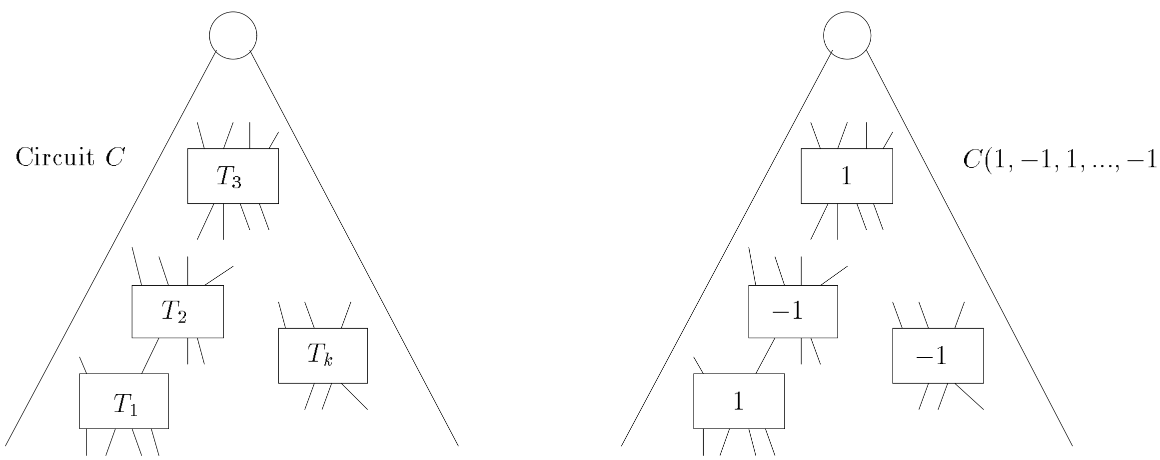

Figure 3: The circuit $C(b_1,\ldots,b_k)$ is obtained from the circuit C by replacing each threshold gate T_i by the bit b_i.

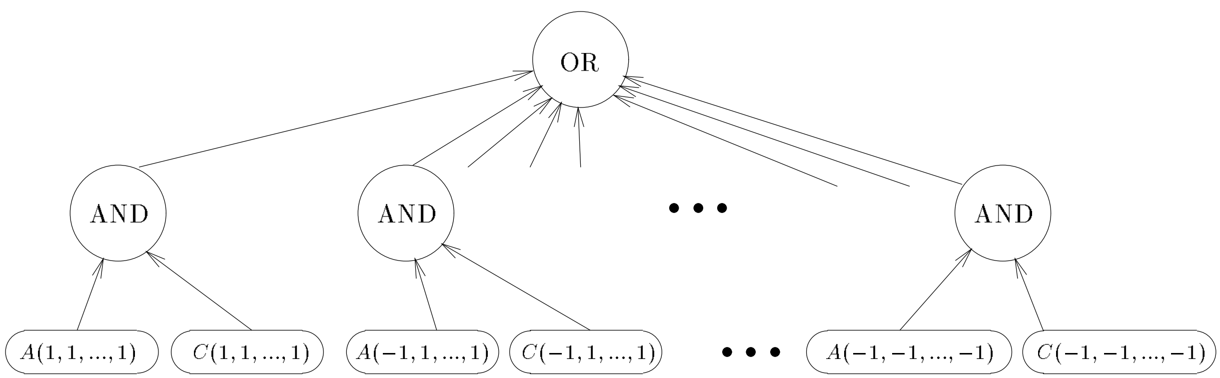

Figure 4: The circuit C can be computed as the OR over all bit sequences $b_1, \ldots, b_k$ of the AND of $A(b_1, \ldots, b_k)$ and $C(b_1, \ldots, b_k)$.

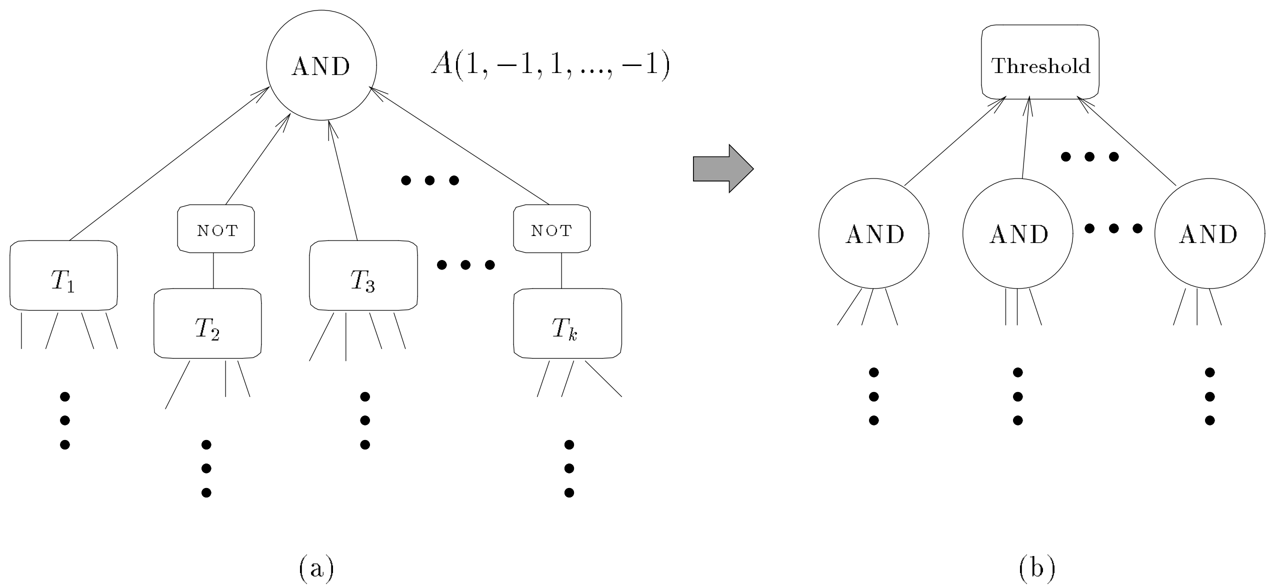

Figure 5: The circuit $A(b_1, \ldots, b_k)$. (a) $A(b_1, \ldots, b_k)$ outputs true if threshold gate T_i outputs b_i, for $i = 1, \ldots, k$. The inputs to each T_i are AND-OR circuits. (b) $A(b_1, \ldots, b_k)$ is converted to a perceptron by pushing any negations past the threshold gates and then applying Theorem 27.

gates. This can be converted, by Corollary 28, to a perceptron with top fanin $2^{O\left(2^k k^3 (\log k)^2 (\log s)^4\right)}$, size $2^{O\left(2^k k^3 (\log k)^2 (\log s)^4\right)} + 2^k (k+1)s$, and depth $D+4$. ∎

For circuits of size $2^{\operatorname{polylog} n}$, our best result is for $O(\log\log n)$ threshold gates.

Corollary 30. *Consider any threshold circuit C having size $2^{\operatorname{polylog} n}$, depth D, and $O(\log\log n)$ threshold gates. There is a perceptron having top fanin $2^{\operatorname{polylog} n}$, size $2^{\operatorname{polylog} n}$, and depth $D+4$ which computes the same function as C.* ∎

To obtain our lower bound for the number of threshold gates in a threshold circuit which computes parity, we will make use of the following theorem of Fred Green.

Theorem (F. Green[15]). *For any $D > 2$ there exists a constant c such that the following is true. Consider any perceptron with top fanin f, with depth $D+1$ and with subcircuits each of size $2^{n^{1/D^2}}$. If the circuit computes parity of n variables correctly, then*

$$f \geq 2^{cn^{(D+1)/D^2}-1}.$$

Theorem 31. *Let C be a threshold circuit having size $2^{n^{o(1)}}$, depth $O(1)$, and only $o(\log n)$ threshold gates. Then C does not compute the parity function of n inputs.*

Proof: By Lemma 29, C can be simulated by a perceptron having top fanin $2^{n^{o(1)}}$, size $2^{n^{o(1)}}$, and depth $O(1)$. By F. Green's theorem, such a perceptron cannot compute parity of n inputs. ∎

A *multilinear gate* evaluates a multilinear polynomial of its inputs (represented as -1 and 1) and outputs true if and only if the result is positive. Note, by Lagrange's interpolation formula, that every symmetric gate is a multilinear gate. Lemmas 22 and 25 yield the following simulation result:

Theorem 32. *If g is computed by a depth-D circuit with a multilinear gate at the output,* $\operatorname{polylog} n$ *threshold gates at the next level, $2^{\operatorname{polylog} n}$ AND, OR, and NOT gates at the remaining levels, and $2^{\operatorname{polylog} n}$ wires then g is computed by a perceptron that has size $2^{\operatorname{polylog} n}$ and depth $D+2$.*

7. Query Hierarchies over PP

We have shown that $O(\log n)$ queries to a PP oracle are no better than one when solving decision problems. In this section we present two results that contrast with that one. First, in relativized worlds, we show that if $f(n) \neq O(\log n)$ then $f(n)$ queries to a PP oracle are better than one when solving decision problems.

Here, and for the remainder of the section, f denotes a polynomial-time computable function. Second, if $\mathrm{P} \neq \mathrm{PP}$ we show that $k+1$ queries to a PP oracle are better than k when computing functions.

The class of languages polynomial-time reducible to a set A with at most $f(n)$ adaptive queries is denoted $\mathrm{P}^{A}_{f(n)\text{-T}}$. The class of languages polynomial-time reducible to a set A with at most $f(n)$ nonadaptive queries is denoted $\mathrm{P}^{A}_{f(n)\text{-tt}}$. The analogous classes of functions are $\mathrm{PF}^{A}_{f(n)\text{-T}}$ and $\mathrm{PF}^{A}_{f(n)\text{-tt}}$. By varying the bound f, we obtain the query hierarchies over A.

In Section 4 we showed that $\mathrm{P}^{\mathrm{PP}}_{O(\log n)\text{-T}} = \mathrm{PP}$. We wonder whether that result is tight or whether $\mathrm{P}^{\mathrm{PP}}_{f(n)\text{-T}} = \mathrm{PP}$ for some function $f(n) \neq O(\log n)$. Since our proof techniques are valid for computation relative to an oracle, we turn to relativized complexity for insight. There is an oracle A for which $\mathrm{P}^{\mathrm{NP}^A}_{f(n)\text{-T}} \subseteq \mathrm{PP}^A$ if and only if $f(n) = O(\log n)$ [5]. Since $\mathrm{NP}^B \subseteq \mathrm{PP}^B$ for all B, with that same A we have

$$\mathrm{P}^{\mathrm{PP}^A}_{f(n)\text{-T}} = \mathrm{PP}^A \iff f(n) = O(\log n).$$

This is circumstantial evidence that our collapse is the best possible, which is surprising, because we have come to expect that collapses translate upward. (For example, if $\mathrm{P}^{\mathrm{NP}}_{2\text{-T}} = \mathrm{NP}$ then $\mathrm{PH} = \mathrm{NP}$ [20]. On the other hand, it is open whether $\mathrm{P}^{\mathrm{NP}}_{O(\log n)\text{-T}} = \mathrm{P}^{\mathrm{NP}}_{3\text{-T}} \Rightarrow \mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}_{3\text{-T}}$.)

Similar results hold relative to random oracles. It was shown in [2] that $\mathrm{PARITYP}^R \not\subseteq \mathrm{PP}^R$ for almost all R. Since $\mathrm{PARITYP}^B \subseteq \mathrm{P}^{\mathrm{PP}^B}$ for all oracles B, it follows that $\mathrm{P}^{\mathrm{PP}^R} \neq \mathrm{PP}^R$ for almost all R. In fact, from the tight lower bounds of [2] for approximating parity, it follows that if $f(n) \neq O(\log n)$ then $\mathrm{P}^{\mathrm{PP}^R}_{f(n)\text{-T}} \neq \mathrm{PP}^R$ for almost R. Since the proof of Theorem 17 relativizes, we conclude that $\mathrm{P}^{\mathrm{PP}^R}_{f(n)\text{-T}} = \mathrm{PP}^R$ for almost all R if and only if $f(n) = O(\log n)$.

By combining the results of this paper with some previously known lower bounds we can prove relativized separations all the way up the query hierarchies.

Theorem 33. *Assume that* $\log n \leq f(n) = n^{O(1)}$ *and* $g(n) \neq O(f(n))$. *Then there is an oracle* A *such that*

$$\mathrm{P}^{\mathrm{NP}^A}_{f(n)\text{-T}} \not\subseteq \mathrm{P}^{\mathrm{PP}^A}_{g(n)\text{-T}}.$$

Proof: We take the following definitions from [5].

- $\mathrm{CHUNK}(A, n, f)$ is the set containing the lexicographically first $f(n)$ strings, starting from 0^n, that belong to A.
- $\max(B)$ is the lexicographically greatest element of B, if it exists; the empty string, otherwise.
- $\text{ODD-MAX-ELEMENT}^A_f$ is $\{0^n : \max(\mathrm{CHUNK}(A, n, f)) \text{ ends in a } 1\}$.
- ODD-MAX-BIT is the set of all strings over $\{0,1\}^*$ whose rightmost 1 is in an odd-numbered position, i.e., the set of strings of the form $x10^k$ where the length of x is even.

Note that ODD-MAX-ELEMENT$^{A}_{2^{f(n)}-1}$ belongs to $\mathrm{P}^{\mathrm{NP}^A}_{f(n)\text{-T}}$ via a binary-search algorithm.

Suppose that for every oracle A, ODD-MAX-ELEMENT$^{A}_{2^{f(n)}-1}$ belongs to $\mathrm{P}^{\mathrm{PP}^A}_{g(n)\text{-T}}$. Then by standard diagonalization techniques of [12], $(2^{f(n)}-1)$-bit instances of ODD-MAX-BIT can be decided by depth-3 stratified circuits having the following form: the root is an OR-gate having fanin $2^{g(n)}$; the second level consists of AND-gates having fanin $g(n)$; the bottom level consists of threshold-gates having fanin $n^{O(1)}$.

If such a circuit is converted to a perceptron by the techniques of the note following Lemma 25, the resulting circuit has the following form: the root is a threshold gate having fanin $2^{n^{O(1)}2^{O(g(n))}}$; the remaining level consists of AND-gates having fanin $n^{O(1)}2^{O(g(n))}$. However, it was shown in [5] that such a circuit deciding m-bit instances of ODD-MAX-BIT requires AND-gates with fanin $m^{\Omega(1)}$. Therefore

$$n^{O(1)}2^{O(g(n))} = 2^{\Omega(f(n))}.$$

In particular if $f(n) \geq \log n$, then $g(n) = \Omega(f(n))$. ∎

Combining the parity lower bounds of [2] with our work one can prove a separation relative to almost all oracles.

Theorem 34. *Assume that* $\log n \leq f(n) = n^{O(1)}$ *and* $g(n) \neq O(f(n))$. *Then for almost all oracles R*

$$\mathrm{P}^{\mathrm{PP}^R}_{f(n)\text{-T}} \not\subseteq \mathrm{P}^{\mathrm{PP}^R}_{g(n)\text{-T}}.$$

The behavior of the query hierarchies over PP is quite different when it comes to functions.

Theorem 35. *If* $\mathrm{P} \neq \mathrm{PP}$ *then* $(\forall X)[\mathrm{PF}^{\mathrm{PP}}_{(k+1)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{k\text{-T}}]$.

Proof: It is known that if A is a self-reducible set that is not in P and if there exists i such that for all $j > i$ we have $\mathrm{P}^{A}_{j\text{-T}} = \mathrm{P}^{A}_{i\text{-T}}$, then A must be p-superterse, i.e., for all k and all sets X we must have $\mathrm{PF}^{A}_{(k+1)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{k\text{-T}}$ [4]. Let A be one of the standard $\leq^p_m$-complete sets for PP, which are known to be self-reducible. Then $\mathrm{P}^{A}_{j\text{-T}} = \mathrm{PP} = \mathrm{P}^{A}_{1\text{-T}}$, so $\mathrm{PF}^{A}_{(k+1)\text{-tt}}$ must not be contained in $\mathrm{PF}^{X}_{k\text{-T}}$ for any k and X unless $\mathrm{P} = \mathrm{PP}$. ∎

In particular $\mathrm{PF}^{\mathrm{PP}}_{(k+1)\text{-T}} \neq \mathrm{PF}^{\mathrm{PP}}_{k\text{-T}}$. This is the opposite of what happens to the query hierarchy for decision problems.

Higher levels in the query hierarchy of functions over PP are distinct, assuming that $\mathrm{PP} \not\subseteq \mathrm{PH/poly}$. The following results are immediate from [1].

- Let f be a nondecreasing function such that $f(n) = O(\log n)$. If $\mathrm{PP} \not\subseteq \mathrm{NP/poly} \cap \text{co-NP/poly}$ then

$$(\forall X)[\mathrm{PF}^{\mathrm{PP}}_{f(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(f(n)-1)\text{-T}}].$$

- Let ϵ be a positive real number, and let g be a nondecreasing function such that $g(n) \leq n^{1-\epsilon}$. If PP $\not\subseteq \Delta_3^p$/poly then

$$(\forall X)[\mathrm{PF}^{\mathrm{PP}}_{g(n)\text{-tt}} \not\subseteq \mathrm{PF}^{X}_{(g(n)-1)\text{-T}}].$$

Note that by Toda's theorem [28] and the Karp-Lipton theorem [18], each of the hypotheses above follows from the assumption that the polynomial hierarchy does not collapse.

8. Concluding Remarks

Paturi and Saks [23] have also used rational approximations in their study of threshold circuits. We are grateful to them for sharing with us a preprint of their paper, in which we discovered Newman's theorem. That theorem was the inspiration for our proofs that PP is closed under symmetric reductions and under multilinear reductions (and the corresponding circuit simulations).

Our interest in the current research topic was inspired by discussions with Fred Green of the query hierarchies over PP. The idea of looking at multivariate polynomials germinated during a visit from Gerd Wechsung.

We would like to thank the following people for helpful comments, advice, and suggestions: Mike Fischer, Lance Fortnow, Bill Gasarch, John Gill, Fred Green, Johan Håstad, Russell Impagliazzo, Simon Kasif, Rao Kosaraju, Ramamohan Paturi, Steven Rudich, Roman Smolensky, Martin Tompa, Gerd Wechsung, and Andy Yao.

We thank Bill Gasarch for proofreading beyond the call of duty. We also thank the referees for their very helpful comments.

References

[1] Amihood Amir, Richard Beigel, and William I. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proceedings of the 5th Annual Conference on Structure in Complexity Theory*, pages 232–243, 1990.

[2] Jim Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 402–409, 1991.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, New York, 1988.

[4] Richard Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84:199–223, 1991.

[5] Richard Beigel. Polynomial interpolation, threshold circuits, and the polynomial hierarchy. YALEU/DCS/TR 843, Yale University, Dept. of Computer Science, January 1991.

[6] Richard Beigel. When do extra majority gates help? Polylog(n) majority gates are equivalent to one. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 1992. To appear.

[7] Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 1992. To appear.

[8] Richard Beigel, Lane A. Hemachandra, and Gerd Wechsung. Probabilistic polynomial time is closed under parity reductions. *Information Processing Letters*, 37:91–94, 1991.

[9] Richard Beigel, Nick Reingold, and Daniel Spielman. Sign approximators. Manuscript.

[10] Steven Fenner, Lance Fortnow, and Stuart Kurtz. Gap-definable counting classes. In *Proc. of the 6th Annual Conference on Structure in Complexity Theory*, pages 30–42. IEEE Computer Society Press, July 1991.

[11] Lance Fortnow and Nick Reingold. PP is closed under truth-table reductions. In *Proc. of the 6th Annual Conference on Structure in Complexity Theory*, pages 13–15. IEEE Computer Society Press, July 1991.

[12] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

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

[14] John T. Gill III. *Probabilistic Turing machines and complexity of computation.* PhD thesis, Department of Mathematics, University of California, Berkeley, Berkeley, California, 1972.

[15] Fred Green. An oracle separating ParityP from PP(PH). *Information Processing Letters*, 37:149–153, 1991.

[16] T. Gundermann, N. Nasser, and G. Wechsung. A survey of counting classes. In *Proceedings of the 5th Annual Conference on Structure in Complexity Theory*, pages 140–153. IEEE Computer Society Press, July 1990.

[17] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences.* To appear.

[18] R. M. Karp and R. J. Lipton. Turing machines that take advice. *L'Enseignement Mathematique*, 28:191–209, 1982.

[19] Johannes Köbler, Uwe Schöning, Seinosuke Toda, and Jacobo Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44, 1992. To appear.

[20] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[21] Marvin L. Minsky and Seymour A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1988. Expanded Edition. The first edition appeared in 1968.

[22] D. J. Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11:11–14, 1964.

[23] Ramamohan Paturi and Michael E. Saks. On threshold circuits for parity. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 397–404, 1990.

[24] Theodore J. Rivlin. *An Introduction to the Approximation of Functions*. Dover Publications, Inc., 1969.

[25] D. A. Russo. *Structural Properties of Complexity Classes*. PhD thesis, University of California at Santa Barbara, March 1985.

[26] Janos Simon. *On Some Central Problems In Computational Complexity*. PhD thesis, Cornell University, Ithaca, New York, 1975. Dept. of Computer Science, TR 75-224.

[27] Kai-Yeung Siu, Vwani Roychowdhury, and Thomas Kailath. Computing with almost optimal size threshold circuits. Manuscript, 1990. A preliminary version appeared as a technical report in the Information Systems Laboratory, Stanford University, Stanford, CA 94305.

[28] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[29] Seinosuke Toda. Restricted relativizations of probabilistic polynomial time. *Theoretical Computer Science*, 1991. To appear.

[30] Jacobo Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):753–774, July 1991.

[31] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.