Abstract

Applications of Polynomial and Rational Approximation to Complexity Theory

Nicholas Frederick Reingold
Yale University
1992

A language is in PP if there is a polynomial-time bounded probabilistic Turing machine that correctly determines whether a string is in the language with probability greater than 1/2. In his seminal paper on probabilistic Turing machines, John Gill asked whether PP was closed under intersection. Using ideas from rational approximation theory, we prove that it is. Extending the basic technique, we also prove closure under polynomial-time truth-table reductions and even constant-round truth-table reductions.

A modification of these techniques proves that the AND of several threshold gates can be computed by a single threshold gate whose inputs are ANDs of some of the original inputs. We also prove that any circuit containing boolean gates and few threshold gates can be simulated by a circuit with a single threshold gate at the root (although the depth increases by a constant, and the size increases doubly exponentially in the number of threshold gates). Using this and some known results on computing parity by a perceptron, we prove a lower bound on the number of threshold gates needed to compute parity in a circuit containing both boolean gates and threshold gates.

We also prove that the OR function can be simulated by a low-degree probabilistic polynomial. From this we show that any function computed by a family of bounded-depth, polynomial-size AND-OR circuits can be computed by a low-degree probabilistic polynomial. We also use this result to give a new proof that constant-depth boolean circuits computing parity require exponential size.

Applications of Polynomial and Rational Approximation to Complexity Theory

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Nicholas Frederick Reingold
December 1992

Acknowledgements

I wish to thank my advisor, Richard Beigel, who supported, encouraged, and collaborated in the work presented here. Dan Spielman and Lance Fortnow were also co-authors on much of the present material. Working with them was a pleasure and a privilege.

The Theory Group at Yale University provided me with an excellent environment in which to do research. I especially wish to thank Jeff Westbrook and Lenore Zuck. My collaborations with them, and advice and friendship from them, made my time at Yale richer.

Thanks to the friends I made while at Yale, especially Dan Rabin, for all the references; Rebecca Wright, for sharing such a small office with me; and Michael Black and Lee Millward, for occasionally letting me make dessert.

Special thanks to Joan Feigenbaum and Bill Gasarch for help and advice, both technical and otherwise.

Thanks to pre-Yale friends who never gave up on me: David Gillman, Dan Goldstein, Tom Hunter, David Grant (Rosenberg), Lenny Cassuto (Rosenberg), Debra Osofsky, and martica campos (*not* a Rosenberg).

The research in this dissertation was supported in part by National Science Foundation grants CCR-8808949 and CCR-8958528.

Contents

List of Figures

Chapter 1

Introduction

1.1 Summary of Thesis

In this dissertation we present several applications of polynomial and rational approximation to complexity theory. The applications fall into three main categories: closure properties for the class PP, combining threshold gates, and approximation by low-degree probabilistic polynomials.

Closure Properties for PP

The class PP was defined in 1972 by John Gill [19, 20] and independently in 1974 by Janos Simon [37]. 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 [46] and is slightly different from, but equivalent to, the usual definition; see Section 2.1). Gill noted that PP is closed under complementation but stated that it was not known whether PP is closed under intersection and union.

Since Gill's paper, PP and related counting classes have been studied extensively by numerous researchers [3,9,23,26,36,41,42,44,46], although few closure properties have been shown for the class. In 1985 Russo [36] showed that the symmetric difference of two sets in PP is also in PP, and in 1991 Beigel, Hemachandra, and Wechsung [9] 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.

All previous closure properties for PP were proved by building certain polynomials and then combining Turing machine computations according to those polynomials. In Chapter 2 we build certain rational approximations to the *sign* function, and use them in a similar way to 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. These results were joint work with Richard Beigel and Daniel Spielman and appeared in [11].

In Sections 2.6 and 2.7 we extend the basic technique and prove that PP is closed under truth-table and constant round truth-table reductions. These results are joint work with Lance Fortnow and appeared in [17].

The Mathematics of Rational Approximations

In Chapter 3 we examine rational approximations to *sign* in more detail. The results of this chapter are primarily of mathematical interest and have no direct applications to complexity theory. However, they do explain some of the motivation behind the construction of the rational approximations constructed in Chapter 2.

Combining Threshold Gates

The technique for combining PP machines can be modified to apply to threshold gates with polynomial-size weights. We explore this in Chapter 4. In particular, we show how to compute the AND of k threshold gates by a 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. For example, if the original circuit has size $2^{\text{polylog}\, n}$ and only $O(\log\log n)$ threshold gates then the new circuit still has size $2^{\text{polylog}\, n}$.

As an application, we prove that no constant-depth circuit with $2^{n^{o(1)}}$ AND, OR, and NOT gates, $o(\log n)$ threshold gates (in arbitrary positions), and $2^{n^{o(1)}}$ wires can compute the parity of n inputs. 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.* [24] have shown that inner product mod 2 cannot be computed by any polynomial-size depth-2 circuit of threshold gates. Paturi and Saks [32] have shown that a depth-2 circuit of threshold gates that computes the parity of n inputs requires $\Omega(n/\log^2 n)$ threshold gates. Siu, *et al.* [38] have shown that a depth-$(d+1)$ circuit of threshold gates which computes the parity of n inputs requires $\Omega(dn^{1/d}/\log^2 n)$ threshold gates. Recently, Beigel [7], extending our techniques, has shown that no constant-depth circuit with $2^{n^{o(1)}}$ AND, OR, and NOT gates, $n^{o(1)}$ threshold gates, and $2^{n^{o(1)}}$ wires can compute parity.

The results of Chapter 4 appeared in [11].

Probabilistic Polynomials and Perceptrons

AC^0 is the class of predicates computed by polynomial-size circuits consisting of unbounded-fanin AND and OR gates, and having depth bounded by some constant. It has been the subject of much recent research.

Razborov [33] and Smolensky [39] showed that every AC^0 predicate can be approximated by a low-degree polynomial over any finite field. Their upper bounds yield circuit lower bounds related to Haståd's lower bound on computing parity [25]. Applying Haståd's lower bound for parity, Linial, Mansour, and Nisan [27] proved an important upper bound: Every AC^0 predicate can be approximated by a low-degree polynomial over the reals. Their upper bound implies an efficient learning algorithm for AC^0 predicates, under the uniform distribution.

In Chapter 5 we define probabilistic polynomials and show that every AC^0 predicate can be computed *exactly* by a low-degree *probabilistic* polynomial over the reals. We believe this may be a step towards distribution-free learning of AC^0 predicates, an important open problem in computational learning theory.

Perceptrons, depth-2 threshold circuits consisting of a single threshold gate at the root with AND gates at the next level, have been studied since the 1950s (both [28] and [12] contain extensive bibliographies). Although perceptrons were proposed as a powerful component of vision systems, Minsky and Papert [28] proved various limitations on their computational power. Interest in the perceptron as a model of computation dwindled considerably. Subsequently, however, the success of neural nets, which are networks of perceptrons, generated circumstantial evidence in favor of the perceptron. Even individual perceptrons are sometimes used for real computation.

We will show that a single perceptron, with access to a small number of random bits, is powerful enough to recognize every language in AC^0. To be precise, we show that probabilistic perceptrons of size $2^{\mathrm{polylog}\, n}$ and bottom fanin $\mathrm{polylog}\, n$ can simulate AC^0 circuits with small error probability. Highly contrasting results are known for deterministic perceptrons. Minsky and Papert [28] constructed a depth-2 AC^0 predicate that cannot be computed by a deterministic perceptron with bottom fanin less than $\frac{1}{4}\sqrt[3]{n}$.

Our upper bound for AC^0 can be combined with Smolensky's easy proof that parity cannot be approximated by a low degree polynomial to obtain a very elegant proof that constant-depth circuits that compute parity require exponential size (cf. [15,18,25,39,47]). The result of Linial, Mansour, and Nisan [27] depends on prior lower bounds for parity and so could not be applied in this way.

The results of Chapter 5 are joint work with Richard Beigel and Daniel Spielman and appeared in [10]. Similar results were obtained independently by Tarui [40].

1.2 Notation

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 $\mathrm{length}(X)$ 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.

1.3 Facts about Polynomials

We mention a few facts which will be used many times in what follows.

Fact 1.3.1 *If p is a polynomial in k variables then $\mathcal{T}(p) \leq \binom{\deg(p)+k}{k}$.*

Notice that if $\deg(p)$ and k are polynomial in n then so is $\log \mathcal{T}(p)$.

Definition 1.3.2 A polynomial is *multilinear* if it is linear in each variable.

Fact 1.3.3 *If p has k variables and is multilinear then $\mathcal{T}(P) \leq \sum_{i=0}^{\deg(p)} \binom{k}{i} \leq 2^k$.*

Fact 1.3.4 $\mathcal{M}(pq) \leq \mathcal{M}(p)\mathcal{M}(q)\min\{\mathcal{T}(p), \mathcal{T}(q)\}$.

Chapter 2

Closure Properties for PP

The complexity class PP, which stands for *probabilistic polynomial-time*, was defined by Gill [19] (and, independently, by Simon [37]) in his seminal paper on probabilistic computations. Roughly speaking, a language is in PP if there is a probabilistic polynomial-time bounded Turing machine which correctly decides membership in the language with probability at least $1/2$. Gill showed that PP is closed under complementation, and asked whether it is closed under union and intersection.

Although, the class PP has played an important role in complexity theory, few closure properties had been proven for it, prior to the present work. Russo [36] showed that PP is closed under symmetric difference, and Beigel, Hemachandra, and Wechsung [9] showed that PP is closed under polynomial-time parity reductions. In Section 2.4 we will prove that PP is closed under intersection (and therefore under union). In Sections 2.6 and 2.7 we will show that PP is closed under polynomial-time truth-table reductions, and even constant-round truth-table reductions.

In Section 2.1 we define the class PrTIME$(t(n))$ and the class PP. We also review some basic facts about these classes. The previous closure properties for PP were proven by manipulating Turing machine computations according to certain polynomials. Our closure properties will rely on an extension of these ideas, which we develop in Section 2.2, in which we manipulate the computations according to a sequence of rational functions. In Section 2.3 we derive the rational functions we need for our closure properties from certain rational approximations to the *sign* function. In Section 2.4 we show that PP is closed under intersection and some special cases of truth-table reductions. Sections 2.5 through 2.7 are devoted to proving closure under truth-table and constant-round truth table reductions.

2.1 Probabilistic Polynomial Time

In this section we define the complexity classes PrTIME($t(n)$) and PP, and review some of the basic facts about these classes. Gill's original definition of PP was given in terms of probabilistic Turing machines. This is somewhat inconvenient for our purposes, so we will adapt a more amenable, but equivalent, definition due to Wagner [46]. The definitions depends solely on the difference between the number of accepting paths and the number of rejecting paths on a given input. This quantity is called the "Gap" by Fenner, *et al.* [16].

Definition 2.1.1 [16]

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 2.1.2

A language L is in PrTIME($t(n)$) if there exists a $O\left(t(n)\right)$-time bounded nondeterministic Turing machine N such that for all inputs X,

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

Definition 2.1.3 PP = PrTIME($n^{O(1)}$).

Notice that in the definition of PrTIME($t(n)$ the machine N can never have a Gap of 0. Disallowing machines which ever have a Gap of 0 is convenient for later manipulations. Gill [19] observed that, without loss of generality, we could allow $\mathrm{Gap}(N, X) = 0$ for $X \notin L$.

Lemma 2.1.4 (Gill [19]) *If N is a $t(n)$-time bounded nondeterministic Turing machine then there exists a $t(n)$-time bounded nondeterministic Turing machine N' such that for all inputs X*

$$\begin{aligned} \mathrm{Gap}(N, X) > 0 &\implies \mathrm{Gap}(N', X) > 0, \\ \mathrm{Gap}(N, X) \leq 0 &\implies \mathrm{Gap}(N', X) < 0. \end{aligned}$$

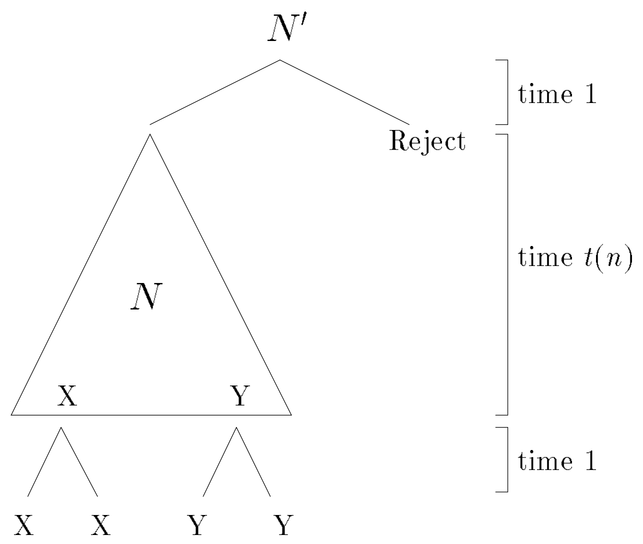

Figure 2.1: Breaking ties in favor of rejecting. The machine N' first nondeterministically either rejects immediately or simulates machine N. After each computation path of the simulation, N' makes one more nondeterministic choice, both paths then accept or reject according to whether the original path accepted or rejected.

Proof On input X, N' first nondeterministically chooses to either halt immediately and reject, or to simulate N on input X. At the conclusion of each computation path of the simulation of N, N' makes one more nondeterministic choice. After each of these choices, N' halts and accepts if and only if the N computation accepted. Apart from the initial possibility of rejecting, N' has exactly twice as many accepting paths as N and also twice as many rejecting paths as N. Therefore, $\mathrm{Gap}(N', X) = 2\mathrm{Gap}(N, X) - 1$. If $\mathrm{Gap}(N, X) > 0$ then $\mathrm{Gap}(N, X) \geq 1$, so that $\mathrm{Gap}(N', X) \geq 1 > 0$. On the other hand, if $\mathrm{Gap}(N, X) \leq 0$ then $\mathrm{Gap}(N', X) < 0$. The running time of N' is $t(n) + 2$. ▮

Gill [19] showed that NP $\subseteq$ PP. His argument extends directly to show

Lemma 2.1.5 (Gill [19]) NTIME($t(n)$) $\subseteq$ PrTIME($t(n)$).

Proof Suppose L is accepted by a $t(n)$-time bounded nondeterministic Turing machine N. Let N' be a nondeterministic Turing machine that works as follows. The machine N' first makes one nondeterministic choice. In either case N' simulates N. In one case, at the end of the simulation, N' accepts if and only if N accepts. In the other case N' accepts regardless of whether N accepts or not. If $X \in L$ then $\mathrm{Gap}(N', X) > 0$ while if $X \notin L$ then $\mathrm{Gap}(N', X) = 0$. By Lemma 2.1.4 L is in PrTIME($t(n)$). ∎

In our definition of PrTIME($t(n)$) all accepting (rejecting) paths are counted equally, regardless of length, as in [46]. Other definitions of PrTIME($t(n)$) either insist that all paths have the same length [6] or weight the paths according to length [19]. For time-constructible $t(n)$ these definitions are equivalent.

Lemma 2.1.6 (Simon [37]) *Suppose N is a $t(n)$-time bounded nondeterministic Turing machine, where $t(n)$ is time-constructible. There exists a $t(n)$-time bounded nondeterministic Turing machine N' such that all computation paths of N' have the same length and*

$$\begin{aligned} \mathrm{Gap}(N, X) > 0 &\implies \mathrm{Gap}(N', X) > 0, \\ \mathrm{Gap}(N, X) < 0 &\implies \mathrm{Gap}(N', X) < 0. \end{aligned}$$

Proof The machine N' first simulates N. Each computation path of N is extended as follows. If the path had length $t(n)$ then N' makes one more nondeterministic choice. Each of the two extended paths accepts if and only if N had accepted. If the N path had length $t(n) - j$, then N' makes $j + 1$ non-deterministic choices. The two rightmost paths both follow Ns original behavior (that is, they both accept if N had accepted, and both reject if N had rejected). Of the other paths, half accept and half reject (based, for example on the last nondeterministic choice). Since half accept and half reject, together these paths cancel in determining the final Gap of N'. Of the remaining paths, each accepting path of N has produced two accepting paths for N' and each rejecting path of N has produced two rejecting paths for N', so the Gap is exactly doubled, thus preserving the sign of the Gap. The running time of the machine described above is $t(n) + 1$ plus the time needed to compute $t(n)$. ∎

2.2 Building Turing Machines from Rational Functions

Beigel and Gill [8] and Gundermann, Nasser, and Wechsung [23] have used polynomials to prove closure properties of various counting classes. In this section we extend the techniques

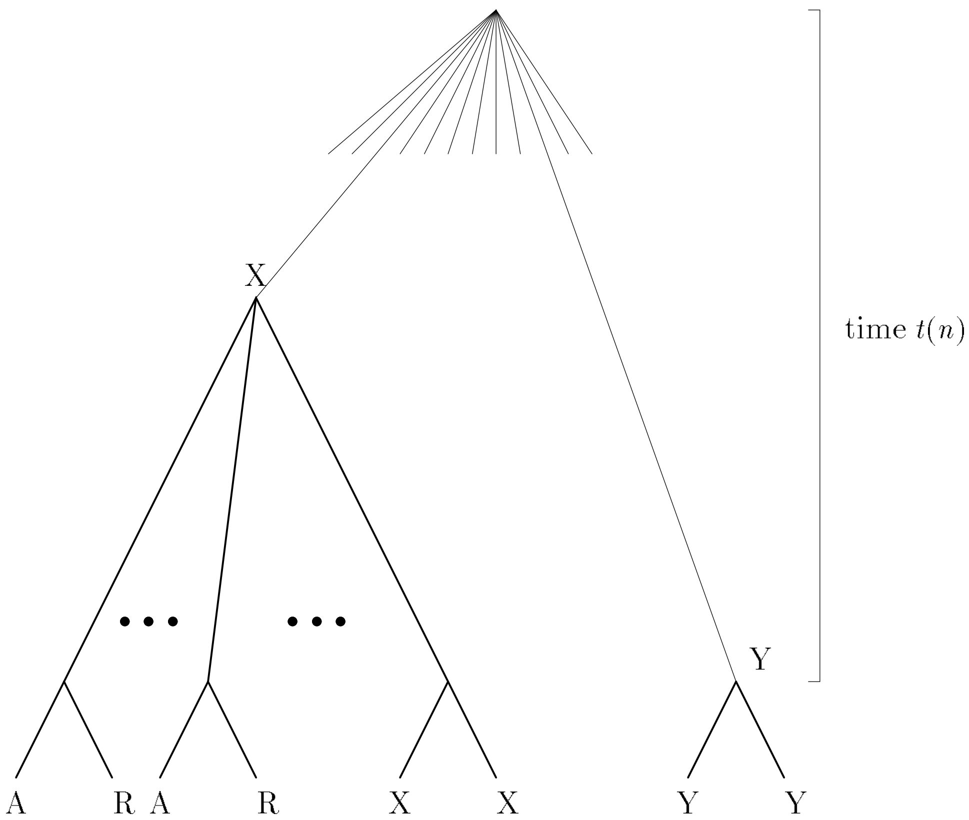

Figure 2.2: Extending all paths to have the same length while preserving the sign of the Gap. In the picture, the thinner lines represent computation paths of the original $t(n)$-time bounded machine, while the thicker lines represent the extensions in the new machine. Each original path of length $t(n)$ is extended to two paths of length $t(n) + 1$. Both paths accept or reject according to whether the original path accepted or rejected. Original paths of length less than $t(n)$ are also extended to have length $t(n) + 1$. Then, the two right-most of the extended paths both accept or reject according to the original path. Of the remaining extended paths, half accept and half reject, thus making no contribution to the Gap of the new machine.

of [8]: 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.

For any nondeterministic Turing machine N, let the complement machine, denoted $\overline{N}$, be the machine that 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_+ that 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 that 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 yields Lemma 2.2.2, below.

Definition 2.2.1 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.

Lemma 2.2.2 *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 $\deg(p_n) = d_n$ and $\mathcal{M}(p_n) = M_n$. 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 $n = \mathrm{length}(X)$ 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 $s(n)$ to determine the degree of p_n and an additional time of $O\left(\log\binom{d_n+k}{k}\right)$ to branch sufficiently. Then N nondeterministically chooses one of c branches, computes the product as described above, and complements if necessary. This takes an additional time $s(n)$ to determine the coefficient c, time $O(\log(M_n))$ to branch sufficiently, and time $d_n t(n)$ to run the machines that occur in the monomial. Figure 2.3 shows the computation tree for machine N. ▮

Example:

Suppose that L_1 and L_2 are in $\mathrm{PrTIME}(t(n))$, and that L is the symmetric difference of L_1 and L_2 (that is, inputs which are in exactly one of L_1 and L_2). Notice that $-x_1x_2$ is

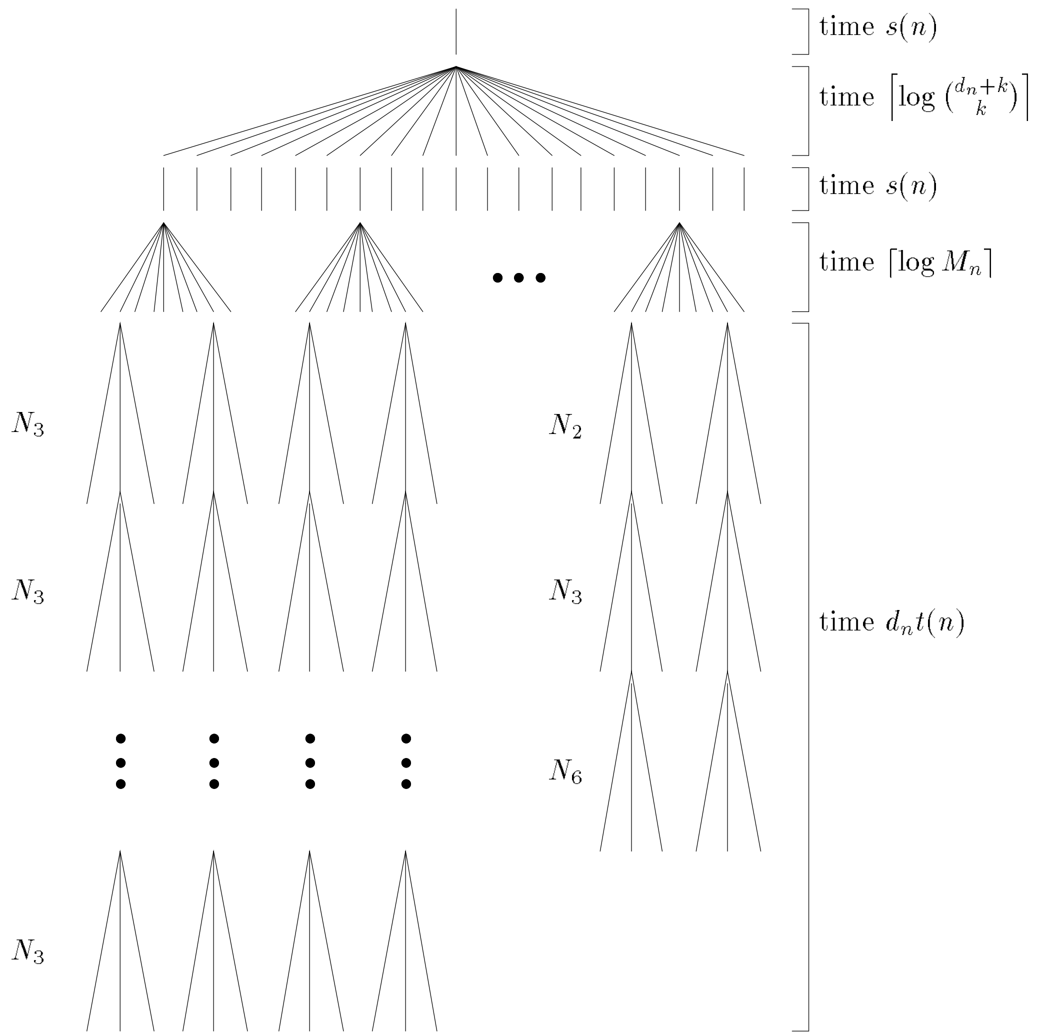

Figure 2.3: The computation tree for a nondeterministic Turing machine N, such that $\mathrm{Gap}(N, X) = p_n(\mathrm{Gap}(N_1, X), \ldots, \mathrm{Gap}(N_k, X))$. In the first stage, N computes the degree of p_n. In the second stage, N nondeterministically chooses a monomial. In the third stage, N computes 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 $x_2 x_3 x_6$.

positive if and only if one of x_1 and x_2 is positive and the other negative. We can show that $L \in \mathrm{PrTIME}(2t(n))$ by applying Lemma 2.2.2 with $p_n(x_1, x_2) = -x_1x_2$. This is essentially the same as Russo's proof [36] that PP is closed under symmetric difference.

If, in Lemma 2.2.2, each of the machines $N_1, \ldots, N_k$ is polynomial-time bounded then to ensure that N is also polynomial-time bounded it suffices to have $s(n)$, $\deg(p_n)$, and $\log \mathcal{M}(p_n)$ bounded by a polynomial in n.

Although we cannot directly apply Lemma 2.2.2 to rational functions (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. Note that in the definition of $\mathrm{PrTIME}(t(n))$ only the sign of the Gap is essential.

Definition 2.2.3 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 2.2.4 *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 $O\left(\log\left(M_n\binom{d_n+k}{k}\right)\right) + 2d_nt(n) + 4s(n)$ such that $\mathrm{Gap}(N, X)$ and $r_n(y_1, \ldots, y_k)$ have the same sign for all X whenever the latter is defined, where $n = \mathrm{length}(X)$ and y_i is $\mathrm{Gap}(N_i, X)$.*

Proof Suppose $r_n = a_n/b_n$, where a_n and b_n are polynomials with integral coefficients. By Lemma 2.2.2 there are Turing machines N_a and N_b whose Gaps are given by a and b. Again by Lemma 2.2.2 there is a Turing machine N whose Gap is the product of the Gaps of N_a and N_b. Since a_nb_n has the same sign as a_n/b_n this is the desired Turing machine. ∎

2.3 Rational Approximation

In this section we define the rational functions $A_n(x, y)$ that 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[1]. In his seminal paper on rational approximation, Newman [29] 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 *sign* function is defined as

$$sign(x) = \begin{cases} 1 & x > 0, \\ 0 & x = 0, \\ -1 & x < 0. \end{cases}$$

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. A more accurate error analysis is given in Section 3.3. The curious expressions above are explained in Section 3.2.

Lemma 2.3.1 *For $n \geq 1$,*

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

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

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

Proof

(a) Since $x \geq 1$, we have that $P_n(x) \geq 0$. Clearly $x - 1 < x + 1$. Also, for $i = 1, \ldots, n$,

$$(x - 2^i)^2 < (-x - 2^i)^2.$$

[1] We remind the reader that lowercase letters such as x and y refer to *real* variables, not Turing machine inputs.

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 $4P_n(x) < -P_n(-x)$.

(b) 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 (a) yield the desired result.

(c) This follows from (b) and the fact that $S_n(x)$ is an odd function. ▮

Note: Although the above lemma only guarantees that the error in approximating *sign* is at most 2/3, it does guarantee *one-sided* error.

Lemma 2.3.2 *Let x and y satisfy $1 \leq |x|, |y| \leq 2^n$.*

(a) $\deg(A_n) = O(n)$.

(b) $\mathcal{M}(A_n) = 2^{O(n^2)}$.

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

(d) 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 2.3.1(b–c). ▮

We now consider analogous rational functions in many variables. Let $h(k)$ be the least odd integer greater than or equal to $\frac{1}{2}\log(2k+1)$. 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 2.3.3

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

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

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

(d) 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 2.3.1(a), $0 \leq (2k+1)\,(P_n(x))^{h(k)} < -\,(P_n(-x))^{h(k)}$. We now proceed as in the proof of Lemma 2.3.1(b). Part (d) follows from (c) and the fact that $S_n^{(k)}(x)$ is an odd function. ▮

Lemma 2.3.4 *Let* x_i *satisfy* $1 \leq |x_i| \leq 2^n$*, for* $i = 1, \ldots, k$.

(a) $\deg(A_n^{(k)}) = O\left(nk \log k\right)$.

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

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

(d) 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 assertions follow from Lemma 2.3.3 just as Lemma 2.3.2(c-d) follow from Lemma 2.3.1. ▮

Notice that given n and k it takes time $O\left(n^2 k \log k\right)$ to compute any coefficient of $A_n^{(k)}$. Thus the $A_n^{(k)}$ are $O\left(n^2 k \log k\right)$-uniform in the sense of Definition 2.2.3.

2.4 Closure Under Intersection

It has been shown that PP is closed under complementation [19], under symmetric difference [36], and under polynomial-time parity reductions [9]. 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. We begin by showing that PP is closed under intersection.

Theorem 2.4.1 *The intersection of finitely many languages in* PrTIME($t(n)$) *is in* PrTIME($t(n)^2$).

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

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

Let c be a constant such that $|\mathrm{Gap}(N_i, X)| \leq 2^{ct(n)}$ for all i and X. Let $r_n(x_1, \ldots, x_k) = A^{(k)}_{ct(n)}(x_1, \ldots, x_k)$. By Lemma 2.3.4, $\deg(r_n) = O\left(t(n) k \log k\right)$ and $\mathcal{M}(r_n) = 2^{O\left(t(n)^2 k \log k\right)}$. Since r_n is $O\left(t(n)^2\right)$-uniform, then by Lemma 2.2.4 there is a nondeterministic Turing machine N whose running time is $O\left(t(n)^2\right)$, and such that for all inputs X, $\mathrm{Gap}(N, X)$ and $r_n(y_1, \ldots, y_k)$ have the same sign, where $n = \mathrm{length}\,(X)$, and, for $i = 1, \ldots, k$, $y_i = \mathrm{Gap}(N_i, X)$. Since $1 \leq |y_i| \leq 2^{ct(n)}$, Lemma 2.3.4 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 $\bigcap_{i=1}^{k} L_i \in$ PrTIME($t(n)^2$). ∎

We now consider several kinds of reductions. We will prove that PP is closed under each of these. Each kind of reduction is a special case of polynomial-time truth-table reduction. We will prove closure under general polynomial-time truth-table reductions in Section 2.6.

Definition 2.4.2 A language A is *polynomial-time truth-table reducible* to a language B, written $A \leq^p_{\mathrm{tt}} B$, 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.

Definition 2.4.3

- 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.

- 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.

- A polynomial-time $O(\log n)$*-Turing reduction* is a polynomial-time Turing reduction that makes $O(\log n)$ queries. We use $\mathrm{P}^{A}_{O(\log n)\text{-}\mathrm{T}}$ to denote the class of languages polynomial-time $O(\log n)$-Turing reducible to a set A.

- 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.

- 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.

Torán [44] asked whether PP is closed under conjunctive reductions, and he noted that an affirmative answer would imply closure under $O(\log n)$-Turing reductions.

Theorem 2.4.4 *The class* PP *is closed under polynomial-time conjunctive reductions and disjunctive reductions.*

Proof For conjunctive reductions we use the rational functions $A_n^{(k)}(x_1, \ldots, x_k)$, just as in Theorem 2.4.1, although k now depends on n, with k bounded by a polynomial in n. The details are as in the previous proof. Closure under polynomial-time disjunctive reductions is obtained by using the rational functions $-A_n^{(k)}(-x_1, \ldots, -x_k)$. ∎

Theorem 2.4.5 PP *is closed under polynomial-time bounded-depth Boolean formula reductions.*

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

Theorem 2.4.6 PP *is closed under polynomial-time* $O(\log n)$*-Turing reductions. In other words,* $\mathrm{P}^{\mathrm{PP}}_{O(\log n)\text{-T}} = \mathrm{PP}$.

Proof Every $O(\log n)$-Turing reduction can be written as a CNF formula over the query answers. Thus, we can convert the polynomial-time $O(\log n)$-Turing reduction to a polynomial-time depth-2 Boolean formula reduction. ∎

Previously it had been shown that $\mathrm{P}^{\mathrm{NP}}_{O(\log n)\text{-T}} \subseteq \mathrm{PP}$ (see [9, 42]), and $\mathrm{P}^{\mathrm{C_=P}}_{O(\log n)\text{-T}} \subseteq \mathrm{PP}$ (see [23]). (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 $\mathrm{Gap}(N, X) = 0$.) By Lemma 2.1.5 $\mathrm{NP} \subseteq \mathrm{PP}$. Also, Russo [36] has shown that $\mathrm{C_=P} \subseteq \mathrm{PP}$. (For a proof, use the fact that for any Turing machine N there is an N' with $\mathrm{Gap}(N', X) = 1 - 2(\mathrm{Gap}(N, X))^2$.) Thus our theorem above implies both prior inclusions.

Theorem 2.4.7 PP *is closed under polynomial-time threshold reductions.*

Proof Define

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

Assume that $1 \le |x_i| \le 2^n$ for $1 \le i \le k$, where $k = n^{O(1)}$. 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 $\mathcal{M}(T_n^{(k)}) = 2^{O\left(n^2 k \log k\right)}$. Also, $T_n^{(k)}$ is $O\left(n^2 k \log k\right)$-uniform. The result now follows from Lemma 2.2.4. ∎

Since every symmetric function can be computed as a threshold of thresholds [24], this immediately implies

Corollary 2.4.8 PP *is closed under polynomial-time symmetric reductions.* ∎

2.5 Gap-P and Gap-Q Functions

In the previous section we were careful to give explicit bounds on the degrees and coefficient for the rational functions involved in the proofs of the closure properties. In the proofs of closure under polynomial-time truth-table reductions and constant round truth-table reductions, it will be convenient to be less explicit. To this end, we will use the notion of Gap-P functions from Fenner *et al.* [16] and define the analogous class of Gap-Q functions. The Turing machine manipulations of Section 2.2 correspond to closure properties of Gap-P and Gap-Q functions.

Definition 2.5.1 A function $f(X)$ is a *Gap*-P *function* if there is a nondeterministic Turing machine M such that for all inputs X, $f(X) = \mathrm{Gap}(M, X)$.

Since the Turing machines for languages in PP may be assumed never to have a Gap of zero (Lemma 2.1.4), we can characterize PP as follows.

Fact 2.5.2 $L \in \mathrm{PP}$ *if and only if there is a Gap*-P *function* f *such that*

$$\begin{aligned} X \in L &\implies f(X) > 0, \\ X \notin L &\implies f(X) < 0. \end{aligned} \tag{2.1}$$

If χ_L is the characteristic function of L then (2.1) is equivalent to

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

We now state some closure properties for the class of Gap-P functions. Proofs can be given based on the discussion preceding Lemma 2.2.2 (and can also be found in [16]).

Lemma 2.5.3 *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* $\mathrm{length}\,(Y) \leq q(\mathrm{length}\,(X))$.

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

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

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

Definition 2.5.4 A *Gap*-Q *function* is the quotient of two Gap-P functions.

Fact 2.5.5 $L \in \mathrm{PP}$ *if and only if there is a Gap*-Q *function g such that*

$$\begin{aligned} X \in L &\implies g(X) > 0, \\ X \notin L &\implies g(X) < 0. \end{aligned}$$

Gap-P functions are closed under (uniform) exponential-size sums by Lemma 2.5.3 (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.

Lemma 2.5.6 *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 each i, $r(i, X)$ is a Gap*-Q *function. Then the function $g(X) = P_n(r(i, X), \ldots, r(k_n, X))$, where $n = \mathrm{length}\,(X)$, 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,

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

$$= \sum_{\vec{\alpha}} c_\alpha \frac{\prod_i h^{\alpha_i}(i,X)}{t^{\sigma(\vec{\alpha})}(X)}$$
$$= \left(\frac{1}{t^{d_n}(X)}\right) \sum_{\vec{\alpha}} c_{\vec{\alpha}} t^{d_n - \sigma(\vec{\alpha})}(X) \prod_i h^{\alpha_i}(i,X).$$

Clearly $t^{d_n}(X)$ is a Gap-P function. Also, the sum is a Gap-P function, by Lemma 2.5.3. ∎

2.6 Truth-Table Reductions

Definition 2.6.1 A language A is *polynomial-time truth-table reducible* to a language B, written $A \leq^p_{\mathrm{tt}} B$, 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 Section 2.4. We start with a few lemmas.

Lemma 2.6.2 *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. ∎

Lemma 2.6.3 *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 we have $1 \leq |c_j| < 1 + 1/2h$ by Lemma 2.3.3. 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,

$$\begin{aligned}
tM\rho(\vec{\alpha}) &= tM \prod_i |c_i| \\
&< tM \left(\frac{(1 + 1/2h)^{d-1}}{2h} \right) \\
&< tM \left(\frac{(1 + 1/2h)^{d}}{2h} \right) \\
&< tM \left(\frac{1}{2h - d} \right) \qquad \text{(by Lemma 2.6.2)} \\
&\leq \frac{tM}{dtM(1/\epsilon)} \\
&\leq \epsilon.
\end{aligned}$$

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

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

$$\begin{aligned}
tM\rho(\vec{\alpha}) &= tM\left|\prod_i c_i^{\alpha_i} - 1\right| \\
&< tM\left((1+1/2h)^d - 1\right) \\
&< tM\left(\frac{2h}{2h-d} - 1\right) \qquad \text{(by Lemma 2.6.2)} \\
&= tM\left(\frac{d}{2h-d}\right) \\
&\leq \frac{tMd}{dtM(1/\epsilon)} \\
&= \epsilon.
\end{aligned}$$

∎

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

Theorem 2.6.4 PP *is closed under polynomial-time truth-table reductions.*

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

$$\chi_B(X) = \frac{1 + sign(\mathrm{Gap}(N, X))}{2}. \tag{2.2}$$

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,

$$\begin{aligned}
X \in A &\implies f(\chi_B(z_1), \ldots, \chi_B(z_k)) = 1, \\
X \notin A &\implies f(\chi_B(z_1), \ldots, \chi_B(z_k)) = 0.
\end{aligned} \tag{2.3}$$

Since f maps $\{0,1\}^k$ to $\{0,1\}$, f has a unique multilinear extension $P(x_1, \ldots, x_k)$ that maps $\mathbf{R}^k$ to $\mathbf{R}$. 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 $\deg(P) \le k$, $\mathcal{M}(P) \le 2^k$ (since each summand above can contribute -1, 0, or 1 to the final coefficient of any monomial), and $\mathcal{T}(P) \le 2^k$ (since P is multilinear).

Substituting (2.2) into (2.3) and replacing f by P gives

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

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

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

$$\begin{aligned} l &= \max_i T(|Y_i|), \qquad \text{and} \\ h &= \lceil (k/2)(1+2\mathcal{T}(P)\mathcal{M}(P)) \rceil . \end{aligned}$$

Since the length of each Y_i is bounded by a polynomial in n, l and $\log k$ are bounded by some polynomial in n. Define

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

Lemma 2.6.3 shows that Q and $\hat{Q}$ differ by less than $1/2$. Also, Lemma 2.5.6 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

$$\begin{aligned} X \in A &\Rightarrow Q(Y_1,\ldots,Y_k) = 1 \Rightarrow \hat{Q}(Y_1,\ldots,Y_k) > \tfrac{1}{2} \Rightarrow G(X) > 0, \\ X \notin A &\Rightarrow Q(Y_1,\ldots,Y_k) = 0 \Rightarrow \hat{Q}(Y_1,\ldots,Y_k) < \tfrac{1}{2} \Rightarrow G(X) < 0. \end{aligned}$$

∎

2.7 Constant Round Truth-Table Reductions

Definition 2.7.1 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 at most $p(n)$. 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 rth 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_{m+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 2.4. If language A is reducible to language B by a k-round polynomial-time truth-table reduction we write $A \leq^p_{k\text{-round}} B$.

In this section will prove that PP is closed under k-round truth-table reductions. We will need one more fact about Gap-P functions. The following lemma on the "arithmetization" of Gap-P functions is from Babai and Fortnow [5], who attribute the result to Fenner, Fortnow, and Kurtz [16]. We state the result purely in terms of uniform polynomials, rather than the "straight line programs" of Babai and Fortnow.

Lemma 2.7.2 (Babai-Fortnow [5]) *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 $n = \text{length}\,(X)$ and X_i is the ith bit of X.

Theorem 2.7.3 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 2.6.4. Suppose that $A \leq^p_{k\text{-round}} B$ 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 $Y_1^{(2)}, \ldots, Y_m^{(2)}$. 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 $\chi_B(z) = (1 + \mathit{sign}(h(z)))/2$. By Lemma 2.7.2 there is a uniform sequence of polynomials P_m, with $\deg(P_m)$ and $\log \mathcal{M}(P_m)$ polynomial in $m = \text{length}(z)$, 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

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

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 2.6.4 $L \in$ PP, and so there is a Gap-P function $r_i(X,j)$ such that

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

Consider the two functions

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

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

By Lemma 2.6.3 a sufficiently good approximation can be obtained by choosing h and l so that $|r_i(X,j)| \leq 2^l$ and $h \geq (\deg(P_m)/2)(1 + 2\mathcal{T}(P_m)\mathcal{M}(P))$. Since $\deg(P_m)$ and the number of variables in P_m are both polynomial in n, $\log \mathcal{T}(P_m)$ 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 2.5.6, that $H(X)$ is a Gap-Q function. So, there exist Gap-P functions $u_i(X)$, $v_i(X)$ such that $2H(X)-1 = u_i(X)/v_i(X)$. 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, $C_i \in \mathrm{PP}$. Since we have assumed that B is m-complete for PP, $C_i \leq_m^P B$. 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

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

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

Chapter 3

Mathematical Extras

In this chapter we present some mathematical results about approximations of the *sign* function. While not of direct relevance to the complexity issues, they are interesting in their own right.

3.1 Intersector Polynomials

The proof that PP is closed under intersection relied on the existence of rational functions $A_n(x, y)$ which for non-zero integral values of x and y within the $2^{n+1} \times 2^{n+1}$ square centered at the origin, are positive if and only if x and y are both positive. By clearing denominators, we can construct polynomials of degree $4n + 1$ which takes on the correct sign for such x and y. In order for the proof to work, we needed polynomials which take on the correct sign within a square whose side is exponential in the degree.

Definition 3.1.1 A function of two variables $p(x, y)$ is an *M-intersector* if for all integers x, y such that $1 \leq |x|, |y| \leq M$

(a) if $x > 0$ and $y > 0$ then $p(x, y) > 0$, and

(b) if $x < 0$ or $y < 0$ then $p(x, y) < 0$.

A function of two variables $p(x, y)$ is an *∞-intersector* if for all non-zero integers x, y

(a) if $x > 0$ and $y > 0$ then $p(x, y) \geq 0$, and

(b) if $x < 0$ or $y < 0$ then $p(x, y) \leq 0$.

The function $sign(x)+sign(y)-1$ is an ∞-intersector. However, Minsky and Papert [28] have shown the following.

Theorem 3.1.2 *Any polynomial ∞-intersector is identically zero.*

In fact, a careful analysis of Minsky and Papert's proof shows that if $p(x, y)$ is a polynomial M-intersector of degree d then $M \leq \mathcal{M}(p)d^d$. While no single polynomial can "cover" the whole plane, is it possible to find a family of polynomials, all of degree d, which cover the plane? In order for such a family to exist, it must be the case that for every M there is a degree d M-intersector. We show that this cannot happen.

Theorem 3.1.3 *For every d there is an M such that no polynomial $p(x, y)$ of degree d is an M-intersector.*

Proof Suppose to the contrary that for every M there is a polynomial $p_M(x, y)$ of degree d which is an M-intersector. We now consider any degree d polynomial in two variables as a point in $\mathbf{R}^n$ where $n = \binom{d+2}{2}$: a polynomial corresponds to the sequence of its coefficients. By dividing if necessary, we may assume that each coefficient of p_M has absolute value at most 1 and some coefficient has absolute value exactly 1. Therefore all such polynomials lie on the boundary of the cube of side 2 centered at the origin. By compactness, there must be a subsequence $\{p_{M_i}\}$ of these polynomials which converges to some point, say $P(x, y)$, on the boundary of the cube. Clearly, $P(x, y)$ is a polynomial ∞-intersector. However, $P(x, y)$ cannot be identically zero since some coefficient is non-zero. ∎

For each d let $M(d) = \inf\{M :$ no polynomial of degree d is an M-intersector$\}$. The above theorem shows that $M(d)$ is finite. We also know that $M(d) = 2^{\Omega(d)}$, since the polynomials obtained from the A_n of Section 2.3 have degree $4n + 1$ and the corresponding M is at least 2^n. We do not have any upper bound on $M(d)$ which depends only on d. We will return to this problem at the end of the chapter.

3.2 Sign-Approximators

3.2.1 From Approximators to Intersectors

Definition 3.2.1 For $\epsilon > 0$ and $M > 1$ an (ϵ, M) *sign-approximator* is a function $S(x)$ such that

(a) If $1 \leq x \leq M$ then $1 \leq S(x) < 1 + \epsilon$,

(b) If $-M \leq x \leq -1$ then $-1 - \epsilon < S(x) \leq -1$.

The rational function $S_n(x)$ built in Section 2.3 is a $(2/3, 2^n)$ sign-approximators of degree $2n + 1$. As we saw in Section 2.3, S_n can be used to build a 2^n-intersector. In fact, whenever $\epsilon < 1$, an (ϵ, M) sign-approximator can be used to build an M-intersector.

Lemma 3.2.2 *Suppose that $S(x)$ is an (ϵ, M) sign-approximator and that $\epsilon < 1$ then $S(x)+S(y) - 1$ is an M-intersector.*

Proof If $1 < x, y < M$ then $S(x) + S(y) - 1 \geq 1 + 1 - 1 = 1 > 0$. If $-M < x < -1$ then

$$\begin{aligned} S(x) + S(y) - 1 &\leq -1 + 1 + \epsilon - 1 \\ &= \epsilon - 1 \\ &< 0. \end{aligned}$$

∎

If $S(x)$ is a polynomial of degree d then the resulting intersector is also a polynomial of degree d. If $S(x) = u(x)/v(x)$ is a rational function then $S(x) + S(y) - 1 = p(x, y)/q(x, y)$ where p and q are polynomials of degree at most $2d$. Since pq has the same *sign* as p/q, pq is an M-intersector of degree $4d$. In fact, we can find an M-intersector of degree $2d$. Consider the four regions

$$\begin{aligned} &\{(x, y) : 1 \leq x, y \leq M\}, \\ &\{(x, y) : 1 \leq -x, y \leq M\}, \\ &\{(x, y) : 1 \leq x, -y \leq M\}, \\ &\{(x, y) : 1 \leq -x, -y \leq M\}. \end{aligned} \tag{3.1}$$

Suppose a polynomial $p(x, y)$ has constant sign in each of these regions, and is positive in an odd number (*i.e.* 1 or 3) of them. Then one of $\pm p(\pm x, \pm y)$ must be an M-intersector. Let us say that such a polynomial is *almost* an M-intersector.

Lemma 3.2.3 *Suppose that $S(x) = u(x)/v(x)$ is a rational function of degree d, and that for some positive $\epsilon < 1$ $S(x)$ is an (ϵ, M) sign-approximator. Let $S(x) + S(y) - 1 = p(x, y)/q(x, y)$. Then one of $p(x, y)$ and $q(x, y)$ is almost an M-intersector.*

Proof We may assume that u and v have no common factors. If either of u or v had a root in $1 \leq x \leq M$ then the other polynomial would necessarily have a root at the same place, but then u and v would have a factor in common. So both u and v have constant sign in $1 \leq x \leq M$. Similarly, both have constant sign in $-M \leq x \leq -1$. Consequently, $q(x, y) = v(x)v(y)$ has constant sign in each of the four regions (3.1). The same must also be true of $p(x, y)$.

Now, suppose both $p(x, y)$ and $q(x, y)$ are positive in an even number of the regions (3.1). Then their product would have to be positive in an even number of regions, which contradicts the fact that the product is an M-intersector. Therefore, one of p and q is positive in an odd number of regions, and so is almost an M-intersector. ∎

3.2.2 Polynomial Sign-Approximators

The intersectors used in the complexity proofs were obtained from rational *sign* approximators. Recall that our proof of closure under intersection required the construction of an (ϵ, M) sign-approximator of degree d with M exponential in d. We now show that no such polynomial *sign* approximations exist. We will need the following lemma from approximation theory. The lemma is implicit in [35], and was recently rediscovered by Nisan and Szegedy [30].

Lemma 3.2.4 *Suppose $p(x)$ is a degree d polynomial such that*

$$|p(i)| \leq K \qquad i = 0, 1, \ldots, m \tag{3.2}$$

for some $m > d^2$. Then

$$|p'(x)| \leq \frac{2d^2 K}{m - d^2} \qquad \text{for all } x \in [0, m], \tag{3.3}$$

Theorem 3.2.5 *Let ϵ be any constant satisfying $0 < \epsilon < 1$. If $p(x)$ is a degree d polynomial such that $|p(x) - sign(x)| < \epsilon$ for $|x| = 1, 2, \ldots, M$ then*

$$d \geq \left(\sqrt{\frac{1-\epsilon}{3+\epsilon}}\right)\sqrt{M} \tag{3.4}$$

Proof Since $(p(-x) - p(x))/2$ is at least as good an approximation to *sign* as $p(x)$ is, we may assume that $p(0) = 0$. Since $p(1) \geq 1 - \epsilon$, there is a $\theta \in [0, 1]$ such that $p'(\theta) \geq 1 - \epsilon$. Suppose that (3.4) is false, then $d < \sqrt{M}$. Since each of $|p(0)|, \ldots, |p(M)|$ is at most $1 + \epsilon$, Lemma 3.2.4 implies that

$$\begin{aligned} 1 - \epsilon &\leq \frac{2d^2(1+\epsilon)}{M - d^2} \\ (1-\epsilon)M - (1-\epsilon)d^2 &\leq 2d^2(1+\epsilon) \\ \left(\frac{1-\epsilon}{3+\epsilon}\right)M &\leq d^2, \end{aligned}$$

but this contradicts the assumption that (3.4) fails. ∎

Paturi's recent work on approximation [31] gives an even stronger result. It follows from that work that if p is a degree d polynomial such that $|p(x) - sign(x)| < 1/3$ for $|x| = 1, 2, \ldots, M$ then $\deg(p) = \Omega(M)$.

3.2.3 Rational Sign-Approximators

For the reader who thinks that the construction of the sign-approximators $S_n(x)$ given in Section 2.3 is artificial, or that the polynomials P_n and the form of S_n are mysterious, we offer the following theorems which show that this is essentially the only way to construct sign-approximators.

The first lemma, well know to approximation theorists (see [34], Exercise 1.1) shows that $(f(x) - f(-x))/2$ is at least as good an approximation to *sign* as $f(x)$. Thus, we may as well use an odd function to approximate *sign*.

Lemma 3.2.6 *For $0 \leq a \leq b$ let $\Delta = \{x : a \leq |x| \leq b\}$. For any function f, and any odd function g*

$$\sup_{x\in\Delta}\left|\frac{f(x) - f(-x)}{2} - g(x)\right| \leq \sup_{x\in\Delta}|f(x) - g(x)|.$$

Proof Let $\mu = \sup_{x \in \Delta} |f(x) - g(x)|$. Then for any $x \in \Delta$,

$$
\begin{aligned}
\left|\frac{f(x)-f(-x)}{2} - g(x)\right| &= \left|\frac{f(x)-g(x)}{2} + \frac{-f(-x)-g(x)}{2}\right| \\
&= \left|\frac{f(x)-g(x)}{2} + \frac{-f(-x)+g(-x)}{2}\right| \\
&\leq \frac{1}{2}|f(x)-g(x)| + \frac{1}{2}|f(-x)-g(-x)| \\
&\leq \frac{1}{2}\mu + \frac{1}{2}\mu \\
&= \mu.
\end{aligned}
$$

∎

The next lemma shows that the mysterious form of $S_n(x)$ is not so mysterious after all. This normal form for odd rational functions is used implicitly in several works on rational approximation (see [14]). Newman [29] gives a similar normal form for even rational functions.

Lemma 3.2.7 *If s is a rational odd function of degree d then there is a polynomial p of degree at most d such that*

$$s(x) = \frac{p(-x)-p(x)}{p(-x)+p(x)}.$$

Proof Suppose that $s(x) = q(x)/r(x)$, where q and r are polynomials of degree at most d. Since s is odd,

$$s(x) = \frac{q(x)}{r(x)} = \frac{-q(-x)}{r(-x)} = \frac{q(x)-q(-x)}{r(x)+r(-x)}.$$

Let $p(x) = q(-x) - q(x) + r(x) + r(-x)$. Since $p(-x) - p(x) = 2(q(x) - q(-x))$, and $p(-x) + p(x) = 2(r(x) + r(-x))$,

$$s(x) = \frac{p(-x)-p(x)}{p(-x)+p(x)}.$$

∎

Theorem 3.2.8 *Let $0 < a < b$ and $c > 1$. The following are equivalent.*

(a) There exists a polynomial p of degree at most d which is not identically zero such that

$$a \leq x \leq b \implies \text{Either } p(x) = 0 \text{ or } \frac{-p(-x)}{p(x)} > c.$$

(b) There exists a rational function r of degree at most d such that

$$\begin{aligned} -b \le x \le -a &\implies -\left(\frac{c+1}{c-1}\right) < r(x) \le -1, \\ a \le x \le b &\implies 1 \le r(x) < \left(\frac{c+1}{c-1}\right). \end{aligned}$$

(c) There exists a rational function r of degree at most d such that

$$\begin{aligned} -b \le x \le -a &\implies r(x) \le -(c^2-1), \\ a \le x \le b &\implies 0 \le r(x) < 1. \end{aligned}$$

Proof

(a) implies (b). For the moment, we will assume $p(-x)+p(x) \neq 0$ when $a \le x \le b$. Let

$$r(x) = \frac{p(-x)-p(x)}{p(-x)+p(x)}.$$

Fix x such that $a \le x \le b$. If $p(x)=0$ then $r(x)=1$. If $p(x) \neq 0$ then

$$\begin{aligned} r(x) &= \frac{p(-x)-p(x)}{p(-x)+p(x)} \\ &= 1 + \frac{2}{\left(\frac{-p(-x)}{p(x)}\right) - 1}. \end{aligned}$$

By (a), $-p(-x)/p(x) \ge c > 1$, so $r(x) \ge 1$. Also,

$$\begin{aligned} r(x) &= 1 + \frac{2}{\left(\frac{-p(-x)}{p(x)}\right) - 1} \\ &< 1 + \frac{2}{c-1} \\ &= \frac{c+1}{c-1}. \end{aligned}$$

Since $r(x)$ is an odd function, r satisfies all the conditions of (b).

Now suppose that $p(-x_0)+p(x_0)=0$ for some x_0 with $a \le x_0 \le b$. If $p(x_0) \neq 0$ then $-p(-x_0)/p(x_0) = 1 < c$, but by (a), $-p(-x_0)/p(x_0) \ge c$, which is a contradiction. This implies that $p(x_0)=0$ and so also $p(-x_0)=0$. Therefore $p(x) = (x^2-x_0^2)p'(x)$ for some polynomial p'. Notice that $p'(x)$ satisfies (a). In particular, $p'(x)$ cannot be a constant. Thus, if we continue to divide by factors of the form $x^2-x_i^2$ we eventually obtain a polynomial q such that $q(-x)+q(x)$ is never zero. Thus we may still take r as above with the understanding that we remove factors common to the numerator and denominator.

(b) implies (a). By Lemma 3.2.6 we may assume that r is an odd function. By Lemma 3.2.7 there is a degree d polynomial p such that

$$r(x) = \frac{p(-x)-p(x)}{p(-x)+p(x)}.$$

We claim that p satisfies (a). If $p(x) \neq 0$ then

$$\begin{aligned} r(x) &= \frac{p(-x)-p(x)}{p(-x)+p(x)} \\ &= 1 + \frac{2}{\left(\frac{-p(-x)}{p(x)}\right) - 1}. \end{aligned}$$

The condition on r now implies the desired condition on p.

(b) if and only if (c). Consider the bilinear transformation

$$f(x) = \frac{(c^2-1)x-(c^2-1)}{(c-1)x+(c+1)}.$$

Except at its simple pole, f has positive derivative. It is therefore easy to check that $f(x)$ satisfies

$$\begin{aligned} -\left(\frac{c+1}{c-1}\right) < x \leq -1 &\implies f(x) \leq -(c^2-1), \\ 1 \leq x < \left(\frac{c+1}{c-1}\right) &\implies 0 \leq f(x) < 1. \end{aligned}$$

Thus if r satisfies (b) then $f(r)$ satisfies (c), and $f(r)$ has the same degree as r.

Since f has an inverse which is also bilinear, the same technique shows that (c) implies (b). ∎

3.2.4 From Intersectors to Approximators

The rational intersectors $A_n(x,y)$ we built have a special form, namely $A_n(x,y)$ can be written $Q(x)+Q(y)$, where $Q(x) = S_n(x) - 1/2$ is a rational function of degree $2n+1$. The next theorem shows that any such intersector gives rise to a *sign*-approximator.

Theorem 3.2.9 *Suppose Q is a rational function of degree d and that Q is an M-intersector. Then there is rational function S which is a $(2+2\sqrt{2}, M)$ sign-approximator of degree d.*

Proof Let $s = \sup\{Q(x) : 1 \leq x \leq M\}$, and $t = \sup\{Q(x) : -M \leq x \leq -1\}$. If $1 \leq x \leq M$ then $Q(x) + Q(x) > 0$, and so $Q(x) > 0$. Thus $s > 0$. Also, $s + t < 0$, so $-t > s > 0$. Now,

$$\begin{aligned} 1 \leq x \leq M &\implies 0 \leq \frac{Q(x)}{-t} < 1, \\ -M \leq x \leq -1 &\implies \frac{Q(x)}{-t} \leq -1. \end{aligned}$$

Theorem 3.2.8(*c implies b*) gives the desired result. ∎

The error is greater than 1, so this is not quite good enough. However, there is a general way to reduce the error.

Lemma 3.2.10 *Let S be an (ϵ, M) sign-approximator of degree n and $0 < \alpha < \epsilon$. If k is the least odd integer greater than or equal to $\log(1+\frac{2}{\alpha})/\log(1+\frac{2}{\epsilon})$. Then there is an (α, M) sign-approximator of degree kn.*

Proof By Lemmas 3.2.6 and 3.2.7, we may assume that

$$S(x) = \frac{p(-x) - p(x)}{p(-x) + p(x)}.$$

Then

$$S(x) = 1 + \frac{2}{R(x) - 1},$$

where $R(x) = -p(-x)/p(x)$.

Let k be as in the statement of the lemma, and let $\hat{R}(x) = (R(x))^k$. Since S is an (ϵ, M) sign-approximator $R(x) > 1 + (2/\epsilon)$ whenever $p(x) \neq 0$. Thus, $\hat{R}(x) > 1 + (2/\epsilon)$ for such x. Since $\hat{R} = -p^k(-x)/p^k(x)$, Theorem 3.2.8(*a implies b*) implies the existence of the desired approximator. ∎

3.3 The Error Term in Approximating *sign*(x).

In Section 2.3, we defined the rational function $S_n(x)$, which approximates the *sign* function for $1 \leq |x| \leq 2^n$. The error bound given there was sufficient for our purposes, but very weak. In this section we give a better analysis of the error term.

Recall the definition of P_n and S_n:

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

Let $E_n(x) = |S_n(x) - sign(x)|$ be the error term. Since both S_n and *sign* are odd functions, we need only consider the error term for positive x. As we saw in Theorem 3.2.8 a lower bound on $R_n(x) = \frac{-P_n(-x)}{P_n(x)}$ will yield an upper bound on the error term.

Theorem 3.3.1 *For sufficiently large n ($n \geq 19$ suffices),*

$$1 < x < 2^n \Longrightarrow E_n(x) < \frac{1}{476.5}.$$

Proof Let $b_n = \prod_{i=1}^{n}(2^i+1)/(2^i-1)$. Since $(2^n+1)/(2^n-1)$ decreases exponentially fast to 1, the $\lim_{n\to\infty} b_n$ exists. For n sufficiently large ($n \geq 18$ suffices), $b_n \geq 8.2559$. We will show that for n sufficiently large, $R_n \geq 14b_{n-1}^2$, which will give the desired bound.

We consider two cases, $1 < x < 2$ and $2 < x < 2^n$ (we need not consider $x = 2$, since $E_n(2) = 0$).

Case I: $1 < x < 2$

$$
\begin{aligned}
R_n(x) &= \frac{-P_n(-x)}{P_n(x)} \\
&= \frac{x+1}{x-1}\prod_{i=1}^{n}\left(\frac{x+2^i}{x-2^i}\right)^2 \\
&= \frac{(x+1)(x+2)^2}{(x-1)(x-2)^2}\left[\prod_{i=2}^{n}\left(\frac{2^i+x}{2^i-x}\right)\right]^2 \\
&\geq \frac{(x+1)(x+2)^2}{(x-1)(x-2)^2}\left[\prod_{i=2}^{n}\left(\frac{2^i+1}{2^i-1}\right)\right]^2
\end{aligned}
$$

$$= \frac{(x+1)(x+2)^2}{(x-1)(x-2)^2}\left(\frac{1}{9}\right) b_n^2.$$

Elementary calculus shows that for $1 < x < 2$, the function $f(x) = \dfrac{(x+1)(x+2)^2}{(x-1)(x-2)^2}$ is minimized at $x = \sqrt{8/5}$, and the minimum value is slightly larger than $168.661 > 14 * 9$.

Case II: $2 < x < 2^n$

Suppose that $2 \le 2^j < x < 2^{j+1} \le 2^n$, then

$$\begin{aligned}
R_n(x) &= \frac{-P_n(-x)}{P_n(x)} \\
&= \frac{x+1}{x-1}\prod_{i=1}^{n}\left(\frac{x+2^i}{x-2^i}\right)^2 \\
&= \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left[\prod_{i=1}^{j-1}\left(\frac{x+2^i}{x-2^i}\right)\prod_{i=j+2}^{n}\left(\frac{2^i+x}{2^i-x}\right)\right]^2 \\
&\ge \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left[\prod_{i=1}^{j-1}\left(\frac{2^n+2^i}{2^n-2^i}\right)\prod_{i=j+2}^{n}\left(\frac{2^i+2^j}{2^i-2^j}\right)\right]^2 \\
&= \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left[\prod_{i=1}^{j-1}\left(\frac{2^{n-i}+1}{2^{n-i}-1}\right)\prod_{i=j+2}^{n}\left(\frac{2^{i-j}+1}{2^{i-j}-1}\right)\right]^2 \\
&= \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left[\prod_{m=n-j+1}^{n-1}\left(\frac{2^m+1}{2^m-1}\right)\prod_{m=2}^{n-j}\left(\frac{2^m+1}{2^m-1}\right)\right]^2 \\
&= \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left[\prod_{m=2}^{n-1}\left(\frac{2^m+1}{2^m-1}\right)\right]^2 \\
&= \frac{(x+1)(x+2^j)^2(x+2^{j+1})^2}{(x-1)(x-2^j)^2(x-2^{j+1})^2}\left(\frac{1}{9}\right) b_{n-1}^2.
\end{aligned}$$

Let $a = 2^j$, and let $f(x) = \dfrac{(x+1)(x+a)^2(x+2a)^2}{(x-1)(x-a)^2(x-2a)^2}$. It remains to show that $f(x) \ge 14 * 9 = 126$ for $2 \le a < x < 2a$.

Now, $f(x) \ge 126$ if and only if for $a < x < 2a$,

$$(x+1)(x+a)^2(x+2a)^2 - 126(x-1)(x-a)^2(x-2a)^2 \ge 0. \tag{3.5}$$

Let $x = a(1+t)$ so that the condition $a < x < 2a$ becomes $0 < t < 1$. Substituting into (3.5) and simplifying yields

$$36 + 36a + 60t + 163t^2 + 127t^4 + (69at - 29at^2) + (173at^3 - 242t^3) + (137at^4 - 125at^5) \geq 0.$$

It is easy to see that, for $0 < t < 1$, each of the parenthesized expressions is positive (for the second expression we use the fact that $a \geq 2$), so that (3.5) is verified. ∎

Note: The error term given here does not depend on n. In fact there is an absolute lower bound on the error term for any rational function approximating *sign*. This follows from the following remarkable theorem of Gončar [21].

Gončar's Theorem

(a) For every $M > 1$ there is a rational function $r(x)$ of degree d such that for $1 \leq |x| \leq M$

$$|r(x) - sign(x)| \leq \frac{2}{\exp(\frac{\pi^2 d}{2 \ln M}) - 1}.$$

(b) For all rational functions r of degree d and all $M > 1$ there is a real number x_0 with $1 \leq |x_0| \leq M$ such that

$$|r(x_0) - sign(x_0)| \geq \frac{2}{\exp(\frac{\pi^2 d}{2 \ln M}) - 1}.$$

Applying *(b)* to $r(x) = S_n(x)$, $d = 2n+1$, $M = 2^n$ shows that there is an x_0 with $1 \leq |x_0| \leq 2^n$ such that

$$E_n(x_0) \geq \frac{2}{\exp(\frac{\pi^2(2n+1)}{2n}) - 1} \longrightarrow \frac{2}{\exp(\pi^2) - 1}.$$

We now return to the following question: How large can M be if there exists a polynomial M-intersector of degree d? While we do not have a complete answer to this question, we can prove the following.

Corollary 3.3.2 *If there exists a rational function Q of degree d such that for some $M > 1$ and all x, y with $1 \leq |x|, |y| \leq M$,*

(a) if $x > 0$ and $y > 0$ then $Q(x) + Q(y) > 0$,

(b) if $x < 0$ or $y < 0$ then $Q(x) + Q(y) < 0$.

then $M \leq exp(\frac{5\pi^2 d}{4\ln 2})$.

Proof By Theorem 3.2.9 and Lemma 3.2.10 the existence of an intersector of the given form implies the existence of a $(2/3, M)$ sign-approximator of degree $5d$. By Gončar's Theorem,

$$\frac{2}{3} \geq \frac{2}{\exp\left(\frac{\pi^2 5d}{2\ln M}\right) - 1}.$$

Simple algebra yields the stated bound. ∎

If we could show that the existence of a polynomial or rational M-intersector of degree d implies the existence of a $(2/3, M)$ sign-approximator of degree $f(d)$ then the above argument would show that $M \leq \exp\left(O\left(f(d)\right)\right)$. Although we cannot prove such a result for any f, we conjecture that it is true for $f(d) = O\left(d\right)$.

Chapter 4

Circuits with Threshold Gates

In this chapter we will modify the techniques used in Chapter 2 to study circuits containing threshold gates. The analogue of the closure under intersection is the simulation of the AND of several threshold gates by threshold gate whose inputs are depth-2 boolean circuits whose inputs are the inputs of the original threshold gate.

The ability to combine threshold gates in this way can be extended to show that any constant-depth AND-OR circuit containing a small number of threshold gates (anywhere in the circuit) can be simulated by a circuit consisting of a single threshold gate whose inputs are constant-depth AND-OR circuits. Although the new circuit is larger than the original circuit, when the number of threshold gate in the original circuit is small, the increase in size is quite modest. For example, 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 $2^{n^{o(1)}}$ AND, OR, and NOT gates, $o(\log n)$ threshold gates (in arbitrary positions), and $2^{n^{o(1)}}$ wires can compute the parity of n inputs. Previous lower bounds for circuits with threshold gates were primarily for *threshold circuits*, circuits having *only* threshold gates (see [24, 32, 38]), or for circuits consisting of a single threshold gate whose inputs are constant depth boolean circuits (see [28,22,3]).

4.1 Threshold Gates and Perceptrons

A *threshold gate* with weights $w_1, \ldots, w_n$ and threshold value t is a boolean function of n inputs which outputs TRUE if a certain "weighted sum" of the inputs is greater than the threshold value. Specifically, the n inputs are converted from TRUE/FALSE to n numbers $x_1, \ldots, x_n$, and the gate returns TRUE if $\sum_{i=1}^{n} x_i w_i > t$, and returns FALSE otherwise. In this chapter we will consider threshold gates in which TRUE is converted to 1 and FALSE to -1. Notice that by changing the threshold value we could use any other pair of values to represent TRUE and FALSE. The $\{1, -1\}$ representation will be particularly convenient in this chapter, while the (somewhat more common) $\{1, 0\}$ representation will be used in the next chapter. We will often abuse terminology by referring to the converted values as the actual inputs.

We will restrict our weights to be chosen from $\{-1, 1\}$. The usual definition of threshold gate has no restriction on the size of the weights; although in the definition of TC^0 (see [24]), all weights are assumed to be bounded by some polynomial. An input to a threshold gate with an arbitrary integral weight w can be replaced by $|w|$ copies of the input, each of weight 1 if w is positive or -1 if w is negative. The techniques in this chapter can therefore be applied to threshold gates with polynomially bounded weights.

There is a strong analogy between threshold gates and PP computations. To see this, call a number i "accepting" if $x_i w_i = 1$ and "rejecting" if $x_i w_i = -1$. 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.

Definition 4.1.1 A *perceptron* is a depth-2 circuit consisting of a threshold gate whose inputs are AND gates.

Definition 4.1.2 By a *generalized 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 generalized perceptron to be the number of wires in the circuit. The depth of a generalized perceptron is defined as for general circuits. The *top fanin* of a generalized perceptron is the fanin of the threshold gate.

Building circuits from polynomials and rational functions is similar to building Turing machines. The following two lemmas are analogous to the lemmas of Section 2.1.

Lemma 4.1.3 *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 generalized perceptron whose inputs are the union of the inputs to all the $C_{i,j}$, with top fanin at most $Mf^d\binom{d+k}{k}$, size at most $\left(M + (d+1)2^{d-1}\right) f^d\binom{d+k}{k} + kfs$, 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 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 that determines whether $p(s_1, \ldots, s_k) > 0$. The circuit consists of a threshold of XORs. 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 XORs 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-$(D+2)$ generalized 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 generalized 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 $\left(M + (d+1)2^{d-1}\right) f^d\binom{d+k}{k} + kfs$. ∎

Note. It is also possible to build a depth-$(D+1)$ generalized perceptron that outputs 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 generalized 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 generalized perceptron is $(d+1)M2^d f^d \binom{d+k}{k} + kfs$.

For rational functions, we build a circuit whose output is determined by the *sign* of the rational function.

Lemma 4.1.4 *For $1 \leq i \leq k$, $1 \leq j \leq f$, 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 $s_i = \sum_{1 \leq j \leq f} c_{i,j}$. 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 generalized perceptron whose inputs are the union of the inputs to all the $C_{i,j}$, with top fanin at most $M^2 f^{2d} \binom{d+k}{k}^2$, size at most $\left(M^2 + (2d+1)2^{2d-1}\right) f^{2d} \binom{d+k}{k}^2 + 2kfs$, and depth $D+2$ such that when $r(s_1, \ldots, s_k)$ is defined the generalized perceptron outputs TRUE if and only if $r(s_1, \ldots, s_k)$ is positive.*

Proof Suppose that $r = a/b$ where a and b are polynomials with integer coefficients. By the proof of the above lemma, there is a circuit C_a, consisting of a threshold of XOR gates that outputs TRUE if and only if $a > 0$ (the inputs to the XOR gates are various $C_{i,j}$s). Similarly, there is such a circuit C_b for the polynomial b. We build another such circuit for the product ab. For each pair of XOR gates, one from C_a and one from C_b, we have one new XOR gate whose inputs are the union of the inputs to original XOR gates. The number of wires from a new XOR gate to the new threshold gate equals the product of the number of wires from each old XOR to the old threshold gates. There are at most $\left(f^d \binom{d+k}{k}\right)^2$ new XOR gates. Each has fanin at most $2d$, and at most M^2 wires leading to the new threshold gate. As in Lemma 4.1.3, we convert each XOR to an OR of ANDs. ∎

4.2 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 circuits. We first apply the techniques of the previous section to show that the AND of k generalized perceptrons can be computed by a single generalized perceptron of slightly larger size and

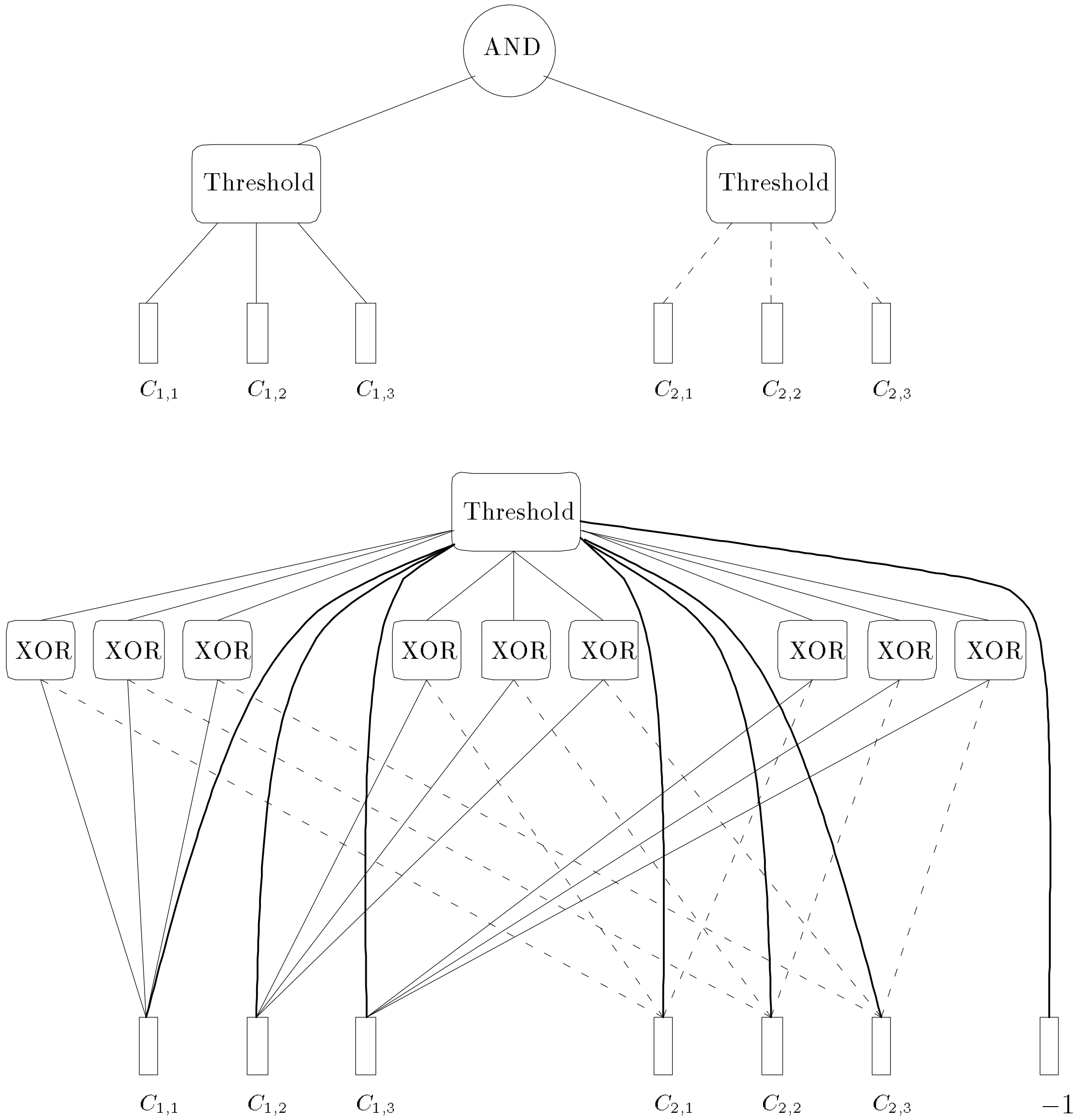

Figure 4.1: The AND of two thresholds as the threshold of XORs. In the bottom circuit, we have used the polynomial $x_1x_2 + 2(x_1 + x_2) - 2$, which works for this small case. The heavier lines (that go directly from the $c_{i,j}$'s or -1 to the threshold gate) represent a *pair* of wires.

depth. Next, we show how to simulate a circuit containing many threshold gates by a generalized perceptron of only slightly higher depth.

Siu, *et al.* [38] have shown that a depth-$(d+1)$ circuit consisting entirely of threshold gates that computes parity must have $\Omega(dn^{1/d}/\log^2 n)$ threshold gates. Combining our simulation result with Green's lower bound on the fanin of a generalized perceptron that 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 Impagliazzo. (Recently, Beigel [7] 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 4.2.1 *Consider k generalized perceptrons having top fanin f, size s, and depth D. The AND of these k generalized perceptrons can be computed via a generalized 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 4.1.4 and Lemma 2.3.4 with $n=\lceil\log f\rceil$. ∎

Corollary 4.2.2 *Consider k generalized perceptrons having top fanin f, size s, and depth D. The OR of these k generalized perceptrons can be computed via a generalized 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 4.2.3 *Consider any threshold circuit C having size s, depth D, and only k threshold gates. We allow the threshold gates to occur anywhere in the circuit. There is a generalized 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 $D+4$ that 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 4.2). 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 4.4(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 4.3).

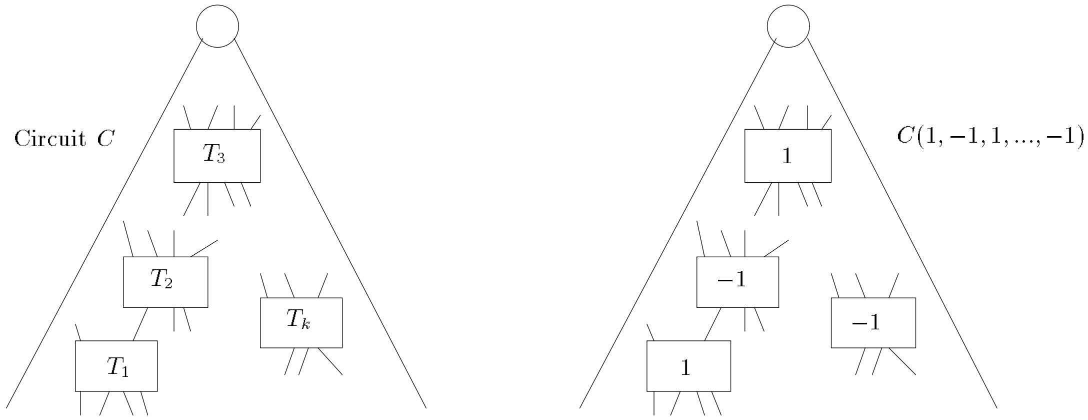

Figure 4.2: The circuit $C(b_1, \ldots, b_k)$ is obtained from the circuit C by replacing each threshold gate T_i by the bit b_i.

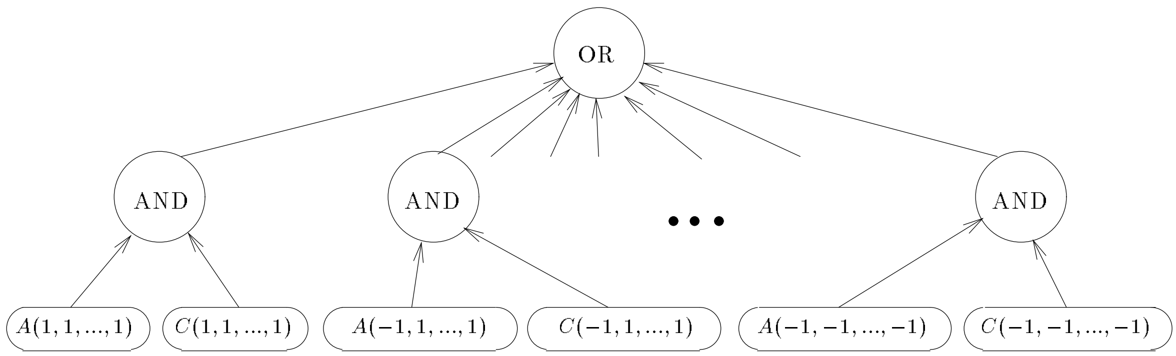

Figure 4.3: The circuit C can be computed as the OR over all bit sequences $b_1, \ldots, b_k$ of the AND of $A(b_1, \ldots, b_k)$ and $C(b_1, \ldots, b_k)$.

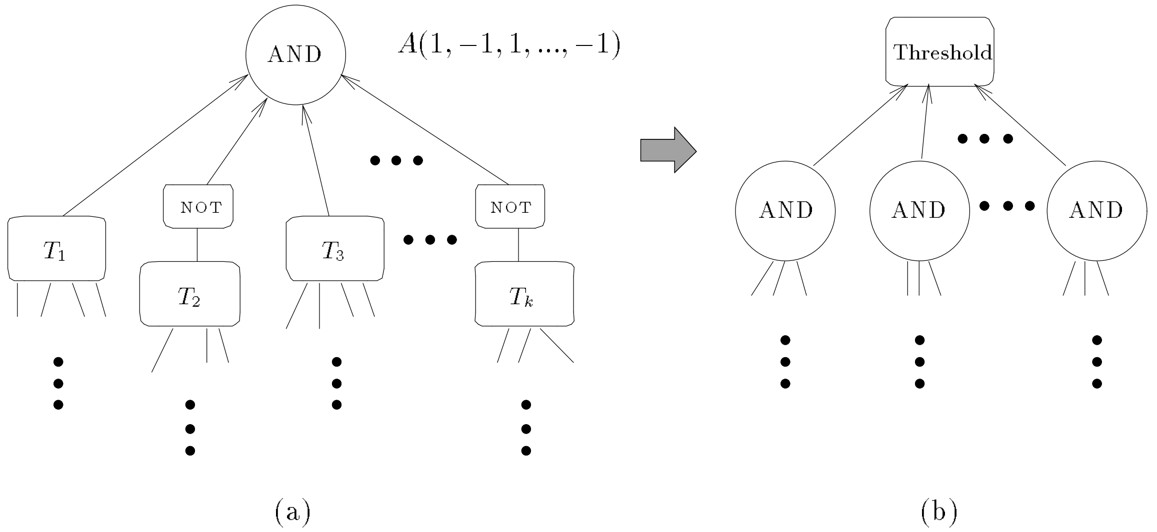

Figure 4.4: The circuit $A(b_1,\ldots,b_k)$. (a) $A(b_1,\ldots,b_k)$ outputs true if threshold gate T_i outputs b_i, for $i=1,\ldots,k$. The inputs to each T_i are AND-OR circuits. (b) $A(b_1,\ldots,b_k)$ is converted to a generalized perceptron by pushing any negations past the threshold gates and then applying Theorem 4.2.1.

Negations can be pushed to the leaves, so $A(b_1,\ldots,b_k)$ can be evaluated as the AND of k generalized perceptrons. Since each of these generalized perceptrons has fanin bounded by s, Theorem 4.2.1 implies that $A(b_1,\ldots,b_k)$ can be computed by a generalized 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 4.4(b)).

The AND of $C(b_1,\ldots,b_k)$ and the generalized perceptron that computes $A(b_1,\ldots,b_k)$ can be computed by a generalized 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 gates. Finally, the OR of these perceptrons can be converted, by Corollary 4.2.2, to a generalized 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 $2^{\mathrm{polylog}\, n}$, our best result is for $O(\log\log n)$ threshold gates.

Corollary 4.2.4 *Consider any threshold circuit C having size $2^{\mathrm{polylog}\, n}$, depth D, and $O(\log\log n)$ threshold gates. There is a generalized perceptron having top fanin $2^{\mathrm{polylog}\, n}$,* *size* $2^{\mathrm{polylog}\, n}$*, and depth* $D+4$ *that computes the same function as* C. ∎

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

Theorem (Green [22]) *For any* $D > 2$ *there exists a constant* c *such that the following is true. Consider any generalized 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 4.2.5 *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 4.2.3, C can be simulated by a generalized perceptron having top fanin $2^{n^{o(1)}}$, size $2^{n^{o(1)}}$, and depth $O(1)$. By Green's theorem, such a generalized perceptron cannot compute parity of n inputs. ∎

Chapter 5

Perceptrons

AC^0 is the class of predicates computed by polynomial-size circuits consisting of unbounded-fanin AND and OR gates, and having depth bounded by some constant. Razborov [33] and Smolensky [39] showed that every AC^0 predicate can be approximated by a low-degree polynomial over any finite field. These upper bounds can be used to obtain circuit lower bounds for computing parity. Building on some ideas and techniques from Håstad's work [25], Linial, Mansour, and Nisan [27] proved an important upper bound: every AC^0 predicate can be approximated by a low-degree polynomial over the reals.

In this chapter we consider approximations by *probabilistic* polynomials. Such polynomials have, in addition to their regular variables, probabilistic variables which are independent and uniformly distributed in $\{0, 1\}$. A function is approximated by such a polynomial if for every input there is high probability that the value of the polynomial is exactly the value of the function. We will show that every AC^0 predicate can be approximated in this sense by a low-degree probabilistic polynomial over the reals.

Our result has an important consequence for neural nets: every AC^0 predicate can be computed by a probabilistic perceptron. This is related to Allender and Hertrampf's [1,2] and Yao's [48] results for circuits with mod-gates.

Our approach is inspired by Allender [1], who showed that every AC^0 circuit can be simulated by a probabilistic depth-2 circuit with a parity gate at the root and $2^{\mathrm{polylog}\, n}$ AND gates of fanin $\mathrm{polylog}\, n$ on the bottom. Allender and Hertrampf [2] extended this result by reducing the number of random bits used. We also build on techniques of Toda and Ogiwara [43].

Finally, our upper bound for AC^0 can be combined with Smolensky's easy proof that parity cannot be approximated by a low-degree polynomial to obtain a new and simple proof that constant-depth boolean circuits computing parity require exponential size (cf. [13,15, 18,25,39,47]).

5.1 Probabilistic Polynomials and Circuits

Definition 5.1.1 A *probabilistic circuit* is a circuit with two types of inputs: actual inputs $x_1, \ldots, x_n$ from $\{0,1\}$, and probabilistic inputs $\rho_1, \ldots, \rho_l$ where each ρ_i is chosen uniformly and independently from $\{0,1\}$. If C is a probabilistic circuit, then for any given $x_1, \ldots, x_n$ we think of the output $C(x_1, \ldots, x_n)$ as a random variable. Let C be a probabilistic circuit, and let f be a function of n inputs. We say that C *computes f with error ϵ* if for every $x_1, \ldots, x_n$, the probability that $C(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ is at least $1 - \epsilon$.

Definition 5.1.2 A *probabilistic polynomial* is a polynomial in two types of variables: actual variables $x_1, \ldots, x_n$ from $\{0,1\}$, and probabilistic variables $\rho_1, \ldots, \rho_l$ where each ρ_i is chosen uniformly and independently from $\{0,1\}$. If P is a probabilistic polynomial, then for any given $x_1, \ldots, x_n$ we think of $P(x_1, \ldots, x_n)$ as a random variable. Let P be a probabilistic polynomial, and let f be a function of n variables. We say that P *computes f with error ϵ* if for every $x_1, \ldots, x_n$, the probability that $P(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ is at least $1 - \epsilon$.

We will make use of the following result due to Valiant and Vazirani [45]:

Theorem 5.1.3 (Valiant-Vazirani [45]) *Let $S \subseteq \{0,1\}^k$ be nonempty. Suppose that $w_0, w_1, \ldots, w_k$ are randomly chosen from $\{0,1\}^k$. Let $S_0 = S$ and let S_i be*

$$\{v \in S : v \cdot w_0 \equiv v \cdot w_1 \equiv \cdots \equiv v \cdot w_i \equiv 0 \pmod 2\},$$

for each $i \in \{1, \ldots, k\}$, where $\cdot$ denotes inner product. Let $\Pr_k(S)$ be the probability that $|S_i| = 1$ for some $i \in \{0, \ldots, k\}$. Then $\Pr_k(S) \geq 1/4$.

Lemma 5.1.4 *For any $\epsilon > 0$, there exists a probabilistic polynomial of degree $O\left(\log(1/\epsilon)\log^2 n\right)$, having n actual variables and $O\left(\log(1/\epsilon)\log^2 n\right)$ probabilistic variables, that computes the OR of the n actual variables with error ϵ.*

Proof Let $b_0, \ldots, b_{n-1}$ be the actual variables and let $k = \lceil \log n \rceil$. Consider elements of $\{0,1\}^k$ as numbers from 0 to $2^k - 1$ represented in binary. Let $S = \left\{v \in \{0,1\}^k : b_v = 1\right\}$, and let the S_i's be as in Theorem 5.1.3. Notice that the S_i depend on the b_i. The bits of each w_i are taken from distinct sets of the probabilistic bits of size k. Notice that $|S_i|$ can be computed as

$$\sum_{v \in \{0,1\}^k} (b_v \wedge (v \cdot w_0 \equiv v \cdot w_1 \equiv \cdots \equiv v \cdot w_i \equiv 0 \pmod 2))). \tag{5.1}$$

Each summand in (5.1) can be computed as a polynomial in $O\left(\log^2 n\right)$ variables, where b_v and each bit in the w_i's are considered separate variables. Thus, for each $i \in \{0, \ldots, k\}$, we can compute $P_i = |S_i|$ as a polynomial with the inputs and the probabilistic bits as its variables. Consider $P = \prod_{i=1}^{k}(1 - P_i)$. P is a probabilistic polynomial with the b_i's as its actual variables and the bits of the w_i's as its probabilistic variables. If every b_i is zero, then every $|S_i| = 0$, so $P(b_0, \ldots, b_{n-1}) = 1$. If at least one b_i is 1, then, by Theorem 5.1.3, $P(b_0, \ldots, b_{n-1}) = 0$ with probability at least 1/4. Therefore P computes the NOR of $b_0, \ldots, b_{n-1}$ with probability at least 1/4.

We can amplify the probability by repeating the test, each time using new probabilistic bits. For each test, we construct a new probabilistic polynomial P that differs only in the probabilistic bits used. Let P' be the probabilistic polynomial obtained by multiplying together $\lceil \log_4(1/\epsilon) \rceil = O(\log(1/\epsilon))$ different such P's. Notice that if every b_i is zero, then $P'(b_0, \ldots, b_{n-1}) = 1$, and if at least one b_i is one, then $P'(b_0, \ldots, b_{n-1}) = 0$ with probability at least $1 - \epsilon$. Since we repeat the test $O(\log(1/\epsilon))$ times, we use $O\left(\log(1/\epsilon)\log^2 n\right)$ probabilistic bits in all. Since P' computes NOR with probability at least $1 - \epsilon$, the polynomial $1 - P'$ computes OR with probability at least $1 - \epsilon$. ∎

Lemma 5.1.5 *For any $\epsilon > 0$ and any function f computed by a depth-d, size-s AND-OR circuit with n inputs, there exists a probabilistic polynomial of degree $O\left(\left(\log(s/\epsilon)\log^2 s\right)^d\right)$, having n actual variables and $O\left(\log(s/\epsilon)\log^2 s\right)$ probabilistic variables, that computes f with error ϵ.*

Proof Change each AND gate in the circuit to the negation of an OR gate of the negations of its inputs. For each OR gate in the circuit, we apply Lemma 5.1.4 to obtain a probabilistic polynomial that computes the value of the gate as a function of its inputs and the probabilistic bits with error ϵ/s. Since each OR gate has fanin at most s, each such probabilistic polynomial has degree $O\left(\log(s/\epsilon)\log^2 s\right)$. The composition of these polynomials has degree $O\left(\left(\log(s/\epsilon)\log^2 s\right)^d\right)$ and computes f with error ϵ. ∎

Definition 5.1.6 A set of gates G is *discerning* if, for any sequence of weights $w_1, \ldots, w_f$, there exists a gate $g \in G$ of fanin f such that for all $x_1, \ldots, x_f$, if $\sum_{i=1}^{f} x_i w_i = 0$ then $g(x_1, \ldots, x_f) = 0$ and if $\sum_{i=1}^{f} x_i w_i = 1$ then $g(x_1, \ldots, x_f) = 1$. For example, the collection of all (weighted) threshold gates is discerning.

Theorem 5.1.7 *For any $\epsilon > 0$, any discerning set of gates G, and any function f computed by a depth-d, size-s AND-OR circuit with n inputs, there exists a probabilistic depth-2 circuit of size $2^{O\left(\log n\left(\log(s/\epsilon)\log^2 s\right)^d\right)}$ composed of a gate from G at the root and AND gates of fanin $O\left(\log n\left(\log(s/\epsilon)\log^2 s\right)^d\right)$ that uses $O\left(\log(s/\epsilon)\log^2 s\right)$ probabilistic bits and computes f with error ϵ.*

Proof By Lemma 5.1.5, there exists a probabilistic polynomial $\hat{P}$ of degree u, where

$$u = O\left(\left(\log(s/\epsilon)\log^2 s\right)^d\right),$$

such that $\hat{P}$ computes f with error ϵ. Any polynomial of degree u in n variables has $2^{O(u\log n)}$ monomials. Each monomial can be computed by an AND gate of fanin u. Thus, f can be computed with error ϵ by a gate from G with these AND gates as inputs. ∎

Definition 5.1.8 AC^0 is the class of predicates computed by constant-depth, polynomial-size circuits consisting of unbounded fanin AND, OR, and NOT gates.

Definition 5.1.9 A *perceptron of size s and order m* is a depth-2 circuit with a threshold gate at the top and s AND gates of fanin m on the bottom.

Note: The definition of the size of a perceptron differs slightly from the definition of the size of a generalized perceptron (Definition 4.1.2).

If we take the discerning set to be all weighted threshold gates, and if the original AND-OR circuit is of polynomial size (*i.e.* an AC^0 circuit) then the resulting probabilistic circuit is a depth-2 probabilistic perceptron of quasi-polynomial size (*i.e.* size $2^{\mathrm{polylog}\, n}$). In fact, even if the original circuit is of quasi-polynomial size, the resulting circuit is a quasi-polynomial size, depth-2 perceptron.

Corollary 5.1.10 *For any polynomial p and any function f computed by a depth-d AC^0 circuit with n inputs, there exists a probabilistic perceptron of size $2^{O\left(\log^{2d+1} n\right)}$ and order $O\left(\log^{2d+1} n\right)$ that uses $O\left(\log^3 n\right)$ probabilistic bits and computes f with error $1/p(n)$.*

Corollary 5.1.11 *For any function f of n inputs computed by a depth-d AND-OR circuit of size-$2^{\log^k n}$, there exists a probabilistic perceptron of size $2^{O\left(\log^{3dk+1} n\right)}$ and order $O\left(\log^{3dk+1} n\right)$ that uses $O\left(\log^{3k} n\right)$ probabilistic bits and computes f with error $1/2^{\log^k n}$.*

Definition 5.1.12 A *symmetric gate* is a gate with inputs $x_1, \ldots, x_n$ that computes $h(\sum_{i=1}^n x_i)$ for some underlying function h. We say that two symmetric gates are of the same *type* if the two underlying functions are the same.

Theorem 5.1.13 *For any $\epsilon > 0$ and any function f computed by a depth-$(d+1)$ circuit of n inputs with a symmetric gate G at the root and m size-s AND-OR subcircuits, there exists a probabilistic depth-2 circuit with a symmetric gate of the same type as G at the root and $2^{O\left(\left(\log(ms/\epsilon)\log^2 s\right)^d\right)}$ AND gates of fanin $O\left(\left(\log(ms/\epsilon)\log^2 s\right)^d\right)$ on the bottom that uses $O\left(\log(ms/\epsilon)\log^2 s\right)$ probabilistic bits and computes f with error ϵ.*

Proof By Lemma 5.1.5, for each subcircuit that computes an input to G, there exists a polynomial of degree $O\left(\left(\log(ms/\epsilon)\log^2 s\right)^d\right)$ that computes the value of the subcircuit with error ϵ/m. As in Theorem 5.1.7, each polynomial can be computed by a weighted sum of AND gates. So, there is a symmetric gate of the same type as G with these AND gates as inputs that computes f with error ϵ. ∎

It is a routine matter to verify that all the polynomials constructed above have coefficients of size at most $2^{\mathrm{polylog}(n/\epsilon)}$; therefore the threshold gates used in the circuits built above have weights of size at most $2^{\mathrm{polylog}(n/\epsilon)}$ as well.

5.2 Parity is not in AC^0: A New Proof

In Chapter 4 we showed that no constant-depth circuit consisting of AND, OR, NOT, and threshold gates, with $2^{n^{o(1)}}$ wires but only $o(\log n)$ threshold gates can compute the parity function of n inputs. There have been many proofs of lower bounds on the size of constant depth boolean circuits with no threshold gates that compute parity [13,4,15,25, 39,47] In this section we present a new proof based on our result result about probabilistic perceptrons. The idea is that if parity is in AC^0 then by Lemma 5.1.5 there must be a low-degree probabilistic perceptron that computes parity with small error. We can then apply an important observation due to Smolensky [39] to show that a certain class of polynomials is smaller than expected. An alternative approach would be to apply the result from [3] that a small order perceptron cannot even approximate parity. We use Smolensky's approach to be more self-contained.

Theorem 5.2.1 *If the parity of n binary inputs is computed by an AND-OR circuit having size s and depth d, then*

$$s = 2^{n^{\Omega(1/d)}}.$$

Proof If such a circuit exists, then by Lemma 5.1.5 there exists a probabilistic polynomial of degree $r = (\log s)^{kd}$, for some constant k, that computes parity with error probability at most $1/4$. Therefore there exists a setting of the random inputs such that the resulting polynomial computes parity on at least $3/4$ of all inputs. Let that polynomial be $q(x_1, \ldots, x_n)$.

Our polynomial has domain $\{0,1\}^n$. It will be more convenient to use domain $\{-1,1\}^n$, however, since over that domain the parity of a set of variables is just the product of the variables. To make the change, we work over the integers mod 3. The function $t(x) = x - 1$ maps -1 to 1 and 1 to 0, so instead of using $q(x_1, \ldots, x_n)$ with domain $\{0,1\}^n$, we take $q(t(x_1), \ldots, t(x_n))$ and domain $\{-1,1\}^n$.

We now apply Smolensky's idea [39]. Consider a polynomial mapping $\{-1,1\}^n$ to the integers mod 3. Note that every input satisfies $x^2 = 1$, so no variable need have degree greater than 1. Thus, any monomial can be written as $m(x_1, \ldots, x_n)p(x_1, \ldots, x_n)$, where m is the product of the variables that do *not* appear in the monomial and $p(x_1, \ldots, x_n)$ computes the parity of all the x_is. If the original monomial had degree greater than $n/2$

this new representation of it has degree at most $n/2$ plus the degree of the parity function. Thus, using q instead of parity, we see that every polynomial can be approximated (*i.e.* computed exactly on at least $3/4$ of all inputs) by a polynomial of degree at most $n/2 + r$.

Suppose the conclusion of the theorem is false. Then for every constant c there are infinitely many n such that $r < n^{ck}$. Pick c so that $ck < 1/2$. From now on we restrict our attention to those n for which $r < n^{ck}$. Thus we may assume that $r = o(n^{1/2})$. We now count the number of distinct functions mapping $\{-1,1\}^n$ into the integers mod 3. On the one hand, there are 3^{2^n} such functions. On the other hand, any such function can be computed by a polynomial, and each such polynomial can be approximated by a polynomial of degree at most $n/2 + r$. We will use this to show that the number of such functions is strictly smaller than 3^{2^n} for sufficiently large n.

Fix a polynomial p mapping $\{-1,1\}^n$ into the integers mod 3. On the $3/4$ of the inputs where the approximation to parity is correct, p must agree with some polynomial of degree at most $n/2 + r$, while on the remaining $1/4$ of the inputs p could be arbitrary. Therefore, the total number of polynomials is at most

$$3^{\frac{1}{4}2^n} 3^m, \tag{5.2}$$

where m is the number of monomials of degree at most $n/2 + r$ with degree at most 1 in each of n variables. Each monomial of degree $i \leq n/2 + r$ is determined by which i of the n variables appear in it, so that

$$\begin{aligned} m &= \sum_{i \leq \frac{n}{2}+r} \binom{n}{i} \\ &\leq 2^{n-1} + \sum_{\frac{n}{2} \leq i \leq \frac{n}{2}+r} \binom{n}{i}. \end{aligned}$$

Since the central binomial coefficient is $O\left(2^n/\sqrt{n}\right)$, we have

$$m = 2^{n-1} + (r+1)O\left(\frac{2^n}{\sqrt{n}}\right) = 2^n\left(\frac{1}{2} + o(1)\right).$$

Substituting this into (5.2) shows that the total number of polynomials is

$$3^{(3/4+o(1))2^n},$$

which, for large n, is smaller than 3^{2^n}. ∎

Bibliography

[1] E. Allender. A note on the power of threshold circuits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 580–584. IEEE Computer Society Press, 1989.

[2] E. Allender and U. Hertrampf. On the power of uniform families of constant depth threshold circuits. In *Proceedings of the 15th International Symposium on Mathematical Foundations of Computer Science*, pages 158–164. Lecture Notes in Computer Science 452. Springer-Verlag, 1990.

[3] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 402–409. Association for Computing Machinery, 1991.

[4] L. Babai. A random oracle separates PSPACE from the polynomial hierarchy. *Information Processing Letters*, 26:51–53, 1987.

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

[6] 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.

[7] R. 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.* Association for Computing Machinery, 1992.

[8] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 1992. To appear.

[9] R. Beigel, L. A. Hemachandra, and G. Wechsung. Probabilistic polynomial time is closed under parity reductions. *Information Processing Letters*, 37:91–94, 1991.

[10] R. Beigel, N. Reingold, and D. Spielman. The perceptron strikes back. In *Proceedings of the 6th Annual Conference on Structure in Complexity Theory*, pages 286–291. IEEE Computer Society Press, July 1991.

[11] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 1–9. Association for Computing Machinery, May 1991. Final version to appear in *Journal of Computer and System Sciences.*

[12] H. D. Block. A review of "perceptron". *Information and Control*, 17:501–522, 1970.

[13] R. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 757–804. MIT Press and Elsevier, The Netherlands, 1990.

[14] D. Braess. *Nonlinear Approximation Theory*, volume 7 of *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, 1986.

[15] J. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. *Journal of Computer and System Sciences*, 38:68–85, 1989.

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

[17] L. Fortnow and N. 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.

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

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

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

[21] A. A. Gončar. Estimates of the growth of rational functions and some of their applications. *Math. USSR Sbornik*, 1(3):445–456, 1967.

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

[23] 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.

[24] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 99–110. IEEE Computer Society Press, 1987. Final version to appear in *Journal of Computer and System Sciences.*

[25] J. Haståd. Almost optimal lower bounds for small depth circuits. In Silvio Micali, editor, *Advances in Computing Research 5: Randomness and Computation*, pages 143–170. JAI Press, Greenwich CT, 1989.

[26] Johannes Köbler, Uwe Schöning, Seinosuke Toda, and Jacobo Torán. Turing machines with few accepting computations and low sets for PP. In *Proceedings of the 4th Annual Conference on Structure in Complexity Theory*, pages 208–215. IEEE Computer Society Press, June 1989. Final version to appear in *Journal of Computer and System Sciences.*

[27] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 574–579, 1989.

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

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

[30] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing.* Association for Computing Machinery, 1992.

[31] R. Paturi. On the degree of polynomials that approximate symmetric boolean functions. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 468–474. Association for Computing Machinery, May 1992.

[32] R. Paturi and M. E. Saks. On threshold circuits for parity. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 397–404. IEEE Computer Society Press, 1990.

[33] A. A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. *Math. notes of the Academy of Science of the USSR*, 41(4):333–338, September 1987.

[34] T. J. Rivlin. *An Introduction to the Approximation of Functions.* Dover Publications, Inc., New York, 1969.

[35] T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *J. SIAM Ser. B Numer. Anal.*, 3(2):311–320, 1966.

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

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

[38] K. Siu, V. Roychowdhury, and T. Kailath. Computing with almost optimal size threshold circuits. Manuscript, 1992.

[39] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 77–82. Association for Computing Machinery, 1987.

[40] J. Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science*. Springer-Verlag, 1991.

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

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

[43] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. In *Proceedings of the 6th Annual Conference on Structure in Complexity Theory*, pages 2–12. IEEE Computer Society Press, July 1991.

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

[45] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

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

[47] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, 1985.

[48] A. C. Yao. On ACC and threshold circuits. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 619–627. IEEE Computer Society Press, 1990.