Chapter 12
Large-Depth Circuits

We now consider circuits of depth \( d \geq 3 \). Out of attempts to prove lower bounds for such circuits, two powerful methods emerged.

The first is a “depth reduction” argument: One tries to reduce the depth one layer at a time, until a circuit of depth 2 (or depth 1) remains. The key is the so-called Switching Lemma, which allows us to replace CNFs on the first two layers by DNFs, thus reducing the depth by 1. This is achieved by randomly setting some variables to constants. If the total number of gates in a circuit is not large enough and the initial circuit depth is small enough, then we will end with a circuit computing a constant function, although a fair number of variables were not set to constants. For functions like the Parity function, this yields the desired contradiction.

The second major tool is a version of Razborov’s Method of Approximations, which we have already seen applied to monotone circuits. Given a bounded-depth circuit for a boolean function \( f(x) \), one uses this circuit to construct a polynomial \( p(x) \) of low degree which differs from \( f(x) \) on relatively few inputs, if the circuit does not have too many gates. This immediately implies a lower bound on the circuit size of any boolean function, like the Majority function, which cannot be approximated well by low-degree polynomials.

12.1 Håstad’s Switching Lemma

Recall that a boolean function is a \( t\text{-CNF} \) function if it can be written as an AND of an arbitrary number of clauses, each being an OR of at most \( t \) literals (variables and negated variables). Dually, a boolean function is an \( s\text{-DNF} \) if it can be written as an OR of an arbitrary number of monomials, each being an AND of at most \( s \) literals.

In the “depth reduction” argument, an important step is to be able to transform \( t\text{-CNF} \) into \( s\text{-DNF} \), with \( s \) as small as possible. If we just multiply the clauses we can get very long monomials, much longer than \( s \). So the function itself may not be an \( s\text{-DNF} \). In this case, we can try to assign constants 0 and 1 to some variables and “kill off” all long monomials (that is, evaluate them to 0). If we set some variable
x_i, say, to 1, then two things will happen: the literal \( \neg x_i \) gets value 0 and disappears from all clauses, and all clauses containing the literal \( x_i \) disappear (they get value 1).

Of course, if we set all variables to constants, then we would be done: no monomials at all would remain. The question becomes interesting if we must leave some fairly large number of variables unassigned. This question is answered by the so-called switching lemma.

Recall that a restriction is a map \( \rho \) of the set of variables \( X = \{ x_1, \ldots, x_n \} \) to the set \( \{ 0, 1, * \} \). The restriction \( \rho \) can be applied to a function \( f(x_1, \ldots, x_n) \), then we get the function \( f_\rho \) (called a subfunction of \( f \)) where the variables are set according to \( \rho \), and \( \rho(x_i) = * \) means that \( x_i \) is left unassigned. Note that \( f_\rho \) is a function of the variables \( x_i \) for which \( \rho(x_i) = * \). We can then apply another restriction \( \pi \) of the remaining variables to obtain a subfunction \( f_{\rho \pi} \) of \( f_\rho \), etc.

Suppose that \( p \) is a real number between 0 and 1. A \( p \)-random restriction assigns each variable \( x_i \) a value in \( \{ 0, 1, * \} \) independently with probabilities

\[
\text{Prob}[\rho(x_i) = *] = p
\]

and

\[
\text{Prob}[\rho(x_i) = 0] = \text{Prob}[\rho(x_i) = 1] = \frac{1 - p}{2}.
\]

Thus, on average, such a restriction leaves a \( p \) fraction of variables unassigned. We will sometimes abbreviate the notation and write \( \text{Prob}[\ast] \) rather than \( \text{Prob}[\rho(x_i) = \ast] \). Note that the probability that more than \( s \) variables remain unassigned does not exceed \( \binom{n}{s} p^s \leq \left(e^p / s\right)^s \). This, in particular, is an upper bound on the probability that \( f_\rho \) cannot be written as an \( s \)-CNF.

The Switching Lemma is a substantial improvement of this trivial observation: if \( f \) is a \( t \)-CNF, then \( f_\rho \) will not be an \( s \)-DNF with probability at most \( (8pt)^s \). Important here is that this “error probability” does not depend on the total number of variables. In fact, we have an even stronger statement (see Exercise 1.7 for why this statement is stronger): \( f_\rho \) will have a minterm longer than \( s \) with at most this probability. Recall that a minterm of a boolean function \( f \) is a minimal (under inclusion) subset of its variables such that the function can be converted into the constant-1 function by fixing these variables to constants 0 and 1 in some way. Let \( \text{min}(f) \) denote the length of the longest minterm of \( f \).

**Switching Lemma.** Let \( f \) be a \( t \)-CNF, and let \( \rho \) be a \( p \)-random restriction. Then

\[
\text{Prob}[\text{min}(f_\rho) > s] \leq (8pt)^s.
\]

This version\(^1\) of the Switching Lemma is due to Håstad (1986, 1989). Somewhat weaker versions of this lemma were proved earlier by Ajtai et al. (1983), Furst et al. (1984), and Yao (1985). All these proofs used probabilistic arguments. A novel, non-probabilistic proof was later found by Razborov (1995), and we present it in the next section. Actually, his argument also yields an upper bound

\(^1\)With a smaller constant 5 instead of 8.