6.841 Advanced Complexity Theory	February 14, 2005
Lecture 4	
Lecturer: Madhu Sudan	*Scribe: Elena Grigorescu*

1 Overview

In this lecture we will present Barrington's Theorem and the proof due to Ben-Or & Cleve. We will also discuss some lower bounds on the depth of a circuit, and introduce the Switching Lemma.

2 Review

Recall the following definitions from the last lecture :

1. **Branching Program(BP)**:
   - DAG with 1 source, 2 sinks labeled 0/1
   - the non-sink vertices are labeled by $\{x_1, \ldots, x_n\}$ and have 2 out-edges labeled 0/1.
2. **Layered Branching Program(LBP)**:
   - the edges go only from one layer to the next
   - the width of a LBP is the max number of vertices in one layer

Notice that the notion of width of a LBP can be connected to that of space in a non-uniform setting. Recall also that constant-width BP's can compute *counting mod k*, but it is not clear if, for example, the *majority function* can be computed by such BP's.

3 Arithmetic Circuits and Register Machines

An **arithmetic circuit** computes an arithmetic function of its inputs. Instead of gates computing boolean operations, we now have gates capable of addition and multiplication over some field of elements (F_2, F_5, etc). Similarly, one can define arithmetic formulas.
A **register machine** is a way of viewing a circuit as a program. For example, the following code represents a register machine with an unbounded amount of space:

$$
\begin{gathered}
G_1 \leftarrow x_1 + x_2 \\
G_2 \leftarrow x_3 * x_4 \\
G_3 \leftarrow x_5 - G_1 \\
\vdots \\
G_m \leftarrow \ldots \\
\vdots
\end{gathered}
$$

We can restrict the amount of memory allowed, by defining registers $R_1, \ldots, R_l$ and writing the circuit in terms of the content of each register at each step. Example:

$$
\begin{aligned}
R_1 &\leftarrow x_1 + x_2 \\
R_2 &\leftarrow x_3 * x_4 \\
R_3 &\leftarrow x_5 - R_1 \\
&\vdots
\end{aligned}
$$

Exercise: Show that if all operations in an l-Register Machine are of the form $R_i \leftarrow R_j + x_l * R_k$ (over F_2), then the Register Machine can be simulated by a width 2^l BP.

In the next section we address the converse problem, namely that of proving that depth $O(d)$ formulas can be converted into arithmetic formulas.

4 Barrington's Theorem, Ben-Or & Cleve's proof

In the '80s, it was conjectured that the *majority of n bits function* does not have $O(1)$-width and poly-size BP's. However, Barrington('86) showed that the conjecture was false.

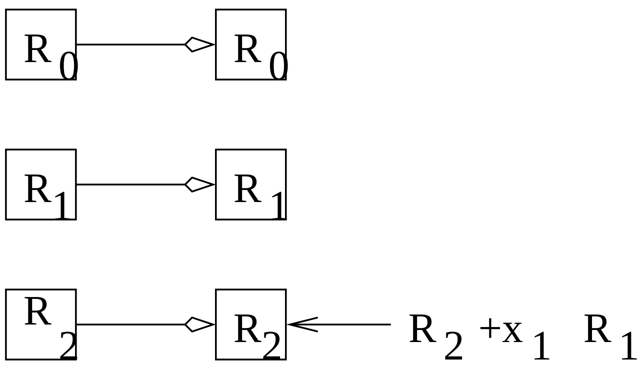
Theorem 1 (Barrington) *Any depth-d formula over $\{NOT, AND, OR\}$ has a BP of size 4^d, and width 5.*

The proof of Barrington's theorem involves the notion of *permutation BP* and uses group-theoretic results. We will omit the original proof in this presentation, and instead show Ben-Or & Cleve's proof of an equivalent result.

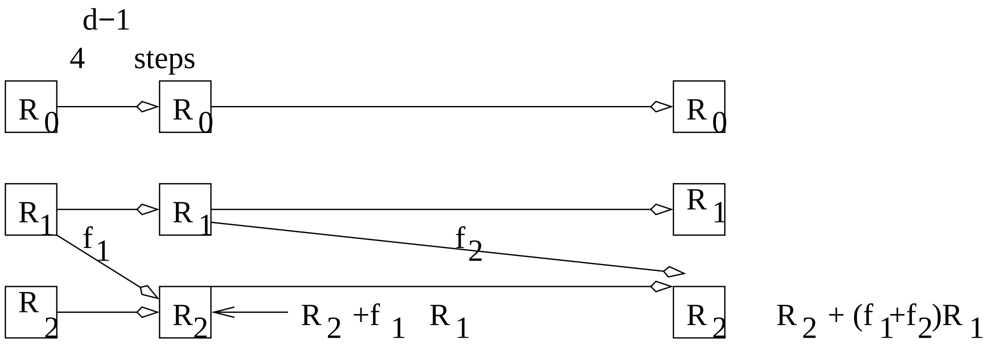
Theorem 2 (Ben-Or, Cleve) *Any depth-d arithmetic formula has a 3-register, 4^d-size Register Machine with instructions of the form $R_i \leftarrow R_j + x_l * R_k$.*

Proof Inductive claim: If f is a depth-d formula, then there exists a 4^d-size machine which starts with R_0, R_1, R_2 and computes $R_2 + fR_1$.

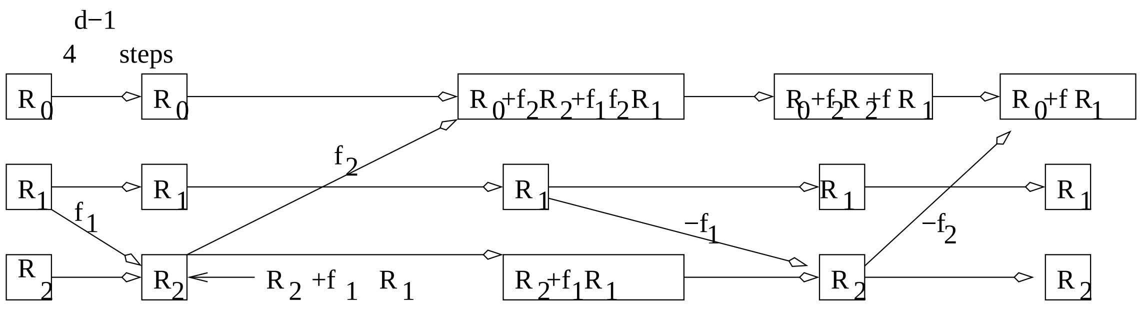
Case: $f(x_1, x_2, \ldots, x_n) = x_1$



Case: $f(x_1, x_2, \ldots, x_n) = f_1(x_1, \ldots, x_n) + f_2(x_1, \ldots, x_n)$, where f_1, f_2 have each depth $d-1$. By induction we know $R_2 + f_1R_1$ and $R_2 + f_2R_1$. The following figure shows the register assignments.



Case: $f = f_1 f_2$ (trickier). The figure below shows how the computation proceeds after each 4^{d-1} steps. Along the way some registers are renamed. Note that the total number of steps of the computation is 4^d.



■

5 Concluding facts about BPs

- We could deduce a powerful upper bound
- We do not know of significant lower bounds
- Open Question: Are poly sized BP's the same as bounded width poly sized BP's?
- Open Question: Is there an explicit function which does not have poly-size BP's?
- Ajtai showed an explicit function that requires super linear depth (time), unless width/size $\geq 2^{\epsilon n}$. In a future lecture we will present a version of this result in the uniform setting.

6 Circuits

In the case of circuits over {2-OR, 2-AND, NOT}-basis, there exists an explicit construction of an f with circuit-size $\geq 4.5n - o(n)$ [Raz, '02].

We are interested in two types of circuits:

- Monotone circuits- those whose output does not decrease as an input changes from 0 to 1.(Eg: the circuit which, given the adjacency matrix of a graph, decides whether the graph has a click of size k).
- $(\infty - OR, \infty - AND, NOT)$-circuits of bounded depth(AC^0-circuits) (Note: in general AC^i circuits have poly size and $(logn)^i$ depth.)

Known results regarding AC^0 circuits:

- AC^0 circuits cannot compute the parity of n bits.
- Furst, Saxe, Sipser ('81) showed that the parity function requires circuits of super poly size.
- Ajtai obtains independently the same results.
- Yao ('85) obtains exponential lower bounds on size.
- Hastad ('87) obtains similar results as Yao, but in a cleaner way, introducing the *Switching Lemma*.
- Razborov-Smolensky prove similar results, using algebraic and probabilistic approaches.

Note: We will discuss the last two papers in future lectures.

7 Switching Lemma

Our goal is to prove that the parity function does not have a small depth circuit.

The idea is that there is a way to convert a DNF- formula(OR of AND's) into a CNF-formula (AND of OR's), and vice versa. However, in this conversion the sizes do not remain proportional. A way of controlling the sizes is by imposing some restrictions on the variables. For example, consider a random restriction obtained by tossing a 3-sided coin, assigning to each variable in a clause the value 0 w.p. $\frac{(1-p)}{2}$, 1 w.p. $\frac{(1-p)}{2}$, and leaving the variable unchanged w.p. p. In this way, we could reduce the depth of the circuit inductively.

We can now introduce the Switching Lemma. Define a k-DNF(k-CNF) to be a DNF(CNF) formula that contains at most k literals in each clause.

Lemma 3 (Switching Lemma) *If F is an l- CNF formula on n variables and we apply to it a random restriction with parameter $p \leq \frac{1}{2}$, then the resulting function has a k-DNF form with probability$\geq 1 - (7pl)^k$.*

We will show a proof of the lemma in the next lecture.