CS5110: Complexity Theory

Shreyas Havaldar
CS18BTECH11042

November 24, 2020

1 Exercise Set 15, Lecture 21

1.1 To show that parity in NC^1

Parity is defined as set of all binary strings with odd number of 1's. So basically we need to find the sum of the total number of ones in the string and return the remainder on division with 2. For $x = x_1 \ldots x_n$

We simply construct a binary tree type of circuit where each Boolean gate is a gate called say P, representing $((\neg a \wedge b) \vee (\neg b \wedge a))$ (ie. the XOR gate), say represented by where a and b are the two inputs to the gate. We have $\log n$ levels in the binary tree where n is the number of bits in the string input, and each higher level receives the output of one two gates from the level exactly below as input and so on until we have only one gate and we output its result as the result of the circuit. This gate

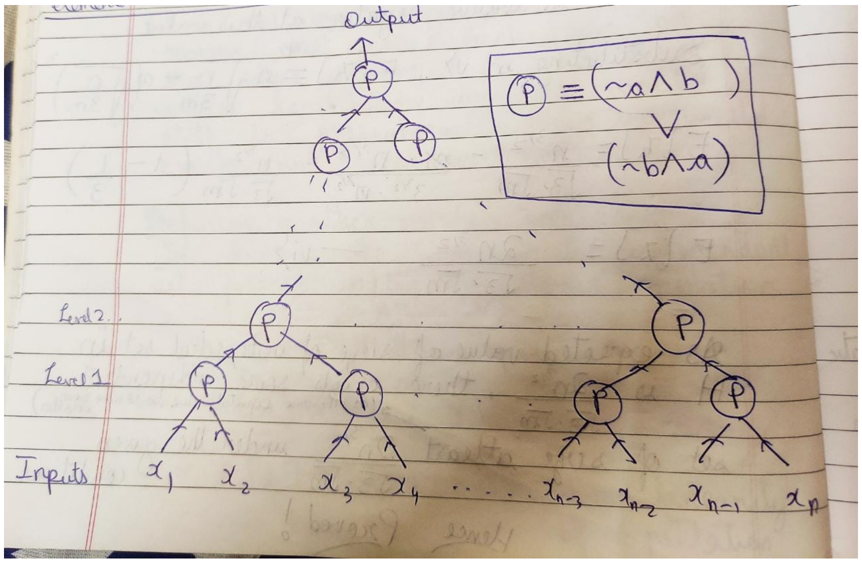

Figure 1: The NC^1 circuit

Boolean function is correct as for both of its inputs being the same, both ones or both zeros it outputs 0, maintaining the parity and for one output that is one and other is zero it outputs one. Thus (a+b) = 1 (mod 2) iff $(\neg a \wedge b) \vee (\neg b \wedge a)$ evaluates to 1.

The circuit is thus of fan-in 2, thus bounded fan-in and constructed using AND, OR and NEGATION and depth $O(\log n)$ and has poly size gates as $(\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \frac{n}{16} \ldots + 1)$ = n-1 gates, thus poly gates and correctly returns the result for parity thus parity $\in NC^1$

1.2 To show that any polynomial over R representing AND requires degree of n

Note: Assuming the question intended n as the degree of polynomial in the function being equal to n and not the number of variables x_i that are being used in the polynomial function being equal to n for the proof.

Assuming we have a function representing AND in degree $d < n$. If the function outputs 1 or 0 for a certain sets of input $x_1, \ldots x_n$, we need show that for any number of variables it cannot be correctly represented by a polynomial of strictly lower degree than n.

We shall proceed by using induction on the number of variables represented in the polynomial. Induction hypothesis is that $f(x_1 \ldots x_n) = x_1 \cdot \ldots \cdot x_n$

For base case when n = 1, we cannot represent in less than 1, that is 0 degree polynomial, as constants and need a linear polynomial in x_1. Assuming $f(x) = ax + b$ where a and b are constants, substituting values for x = 0 and x = 1 make us realise $f(x_1) = x_1$.

For proving our induction hypothesis, for some k, $f(x_1 \ldots x_k) = x_1 \cdot \ldots \cdot x_k$, and now trying to find $f(x_1 \ldots x_{k+1})$, we can show that for no other linear combination on the terms $f(x_1 \cdot \ldots \cdot x_k)$ and x_{k+1} would give us the correct answer by substituting 0 and 1 for x_{k+1} corresponding to $x_1 \cdot \ldots \cdot x_k$ being 0 or 1, so we have essentially 4 cases to check, on $f(x_1 \ldots x_{k+1}) = a \cdot f(x_1 \cdot \ldots \cdot x_k) + b \cdot x_{k+1} + c \cdot f(x_1 \cdot \ldots \cdot x_k) \cdot x_{k+1}$ AND on all these k+1 variables is 1 only when all k+1 variables are 1 and thus only for case when both terms ($f(x_1 \ldots x_k)$ and x_{k+1}) are 1, our function should output 1. For the remaining 3 cases we need the output to be 0.

Likewise we find that a=0, b=0 and c=1 for our outputs to match and no other combination achieves the same results as the AND on these k+1 variables for each possible assignment. Thus $f(x_1 \ldots x_{k+1}) = \cdot f(x_1 \cdot \ldots \cdot x_k) \cdot x_k = x_1 \cdot \ldots \cdot x_{k+1}$ indeed!

So for some input configuration we have an incorrect mapping between the poly function $f(x_1 \ldots x_n)$ and the AND gate, if the degree is less than n, and our induction hypothesis is correct and indeed we require n degrees to represent AND of n variables.

Thus AND cannot be represented in a polynomial of strictly less than n degree, hence proved!