CS221: Computational Complexity Prof. Salil Vadhan

Lecture 22: **AC**

11/15 *Scribe: Arthur Rudolph*

Contents

1 Announcements **1**

2 Recap of NC **1**

3 AC **2**

4 Computing PARITY **2**

1 Announcements

- Pick up survey and solution set 3.
- Section monday as 6pm.
- No office hours for Salil monday (11/18)

2 Recap of NC

Recall that we defined

$$\mathbf{NC_k} = \text{uniform circuits of depth } O(\log^k n) \text{ and poly size}$$

Last time we showed:

Theorem 1

1. $\mathbf{NC}_1 \subseteq \mathbf{L}$
2. $\mathbf{NL} \subseteq \mathbf{NC_2}$

Corollary 2

$$\cup_k \mathbf{SPACE}(\log^k n) = \cup_k \mathbf{DEPTH}(\log^k n),$$

where $\mathbf{DEPTH}(\log^k n)$ *denotes languages decided by uniform circuits of depth* $O(\log^k n)$*. (This is not the same as* $\mathbf{NC_k}$ *for* $k > 1$ *because there is no polynomial bound on the size of the circuits.)*

Proof of Corollary: From 1 above and the fact that size $\leq 2^{\text{depth}}$, we have

$$\mathbf{DEPTH}(\log n) = \mathbf{NC_1} \subseteq \mathbf{SPACE}(\log n)$$

Via padding, we conclude

$$\mathbf{DEPTH}(\log^k n) \subseteq \mathbf{SPACE}(\log^k n)$$

Note additionally this shows us that $\mathbf{NC_2} \subseteq \mathbf{L}^2$, so that we have $\mathbf{NL} \subseteq \mathbf{NC_2} \subseteq \mathbf{L}^2$, a strengthening of Savitch's Theorem.

From 2 above we have that:

$$(\mathbf{N})\mathbf{SPACE}(\log n) \subseteq \mathbf{NC_2} \subseteq \mathbf{DEPTH}(\log^2 n),$$
$$\textit{so } \mathbf{SPACE}(\log^k n) \subseteq \mathbf{DEPTH}(\log^{2k} n)$$

■

In conclusions, circuit depth, space, and formula size are all closely related complexity measures.

3 AC

Given the depth restrictions we have looked at, it is natural to ask about circuits restricted to constant depth. Because we have restricted the fan-in (i.e. the number of input to each gate) to a constant in all of our circuits, this also means the circuits have constant size, and thus the output can depend on only a constant number of the input bits. Because of this, it is not a very useful class, as typically we want to compute functions that depend on all of their input.

Instead we consider circuits with *unbounded fan-in* $\wedge$ and $\vee$ gates (in addition to $\neg$ gates of fan-in 1). We define

$$\mathbf{AC_k} = \text{uniform, unbounded fan-in circuits of poly-size and depth } O(\log^k n)$$

The "**AC**" stands for "Alternating circuit." The reason for this name is that we can think of $\wedge$ nodes as analogous to universal configurations in an alternating TM and $\vee$ nodes with existential configurations. In this way **AC** can be seen as the polynomial hierarchy, scaled down by an exponential.

Proposition 3 $\mathbf{NC_k} \subseteq \mathbf{AC_k} \subseteq \mathbf{NC_{k+1}}$ *(In particular,* $\mathbf{AC} = \mathbf{NC}$*.)*

To see this, realize that because of the polynomial size restriction, the fan-in to any node is equal to some $m \leq \textit{poly}(n)$. We can expand this node to a binary tree of depth $\log m = O(\log n)$.

Going back to the question of constant-depth circuits, constant-depth circuits with unbounded fan-in *are* interesting; in particular, they can compute every function (albeit with exponential size), because we can use the CNF representation and use our unbounded fan-in to use depth two, with just one "and" node at the top collecting all the clauses.

4 Computing Parity

The Parity function is $\text{Par} : \{0,1\}^* \to \{0,1\}$ defined by $\text{Par}(x) = \oplus_i x_i$. It turns out this simple function is very hard for constant-depth (unbounded fan-in) circuits. The smallest constant-depth circuits for this function are essentially given by the following.

Proposition 4 $\forall d$ PARITY *can be computed by a depth d circuit of size* $2^{n^{O(1/d)}}$.

Proof: First set $m = n^{1/d}$. We can think of computing parity recursively, from the top down, with m recursive calls at each level. At the bottom of our recursion, we compute parity on m input nodes by using the CNF representation, using depth two, as above. We then, one level up, can consider the results of the recursive calls as input, and use two more levels to finish the computation on m^2 input nodes with a depth four circuit. In total, the depth of our circuit is $2d$. The size is $O(n2^{n^{1/d}})$. We can change d so as to achieve the bound above. ∎

Now (and next lecture) we will see that the above bound is essentially optimal.

Theorem 5 (Furst–Saxe–Sipser, Ajtai, Yao, Hästad) PARITY *is not in* $\mathbf{AC_0}$. *In fact any family of depth d circuits for* PARITY *requires size* $\geq 2^{n^{\Omega(1/d)}}$

We will see a proof of this due to Smolensky, which uses a technique called the "Approximation Method" for circuit lower bounds. To prove f is hard for circuit class $\mathcal{C}$ we show two things:

1. Any function computed by a small circuit for $\mathcal{C}$ can be approximated by a "simple" function (i.e. one that has more structure, so that we can say more concrete things about them).

2. f cannot be approximated by any simple function.

Through this method, we've gotten circuits, which are hard to discuss, out of the picture. (This method has been used to prove other lower bounds, notably an exponential lower bound on monotone circuits for the CLIQUE problem [Razborov, in Papadimitriou 14.4]).

As it concerns us, $\mathcal{C}$ above means constant depth circuits with unbounded fan-in, and "simple" functions are low degree polynomials. Specifically, they are polynomials over $\mathbb{Z}_3 = \mathbb{Z}/(3)$ (the integers modulo 3). We say that $g : \mathbb{Z}_3^n \to \mathbb{Z}_3$ *α-approximates* $f : \{0,1\}^n \to \mathbb{Z}_3$ if

$$\frac{\#\{x \in \{0,1\}^n \mid f(x) = g(x)\}}{2^n} \geq \alpha$$

Notice we only measure the approximation on $\{0,1\}^n \subset \mathbb{Z}_3^n$.

Fact 6 *Every function $f : \mathbb{Z}_3^n \to \mathbb{Z}_3$ is a polynomial.*

Proof: This follows because for any $(a_1, \ldots, a_n) \in \mathbb{Z}_3^n$, we can define the polynomial:

$$\begin{aligned}\delta_{(a_1,\ldots,a_n)}(x_1, \ldots, x_n) &= \prod_{i=1}^{n} \prod_{a \in \mathbb{Z}_3 \setminus \{a_i\}} \frac{(x_i - a)}{(a_i - a)} \\ &= \begin{cases} 1 & (x_1, \ldots, x_n) = (a_1, \ldots, a_n) \\ 0 & \text{o.w.} \end{cases}\end{aligned}$$

Note that $\delta_{\vec{a}}$ is a polynomial of total degree at most $2n$. We can now write any function f as a linear combination of the $\delta_{\vec{a}}$'s as follows:

$$f(\vec{x}) = \sum_{\vec{a} \in \mathbb{Z}_3^n} f(\vec{a})\delta_{\vec{a}}(\vec{x}).$$

(This is a form of Polynomial Interpolation, a result attributed to Lagrange though originally published by Waring.) ∎

The first step of the approximation method for our problem is given by the following:

Lemma 7 *If $f : \{0,1\}^n \to \{0,1\}$ is computed by a depth d, size s, unbounded fan-in circuit, then $\exists$ a polynomial $g : \mathbb{Z}_3^n \to \mathbb{Z}_3$ of degree $(\log s)^{O(d)}$ which 99%-approximates f.*

Consider following special case of the function $\mathrm{AND}(x_1, \ldots, x_n)$. An error-less polynomial for this is the monomial $x_1 x_2 \cdots x_n$. This has degree n, so we need some way to decrease our degree by lessening our accuracy in some controlled way.