CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 10

March 3, 2010 **Scribe: Jonathan Ullman**

1 Circuit Size Bounds

Proposition 1 *Every function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of size $O(n2^n)$ over the basis $S = \{\wedge, \vee, \neg\}$.*

Proof: We can write the truth table of f in DNF:

$$f(x) = \bigvee_{\alpha : f(\alpha)=1} \left(\bigwedge_i (x_i = \alpha_i) \right)$$

and observe that this formula has size $O(n2^n)$. ■

It is possible to improve this bound to compute any boolean function by circuits of size $(1 + o(1))2^n/n$. The next result will show that the improved bound is tight.

Theorem 2 *For every n, there exists a function $f : \{0,1\}^n \to \{0,1\}$ with $\text{size}_{B_2}(f) \geq (1 - o(1))2^n/n$, where B_2 is the basis consisting of all binary boolean gates.*

Proof: The proof of this theorem is a counting argument. The number of boolean functions on n inputs is 2^{2^n}. We can also bound the number of circuits of size S over B_2:

$$\#\text{ circuits of size S} \leq (S^2)^S \cdot |B_2|^S \cdot S^n \cdot S \cdot \frac{1}{S!}$$

The first term comes from choosing the inputs to each gate, the second is the number of possible operations for each gate, the third and fourth are the number of choices for the input and output nodes, and the final term is because the indexing of the gates in the circuit is arbitrary, so we divide by $S!$ to avoid over-counting isomorphic circuit diagrams. Continuing our calculation, and choosing $S = (1 - \epsilon)2^n/n$ we get

$$\#\text{ circuits of size S} \leq 2^{S \log S + O(S + n \log S)} \leq 2^{(1-\epsilon)2^n + O(2^n/n)} \ll 2^{2^n},$$

even for $\epsilon = c/n$ for a sufficiently large constant c. ■

Examining the proof further, we can actually conclude that almost every boolean function requires exponential-sized circuits, but we do not know any such function in **NP**. In fact, the best known lower bounds for circuit complexity of functions in **NP** is $cn + o(n)$ where $c = 5$ or 6 depending on whether you count the inputs. (Our definition does count the input nodes in the size.)

2 Circuit Depth Bounds

Circuit depth is naturally related to the complexity of parallel computation, and is also interesting for complexity theory because we know non-trivial lower bounds for bounded-depth circuits.

We can define depth properties analogously to size properties. Specifically we write:

- depth(C) = length of the longest path from any input to any output.
- $\text{depth}_B(f)$ = the minimum depth of any circuit over basis B that computes f.
- $\mathbf{DEPTH}_B(d(n))$ = the class of functions computable by circuits of depth $O(d(n))$.

For circuits with fan-in 2 we have $\text{depth}(C) \leq \text{size}(C) \leq 2^{\text{depth}(C)+1}$ and either of these inequalities can be tight. For instance a circuit with one computation path satisfies $\text{depth}(C) = \text{size}(C)$ and a balanced tree satisfies $\text{size}(C) = 2^{\text{depth}(C)+1} - 1$. As a result, we can get a corollary to Theorem 2 for depth.

Corollary 3 *For every n, there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that* $\text{depth}_{\text{B}_2}(\text{f}) \geq n - \log n - O(1)$. *Every function can be computed in depth $n + \log n + O(1)$ (via the DNF representation).*

3 Boolean Formulas

Definition 4 *A* boolean formula *is a boolean circuit where every gate has fan-out* 1 *and inputs can be copied multiple times.*

We can define fsize(C) and fsize(f) for boolean formulas analogously to size(C) and size(f) for circuits. Since we can't "reuse" outputs in a boolean formula the way we can in a boolean circuit, we can get a tighter relationship between size and depth:

Theorem 5 *For every $f : \{0,1\}^n \to \{0,1\}$ over basis B_2*

1. $\mathit{fsize}(f) \leq 2^{\mathit{depth}(f)+1}$
2. $\mathit{depth}(f) \leq O(\log \mathit{fsize}(f)) + O(1)$

Proof: For 1, given a boolean circuit, traverse the tree making copies of any subtree that gets reused. This transformation will result in a boolean formula of the same depth, so we can use the same bound we had for circuits.

For 2, given a formula F, we want to construct an equivalent circuit of small depth. Let $d(S)$ be the smallest depth that will suffice for every formula of size S.

Claim 6 $d(S) \leq d(2S/3) + O(1)$

Proof: [Claim 6] First we observe that given any boolean formula of size S, we can always find a gate v such that the subformula rooted at v has size in the interval $[S/3, 2S/3 + 1]$. To see this, start at the root and move to the child rooting the larger subtree until we reach a subtree whose size is in the interval $[S/3, 2S/3+1]$. If the size of the subformula we choose is m, then the previous subformula is of size at most $2m + 1$, so we can't skip over the entire interval $[S/3, 2S/3 + 1]$.

Consider a gate v and the subformula F_v. Now consider the formulas $F\mid_{v=0}$ and $F\mid_{v=1}$. Then we have

$$F = (F_v \wedge F\mid_{v=1}) \vee (\neg F_v \wedge F\mid_{v=0})$$

with all three subformulas of size at most $2S/3+1$ and the depth of the formula F is the maximum over the three subformulas plus a constant number of additional layers. ■

Part 2 follows directly from the recursion in claim 6. ■

4 Simultaneous Size and Depth Bounds

We can define restricted classes of circuits with simultaneously bounded size and depth:

- $\mathbf{NC^k}$ = {Languages decidable by circuits of size poly(n) and depth $O(\log^k n)$}
- $\mathbf{NC} = \bigcup_k \mathbf{NC^k}$
- $\mathbf{AC^k}$ = {Langauges ... circuits of size poly(n) and depth $O(\log^k n)$ with *unbounded fan-in* $\wedge, \vee$ gates}
- $\mathbf{AC} = \bigcup_k \mathbf{AC^k}$

In general people are not consistent about uniform vs. non-uniform **AC** and **NC** classes, so we will try to be explicit about it. Uniform **NC** was a popular theoretical model for parallel computation in the 80s, since it consists of the problems solvable on parallel computers in polylogarithmic time, using polynomially many processors (with either a shared memory or an efficient communication network). Now that there is a renewed interest in parallel computation, due to the advent of multicore machines, there may be a renewed interest in **NC**.

We have the following relationship between $\mathbf{NC^k}$ and $\mathbf{AC^k}$ classes.

Proposition 7 $\mathbf{NC^k} \subseteq \mathbf{AC^k} \subseteq \mathbf{NC^{k+1}}$.

Proof: The first inclusion is by definition. The second inclusion is because, given an $\mathbf{AC^k}$ circuit on n inputs, we can replace an $\wedge$ or $\vee$ gate of unbounded fan-in with a binary tree of gates of constant fan-in, with depth $O(\log n)$. ■

The above inclusions hold for both uniform and non-uniform classes.

Let's look at the low levels of this hierarchy:

- $\mathbf{NC^0}$ = functions that depend on only a constant number of input bits. Not very interesting for languages, but it turns out to be quite rich functions with many output bits. Indeed, a recent breakthrough (Applebaum-Ishai-Kushilevitz 2004) showed that many cryptographic primitives can be realized in $\mathbf{NC^0}$.
- $\mathbf{AC^0}$ = can compute functions that depend nontrivially on all input bits. However, it is known that Parity and Majority are not in (even nonuniform) $\mathbf{AC^0}$. These beautiful results are close to the frontier of circuit lower bounds.
- $\mathbf{NC^1} = \mathbf{DEPTH}(O(\log n)) = \{\text{poly} - \text{sizedformulas}\}$. Very rich. We don't know how to prove that $\mathbf{NP} \neq \mathbf{NC^1}$. The best known formula-size lower bound for an explicit function is roughly n^2 (note that this is much better than the linear lower bounds known for circuit size).

We can also relate **NC** and **AC** to space-bounded computation in the following way

Theorem 8 *1.* uniform $\mathbf{NC^1} \subseteq \mathbf{L}$

2. $\mathbf{NL} \subseteq$ uniform $\mathbf{AC^1} \subseteq$ uniform $\mathbf{NC^2}$

Proof: For part 2 we state without proof that it is sufficient to show that PATH $\in$ uniform $\mathbf{AC^1}$. (A bit more work is needed to complete the proof, since it is not obvious that $\mathbf{AC^1}$ is closed under logspace reductions.)

To demonstrate that we can solve PATH with uniform $\mathbf{AC^1}$ circuits, consider solving connectivity by computing the n-th boolean power of the adjacency matrix, A^n. We can compute any entry of $A \cdot B$ using an $\mathbf{AC^0}$ circuit and we can compute an entry of A^n using $O(\log n)$ applications of that $\mathbf{AC^0}$ circuit, which would give us an $\mathbf{AC^1}$ circuit.

For part 1 we can demonstrate the following procedure: On input x of length n, we can "construct" in logspace a circuit C_n of depth $O(\log n)$, where as usual, instead of actually constructing the circuit, we reproduce the part of the circuit we need every time, to avoid using too much space. Now we hardwire the input x into $C_n(x)$ to get a circuit with no variables. Then we convert this to a polynomial-sized formula in prefix notation ($F = \mathrm{op}F_1F_2$). Finally we recursively evaluate the formula F using $O(1)$ space per level of recursion and $O(\log n)$ depth, so $O(\log n)$ total space. ■