Lecture 8

Circuit Reducibility, Depth Reduction, and Parallel Arithmetic

April 24, 2008
Lecturer: Paul Beame
Notes: Gyanit Singh

The following questions are open.

- Is $\mathsf{P} \subseteq \mathsf{NC}^1$?
- Is $\mathsf{NP} \subseteq \mathsf{NC}^1$?
- Is $\mathsf{PH} \subseteq \mathsf{NC}^1$?
- Find an explicit $f \in \mathsf{NP}$, such that $f \notin \mathsf{SIZE\text{-}DEPTH}(O(n), O(\log n))$.

Definition 8.1 (Circuit reduction) Given a circuit complexity class C and functions $f, g : \{0,1\}^* \to \{0,1\}^*$ we say that $f \leq_{\mathsf{C}} g$ iff $\exists$ circuit family meeting the structural constraints of the circuit complexity class C that computes f using gates for g added to the basis. (This means that gates for g will have unit size and depth cost except for classes where the number of input edges between gates are counted.)

Theorem 8.2 $f \leq_{\mathsf{log}} g \Rightarrow f \leq_{\mathsf{NC}} g$.

Proof $L \subseteq \mathsf{NC}^2$. □

Definition 8.3 (P Complete) L is P-complete if and only if

1. $L \in \mathsf{P}$
2. $\forall\, K \in \mathsf{P}$, $K \leq_{\mathsf{NC}} L$.

Theorem 8.4 If L is P-complete and $L \in \mathsf{NC}$ then $\mathsf{P} \subseteq \mathsf{NC}$.

Theorem 8.5 (Ladner) The circuit-value problem $CVP = \{\langle C, x\rangle \mid C(x) = 1\}$ is P-complete.

Proof The reduction produces the circuit that computes the standard Cook-Levin tableau. □

Theorem 8.6 If $f \in \mathsf{NC}^1$ then f can be computed by an AC circuit of polynomial size and depth $O(\frac{\log n}{\log\log n})$

Proof By adding dummy nodes, without loss of generality we can assume that all input to output paths in the circuit are the same length. Divide F into layers of height $\log_2 \log_2 n$. Since the fan-in of the original formula is 2, each layer depends on only $2^{\log_2\log_2 n} = \log_2 n$ values from the output the previous layer. Note that this layer can be replaced by a DNF or CNF formula of size $\log_2 n \cdot 2^{\log_2 n+1} = 2n\log_2 n$. Pushing negations to the inputs (and possibly doubling the size of the circuit) reduces the depth by a $\frac{2}{\log_2\log_2 n}$ factor. (By alternating between the use of DNF and CNF the depth could be reduced by a further factor of 2.) $\square$

Theorem 8.7 (Valiant) For all $\epsilon > 0$ if $f \in \mathsf{SIZE\text{-}DEPTH}(O(n), O(\log n))$ then f can be computed by a depth 3 AC circuit of size $2^{O(\frac{n}{\log\log n})}$ with bottom fan-in at most n^ϵ.

Proof Let $C = C_n$ be a circuit of size m and depth $d \le c\log_2 n$. We will find a small set of edges (wires) to cut which will remove all long paths (length $> \epsilon\log_2 n$) in C. Since circuits are fan-in 2, paths of length less than $\epsilon\log_2 n$ implies that each node depends on the value of at most n^ϵ inputs and removed edges. Suppose that S is the set of edges removed. We can write

$$f(x) = \vee_{\alpha\in\{0,1\}^S}\ f_\alpha(x)$$

where α represents the values carried by the edges in S and $f_\alpha(x)$ is 1 if and only if $f(x) = 1$ and the values of the wires in S on input x are equal to α. We claim that each C_α can be computed by a CNF formula of clause size at most n^ϵ. Since the paths are length $\epsilon\log_2 n$, the value of any gate g in C depends on at most n^ϵ inputs or edges in S. Therefore we can write the gate value C_g write this as a CNF formula of bottom fan-in n^ϵ in the values of these inputs or edges from S. Once we fix α, the values of the edge in S is fixed so the resulting CNF formula $C_{g,\alpha}$ depends only on n^ϵ inputs. Abusing notation by identifying edge edge in S with the gate that computes it, we have that $f_\alpha(x) = C_{o,\alpha}(x) \wedge \bigwedge_{g\in S} C_{g,\alpha}(x)$ where o is the output gate of the circuit. We can combine the levels of $\wedge$ gates to get a CNF formula for C_α This yields a depth 3 circuit of size $2^{|S|}(|S|+1)2^{n^\epsilon+1}n^\epsilon$ of bottom fan-in at most n^ϵ.

We will now show how to pick the set of edges S such that $|S|$ is at most $\frac{km}{\log_2\log_2 n}$ and no paths have length more than $d/2^{k-1}$ where k is an integer parameter we are free to choose. Using this we will choose constant k so that $c/2^{k-1} \le \epsilon$. For this choice of S, we have that the size of our circuit, $2^{|S|}(|S|+1)2^{n^\epsilon+1}n^\epsilon$, is $2^{O(n/\log\log n)}$.

Each edge $= (u,v)$ has endpoints at different depths. Classify each edge according to the most significant bit at which the depth of the endpoints differ. Pick the set I to be the k least popular (minimum number of edges assigned to these bits) classes according to this classification. The set S will be all edges whose associated class is in I.

Let's write $\ell = \lceil\log_2 d\rceil$, so $I \subseteq \{0,\dots,\ell-1\}$. Observe that for any edge (u,v) if (u,v) is not in the class for bit i then if we remove the i-th bit from both the bit strings for the depths $d(u)$ and $d(v)$ producing $d(u)_{\bar{i}}$ and $d(v)_{\bar{i}}$, then $d(u)_{\bar{i}} < d(v)_{\bar{i}}$. (Either $d(u)$ and $d(v)$ agree on the i-th bit or they disagree on a more significant bit than bit i.) More generally, if edge (u,v) is not assigned to any bit in set I then $d(u)_{\bar{I}} < d(v)_{\bar{I}}$. Therefore if $v_1,\dots,v_r$ are the nodes along a path in the circuit none of which is classified as a bit in I then $d(v_1)_{\bar{I}} < d(v_2)_{\bar{I}} < \cdots < d(v_r)_{\bar{I}}$. This sequence consists of r distinct bit strings of $\ell - k$ bits so $r \leq 2^{\ell-k} < 2d/2^k = d/2^{k-1}$.

□

Note: For depth 2 circuits we know a size lower bound of $n2^n$ as parity requires CNF and DNF formulas of that size. To make use of Valiant's Lemma [8.7] only requires that we extend this to a similar bound for depth 3.

Last time we saw that $\mathsf{NL} \in \mathsf{NC}^2$. We now see that NC^1 contains many other natural problems.

Theorem 8.8 *Parity* $\in \mathsf{NC}^1$, and Integer-Addition $\in \mathsf{AC}^0$.

Proof The proof for parity is based on the balanced circuits discussed in the last lecture.

We can write sum of two n-bit numbers x, y using the standard binary computation using sum and carry bits. The sum i-th bit of the sum $s_i = x_i \oplus y_i \oplus c_i$ where c_i is the carry bit. The i-th carry bit c_i is 1 iff it is generated in some column j to the right of column i, and is propagated in every column between j and i. That is,

$$c_i = \vee_{j<i}[(x_j \wedge y_j) \wedge \wedge_{j<k<i}(x_k \vee y_k)]$$

which yields a polynomial-size constant depth unbounded fan-in circuit. □

Theorem 8.9 Iterated-Addition of n n-bit integers $\in \mathsf{NC}^1$.

Proof The so-called Wallace tree circuits can achieve this. Observe that For $x + y + z = u + v$ where u and v are $n+1$-bit integers given by expressing $x_i + y_i + z_i$ as the 2-bit binary integer $v_{i+1}u_i$. u and v can be produced from x, y, and z in constant depth fan-in 2 circuits. Using this trick in parallel we can reduce the original sum of n n-bit intgers to the sum of $\lceil 2n/3 \rceil$ integers of $n+1$ bits each. Recursing $\log_{3/2} n$ times we get that in $O(\log n)$ depth and polynomial size the original sum can be reduced equivalent to adding two $n + \log_{3/2} n$-bit integers. We apply the NC^1 translation of the above AC^0 circuit to compute this. □

The usual elementary school method for integer multiplication shows the following and the derived circuits are known as Wallace tree multipliers.

Lemma 8.10 *Multiplication* $\leq_{\mathsf{NC}^0}$ *Iterated-Addition* $\in \mathsf{NC}^1$.

Wallace tree multipliers are conceptually simple but somewhat large. More efficient circuits of $O(\log n)$ size and $O(n \log^2 n)$ depth can be produce using FFTs.

By taking an n-bit input x, producing $x' = x_n 0^\ell x_{n-1} 0^\ell \cdots 0^\ell x_1$ where $\ell > \log_2 n$ and multiplying this by $y' = 10^\ell 10^\ell \cdots 0^\ell 1$ and taking the middle bit of $x' \cdot y'$ we derive the following:

Lemma 8.11 *Parity* $\leq_{\mathsf{NC}^0}$ *Multiplication.*

By using iterated addition on the individual bits and comparing the result to $\lceil n/2 \rceil$ we obtain that *Majority* $\in \mathsf{NC}^1$. In AC^0 we can compare two integers and by counting the results of comparisons using iterated addition and using selection we obtain that *Sorting* $\in \mathsf{NC}^1$.

The last of the basic arithmetic operations is integer division.

Theorem 8.12 (Beame, Cook, Hoover) *Division* $\le_{\mathsf{AC}^0}$ *Iterated-Addition* $\in \mathsf{NC}^1$.

Proof We first show how to reduce integer division to integer powering which is to compute the n-th power of an n-bit integer. Let $2^{r-1} \le y < 2^r$. To determine r we merely need to determine the most significant bit of y which can be done in AC^0. Then

$$\frac{x}{y} = \frac{x}{2^r(1-\hat{y})} = \frac{x}{2^r}(1 + \hat{y} + \hat{y}^2 + \cdots + \hat{y}^{n+1}) + ...$$

Since $\hat{y} \le 1/2$ it suffices to take $O(n)$ terms of this series to compute $\lceil x/y \rceil$. It now remains to compute integer powering using Iterated-Addition.

To do this we will describe a highly non-uniform algorithm since that will be simpler. If x is n bits long then x^n is n^2 bits. Let $p_1, \ldots, p_{n^2}$ be the first n^2 primes. To compute x^n it suffices to compute it modulo $M = \prod_{i=1}^{n^2} p_i$ which is larger than 2^{n^2}. Observe that by the prime number theorem, p_{n^2} is $O(n^2 \log n)$ and therefore only $O(\log n)$ bits long.

One can use the Chinese remainder theorem to compute $x^n \bmod M$ where $M = \prod_{i=1}^{n^2} p_i$ by first computing $a_i \equiv x \bmod p_i$ for $i = 1$ to n^2. We then compute $c_i = a_i^n \bmod p_i$. By the Chinese remainder theorem $x^n \bmod M$ is congruent to $\sum_{i=1}^{n^2} c_i u_i \bmod M$ where u_i is congruent to 1 modulo p_i and 0 modulo p_j for $j \ne i$. (In fact, $u_i = v_i w_i$ where $v_i = M/p_i$ and $w_i = v_i^{-1} \bmod p_i$. We will show that all of these computations can be AC^0 reduced to iterated addition.

First, we can hardwire in the values of M and all the primes since these depend only on the input size n. Second, we can hardwire approximations to $1/p_i$ and using iterated addition compute the product of x and $1/p_i$ to derive each a_i. Assume for now that from the a_i we can compute the c_i values. We can hardwire in the u_i values since these depend only on n. At the end by computing $C = \sum_i u_i c_i$ using a applications of iterated addition to compute each product in parallel and then another iterated addition to compute the sum. we have that $C \equiv x^n \pmod{M}$. However, C may not be between 0 and M. Note, though that $0 \le u_i < M$ so $u_i c_i \le M n^2 \log n$ and thus $0 \le C = \sum_i u_i c_i \le M n^4 \log n$. We can try by subtracting all $n^4 \log n$ multiples of M in parallel to find the value of $C \bmod M$.

It remains to compute each $c_i = a_i^n \bmod p_i$ for each i in parallel. To do so we observe that $\mathbb{Z}_{p_i}^*$ is cyclic with generator g_i since p_i is prime. If $a_i = 0$ we set $c_i = 0$. Otherwise, find b_i such that $a_i = g_i^{b_i} \bmod p_i$ where $0 \le b_i < p$. This is the discrete lagorithm of a_i. Then $c_i = g_i^{nb_i} \bmod p_i$. The circuit will have a table of logarithms for each prime p_i (and its inverse table of exponentiation). The algorithm will find b_i from the logarithm table, compute $d_i = nb_i \bmod (p_i - 1)$ (using hardwired approximations of $1/(p_i - 1)$) and then use the table of exponentiation mod p_i to recover c_i. Lookup tables can be done in AC^0. $\square$

The above circuit is highly non-uniform. It suffices to hard-wire in a single modulus $M = \binom{2m}{m}$ for suitable m that is $O(n^2)$. (One must use prime powers instead of primes in this case. Working modulo prime powers one still has single generators for the multiplicative group except in the case that the prime is 2 in which case one will have two generators.) Subsequently, Chiu, Davida, and Litow showed using multiple Chinese Remainder representations how more complicated versions of these circuits can be made log-space uniform. Finally, Hesse showed how to extend this idea to create highly uniform circuits of constant depth using majority gates.