Lecture 1

Formula Size, Circuit Depth, and Parallel Complexity Classes

April 22, 2008
Lecturer: Paul Beame
Notes: Punyashloka Biswal

Although by Shannon's Theorem almost all functions have exponential circuit size, it is a very difficult to find explicit functions that have large circuit complexity. Although we know by Kannan's Theorem that there are functions in $\Sigma_2^p \cap \Pi_2^p$ that have circuit size $\omega(n^k)$ for any integer k, the best size lower bound known for any explicit function is $5n - o(n)$. However, we can show stronger lower bounds for a more restricted class of circuits.

Definition 1.1 (Formulæ) A *formula* over a basis Ω is a Boolean circuit over Ω that is a tree. The *formula size* of f, denoted $L_\Omega(f)$, is the minimum number of leaves required to compute f in a formula over Ω. When Ω equals $\{\wedge, \vee, \neg\}$, the De Morgan basis, we just write $L(f)$.

Theorem 1.2 (Shannon's Theorem for Formulas) For any finite basis Ω and integer $S \geq n$ there is a constant $c > 0$ such that there are at most c^S functions $f : \{0,1\}^n \to \{0,1\}$ that have formula size $\leq S$. In particular, for any $\epsilon > 0$, finite basis Ω, and sufficiently large n, at least a $1 - \epsilon$ fraction of functions $f : \{0,1\}^n \to \{0,1\}$ have $L_\Omega(f)$ is $\Omega(2^n)$.

Proof As in the corresponding proof for circuits, we count the number of functions represented by formulas of size at most S. Since a chain of unary gates can be replaced by a single

We can remove chains of unary gates by extending the finite basis Ω to some Ω' so htat any formula must have $\leq S$ internal nodes. The tree structure of the formula can be represented by $\leq S$ pairs of nested balanced parentheses, and there are $< 2^{2S}$ such strings. Each node receives one of $|\Omega|$ labels, for a total of $\leq (4|\Omega'|)^S$ possible functions. But the total number of functions $f : \{0,1\}^n \to \{0,1\}$ is 2^{2^n}, so we must have $(4|\Omega'|)^S \geq \epsilon 2^{2^n}$, and the claim follows. □

Definition 1.3 (Restrictions, subfunctions) A *restriction* or partial assignment ρ is a partial function that maps $[n] \to \{0,1\}$, or equivalently, a total function $\rho : [n] \to \{0,1,*\}$. We define $unset(\rho) = \rho^{-1}(*)$.

Given a function $f : \{0,1\}^n \to \{1,0\}$ and a restriction ρ, the *ρ-restriction of f*, denoted $f|_\rho$, is a function mapping $\{0,1\}^{unset(\rho)} \to \{0,1\}$, which for $y \in \{0,1\}^{unset(\rho)}$ is given by $f|_\rho(y) = f(x)$ where

$$x_i = \begin{cases} y_i & i \in unset(\rho) \\ \rho(i) & \rho(i) \in \{0,1\}. \end{cases}$$

Given $B \subseteq [n]$, a *sub-function* of f on B is a function $f|_\rho$ for some ρ such that $unset(\rho) = B$. The set of all subfunctions of f on B is denoted by $SUB(f, B)$.

Theorem 1.4 (Nečiporuk) Let Ω be any finite basis and $B_1, \ldots, B_k$ be a partition of $[n]$. Then there is a constant $c > 0$ such that for any function $f : \{0,1\}^n \to \{0,1\}$,

$$L_\Omega(f) \geq \sum_{i=1}^{k} \log_c |SUB(f, B_i)|.$$

Proof Let Ω' be the set of all subfunctions of elements of Ω (so we don't need constant inputs any more). Let L_i be the number of leaves of F that are labelled by variables x_j such that $j \in B_i$. $L_\Omega(F) \geq \sum_{i=1}^{k} L_i$.

Consider restrictions ρ such that $unset(\rho) = B_i$. We want to count the leaves of a formula F for f. Observe that any restriction $f|_\rho$ has a formula $F|_\rho$ over Ω' that can be obtained by modifying F, propagating the inputs set by ρ. Clearly, $F|_\rho$ has at most L_i leaves. We conclude that $L_{\Omega'}(g) \leq L_i$ for each $g \in SUB(f, B_i)$. Then, by Theorem 1.2, we must have $|SUB(f, B_i)| \leq c^{L_i}$ so $L_i \geq \log_c |SUB(f, B_i)|$ and therefore $L(f) \geq \sum_{i=1}^{k} L_i = \sum_{i=1}^{k} \log_c |SUB(f, B_i)|$. □

Example 1.5 (Element distinctness) We apply this to a natural example. Define the *element distinctness function* $ED : \{0,1\}^n \to \{0,1\}$ as follows. Write $n = 2N \log_2 N$, and let the input string represent a sequence of integers $(x_1, \ldots, x_N) \in [N^2]^N$. Then $ED(x_1, \ldots, x_n) = 1$ iff all the x_i's are distinct.

Take B_i to be the bit positions of x_i, and let ρ be a restriction such that $unset(\rho) = B_i$. If ρ assigns equal values to some pair of variables x_j, x_k for $j \neq k$, then $f|_\rho$ is identically zero. On the other hand, suppose ρ assigns distinct values to all variables x_j for $j \neq i$ and let $T \subset [N^2]$ be the set of these $n - 1$ values. Then $f|_\rho$ is the function that is 1 iff its input does not lie in T. There are $\binom{N^2}{N-1}$ choices for T, so we have

$$|SUB(ED, B_i)| \geq \binom{N^2}{N-1} + 1 \geq \left(\frac{N^2}{N-1}\right)^{N-1} > N^{N-1}.$$

Then, by Nečiporuk's theorem, we have that $L_\Omega(ED)$ is $\Omega(N^2 \log N)$ which is $\Omega(n^2/\log n)$.

This lower bound is essentially the largest possible using Nečiporuk's Theorem.

Formula lower bounds over the De Morgan basis

Necčiporuk's Theorem is the strongest known explicit lower bound for formula size over a general basis (up to the constant factor). There are stronger lower bounds known over the De Morgan basis:

Krapchenko showed that $L(Parity_n)$ and $L(Majority_n)$ are both $\Omega(n^2)$. The proof of this uses so-called "influence" arguments to prove this result, namely that there are larger sets of inputs $A \subset f^{-1}(0)$ and $B \subseteq f^{-1}(1)$ such that that each input in A differs from many inputs in B in precisely one bit. This is tight since $x \oplus y$ can be represented as a De Morgan formula $(\neg x \wedge y) \vee (x \wedge \neg y)$

and simulating a balanced tree of $\oplus$ gates using a De Morgan formula will have $4^{\log_2 n} = n^2$ binary gates.

Using a different technique based on the fact that random restrictions are effective at simplifying De Morgan formulas, Subotovskaya showed an $\Omega(n^{3/2})$ lower bound on computing $Parity_n$.

Andre'ev combined the parity function with the function $g : \{0,1\}^{n+\log_2 n} \to \{0,1\}$ given by $g_{x_1 \ldots x_n}(x_{n+1}, \ldots, x_{n+\log_2 n})$ where $g_{x_1 \ldots x_n} : \{0,1\}^{\log_2 n} \to \{0,1\}$ is the function whose truth table has x_i as its i-th entry. He obtained obtain an explicit function f for which he showed an even larger lower bound, $\Omega(n^{5/2-\epsilon})$. Andre'ev's function f is defined on $(1 + \log_2 n)n$ bits. We can denote these as $x_{i,j}$ for $0 \leq i \leq \log_2 n$ and $1 \leq j \leq n$.

$$f(x) = g_{x_{0,1} \ldots x_{0,n}}(\oplus_{j=1}^{n} x_{1,j}, \ldots, \oplus_{j=1}^{n} x_{\log_2 n, j}).$$

Using CNF or DNF, any function on $\log_2 n$ bits can be computed by De Morgan formulas with $O(n \log n)$ bits. Therefore, using the above $O(n^2)$ size parity formulas so Andre'ev's function has De Morgan formula size $O(n^3 \log n)$. Finally, Goldmann-Haståd showed that Andre'ev's function requires $\Omega(n^{3-\epsilon})$ size De Morgan formulas nearly matching the best algorithm. This is currently the best lower bound in any complete basis for any explicit function.

Formula size versus depth

Theorem 1.6 (Spira) Let Ω be a finite universal basis and $c > 1$ be its maximum fan-in. Let $f : \{0,1\}^n \to \{0,1\}$ be a function. We have

1. $depth_\Omega(f) \geq \log_c L_\Omega(f)$, and
2. if $c \leq 2$, then $depth_\Omega(f) \leq 1 + 2 \log_{3/2} L_\Omega(f)$.

Proof Take a minimum-depth circuit for f and convert it into a formula by repeating subgraphs as necessary to make its fan-out 1; this operation does not alter the depth and gives a size $\leq c^{depth_\Omega(f)}$. The minimum-size formula is at most this big, so the first part follows.

For the second part, we will need the following claim:

Claim: [$\frac{1}{3}$–$\frac{2}{3}$ Lemma] Any tree of maximum degree ≤ 2 has a node whose subtree has between 1/3 and 2/3 fraction of the total number of leaves.

Proof Let the *weight* of a node be the number of leaves in the subtree rooted at that node, and let N be the number of leaves in the entire tree. Consider the following algorithm:

> Walk down from the root. If the current node has weight $\leq 2N/3$, output it. Otherwise, proceed to the child of larger weight and repeat.

This algorithm terminates, because the weights decrease from N to 1 along the path followed by the algorithm. Let v be the node output by the algorithm, and u be its parent. By construction, we know that v has weight $\leq 2N/3$ and u has weight $> 2N/3$. But we also know that v is the child of larger weight, so its weight is at least half of the weight of u. Therefore, the weight of u must lie in $(1/3, 2/3]$. □

We will prove the second part for the De Morgan basis via a recursive construction; the argument for any other universal basis is similar. Let F be a minimum-size formula for f, v be the $\frac{1}{3}$–$\frac{2}{3}$ split node of F and G be the formula rooted at v. If we write F_0 and F_1 for the formulæ obtained by

replacing G with the constants 0 and 1, respectively, we get an equivalent smaller-depth formula for f, *viz.*

$$f = (G \wedge F_1) \vee (\neg G \wedge F_0).$$

Now G, F_0, F_1 each have size at most $2L(f)/3$, so we can perform this operation recursively on each of them (viewing each sub-formula as a function). There will be $\log_{3/2} L(f)$ levels of recursion until these are reduced to constants or variables. The subcircuit above has depth 2 in $\wedge$ and $\vee$ gates per level of recursion. By pushing negations down to the leaves, we get a formula of the desired depth $1 + 2\log_{3/2} L(f)$. □

Corollary 1.7 A function $f\colon \{0,1\}^n \to \{0,1\}$ has formulæ of size polynomial in n if and only if it has circuits of depth $O(\log n)$.

Definition 1.8 (Parallel circuit complexity classes) Let Ω and $\tilde{\Omega}$ be the De Morgan basis with binary and arbitrary fan-in $\wedge$ and $\vee$ gates, respectively.

- NC^k is defined as the set of all functions $f : \{0,1\}^* \to \{0,1\}^*$ such that there exists a circuit family $\{C_n\}_n$ over Ω computing f and satisfying $size(C_n)$ is $n^{O(1)}$ and $depth(C_n)$ is $O(\log n)^k$.
- AC^k is the analogous class with basis $\tilde{\Omega}$.
- $\mathsf{NC} = \bigcup_k \mathsf{NC}^k$.

Note that NC^0 is the set of functions for which each output bit depends on a constant number of input bits. Also NC^1 is the set of functions with polynomial formula size.

We can also define uniform versions of these classes by requiring that the circuit families be efficiently constructible. One natural uniformity notion is log-space constructibility, namely that the function mapping $1^n \mapsto \langle C_n \rangle$ for each n is computable in $O(\log n)$ space by a Turing machine.

The terminology NC stands for "Nick's Class" after Nick Pippenger. It can be expressed as $\mathsf{SIZE\text{-}DEPTH}(n^{O(1)}, \log^{O(1)} n)$, functions computable by polynomial-time, polylog depth circuits, and is the non-uniform analogue of $\mathsf{TIME\text{-}SPACE}(n^{O(1)}, \log^{O(1)} n)$, functions computable of polynomial-time, polylog space Turing machines. The latter class was studied by Steve Cook (who called it PLOPS) which Pippenger called "Steve's Class". Steve Cook returned the favor for the class Pippenger was studying. AC stands for "Alternating Circuits" since the depth of a circuit is the number of alternations between $\wedge$ and $\vee$ gates.

NC can be thought of as the set of problems with efficient superfast *parallel* algorithms. It is *unknown* whether $\mathsf{NP} \subseteq \mathsf{NC}^1$, and whether uniform-$\mathsf{NC} = \mathsf{P}$.

Theorem 1.9 $\mathsf{AC}^k \subseteq \mathsf{NC}^{k+1}$.

Proof A polynomial-fan-in $\vee$ or $\wedge$ gate can be implemented as an $O(\log n)$-depth binary tree of binary $\vee$ or $\wedge$ gates. This replacement blows up the circuit depth by at most $O(\log n)$, so we get an NC^{k+1} circuit from an AC^k circuit. □

Corollary 1.10 $\mathsf{NC} = \bigcup_k \mathsf{NC}^k = \bigcup_k \mathsf{AC}^k$.

Theorem 1.11 $\mathsf{NL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2$.

Proof We shall show that the NL-complete problem $PATH$ lies in AC^1 (and therefore NC^2, by Theorem 1.9. To do this, let A be the adjacency matrix of the graph G, and observe that there is a path from s to t if and only if the (s,t) entry of $(I \vee A)^m$ is 1 for any $m \geq n$ (here the exponentiation refers to repeated application of Boolean $\wedge$-$\vee$ matrix multiply; that is, $C = A \odot B$ iff $c_{ij} = \vee_{k=1}^{n}(a_{ik} \wedge b_{kj})$. For $m = 2^{\lceil \log_2 n \rceil}$, we can compute this quantity by $\lceil \log_2 n \rceil$ rounds of repeated squaring. □

Note that in the above proof, we used unbounded fan-in $\vee$ gates and binary $\wedge$ gates. Define SAC^k (semi-unbounded AC^k) as the class of problems that have polynomial sized circuits of $O(\log^k n)$ depth. Surprisingly, it can be shown that SAC^k is closed under complement. In particular, this means that we can equivalently define SAC^k using a basis of unbounded fan-in $\wedge$ gates and binary $\vee$ gates. It turns out that the context-free language recognition problem $A_{CFG} = \{\langle G, w\rangle \mid G \Rightarrow^* w\}$ can be computed in SAC^1 and the log-space uniform version of SAC^1 can be shown to be equivalent to LOGCFL, the set of functions log-space reducible to A_{CFG}.

Theorem 1.12 uniform-$\mathsf{NC}^1 \subseteq \mathsf{L}$.

Proof The basic idea of the proof is to evaluate circuit C_n of depth $O(\log n)$ on input x using a depth-first search, evaluating the left sub-tree and then the right if needed. The $O(\log n)$ space algorithm will use the log-space circuit constructor to generate the properties of each gate as needed. Since the tree is binary, the depth-first search will only need to maintain the sequence of left and right moves down the tree (i.e. the node name) as well as the current value of the gate being evaluated. This can be done using $O(\log n)$ storage. □