Lecture 3

Circuit Size versus Uniform Complexity

April 8, 2008
Lecturer: Paul Beame
Notes:

The Karp-Lipton theorem gives a conditional result about the relationship between circuit complexity (non-uniform complexity) and uniform complexity. What unconditional properties do we know about circuit complexity and about its relationship to uniform complexity?

Let $\mathbb{B}_n = \{f : \{0,1\}^n \to \{0,1\}\}$, that is, the set of all Boolean functions on n bits. Observe that $|\mathbb{B}_n| = 2^{2^n}$.

Theorem 3.1 (Shannon) "Most" Boolean functions $f : \{0,1\}^n \to \{0,1\}$, have circuit complexity $size(f) \geq \frac{2^n}{n} - \phi(n)$ where $\phi(n)$ is $o(\frac{2^n}{n})$. More precisely, for any $\epsilon > 0$ and any basis $\Omega \subseteq \mathbb{B}_1 \cup \mathbb{B}_2$ there is a function $\phi_\epsilon : \mathbb{N} \to \mathbb{N}$ such that at least a $(1-\epsilon)$ fraction of functions f have $size_\Omega(f) \geq \frac{2^n}{n} - \phi_\epsilon(n)$.

Proof The proof is a by a counting argument. We will show that the number of circuits of size much smaller than $\frac{2^n}{n}$ is only a negligible fraction of $|\mathbb{B}_n|$, proving the claim.

We first compute the number of circuits of with $S \geq n+2$ gates over n inputs with $\Omega = \{\neg, \wedge, \vee\}$. What does it take to specify a given circuit? A gate labeled i in the circuit is defined by the labels of its two inputs, j and k ($j = k$ for unary gates), and the operation g the gate performs. The input labels j and k can be any of the S gates or the n inputs or the two constants, 0 and 1. The operation g can be any one of the three Boolean operations in the basis $\{\neg, \wedge, \vee\}$. Therefore there are at most $(S+n+2)^2 3$ possibilities for each gate. The circuit description also needs to specify the output gate, so any circuit with at most S gates can be specified by a description of length at most $(S+n+2)^{2S}3^S S$ (where we have added dummy gates if the circuit has fewer than S gates).

Note, however, that such descriptions compute the same function under each of the $S!$ ways of naming the gates. Since $S! \geq (S/e)^S$ for any integer S, the number of different functions computed by circuits of size S is at most

$$\begin{aligned}
\frac{(S+n+2)^{2S}3^S S}{S!} &\leq \frac{(S+n+2)^{2S}(3e)^S}{S^S} \\
&\leq \frac{(2S)^{2S}(3e)^S S}{S^S} \qquad \text{since } S \geq n+2 \\
&\leq (12eS)^{S+1}
\end{aligned}$$

Observe that for general $\Omega \subseteq \mathbb{B}_1 \cup \mathbb{B}_2$, we can assume that $|\Omega| \leq 16$ since constant functions and the unary identity function are not needed and we can replace 3 in the above calculation by 16. Therefore the number of such circuits is at most $(64eS)^{S+1}$. If $(64eS)^{S+1} \leq \epsilon 2^{2^n}$ then at least an $(1-\epsilon)$ fraction of functions in $\mathbb{B}_n$ have at size at least S. This holds if $(S+1)\log_2(64eS) \leq 2^n - \log_2(1/\epsilon)$. Now for $S \leq 2^n/n$, we have $\log_2(64eS) \leq n+8-\log_2 n$ and so $(S+1)\log_2(64eS) \leq (S+1)(n+8-\log_2 n)$. Therefore if $S+1 \leq \frac{2^n}{n+8}$ then $(S+1)\log_2(64eS) \leq 2^n - \frac{2^n \log_2 n}{n+8}) \leq 2^n - \log_2(1/\epsilon)$ for n sufficiently large as a function of $1/\epsilon$. The claim follows by observing that $\frac{1}{n+8} = \frac{1}{n} - \frac{8}{n(n+8)}$ which is $\frac{1}{n} - \Theta(\frac{1}{n^2})$. $\square$

Theorem 3.2 (Lupanov) Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ has $size(f) \leq \frac{2^n}{n} + \psi(n)$ where $\psi(n)$ is $o(\frac{2^n}{n})$.

Proof Proof of this part is left as an exercise. Note that a Boolean function f over n variables can be easily computed in using its canonical DNF or CNF representation and so $size(f) \leq n2^{n+1}$. Bringing it down close to $\frac{2^n}{n}$ is a bit trickier. This gives a fairly tight bound on the size needed to compute most Boolean functions over n variables. $\square$

As a corollary, we get a circuit size hierarchy theorem which is even stronger than the time and space hierarchies we saw earlier; circuits can compute many more functions even when their size is tripled.

Corollary 3.3 (Circuit-size Hierarchy). For any $S, S' : \mathbb{N} \to \mathbb{N}$, if $n \leq 3S(n) \leq S'(n) < 2^n/n$, then $\mathsf{SIZE}(S(n)) \subsetneq \mathsf{SIZE}(S'(n))$.

Proof Let $m = m(n) < n$ be the largest integer such that $S'(n) \geq 2^m/m + \psi(m)$ where $\psi(m)$ is defined as in Theorem 3.2 and is $o(2^m/m)$. Since $2^{m+1}/(m+1) + \psi(m+1) > S'(n) \geq 3S(n)$, we have $S(n) < \frac{1}{3}(\frac{2^{m+1}}{m+1} + \psi(m+1)) = \frac{2}{3}(\frac{2^m}{m+1} + \frac{\psi(m+1)}{2}) = \frac{2}{3}(\frac{2^m}{m} - \frac{2^m}{m(m+1)} + \frac{\psi(m+1)}{2}) < \frac{2^m}{m} - \phi(m)$ where $\phi(m)$ is defined as in Theorem 3.1 for $\epsilon = 1/2$. Consider the set F of all Boolean functions on n variables that depend only on the first m bits of their inputs. By Theorem 3.2, all functions in F can be computed by circuits of size $2^m/m + \psi(m) \leq S'(n)$ and are therefore in $\mathsf{SIZE}(S'(n))$. On the other hand, at least $1/2$ of the functions in F cannot be computed by circuits of size $2^m/m - \phi(m) > S(n)$ and are therefore not in $\mathsf{SIZE}(S(n))$. (Note that with the weaker bound size upper bound based on DNF formulas of $m2^{m+1}$ for m-input functions, a similar argument would yield a separation if $S'(n)$ is $\omega(S(n)\log^2 S(n))$.) $\square$

We now consider how these circuit size classes relate to uniform complexity classes. The Cook-Levin Theorem shows how to simulate any algorithm running $\mathsf{TIME}(T(n))$ on inputs of length n by a circuit of size $O(T^2(n))$ with a constant number of circuit elements for each entry of the $T(n) \times T(n)$ tableau for the time $T(n)$ computation. The following Theorem, whose proof we just sketch, shows that a more efficient simulation is possible.

Theorem 3.4 (Fischer-Pippenger) If $T(n) \geq n$ then $\mathsf{TIME}(T(n)) \subseteq \bigcup_c \mathsf{SIZE}(cT(n)\log_2 T(n))$.

Proof The basic idea of the proof is a variant on the Cook-Levin tableau construction. Observe that the only calculations that take place in each row of this tableau involve the constant number of circuit elements that surround the read/write head. The contents of other entries can just be passed along directly to the next row. Unfortunately, in a typical Turing machine running on inputs of size n the position of the read/write head at a fixed time step can vary based on the input string. We say that a multitape Turing machine is *oblivious* if and only if the positions of its read/write heads only depends on the time step but not on its actual input.

It turns out that Hennie and Stearns [1] showed that there is a simulation of multitape TMs running in time $O(T(n))$ by 2-tape oblivious TMs running in time $O(T(n)\log T(n))$. The circuit we need to construct is just the tabeau circuit for this 2-tape TM. This circuit will have two rows for each time step and only a constant number of circuit elements per row (after the first row). The total number of gates will be $O(T(N)\log T(n))$. $\square$

Theorem 3.5 (Kannan) For all k, $\Sigma_2^p \cap \Pi_2^p \not\subseteq \mathsf{SIZE}(n^k)$.

Proof We know that $\mathsf{SIZE}(n^{k+1}) \not\subseteq \mathsf{SIZE}(n^k)$ by the circuit hierarchy theorem. To prove this theorem we will give a specific example of a language with circuit size at least n^{k+1} that is in $\Sigma_2^p \cap \Pi_2^p \setminus \mathsf{SIZE}(n^k)$.

For each n, let C_n be the lexicographically smallest circuit on n inputs such that $size(C_n) \geq n^{k+1}$ and C_n is minimal; i.e., C_n is not equivalent to any smaller circuit. (For lexicographic ordering on circuit encodings, we'll use $\preceq$ and we assume that if $size(C) \leq size(C')$ then $C \preceq C'$.) Let $\{C_n\}_{n=0}^{\infty}$ be the corresponding circuit family and let A be the language decided by this family.

By our choice of C_n, $A \notin \mathsf{SIZE}(n^k)$. Also, by the circuit hierarchy theorem, $size(C_n)$ is a polynomial $\leq 3n^{k+1}$ and the size of the encoding $|\langle C_n\rangle| \leq n^{k+3}$, say. Note that the factor of 3 is necessary because there may not be a circuit of size exactly n^{k+1} that computes A, but there must be one of size not too much larger than this by the circuit hierarchy theorem. We first show a weaker reswlt.

Claim: $A \in \Sigma_4^p$.

The basic idea of the claim is that we can express the conditions using quantifiers. We define A by guessing the encoding $\langle C_n\rangle$ for inputs of length n as a string and then verifying that C_n satisfies:

- $size(C_n) \geq n^{k+1}$.
- C_n is minimal.
- For all minimal circuits D on n inputs of size at least n^{k+1} (and at most $3n^{k+1}$), $C_n \preceq D$.

Recall from Lecture 1 that the property of a circuit being minimal is a Π_2^p property. That is, a circuit C on n inputs (of size at most $3n^{k+1}$ say) is minimal if and only if

$$\forall \langle C'\rangle \in \{0,1\}^{n^{k+3}} \exists y \in \{0,1\}^n ((size(C') \geq size(C)) \vee (C'(y) \neq C(y))).$$

The third condition for a fixed D of size between n^{k+1} and $3n^{k+1}$ is equivalent to saying that D is not minimal or $C_n \preceq D$, i.e., . This is a Π_2^p condition in $\langle D\rangle$ and $\langle C_n\rangle$:

$$\exists \langle D'\rangle \in \{0,1\}^{n^{k+3}} \forall z \in \{0,1\}^n \ [((size(D') \leq size(D)) \wedge (D'(z) = D(z))) \vee (C_n \preceq D)].$$

Now, we can use the same variable D to represent the candidate circuit in the third condition and in place of the C' in the minimality condition for C_n. Therefore $x \in A$ if and only if

$$\begin{aligned}
&\exists \langle C\rangle \in \{0,1\}^{|x|^{k+3}} \forall \langle D\rangle \in \{0,1\}^{|x|^{k+3}} \exists \langle D'\rangle \in \{0,1\}^{|x|^{k+3}} \exists y \in \{0,1\}^{|x|} \forall z \in \{0,1\}^{|x|} \\
&\qquad (C(x) \\
&\qquad \wedge (size(C) \geq |x|^{k+1}) \\
&\qquad \wedge ((size(D) \geq size(C)) \vee (D(y) \neq C(y))) \\
&\qquad \wedge [(size(D') \geq |x|^{k+1}) \wedge [((size(D) \leq size(D')) \wedge (D(z) = D'(z))) \vee (C \preceq D)]).
\end{aligned}$$

The last three lines of the condition each match an item of the requirements and specifies that the circuit C is precisely $C_{|x|}$. The first line says that $x \in A$ if and only if $C_{|x|}(x)$ is true. This proves the claim and also the weaker conclusion that $A \in \mathsf{PH}$.

We finish the proof of the theorem by analyzing two possible scenarios:

(a) $\mathsf{NP} \subseteq \mathsf{P/poly}$. In this case, by the Karp-Lipton Theorem, $A \in \mathsf{PH} = \Sigma_2^p \cap \Pi_2^p$ because the polynomial time hierarchy collapses, and we are done.

(b) $\mathsf{NP} \nsubseteq \mathsf{P/poly}$. In this simpler case, there is some $B \in \mathsf{NP} - \mathsf{P/poly}$. In particular $B \notin \mathsf{SIZE}(n^k)$ and since $\mathsf{NP} \subseteq \Sigma_2^p \cap \Pi_2^p$, we have $B \in \Sigma_2^p \cap \Pi_2^p - \mathsf{SIZE}(n^k)$.

This finishes the proof of the Theorem. □

Note that this argument is non-constructive: A is an explicit language not in $\mathsf{SIZE}(n^k)$ and if $\mathsf{NP} \subseteq \mathsf{P/poly}$ then A is in $\Sigma_2^p \cap \Pi_2^p$. In the second case we do not have an explicit language B and we also don't explicitly know which case is true. The latter problem would not be an issue: We could define a new language $A \otimes B = \{0x \mid x \in A\} \cup \{1x \mid x \in B\}$, which is at least as hard as both A and B. However, the former problem is much trickier to deal with but we can get an explcit Σ_2^p (or Π_2) language that is not in $\mathsf{SIZE}(n^k)$.

The key to producing an explicit language in $\Sigma_2^p - \mathsf{SIZE}(n^k)$ is the fact that the proof of the Karp-Lipton theorem is constructive. The construction for the Karp-Lipton theorem shows that for any Π_2^p language L defined by an explicit formula $\forall u \in \{0,1\}^{q(|x|)} \exists v \in \{0,1\}^{q(|x|)} R(x,u,v)$ and for any polynomial circuit size bound n^k, there is another explicit formula

$$\exists \langle C' \rangle \in \{0,1\}^{n^{2k}} \forall u \in \{0,1\}^{q(|x|)} R(x,u,C'(x,u))$$

such that if the language L' defined by $\exists v \in \{0,1\}^{q(|x|)} R(x,u,v)$ is in $\mathsf{SIZE}(n^k)$ then the two formulas define the same language. In general the new formula might not define the same language so call this resulting Σ_2^p language $\tau(L)$; this will equal L if $L' \in \mathsf{SIZE}(n^k)$. Similarly, by taking complements, for $\bar{L} \in \Sigma_2^p$ there is an explicit $\tau(\bar{L}) \in \Pi_2^p$ that is equal to $\bar{L}$ if $L' \in \mathsf{SIZE}(n^k)$.

Now let $\bar{L}$ be the Σ_2^p language defined by removing the two initial quantifiers $\exists \langle C \rangle \in \{0,1\}^{|x|^{k+3}} \forall \langle D \rangle \in \{0,1\}^{|x|^{k+3}}$ from the definition of A. Then the Π_2^p language $\tau(\bar{L})$ is equal to $\bar{L}$ if $L' \in \mathsf{SIZE}(n^k)$ where L' is the NP language related to L defined as above. Now define A_0' by $x \in A_0'$ if and only if $\exists \langle C \rangle \in \{0,1\}^{|x|^{k+3}} \forall \langle D \rangle \in \{0,1\}^{|x|^{k+3}} x \in \tau(\bar{L})$. Clearly A_0' is an explicit Σ_3^p language and if $L' \in \mathsf{SIZE}(n^k)$ then $A_0' = A \notin \mathsf{SIZE}(n^k)$. Let $A' = A_0' \otimes L'$. Then A' is in Σ_3^p but $A' \notin \mathsf{SIZE}(n^k)$.

Repeating this construction again with A' instead of A and removing only the initial $\exists \langle C \rangle \in \{0,1\}^{|x|^{k+3}}$ from the Σ_3^p definition of A' we can apply the analogous transformation to the resulting Π_2^p language and convert A' to a $A'' = A_0'' \otimes L''$ that is in Σ_2^p but not in $\mathsf{SIZE}(n^k)$.

References

[1] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, 1966.