Boolean function complexity

Lecturer: *Nitin Saurabh* Meeting: 2
Scribe: *Nitin Saurabh* 24.04.2019

In this lecture we will see asymptotically tight upper bound of $O(2^n/n)$ on the circuit complexity of functions over n variables and lower bounds against circuits using the gate elimination technique.

1 Upper Bounds

In the previous lecture we saw that $\mathsf{C}(n) > 2^n/n$. Recall $\mathsf{C}(n)$ is the smallest number t such that every function on n variables can be computed by a circuit of size at most t. We now show that every function can be computed by a circuit of size at most $O(2^n/n)$.

Let $f\colon \{0,1\}^n \to \{0,1\}$. Consider the following brute-force representation of f,

$$f(x) = \bigvee_{a\colon f(a)=1} [x = a],$$

where $[x = a]$ is a Boolean function that outputs 1 iff $x = a$. We could implement $[x = a]$ by a simple conjunction as follows,

$$[x = a] = x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_i^{a_i} \wedge \cdots \wedge x_n^{a_n},$$

where $x_i^{a_i} = x_i$ if $a_i = 1$, and $x_i^{a_i} = \neg x_i$ otherwise. Thus we obtain a DNF representation of f,

$$f(x) = \bigvee_{a\colon f(a)=1} (x_1^{a_1} \wedge x_2^{a_2} \wedge \cdots \wedge x_i^{a_i} \wedge \cdots \wedge x_n^{a_n}).$$

Clearly, the size of this circuit is at most $O(n2^n)$. Can we do better?

Function Decomposition: Consider the following recurrence:

$$f(x) = (x_n \wedge f(x_1, \ldots, x_{n-1}, 1)) \vee (\neg x_n \wedge f(x_1, \ldots, x_{n-1}, 0)). \tag{1}$$

Suppose we have circuits for $f(x_1, \ldots, x_{n-1}, 1)$ and $f(x_1, \ldots, x_{n-1}, 0)$, then we can construct a circuit for f as in Figure 1. Clearly, from this construction we have $\mathsf{C}(f) \leq \mathsf{C}(f|_{x_n=1}) + \mathsf{C}(f|_{x_n=0}) + 3$, where $f|_{x_i=b} = f(x_1, \ldots x_{i-1}, b, x_{i+1}, x_n)$ for $b \in \{0,1\}$. Recursively, we can apply this decomposition to $f|_{x_n=1}$ and $f|_{x_n=0}$. Therefore, the size of the circuit can be bounded by the following recurrence,

$$T(n) \leq 2 \cdot T(n-1) + 3.$$

Solving the recurrence we obtain that $T(n) \leq 4 \cdot 2^{n-1}$. How much more can we improve? Lupanov established the following tight upper bound.

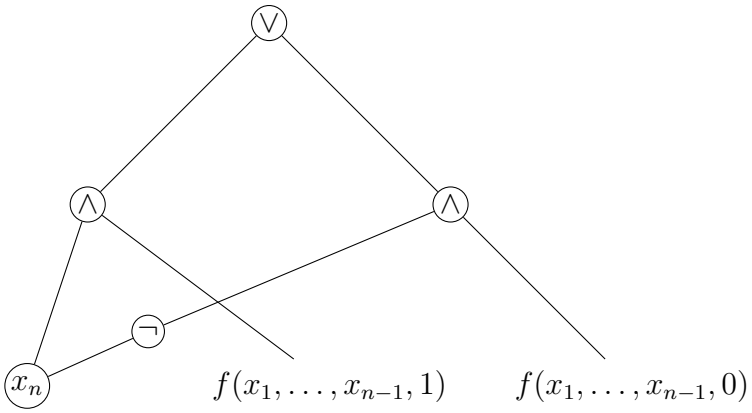

Figure 1: Function decomposition

Theorem 1 (Lupanov 1958)**.** *For every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$,

$$\mathsf{C}(f) \leq (1 + o(1))\frac{2^n}{n}.$$

Proof. We will prove a slightly weaker but almost tight bound of $(4 + o(1))\frac{2^n}{n}$.

Let $\mathsf{All}_k : \{0,1\}^k \to \{0,1\}^{2^{2^k}}$ be the function that computes all Boolean functions over k variables at the same time. That is, $\mathsf{All}_k(y) = (f_1(y), f_2(y), \ldots, f_{2^{2^k}}(y))$.

We now look back at the function decomposition (1). Consider the recursive process at the $(n-k)$-th step; we have a circuit of size at most $3(2^{n-k} - 1)$ with inputs as the last $n-k$ variables, $x_{k+1}, x_{k+2}, \ldots, x_n$, and subfunctions on the first k variables $f_a(x_1, \ldots, x_k) = f(x_1, \ldots, x_k, a_1, \ldots, a_{n-k})$ where $a \in \{0,1\}^{n-k}$. Note that if we restrict the k-th variable according to the recurrence (1), then we increase the size of the circuit in the next step by $3 \cdot 2^{n-k}$. One the other hand, instead of recursing on x_k, we could also plug in the circuit for All_k. This is because the subfunctions on the first k variables are included in the outputs of All_k. In this case we will increase the size by $\mathsf{C}(\mathsf{All}_k)$. If it were the case that $3 \cdot 2^{n-k} > \mathsf{C}(\mathsf{All}_k)$, then we could achieve reduction in the size by computing all functions on k variables. Therefore, the total size of the circuit will be at most $3 \cdot 2^{n-k} + \mathsf{C}(\mathsf{All}_k)$.

To complete the proof we now show that reduction in size is possible when k is roughly $\log n$. The following lemma gives an efficient circuit construction to compute all Boolean functions on k variables.

Lemma 2. *For any* k, $\mathsf{C}(\mathsf{All}_k) \leq 2^{2^k}(1 + o(1))$.

We first finish the proof of the theorem assuming the lemma. Choose $k = \log(n - \log n)$. We can now bound the size of the circuit as follows

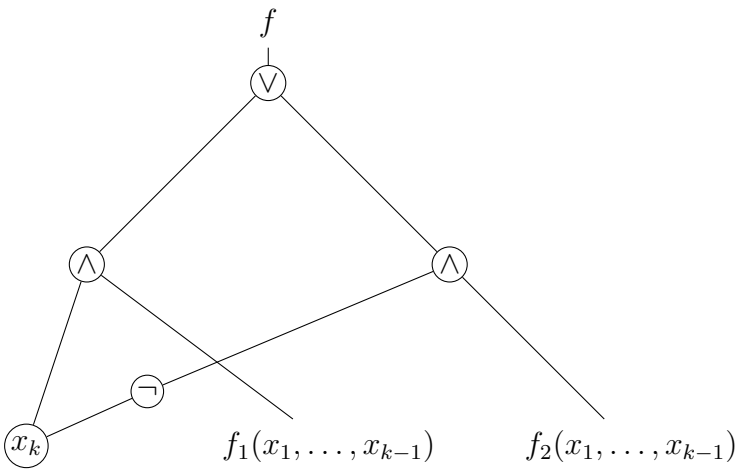

Figure 2: Inductive construction for All_k

$$\begin{aligned}
\mathsf{C}(f) &\leq 3 \cdot 2^{n-k} + 2^{2^k}(1 + o(1)) \\
&\leq 3 \cdot \frac{2^n}{n - \log n} + \frac{2^n}{n}(1 + o(1)) \\
&\leq \frac{2^n}{n}(4 + o(1)).
\end{aligned}$$

□

We now prove Lemma 2 to complete the construction.

Proof of Lemma 2. A function f on k variables can be recursively defined as follows

$$f(x) = (x_k \wedge f_1(x_1, \ldots, x_{k-1})) \vee (\neg x_k \wedge f_2(x_1, \ldots, x_{k-1}))\,,$$

where f_1 and f_2 are functions on $k-1$ variables. Therefore, using the above decomposition we can inductively construct a circuit for All_k if we have a construction for All_{k-1} as shown in Figure 2. Therefore, total size required to realize all 2^{2^k} functions

$$\begin{aligned}
\mathsf{C}(\mathsf{All}_k) &\leq \mathsf{C}(\mathsf{All}_{k-1}) + \{\#\text{ gates required to compute all } k\text{-variate functions from } \mathsf{All}_{k-1}\}, \\
&\leq \mathsf{C}(\mathsf{All}_{k-1}) + \ \#\text{ of } \wedge \text{ gates } + \ \#\text{ of } \vee \text{ gates }, \\
&\leq \mathsf{C}(\mathsf{All}_{k-1}) + 2 \cdot 2^{2^{k-1}} + 2^{2^{k-1}} \cdot 2^{2^{k-1}}, \\
&\leq \mathsf{C}(\mathsf{All}_{k-1}) + 2 \cdot 2^{2^{k-1}} + 2^{2^k}.
\end{aligned}$$

Solving the above recurrence with $\mathsf{C}(\mathsf{All}_0) = 2$, we obtain the required upper bound on the size of the circuit computing all functions,

$$\mathsf{C}(\mathsf{All}_k) \leq 2^{2^k}\left(1 + \frac{3\sum_{i=0}^{k-1} 2^{2^i}}{2^{2^k}}\right) = 2^{2^k}(1 + o(1)).$$

□

Analogous to time/space hierarchy theorem in Turing machine model, we can show that larger circuits can compute strictly more functions than the smaller ones.

For $t\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{size}(t)$ denote the set of Boolean functions $f\colon \{0,1\}^* \to \{0,1\}$ such that $\mathsf{C}(f_n) \leq t(n)$ for all $n \in \mathbb{N}$.

Theorem 3. *Let $t(n)$ be such that $n \leq t(n) \leq 2^{n-2}/n$. Then,*

$$\mathsf{size}(t(n)) \subsetneq \mathsf{size}(4t(n)).$$

Proof. Pick $m \leq n$ such that $t(n) \leq 2^m/m \leq 2 \cdot t(n)$. Consider the set $\mathcal{S}$ of all Boolean functions over n variables that depend only on m variables. By Shannon's lower bound (Theorem 7 in Lecture 1), there exists $f\colon \{0,1\}^n \to \{0,1\}$ in $\mathcal{S}$ such that $\mathsf{C}(f) > 2^m/m \geq t(n)$. On the other hand, by Lupanov's upper bound (Theorem 1), we have $\mathsf{C}(f) \leq 2 \cdot 2^m/m \leq 4t(n)$. □

The lower bound results we have seen until now are not quite satisfactory. We know that almost all Boolean functions are hard (Shannon's lower bound), however no "explicit" function is known to be hard. Proving super-polynomial lower bound against "explicit" functions, e.g., functions in NP, is the main goal of the circuit complexity. Unfortunately at the moment we can not even prove a lower bound better than $5n$. In the following section we will see some lower bounds against "explicit" functions, e.g., Th^n_k, Parity, etc.

2 Gate Elimination

Existing lower bounds for circuits are proved by what is known as the *gate elimination* technique. The overall idea behind this technique is as follows.

Given a circuit that computes a particular function, we first argue that some variable must be an input to many gates. Then, argue that setting this variable to a constant will eliminate many gates. Repeat this process to conclude that the initial circuit must have had large size.

We now prove a linear lower bound for the threshold function $\mathsf{Th}^n_2\colon \{0,1\}^n \to \{0,1\}$. Recall, $\mathsf{Th}^n_2(x) = 1$ iff $\sum_{i=1}^n x_i \geq 2$. We also recall that $\mathcal{B}_2$ is the set of all 16 Boolean functions over 2 variables.

Theorem 4. *For every n, Th^n_2 requires at least $2n-4$ gates over circuits with gates in $\mathcal{B}_2$.*

Proof. The proof is by induction on n.

Base case: $n = 3$. The bound is trivial since Th^3_2 depends on all 3 variables. (Also see Exercise 2 in Problem Set 1.)

For the induction step, assume that the theorem holds for Th^{n-1}_2. We would now prove it for n. Consider an optimal circuit $\mathcal{C}$ for Th^n_2. Let g be a gate in $\mathcal{C}$ such that the two gates feeding into g are input gates. Wlog, we assume that the input gates are labelled by distinct variables x_i and x_j. We now claim the following.

Claim 2.1. *Either x_i or x_j is an input to another gate h (distinct from g).*

We complete the proof assuming the claim. Suppose x_i is the variable that feeds into h. Set $x_i = 0$ in the circuit $\mathcal{C}$. The resulting circuit computes Th_2^{n-1}. Furthermore, setting $x_i = 0$ has eliminated both the gates g and h from the new circuit. This is because, upon setting x_i, both g and h are functions of at most one child. So we can rewire these children to be an input to the gates where g or h were feeding. Thus eliminating the need of g and h in the process. Since the resulting circuit computes Th_2^{n-1}, by induction hypothesis it has at least $2(n-1)-4$ gates. Therefore, the original circuit must have had at least $2(n-1)-4+2 = 2n-4$ gates. This concludes the proof of the theorem. We now prove the claim.

Proof of Claim 2.1: We prove the claim by contradiction. Suppose both x_i and x_j are inputs to the gate g alone. We now consider the four possible settings to x_i and x_j. Under these settings the circuit simplifies to two distinct circuits, namely one where g evaluates to 0 and the second where g evaluates to 1. On the other hand, the function Th_2^n simplifies to three distinct subfunctions, namely Th_0^{n-1}, Th_1^{n-1} and Th_2^{n-1}. Thus we have a contradiction. □

Until very recently, the best known lower bound over the basis $\mathcal{B}_2$ was $3n - o(n)$ [Blu83]. In a recent work this was improved to $(3 + \frac{1}{86})n - o(n)$ [FGHK16].

Restricting the basis to De Morgan basis, i.e., $\vee$, $\wedge$, and $\neg$, allows us to prove slightly better lower bounds.

Theorem 5 ([Sch76])**.** *The minimal number of $\wedge$ and $\vee$ gates in a circuit over De Morgan basis computing Parity_n or $\neg\,\mathsf{Parity}_n$ is exactly $3(n-1)$.*

Proof. **Upper bound:** We can compute the parity on two variables x_1 and x_2 as follows: $x_1 \oplus x_2 := \mathsf{Parity}_2(x_1, x_2) = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$. Thus using the recursion, $\mathsf{Parity}_n(x) = \mathsf{Parity}_{n-1}(x_1, \ldots, x_{n-1}) \oplus x_n$, we obtain a circuit of size $3(n-1)$ computing Parity_n.

Lower bound: The proof is by induction on n. Base case is $n = 2$. It easily seen that we need at least 3 gates in this case. Suppose not. That is, there exists a circuit with 2 gates that computes Parity_2. Clearly one of the two variables x_1 or x_2 is an input to the output gate. Since the output gate is either $\vee$ or $\wedge$ we can make it a constant by setting the variable that feeds into it appropriately. For example, if the output gate is an $\vee$ gate and x_2 (or, $\neg x_2$) feeds into it, then by setting x_2 to 1 (or, 0) we can make the output gate constant. Analogously when the output gate is an $\wedge$ gate. Since parity doesn't reduce to a constant unless all variables are set, we obtain a contradiction to the fact that the circuit computes parity.

For the induction step, we assume that the theorem holds for $n-1$. We now prove it for n. Let us examine a minimal size circuit $\mathcal{C}$ that computes Parity_n. Let g be a gate in $\mathcal{C}$ such that the two gates feeding into g are input gates. Wlog, we assume that the input gates are labelled by distinct variables x_i and x_j.

Claim 2.2. *x_i is also an input to another gate h (distinct from g).*

Proof. Suppose not. That is, x_i only feeds into the gate g. Then, there is a setting to the other variable x_j such that g evaluates to a constant. Thus making the circuit independent of x_i. However, the resulting circuit computes either Parity_{n-1} or $\neg\,\mathsf{Parity}_{n-1}$. Therefore, we obtain a contradiction. □

Claim 2.3. *h is not the output gate of the circuit $\mathcal{C}$.*

Proof. This is because if it were so, then we can make the output gate constant by just setting the variable x_i which is a contradiction to the fact that it computes Parity_n. □

Therefore, we know that h feeds into some other gate h' in the circuit. We now claim that h' must be distinct from g.

Claim 2.4. *h' is distinct from g.*

Proof. This follows from the choice of g. Recall that g is such that the two gates feeding into it are input gates. Whereas a non-input gate h feeds into h'. □

Thus, we have three distinct gates g, h, and h' in the circuit $\mathcal{C}$. We now observe that there is a setting to $x_i \in \{0,1\}$ such that the aforementioned three gates are eliminated in the resulting circuit.

Claim 2.5. *There exist $b \in \{0,1\}$ such that when $\mathcal{C}$ is restricted to $x_i = b$ the three gates g, h and h' are eliminated from the resulting circuit.*

Proof. Since $h \in \{\vee, \wedge\}$, there is a setting to x_i such that h evaluates to a constant. On this setting, we claim that the three gates are eliminated. Clearly h has been eliminated. Moreover, the other two gates g and h' are now functions over one variable and hence we can rewire the edges to eliminate the need for g and h' (as was done in Theorem 4). □

We restrict the circuit $\mathcal{C}$ as per Claim 2.5 while eliminating at least 3 gates. The resulting circuit computes Parity_{n-1} or $\neg\,\mathsf{Parity}_{n-1}$. Therefore, by induction hypothesis, it still has $3(n-2)$ gates. Hence, the original circuit $\mathcal{C}$ must have had at least $3(n-2)+3 = 3(n-1)$ gates to begin with. This concludes the proof of the lower bound. □

The current best lower bound known over the De Morgan basis is $5n - o(n)$ [IM02].

References

[Blu83] Norbert Blum. A boolean function requiring 3n network size. *Theoretical Computer Science*, 28(3):337 – 345, 1983.

[FGHK16] Magnus G. Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 89–98, 2016.

[IM02] Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of 5n - o(n) for boolean circuits. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, MFCS '02, pages 353–364, 2002.

[Sch76] C. P. Schnorr. The combinational complexity of equivalence. *Theoretical Computer Science*, 1(4):289 – 295, 1976.