ITCS:CCT09 : Computational Complexity Theory	Apr 13, 2009
Lecture 10	
Lecturer: Jayalal Sarma M.N.	*Scribe: Youming Qiao*

In the last few lectures, we saw the circuit model of computation, various resources associated with them, and how some of them compare with the complexity classes defined based on Turing machines. In this lecture we will to give an introduction to a "holy grail" of complexity theory; namely proving circuit lower bounds.

1 Explicit Lower Bound question

With the famous P vs NP question in mind, the aim is to prove lower bounds for circuit parameters (like size, depth etc) required for any *uniform* circuit family to compute specific Boolean functions (say in NP). For example, if we prove a super-polynomial size lower bound[1] for any *uniform* circuit family computing a specific Boolean functions (say in NP) then we separate P from NP.

An even stronger aim (hence may be difficult to achieve) can be is to relax the constraint of uniformity. Let us say that we want to argue that for an explicit Boolean function f in NP there does not exist a poly size circuit family at all (so no uniformity machine, no-matter how powerful it is, cannot find it anyway). This relaxation is done more because, we do not know how to effectively exploit the *uniformity* constraint even if it is included.

The circuit lower bounds against non-uniform circuits does imply lower bounds against uniform circuit families as well. But we will relax this question further. Do we know if there exist Boolean functions that requires super-polynomial size circuits? Yes, and this is an important result which goes back to Shannon (1948). We will see this counting argument today in some detail. In fact it indicates that *most of* Boolean functions defined actually require super-polynomial size circuits. Now, can we find some specific ones? This leads to the explicit Lower bound question.

Definition 1 (Explcit Lower Bound Question) *Is there an explicit Boolean function*[2] $f : \{0,1\}^n \to \{0,1\}$*, for which any circuit family* $\{C_n\}_{n\geq 0}$ *computing it must have super-polynomial size?*

As we argued, this is stronger than P vs. NP question which is is not answered yet. But we

[1] We call this a lower bound against polynomial size Boolean circuit families.

[2] preferably one in NP, but in this case the notion of explicitness is weaker, because if "exists" a function in NP that requires super-poly size circuits, then so does the SAT function, which is explicit.

believe that this formalism is more useful for the techniques that we have and that provides the motivation to study the circuit lower bound question. However, despite lot of efforts by the best minds, we do not have strong reasons to believe that we are close to proving it or disproving it.

There has been a lot of progress if we restrict the circuit family that we consider. This line of research was very successful in the 1980's, and several distinguished results, including PARITY $\notin \mathsf{AC}^0$ were proved. We will see several of these results in the next few lectures.

The specific agenda for today is to present (1) a conditional answer to the explicit lower-bound question (2) an unconditional answer to the lower-bound question, but in the non-explicit way.

2 Conditional Lowerbound (Karp-Lipton-Sipser Theorem)

Karp, Lipton and Sipser showed that NP is unlikely to have polynomial sized circuit, by showing that a collapse of the PH is implied by it. We believe that this is not the case. A way to view this is as s conditional lower bound, where the condition is widely believed to be satisfied. We will present this result in this section.

Before going into the main theorem, we will review the three aspects of a language being computed by polynomial sized circuit.

2.1 Three Descriptions of Polynomial Circuits

Some notations: for a language $L \in \{0,1\}^*$ denote $L_n = \{x \in L : |x| = n\}$. And $\mathbb{N}$ is denoted as the set of natural numbers.

Definition 2 (Polynomial circuits) *A language L is said to be computable by a polynomial circuit family if there exists a family of circuits $\{C_i\}_i \in \mathbb{N}$ such that $|C_n| \leq p(n)$ for some polynomial $p(n)$ and C computes L : that is, $\forall x : x \in L \iff C_{|x|}(x) = 1$.*

Definition 3 (Sparse sets) *A set $S \subset \{0,1\}^*$ is called a sparse set if there exists a polynomial $p(n)$ such that $|S_n| \leq p(n)$.*

Definition 4 (P-time TMs with poly advice) *A language $L \in \mathrm{P/poly}$ if there exist $L' \in \mathrm{P}$ and a polynomial bounded function $l : \mathbb{N} \to \{0,1\}^M$, $|l(n)| \leq p(n)$ for some polynomial $p(n)$, such that:*

$$x \in L \iff \langle x, l(|x|)\rangle \in L'$$

and l is called the advice function.

Proposition 5 *For a language $L \in \{0,1\}^*$, the following three propositions are equivalent:*

1. *L has polynomial circuits $\{C_n\}_{n \in N}$;*
2. *$L \in \mathrm{P/poly}$;*
3. *$L \in \mathrm{P}^S$ for some sparse set S.*

Proof We will prove (1) $\Longleftrightarrow$ (2), and (1) $\Longleftrightarrow$ (3).

(1) $\Rightarrow$ (2) Given input x, the advice function just maps $n = |x|$ to the description of C_n, and the Turing machine will simulate the computation of C_n on x.

(2) $\Rightarrow$ (1) We need to construct a circuit family $\{C_n\}_{n \in N}$ computing L. To construct C_n we can hardwire the result of $l(n)$ (we can do so since we just need to show the existence of such a circuit family) and then simulate the computation of the corresponding polynomial Turing machine that accepts $\langle x, l(|x|)\rangle$. (This can be done since we already know P has polynomial circuits, and $l(n)$ is polynomially bounded.)

(3) $\Rightarrow$ (1) For Turing machine M computing $L \in \mathrm{P}^S$, on input x it would only query strings in S the lengths of which are polynomial of n. And the number of queries is bounded by some polynomial, too. So given L_n, the total length of queries made by M for L_n is a polynomial of n. Then we just need to hardwire those queries into the circuit and the circuit would simulate the computation of the Turing machine with oracle without difficulty.

(2) $\Rightarrow$ (3) We would turn the advice function $l(n)$ into a sparse set S, such that the machine would get $l(n)$ from S by querying oracles.

The trick is to construct a set S as follows (suppose $|l(n)|$ is bounded by $p(n)$:

$$S = \{\langle 1^{p(n)}, p_n\rangle \mid p_n \text{ is a prefix of } l(n), n \in N\}$$

It is easy to see that S is a sparse set. Then the machine would operate as follows: given x of length n, to get $l(n)$, it would query $\langle 1^{p(n)}, 0\rangle$ and $\langle 1^{p(n)}, 1\rangle$ to get the first bit of $l(n)$. Having found a prefix p of $l(n)$, the machine would query $\langle 1^{p(n)}, p0\rangle$ and $\langle 1^{p(n)}, p1\rangle$ to determine the next bit, until the machine find that $l(n)$ is already fully extended. ■

Theorem 6 (Karp-Lipton-Sipser Theorem) *If* NP $\subset$ P/poly*, then* PH $= \Sigma_2 \cap \Pi_2$.

Another way to state KLS theorem would give us the conditional circuit lowerbound theorem, given Proposition 5.

Corollary 7 (Conditional circuit lowerbound) *If* PH $\neq \Sigma_2 \cap \Pi_2$*, then* SAT *function requires super-polynomial size for any circuits computing it.*

Remark Note that in Proposition 5, the assumption is that L is Turing-reduced to a sparse set. A natural question is whether a stronger assumption would imply a stronger consquence. For many-one reductions, which is a stronger condition, Mahaney proved in 1982 that a stronger consquence can be derived. **Mahaney's Theorem:** *if* SAT *can be Karp (many-one) reduced to a sparse set, then* NP = P. Ogihara and Watanabe (1991) strengthened Mahaney's theorem as : if SAT can be Turing reduced to a sparse set where the reduction asks only constant number of queries to the sparse set, then $\mathsf{P} = \mathsf{NP}$. It is open if this is true for $O(\log n)$ queries.

2.2 Proof of KLS Collapse Theorem

In this section we prove Theorem 6.

Proof We will prove $\Pi_2 \subset \Sigma_2$, which implies PH collapses to $\Sigma_2 \cap \Pi_2$.

Starting with $L \in \Pi_2$, there exists a polynomial Turing machine T such that

$$x \in L \iff \forall y \exists z [T(x, y, z) = 1]$$

Then using Cook's theorem, which shows that SAT is NP-complete, the $\exists z[T(x, y, z) = 1]$ part can be replaced with satisfiable, polynomial-size CNF formula ψ that takes (x, y) as input. So we have

$$x \in L \iff \forall y [\psi(x, y) \in \text{SAT}]$$

Since $\psi(x, y)$ is of polynomial size, say $p(n)$, the assumption NP $\subset$ P/poly would give us a polynomial-size circuit C that solves SAT of size $p(n)$. So one can turn the above Π_2 computation into a Σ_2 computation by guessing C first, then use C to solve the $\psi(x, y) \in$ SAT part. That is:

$$\begin{aligned} x \in L &\Rightarrow \forall y[\psi(x,y) \in \text{SAT}] \\ &\Rightarrow \exists C \forall y[C(\psi(x,y)) = 1] \end{aligned}$$

However, for the inverse direction, things are a bit different. At this time we requires all circuits of size $|C|$ should reject the unsatisfiable formulaes. That is:

$$\begin{aligned} x \notin L &\Rightarrow \exists y[\psi(x,y) \notin \text{SAT}] \\ &\Rightarrow \forall C \exists y[C(\psi(x,y)) = 0] \end{aligned}$$

The good news is that, if some circuit C claims ψ to be correct, we can construct another circuit C' making use of C to find the satisfying assignment. This is called the "self-reducibility" property of SAT.

So given C that claims to solve SAT of size $\leq p(n)$, C' operate as follows: on input ψ, C' feeds C with ψ, and if C claims ψ to be satisfiable, C' would feed C with $\psi_1 = \psi_{x_1=T}$ and with $\psi_2 = \psi_{x_1=F}$. If C claims one of them to be satisfiable, C' would set ψ to be the satisfiable one and recursively do the above procedure. In the end, C' would get some assignment s and check whether s makes ψ satisfiable. Only if this is true C' would accept ψ, otherwise C' just rejects.

So the above procedure makes sure that those "bad" circuits would not affect our computation. And to get C' from C is efficient. To conclude we have:

$$x \in L \iff \exists C \forall y[C'(\psi(x,y)) = 1]$$

And the proof is complete. ■

3 Unconditional Lowerbound (Shannon's Theorem)

Some notations: $\mathbb{B}^n$ denotes $\{0,1\}^{\{0,1\}^2}$, the set of all n-ary Boolean functions. For an integer $n \in N$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. We will now prove Shannon's Theorem.

Theorem 8 *Let $\varepsilon > 0$. The ratio of $\mathbb{B}^n$ that can be computed by circuits over $\mathbb{B}^2$ with $(1-\varepsilon)\frac{2^n}{n}$ gates approaches 0 as $n \to \infty$.*

Proof For $f \in \mathbb{B}^n$, define $S(f) \doteq \min\{S(C) \mid C \text{ is a circuit over } \mathbb{B}^2 \text{ computing } f\}$. For $q, n \in N$ define $N_{q,n} \doteq |\{f \in \mathbb{B}^n \mid S(f) \leq q\}|$. So the claim becomes for $q = (1-\varepsilon)\frac{2^n}{n}$, the following holds:

$$\lim_{n\to\infty} \frac{N_{q,n}}{2^{2^n}} = 0$$

Next we will use some combinatorial method to bound $N_{q,n}$. The crucial point is to realize that a circuit C of size q can be encoded by the following map:

$$\tau : [q] \to ([q] \times \{x_i\}_{i\in[n]})^2 \times [16]$$

This is because every internal gate has two predecessors, which can be other internal nodes or the input nodes. Also every internal gate can be of 16 types since the circuit is over $\mathbb{B}^2$. Denote $T_{q,n}$ as the set of all such maps and $|T_{q,n}| = (16 \cdot (n+q)^2)^q$.

However, two points needs some consideration: the first point is that some maps of this form are not valid circuits. We can ignore this point since we want to show an upper bound. The second point is that some different maps may describe the same circuit. We deal this in the following.

Let $\sigma \in S_q$ be a permutation on q elements, and define $\sigma\tau$ in the obvious way. (If $\tau(i) = (j,k,l)$, then $\sigma\tau(i) = (\sigma(j), \sigma(k), l)$, by defining $\sigma(j) = \sigma(j)$ when $j \in [q]$, and $\sigma(j) = j$ when $j \in x_{ii\in[n]}$.)

For a circuit C, in principle every gate v can be viewed as a function over the inputs. Denote this function as C_v. A circuit C is said to be reduced if for every pair of internal nodes v and u, $C_v \neq C_u$. The reduced circuit is important since for the map τ representing it, the only permutation σ that would cause $\sigma\tau = \tau$ is just the identity permutation. So it contributes the most "replications" in $T_{q,n}$. In other words, a circuit can not have more than $q!$ replicas, so we have

$$N_{q,n} \leq q \cdot \frac{T_{q,n}}{q!}$$

The q factor appears since for a reduced circuit, different gate contributes a different function.

Next we just need to work on counting. Let $d = 16e$ and $q \geq n$, then by Stirling's formula we have:

$$\begin{aligned} N_{q,n} &\leq q \cdot \frac{(16 \cdot (n+q)^2)^q}{q!} \\ &\leq q \cdot d^q \cdot \frac{(n+q)^{2q}}{q^q} \\ &\leq q \cdot d^q \cdot \frac{4^q q^{2q}}{q^q} \\ &= q(4dq)^q \\ &\leq (4dq)^{q+1} \end{aligned}$$

Let $c = 4d$. Next we plug in $q = (1-\varepsilon)^{\frac{2^n}{n}}$, and for sufficiently large n ($n \geq c(1-\varepsilon)$) we have:

$$\begin{aligned} N_{q,n} &\leq (2^n)^{(1-\varepsilon)2^n/(n+1)} \\ &= 2^{(1-\varepsilon)2^n+n} \end{aligned}$$

So we get the result. ■

In terms of circuit lower bound, we have:

Corollary 9 *Given $\varepsilon > 0$, most Boolean functions need more than $(1-\varepsilon)2^n/n$-size circuits when n approaches infinity.*

But as we remarked initially, this does not give us any explicit Boolean function which requires this sized circuits computing it. Now, given that Shannon's bound is actually less than the obvious $O(2^n)$ size bound for any boolean function, a natural question is whether the size $\frac{2^n}{n}$ is actually sufficient. This leads us to Lupanov's theorem.

Theorem 10 *Every Boolean function can be computed by circuits with $\frac{2^n}{n} + o(\frac{2^n}{n})$ gates over the basis $\{0, 1, \oplus, \wedge\}$.*

We will skip the details of the proof of this theorem. It uses a nice representation of Boolean fuction as "ring sum expansions". Interested readers are referred to [1].

References

[1] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.