ITCS:CCT09 : Computational Complexity Theory Apr 29, 2009

Lecture 15

Lecturer: Jayalal Sarma M.N. *Scribe: Yu Wu*

In the previous lecture we studied monotone boolean functions and monotone circuit. In this course we will discuss circuits with negation gates. We restrict circuits to have size of poly(n), and restrict the number of negation gates to be M. Remember the following theorem:

Theorem 1 (Razborov) *If M=0, then circuit of polynomial size cannot compute* $CLIQUE_{k,n}$

Generally, we want to ask the following three questions:

1. What is the minimum number of negations needed to compute a function f? (We denote this as M(f))
2. If circuit C computes f using k negations, can we reduce k to (k-1) without increasing the size much?
3. Suppose that f is a monotone function (that means, there exist a monotone circuit which computing f), what is the value R(f), such that any circuit with at most R(f) negations requires super poly-size?

The answer of the first question is from Markov. We present two important theorems following:

Theorem 2 (Markov, 1957) *Any function* $f : \{0,1\}^n \to \{0,1\}$ *can be computed by a circuit that uses at most* $M = O(\log n)$ *negations.*

Theorem 3 (Fiser, 1974) *Any function* $f, f \in P/poly$ *can be computed by a polynomial size circuit that uses at most* $M = O(\log n)$ *negations.*

Proof Take a circuit C, we would be able to push down the negations of the inputs. Thus we could suppose C has size of $2|C|$ and n negations.We use the following notations:

Definition 4 *(chain) A chain in the binary n-cube is an increasing sequence* $y^1 < y^2 < ... < y^k$ *of vectors in* $\{0,1\}^n$.

Definition 5 *(decrease) Given a chain* $Y = y^1 < y^2 < ... < y^k$*, we define the decrease of y on Y to be* $d_Y(f) =$ *the number of i, s.t* $f(y^i) > f(y^{i+1})$*, and the decrease d(f) to be* $d(f) = \max_Y d_Y(f)$.

Actually we can prove that $M(f) = \lceil \log d(f) + 1 \rceil$. We first prove the lower bound:

$$M \geq \lceil \log d(f) + 1 \rceil$$

Choose a chain $Y = y^1 < y^2 < ... < y^k$ such that $d_Y(f) = d(f)$, let $I(f) = \{i | f(y^i) > f(y^{i+1})\}$(hence $|I(f)| = d(f)$). Suppose C computes f using r negation gates. We need to prove $r \geq [\log |I(f)| + 1]$. The idea is to prove by (kind of a) contradiction. Let's look at the first negation of C. Let h be the function computed at the input to this negation gate, and $g = \neg h$. By definition, h is monotone, and $d_Y(g) \leq 1$.

1. $d_Y(g) = 0$. This implies that $g = 0$ or $g = 1$. In either case, we can eliminate the not gate without changing the decrease.

2. $d_Y(g) = 1$. Let us devise I into two sets based on g:

$$I_0 = \{g(y^i) = 0 | i \in I\}$$
$$I_1 = \{g(y^i) = 1 | i \in I\}$$

   One of I_0, I_1 must has size $\geq \frac{|I|}{2}$. **if** $|I_1| \geq \frac{|I|}{2}$ then we replace the negation gate by constant 1, otherwise by constant 0. Computing f^1 using the new circuit (with negation gates one less than C). Note f^1 has the property that

$$d(f^1) \geq d_Y(f^1) \geq \frac{d(f)}{2} \tag{1}$$

Now we repeat the process, and get a sequence of functions: $f, f^1, ..., f^r$. f^r is a function with 0 negation gate. Thus it is a monotone function. Suppose $r < \lceil \log d(f) + 1 \rceil$, following from (), we have $d(f^r) \geq 1$, which contradicts that f^r is a monotone function. Thus $r \geq \lceil \log |I(f)| + 1 \rceil$.

Now let's prove the upper bound:

$$M(f) \leq \lceil \log d(f) + 1 \rceil \tag{2}$$

We prove this by introduction on $l(f) = \lceil \log d(f) + 1 \rceil$.
Basis: If $l = 0, d(f) = 0$, f is monotone. The statement holds.
Suppose that the statement holds for $l(f) \leq k, k > 0$. We define a set S, $S = \{x \in \{0, 1\}^n |$ any chain starting in x, has $d_Y(f) \leq 2^{l(f)-1}\}$.

From this we could conclude that $\forall y \notin S$, any chain that ends in y doesn't has decrease $d_Y(f) \le 2^{l(f)-1}$. (Otherwise there exists a chain that has decrease greater that $d(f)$, which contradicts the definition of $d(f)$.)

Now we introduce two functions f_0, f_1 as following:

$$f_0(x) = \begin{cases} f(x) & x \in S \\ 0 & x \notin S \end{cases}$$

$$f_1(x) = \begin{cases} 1 & x \in S \\ f(x) & x \notin S \end{cases}$$

By definition, we could easily conclude the following:

$$d(f_0) \le 2^{l(f)-1} \tag{3}$$
$$d(f_1) \le 2^{l(f)-1} \tag{4}$$

and

$$l(f_i) \le \log 2^{l(f)-1} < k, \quad i = 0, 1 \tag{5}$$

By the introduction hypothesis, $neg(f_i) \le M(f_i)l(f) - 1$ for both $i = 0, 1$. It is therefore remains to show that

$$neg(f) \le \max\{neg(f_0), neg(f_1)\} + 1 \tag{6}$$

We introduce a connective function $\mu(a, b) : \{0,1\}^n \to \{0,1\}$, which satisfies:

$$\mu(0, 1, x) = f_1(x)$$
$$\mu(1, 0, x) = f_0(x)$$
$$\mu(a, \neg a, x) = f_a(x)$$

Claim 6 *There exists a connector μ for f_0, f_1, $neg(\mu) \le \max\{neg(f_0), neg(f_1)\}$.*

We prove this by introduction on $r = \max\{neg(f_0), neg(f_1)\}$:
Basis: $r = 0$. f_0, f_1 are monotone functions. $\mu(a, b, x) = (a \wedge f_1) \vee (b \wedge f_0)$.
Introduction step: suppose circuit $C_i(x)$ compute f_i using r negation gates. Let's look at the first negation gate of each C_i. Replace the gate by a new variable z we obtain a circuit $C_i'(z, x)$ on (n+1) variables with one negation gate fewer. Let $f_i'(z, x)$ be the function computed by this circuit, and let $h_i(x)$ be the monotone function computed just before the first negation gate in C_i. We have: $f_i(x) = f_i'(\neg h_i(x), x)$.

By the introduction hypotheses, there is a boolean function $\mu'(a, b, z, x)$ such that $\neg(\mu') \le \max\{neg(f_0'), neg(f_1')\} \le r - 1$ and for i=0,1,

$$\mu'(i, \neg i, z, x) = f_i'(z, x) \tag{7}$$

By replacing the variable z by the following function

$$Z(a,b,x) = \neg((a \wedge h_0(x)) \vee (b \wedge h_1(x))) \tag{8}$$

in (7), we can get a connector $\mu(a,b,x)$ of f_0 and f_1. Since h_0 and h_1 are monotone functions, we have $\neg(\mu) \le 1 + neg(\mu') \le r$, as desired.

Let $s(x)$ be the characteristic function of S. Note that $s(x)$ is monotone. Let μ be a connector of f_0, f_1. Then $f(x) = \mu(s(x), \neg s(x), x)$, and by Claim, $neg(f) \le neg(\mu) + 1 \le \max\{neg(f_0), neg(f_1)\} + 1$.

■

Now let's back to Fisher's theorem. The idea of proving this theorem is designing a black box called 'NEGATOR' which takes $x_1, ..., x_n$ as its input and outputs $\neg x_1, ..., \neg x_n$. We will use threshold function and Fact() to complete the proof.

Remember the threshold function:

$$Th_k^n(x_1,...,x_n) = \begin{cases} 1 & \textbf{if} \sum_{i=1}^{n} x_i \ge h \\ 0 & otherwise \end{cases} \tag{9}$$

Fact 7 *Th_k^n has monotone circuit of O(n log n) size.*

Proof We define $NEG(x_1,...,x_n) = (\neg x_1, ..., \neg x_n)$. We understand $\neg x_i$ as a function of x: $f_i(x) = \neg x_i$.

$$\neg x_i(a) = \begin{cases} 0 & \textbf{if} \quad a_i = 1 \\ 1 & \textbf{if} \quad a_i = 0 \end{cases} \tag{10}$$

If we let $Th_{k,i}^n(x) = Th_k^{n-1}(x_1,...,x_{i-1},x_{i+1},...,x_n)$, we have the following expression:

$$f_i(x) = \vee_{k=0}^{n}(\neg Th_k^n(x) \wedge Th_{k,i}^n(x)) \tag{11}$$

It remains to compute the function $\neg T(x) := (\neg T_1^n(x), ..., \neg T_n^n(x))$. Observe that the bits of any input $y \in \{0,1\}^n$ are sorted in decreasing order $y_1 \ge ... \ge y_n$.

Definition 8 $A_{sort} = \{y | y \in \{0,1\}^n, y_1 \ge ... \ge y_n\}$

Claim 9 *There exist a circuit $\hat{C}_n$ of size $O(n)$ which has at most $r = \lceil \log(n+1) \rceil$ negation gates such that $\hat{C}_n = neg(y)$ for all inputs $y \in A_{sort}$.*

Again, we prove this by introduction on r.
Basis: $r = 1$, $\hat{C_1}$ contains one negation and can compute $\neg y_1$.
Introduction step: suppose the claim is true for $r \leq \lceil \log(n+1) \rceil - 1$. Take the middle bit $y_m (m = n/2)$, if $y_m = 1$, we only need to compute $\hat{C_{n/2}}(y_1, ..., y_{m-1})$, and the next (n+1-m) bits of $\hat{C_n}$ are 1. Otherwise the first m bits are 0, and the next bits are $\hat{C_{n/2}}(y_{m+1}, ..., y_n)$. By the introduction hypnosis, we thus compute $\hat{C_n}$ with r negations.

Let $C_2(y)$ be a circuit of size O(n) with $\lceil \log(n+1) \rceil$ negations which computes $neg(y), y \in A_{sort}$. The resulting circuit $C(x) = C_2(C_1(x))$ computes $\neg T(x)$. ■

From proofs above, we could give some answers to question 1 and 2. Now let's considerate the question 3. We give some result:

Claim 10 *If for some f, $R(f) geq \log n$, then $f \notin P/poly$.*

Proof This is implied by Fisher's theorem. ■

Theorem 11 *(A,M) If $M = O(\log \log n)$, then $CLIQUE_{k,n}$ cannot be computed by polynomial size circuit.*

We will not present the proof of this theorem here, but will prove another theorem:

Theorem 12 *$R(f) \geq \log n - O(\log \log n)$:*

Proof $f : \{0,1\}^n \to \{0,1\}$
$C(X,Y) = \{0,1\}^2, f_0(X), f_1(Y), X \bigcap Y = \emptyset$

Claim 13 *If C has one negation gate, then at least one of f_0 or f_1 can be computed by a monotone circuit of same or smaller size.*

We use the notion of *minterm* of a monotone function to prove this claim.

Definition 14 *(minterm) A minterm is a minimal set of variables which, if all assigned the value 1, forces the function to take the value 1 regardless of other valuables.*

Let g be the monotone function computed at the input to the first negation gate. We have two possibilities: either some *minterm* of g lies entirely in Y, or not. In the first case, we assign constant 1 to all the variables in Y. As a result, g turns into a constant 1. Thus we can replace the negation gate by constant 0. Since $X \bigcap Y = \emptyset$, this change does not affect the function f_0. In the second case, we assign constant 0 to all the variables in X, and by a similar argument, we can conclude that f_1 is not affected. In either case we obtain a circuit which computes f_0 or f_1 and contains no negation gate.

Let $f = f(X)$ be a boolean function in m variables $X = \{x_1, ..., x_m\}$, and $n = km$. A function $f_n : \{0,1\}^n \rightarrow \{0,1\}^k$ is a $k-fold\ extension$ of f if it computes k copies of f on disjoint copies $X_1, ..., X_k$ of X. That is, given an input $(a^1, ..., a^k)$ with $a^i \in \{0,1\}^{X_i}$, the function outputs the sequence $(f(a^1), ..., f(a^k))$. Note:

1. The $i-th$ output bit $f(a^i)$ is independent of inputs a^j for $j \neq i$.
2. If f is a monotone function, then f_n is also a monotone function.

Iterating the argument used in the proof of Claim () yields the following:

Claim 15 *If a monotone function f cannot be computed by a monotone circuit of size t, then its k-fold extension cannot be computed by a circuit of size t using* $\lceil \log(k+1) \rceil$ *negation gates.*

■