18.405J/6.841J Advanced Complexity Theory	March 28, 2000

Lecture 14

Lecturer: Dan Spielman — *Scribe: Ted Allison*

This lecture begins the presention of the other great circuit lower bound, Razborov's Theorem. We will be working woth monotone circuits and their ability to compute Clique.

Definition 1 *A function f is* monotone *if for all x and y, $x \leq y \Rightarrow f(x) \leq f(y)$. (This is using component-wise $\leq$.)*

A circuit C is monotone if and only if all of its gates are AND *and* OR *gates. A monotone circuit has no* NOT *gates.*

The definition says that for a monotone function f and an input x for which $f(x) = 1$, if x' differs from x by having a bit changed from a 0 to a 1, then $f(x') = 1$.

Some examples of monotone functions:

AND and OR are monotone functions.

Threshold functions are monotone. A threshold function is

$$thresh_k(x) = \begin{cases} 1, & \text{if } \sum x_i \geq k, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

$Clique_k$ is monotone. $Clique_k(x) = 1$ if the graph described by x has a k-clique. The input x has a bit for every possible edge in the graph. $Clique_k$ is monotone because adding an edge to a graph (i.e., changing a 0 to a 1) doesn't remove a k-clique. Recall that for large enough k, computing $Clique_k$ is NP-hard.

Monotone circuits and monotone functions are related in the obvious way.

Claim 2 *A monotone circuit computes a monotone function, and every monotone function can be computed by a monotone circuit.*

The goal we are heading toward is a lower bound on the monotone circuit size for some function. Observe that such a lower bound doesn't prove that the monotone function doesn't have a more efficient implementation in a non-monotone circuit.

Proof of claim: The first direction is obvious, once you note that the composition of monotone functions is a monotone function.

The second direction is easier to prove with a couple additional concepts.

Definition 3 *The* minterms *of a monotone function f are the inputs x such that $f(x) = 1$ but $\forall y \leq x$ (with $y \neq x$), $f(y) = 0$. Similarly, the* maxterms *of f are the inputs y such that $f(y) = 0$ but $\forall x \geq y, x \neq y, f(x) = 1$.*

Note that the minterms or maxterms of f can be $\emptyset$, as when f is constant.

Now observe that any function f is an OR of ANDs of its minterms. (For each minterm, form an AND of its 1 bits.) There are possibly some trivial cases to be considered, but that is the essential idea of the proof of the second direction. ∎

For an example of minterms and maxterms, consider $Clique_k$. The minterms are sets of $\binom{k}{2}$ edges giving a clique on k vertices. Some of the maxterms are complete $(k-1)$-partite graphs. (There are other maxterms: consider a set of five vertices connected cyclically, with $k = 3$.)

Now we can state Razborov's Theorem.

Theorem 4 *Any monotone circuit that computes $Clique_{logn}$ must have $n^{\Omega(logn)}$ gates.*

(Another result along the same lines is due to Alon and Boppana, and states that $Clique_{n^{2/3}}$ requires $2^{\Omega(\frac{n^{1/3}}{logn})}$ gates.)

To prove Razborov's Theorem we will again be using approximation techniques. The idea will be to approximate a circuit with a function which agrees on most of the minterms and maxterms of $Clique_k$. For the proof we will need a number of additional tools.

Definition 5 *A* clique indicator *on a set of vertices S, which we will denote $ind(S)$, is an* AND *of all the edges between any two vertices in S. The* size *of $ind(S)$ is $|S|$.*

For example, if x_1 represents the edge between vertices v_1 and v_2, x_2 the edge between v_2 and v_3, and x_6 the edge between v_1 and v_3, then x_1 is a clique indicator corresponding to a 2-clique on $\{v_1, v_2\}$ and $x_1x_2x_6$ is a clique indicator corresponding to a 3-clique on $\{v_1, v_2, v_3\}$.

Definition 6 *A* clean function *is an* OR *of at most n clique indicators of size at most $l = k - 1$, where n is the number of vertices in the graph which is the input to $Clique_k$.*

Observe that the inputs to a circuit are clean functions, being the OR of a single indicator of size 2. The idea of the proof is to work upwards, building an approximation of the circuit using clean functions. A problem is that $f \wedge g$ and $f \vee g$, for clean functions f and g, are not necessarily clean. In order to solve this, we will define approx-OR and approx-AND (denoted $f \sqcup g$ and $f \sqcap g$) such that $f \sqcup g$ and $f \sqcap g$ are clean and approximate $f \vee g$ and $f \wedge g$.

The problem which prevents $f \vee g$ from being a clean function is that it could be an OR of as many as $2n$ clique indicators. We need to get rid of some. To do this, we will define an operation which is based one a result from combinatorics known as the Sunflower Theorem.

Definition 7 *A* sunflower *is a set system. A collection of sets $S_1, \ldots, S_m$ is a* sunflower *if*

$$\forall i \neq j, S_i \cap S_j = \bigcap_{k=1}^{m} S_k.$$

This intersection could be $\emptyset$.

The sets S_i are called petals *and the intersection $\bigcap_k S_k$ is called the* core.

The name *sunflower* comes from the image created by a Venn diagram of the sets.

Theorem 8 (Erdos and Rado) *Let $S_1 \ldots S_m$ be sets, each of size at most l. If $m > (p-1)^l l!$, then the system contains a sunflower with p petals.*

The proof will be left for a problem set.

The idea for approx-OR will be to find a sunflower in $f \vee g$ and replace the petals with the core. Let's define a couple operations to do that.

Definition 9 *For a function f which is an* OR *of clique indicators, define $pluck(f)$ to be the result of the following sequence of operations: Write f as $\underset{i=1}{\overset{m}{\text{OR}}} ind(S_i)$. Find a sunflower among the S_i with p petals. Remove the clique indicators which are the petals of the sunflower, and replace them with the indicator corresponding to the core.*

Define $weed(f)$ as the result of repeatedly plucking until at most n indicators remain.

It is clear that plucking (and weeding) reduces the size of a function; the question is, is the result a good approximation of the original? The proof that it is will be left for the next lecture. But assuming that it is, we can define $f \sqcup g$.

Definition 10 *Define approx-*OR *as:* $f \sqcup g \equiv weed(f \vee g)$.

Now consider $f \wedge g$. A little more work is required to make this look like a clean function. First, write $f = \overset{n}{\underset{i=1}{\mathrm{OR}}} ind(S_i)$ and $g = \overset{n}{\underset{j=1}{\mathrm{OR}}} ind(T_j)$. Then, using distributivity, we can write

$$f \wedge g = \overset{n,n}{\underset{i=1,j=1}{\mathrm{OR}}} (ind(S_i) \wedge ind(T_j)).$$

We have combined two clique indicators with a conjunction, but that combination is not a clique indicator; we need to add in the edges between pairs of vertices from different sets. So we define the following operation.

Definition 11 *For the two clean functions f and g expressed as above, define*

$$cover(f \wedge g) = \overset{n,n}{\underset{i=1,j=1}{\mathrm{OR}}} ind(S_i \cup T_j).$$

After we have covered each pair of smaller cliques with one larger one, we have again arrived at an OR of clique indicators. But some of them may be too large. We will solve this problem by simply throwing the oversized indicators away.

Definition 12 *For a function h which is an* OR *of clique indicators, let $trim(h)$ be the* OR *of all the clique indicators in h which have size at most l.*

Now we can define approx-AND.

Definition 13 *Define approx-*AND *as:* $f \sqcap g \equiv weed(trim(cover(f \wedge g)))$.

Now we can reexamine our strategy for proving Razborov's Theorem. Given a monotone circuit, we can replace each AND gate and OR gate in it with an approx-AND and an approx-OR, respectively. The result will be a clean function. The steps we have left to show are that the clean function we get is a good approximation of the circuit (at least on the minterms and maxterms of a small enough circuit), and that clean functions cannot approximate $Clique_k$.

Then we will have the following chain of reasoning: given any small monotone circuit which computes $Clique_k$, we can approximate it with a clean function. But no clean function is an approximation of $Clique_k$, and we have a contradiction. Therefore no small monotone circuit can compute $Clique_k$.

Most of this will be left for the next lecture, but here we will give a glimpse of the reasoning that $cover(f \wedge g)$ is a reasonable approximation of $f \wedge g$.

What we need to show this is two things: that we do not accept any maxterms we didn't accept before we applied cover, and that we don't reject any cliques we didn't reject before.

To show the first, we have the following claim.

Claim 14 *If $(f \wedge g)(x) = 0$, then $cover(f \wedge g)(x) = 0$.*

Proof Cover only increases the size of the ANDs which make up the function. Certainly the newly enlarged ANDs will not accept where the gates they superseded did not. ■

The second part is given by the next claim.

Claim 15 *If x is (exactly) a k-clique, then $(f \wedge g)(x) = 1 \Rightarrow cover(f \wedge g)(x) = 1$.*

Proof If x is a clique, then any edge between any two vertices in the set covered by x is already in x. Thus the operation of adding edges without adding any new vertices which is performed by the union in the definition of cover cannot add any edges which are not already in x. ■