Lecture 2

Topics in Complexity Theory and Pseudorandomness (Spring 2013)
Rutgers University
Swastik Kopparty
Scribes: Amey Bhangale, Mrinal Kumar

1 Overview

In this lecture, we will complete the proof of formula lower bounds via Krapchenko's method and then look at formula lower bounds via Subbotovskaya's method and Andreev's method. We will also see one lower bound on the size of formula over full binary basis given by Nechiporuk. We will finally conlcude with an introduction to the notion of randomized circuits.

2 Krapchenko's method

Krapchenko's method is based upon the idea of formal complexity measure, which is defined as follows.

Definition 1. *A function FC from the set of all boolean functions to the set of real numbers is said to be formal complexity measure if it satisfies the following properties.*

1. $FC(x_i) = 1$ *for all variables* x_i
2. $FC(f) = FC(\neg f)$ *for every boolean function* f
3. *Sub-additivity:* $FC(f \vee g) \leq FC(f) + FC(g)$ *for all boolean functions* f *and* g

We also proved the following theorem in the last lecture which relates the formula complexity of a function to its formal complexity, for any formal complexity measure, over the De Morgan's basis.

Theorem 2. *For any boolean function f and any formal complexity measure FC, the following is true over the De Morgan's basis*

$$FC(f) \leq L(f) \tag{1}$$

We will now define Krapchenko's formal complexity measure and then use it to prove lower bounds.

Definition 3. *(Krapchenko's measure) For a boolean function f, we define*

$$FC(f) = max_{A \subseteq f^{-1}(0), B \subseteq f^{-1}(1)} \frac{e(A,B)^2}{|A||B|} \tag{2}$$

Here, $e(A,B)$ is defined as the number of edges between A and B in the $n-$dimensional boolean hypercube.

Before we go on and show that FC as defined above is indeed a formal complexity measure and satisfies all the properties in definition 1 , we will see some applications.

2.1 Formula Lower Bound for Parity over De Morgan's basis

We will now show any formula computing the PARITY of n bits must be of size $\Omega(n^2)$. We first prove the following lemma.

Lemma 4. *For the formal complexity defined in definition 3, $FC(PARITY) \geq n^2$.*

Proof. To prove the statement, it is sufficient to show a set $A \subseteq PARITY^{-1}(0)$ and a set $B \subseteq PARITY^{-1}(1)$, such that $\frac{e(A,B)^2}{|A||B|} \geq n^2$. Let us take $A = PARITY^{-1}(0)$ and $B = PARITY^{-1}(1)$. Now, for any x, such that $PARITY(X) = 0$, all its n neighbours in the boolean hypercube will have parity 1 and vice versa. Therefore, $e(A, B) = n2^{(n-1)}$, where $|A| = |B| = 2^{n-1}$. Subsituting back in the definition of FC, we obtain $FC(PARITY) \geq n^2$. □

Now, from the lemma above and theorem 2, we have the following theorem.

Theorem 5. *(Krapchenko) Any formula which computes a $PARITY$ of n bits over the De Morgan's basis, must have size $\Omega(n^2)$.*

2.2 Formula lower bound for Majority over De Morgan's basis

We will now show a formula lower vbound for the majority of n bits, via proving that the formal complexity of the majority function is high. More precisely, we will prove the following lemma.

Lemma 6. *For the formal complexity defined in definition 3, $FC(MAJ) \geq \Omega(n^2)$.*

Proof. The proof idea is again very similar to the proof for the parity function. We first check if the sets defined in lemma 4 work. For the majority function, the only strings $x \in MAJ^{-1}(1)$, which have a neighbour in $MAJ^{-1}(0)$ are the ones which have $\frac{n}{2} - 1$ ones. Moreover, the number of such neighbours is precisely $\frac{n}{2} + 1$. For the simplicity of presentation, let us assume that n is even. The asymptotics remain the same even for an odd n. Now, if we take our set $A = MAJ^{-1}(0)$ and set $B = MAJ^{-1}1$, we get $e(A, B) = (\frac{n}{2} + 1)\binom{n}{\frac{n}{2}-1}$. Hence, we get

$$FC(MAJ) \geq \frac{((\frac{n}{2} + 1)\binom{n}{\frac{n}{2}-1})^2}{2^{2(n-1)}} \tag{3}$$

which is just linear in n. So, we do not get anything non trivial. It can be observed that all the edges across A and B are concentrated between the strings which have just less than half $1's$ and the strings which have one more 1 than them. Keeping this in consideration, let us redefine the sets A and B as follows. Let $A = \binom{[n]}{\frac{n}{2}-1}$ and let $B = \binom{[n]}{\frac{n}{2}}$. Clearly the number of edges between A and B remains the same while their sizes becomes smaller. So, we get

$$FC(MAJ) \geq \frac{((\frac{n}{2} + 1)\binom{n}{\frac{n}{2}-1})^2}{\binom{n}{\frac{n}{2}-1}\binom{n}{\frac{n}{2}}} \tag{4}$$

which is $\Omega(n^2)$. □

From the lemma above and theorem 2, we get the following theorem.

Theorem 7. *Any formula which computes a MAJ of n bits over the De Morgan's basis, must have size $\Omega(n^2)$.*

2.3 FC is a formal complexity measure

We will now verify that the function FC, as defined in definition 3 indeed satisfies all the properties mentioned in definition 1. To this end, we will verify the properties sequentially.

1. $FC(x_i) = 1$, for all variables x_i

   Proof. For the function $f(x) = x_i$, $f^{-1}(0) = \{x : x_i = 0\}$ and $f^{-1}(1) = \{x : x_i = 1\}$. We can observe that the number of neighbours that any string in $f^{-1}(0)$ can have in $f^{-1}(1)$ is at most 1, and vice-versa. So, for any sets $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$, $\frac{e(A,B)}{|A|} \leq 1$ and $\frac{e(A,B)}{|B|} \leq 1$. Multiplying together, we have $FC(x_i) \leq 1$. Now,consider $A = \{0^n\}$ and $B = \{0^{n-i}10^{i-1}\}$. Clearly, $e(A, B) = |A| = |B| = 1$. So, we have $FC(x_i) \geq 1$. Consequently, we get $FC(x_i) = 1$. □

2. $FC(f) = FC(\neg f)$ for all boolean functions f.

   Proof. The proof follows from the fact that the definition of FC is symmteric with respect to 0 and 1. □

3. *Sub-additivity:* $FC(f \vee g) \leq FC(f) + FC(g)$ for all boolean functions f and g

   Proof. Let $h = f \vee g$. For the proof, it would suffice if for every pair of sets $A \in h^{-1}(0)$ and $B \in h^{-1}(1)$, we could show the existence of sets $A_f \in f^{-1}(0)$, $A_g \in g^{-1}(0)$, $B_f \in f^{-1}(1)$ and $B_g \in g^{-1}(1)$ which satisfy

   $$\frac{e(A,B)^2}{|A||B|} \leq \frac{e(A_f,B_f)^2}{|A_f||B_f|} + \frac{e(A_g,B_g)^2}{|A_g||B_g|} \tag{5}$$

   We also observe the following relationships.

   - $h^{-1}(0) = f^{-1}(0) \cap g^{-1}(0)$
   - $h^{-1}(1) = f^{-1}(1) \cup g^{-1}(1)$

   Now taking these into consideration, for any $A \in h^{-1}(0)$, take $A_f = A_g = A \subseteq h^{-1}(0) = f^{-1}(1) \cap g^{-1}(1)$ and take $B_f = B \cap f^{-1}(1)$ and take $B_g = B \setminus B_f$. So, B_f, B_g is a partition of B. Therefore, we also have $e(A, B) = e(A, B_f) + e(A, B_g)$.

   Now, invoking the inequality in claim 8, which is proved below with $a = e(A, B_f)$, $b = e(A, B_g)$, $c = |B_g|$ and $d = |B_f|$, we get

   $$(\frac{e(A,B_f)^2}{|B_f|} + \frac{e(A,B_g)^2}{|B_g|}) \geq \frac{(e(A,B_f) + e(A,B_g))^2}{(|B_f| + |B_g|)} \tag{6}$$

Dividing by $|A|$ both sides, we get

$$(\frac{e(A,B_f)^2}{|A||B_f|} + \frac{e(A,B_g)^2}{|A||B_g|}) \geq \frac{(e(A,B_f)+e(A,B_g))^2}{|A|(|B_f|+|B_g|)} \tag{7}$$

which is the same as

$$(\frac{e(A,B_f)^2}{|A||B_f|} + \frac{e(A,B_g)^2}{|A||B_g|}) \geq \frac{(e(A,B)^2}{|A||B|} \tag{8}$$

Hence, we obtain

$$FC(H) \leq FC(f) + FC(g) \tag{9}$$

□

Claim 8. *For any four positive real numbers a, b, c, d,*

$$\frac{(a+b)^2}{(c+d)} \leq \frac{a^2}{d} + \frac{b^2}{c} \tag{10}$$

Proof. Since a, b, c, d are positive real numbers, using the $AM - GM$ inequality on $\frac{c.a^2}{d}$ and $\frac{d.b^2}{c}$, we get

$$\frac{c.a^2}{d} + \frac{d.b^2}{c} \geq 2.\sqrt{(\frac{c.a^2}{d}\frac{d.b^2}{c})} = 2ab \tag{11}$$

Adding $a^2 + b^2$ on both sides of the inequality, we obtain

$$\frac{(c+d).a^2}{d} + \frac{(c+d).b^2}{c} \geq a^2 + b^2 + 2ab \tag{12}$$

Dividing both sides by the positive quantity $(c+d)$, we get

$$\frac{(a+b)^2}{(c+d)} \leq \frac{a^2}{d} + \frac{b^2}{c} \tag{13}$$

□

Exercise: Show that using *Krapchenko's method* one cannot get a lower bound better that $\Omega(n^2)$.

3 Subbotovskaya's method

We will now look at another formula lower bound proved via a different method. Before moving onto the method, let us define a few ideas which will be cruicially used in the rest of the lecture.

Definition 9. Restrictions of a function: *Given a boolean function f, a restriction of f is a function which can be obtained from f by setting a subset of variables to some fixed values.*

Observe that, given the formula for a function f, we can obtain the formula for any restriction of f by setting a subset of variables to some fixed value in the formula. This procedure reduces the size of the formula in following ways.

- A leaf gets set to either 0 or 1 and so the number of leaves which are indexed by variables reduces.
- Setting a variable to some value might directly or convert an existing non trivial gate in the circuit into a trivial gate. Here, a gate is said to be trivial if it computes a constant function, it is non trivial otherwise. If a gate trivializes due to setting one of its input variables, the number of leaves reduces by 2, for every leaf which is set to a constant.

We also observe the following properties of an optimal formula F computing a function.

- Without loss of generality, we can assume that all the leaves in the formula are labelled by input variables. Otherwise, the formula can be reduced in size further, which would contradict optimality of F. For example, $f \vee 1$ can be replaced by 1 and $f \vee 0$ can be replaced by f, and similarly for $\wedge$ gates.
- Let there be an $\wedge$ gate in F such that its output $h = x_i \wedge g$. Then, without loss of generality, we can assume that the sub-formula computing g does not depend upon x_i. The observation follows from the fact that if x_1 is 0, then $h = 0$ and so h depends on g only if $x_i = 1$. So, the subformula for g can be replaced by a formula where every occurence of x_i in the subformula for g is replaced by 1. This entire procedure leads to no increase in size of f. An $\vee$ gate of the form $x_i \vee g$ can be handled similarly.

For the rest of the lecture, we would be working with a formula satisfying these two properties. We will call such a formula to be nice.

Definition 10. Nice formula: *A formula computing a function is said to be nice if it satisfies the above two properties.*

We can observe that an arbitary formula for a function can be converted into a a nice formula without an increase in size.

Intuition: The basic idea of Subbotovskaya's proof is the following. If there is a formula computing a function f, we can compute a formula for any restriction of f from it. Now, if we somehow knew that this process of restriction leads to a drastic decrease in size, yet the restriction obtained is a non trivial function in the remaining variables and hence has a non-trivial size, we will get a lower bound. For any nice formula having s leaves has an input variable labelling at least $\frac{s}{n}$ leaves, if it computes a function of n variables. So, setting this variable to either 0 or 1, reduces the size of the formula by a factor of at least $(1 - \frac{1}{n})$. If we continue restricting the formula, till it depends on just one variable, we get $\Pi_{i=n}^{2}(1 - \frac{1}{i})s \geq 1$, which just gives us a trivial bound $s \geq n$. So, we need to prove a better shrinkage factor if we hope to obtain a nontrivial lower bound. The following lemma helps us achieve this goal.

Lemma 11. Random Restrictions: *For any boolean function f, let F be the optimal sized formula computing f over the De Morgan basis. Consider the following random process. We select a subset of variables of size $n-k$ uniformly at random and independently set each of these variables to 0 or 1, each with probability 0.5. Let us denote this proess by ρ and the resulting formula obtained from this process by $F|_\rho$. Then,*

$$\mathbb{E}[|F|_\rho|] \leq (\frac{k}{n})^{1.5} |F| \tag{14}$$

To prove the lemma, we will analyse the effect of randomly restricting one variable in the formula, chosen uniformly at random. We will first prove the following lemma.

Lemma 12. *For any boolean function f, let F be the optimal sized nice formula computing f over the De Morgan basis. Let us define the following random process ρ. Let us pick one variable uniformly at random and set it to 0 or 1 with probability 0.5 each. Then, the expected size of the formula computing the function $f|_\rho$ is at most $|F|(1-\frac{1.5}{n})$.*

Proof. For each $i \in [n]$, let s_i be the number of leaves labelled with x_i. Now, when we apply the random restriction, the formula F is affected in the following two ways.

- Any variable which is set to a constant leads to a decrease in the number of leaves.
- If one of the inputs of an $\vee$ gate gets set to 1, then it can be replaced by a constant 1 and the number of leaves decreases by at least 2. Similarly, if an input to an $\wedge$ gate gets set to 0, then it can be replaced by 0 and the number of leaves falls by at least 2. Therefore, for every leaf in the original formula, this second decrease of 1 happens with a probability $\frac{1}{2n}$. So, expected decrease in the number of leaves per leaf of the original formula is $2 * \frac{1}{2n} + 1 * \frac{1}{2n} = \frac{3}{2n}$. Therefore, by the linearity of expectations, the expected decrease in the number of leaves in the formula is $s\frac{3}{2n}$. Observe that since we began with a nice formula, so there is no overcounting being done here.

At the end of the above process, the expected number of leaves surviving in the formula is $s(1-\frac{1.5}{n})$, and we have the lemma. □

We now use the above argument sequentially to prove lemma 11.

Proof. of lemma 11:

Now, we will apply the lemma again and again k times to obtain stronger lemma 11. After each step, we will simplify the resulting formula into a nice formula so that the premises of lemma 12 are met. Hence, at the end of $n-k$ steps, the size of the resulting formula is at most $\Pi_{i=n}^{k-1}(1-\frac{1.5}{i})$. For every i, the i^{th} term in the above product is positive and is upper bounded by $(1-\frac{1}{i})^{1.5}$. Therefore, the product is at most $(\frac{k}{n})^{1.5}$. □

3.1 Lower bounds for Parity

Let us now apply lemma 11 to obtain lower bounds for $PARITY$. We invoke the lemma for $k = 1$. Since the parity function depends upon all its input, even when we restrict a formula computing parity to just one live variable, we still have a non trivial function. Now, lemma 11 tells us that there exists a restriction ρ such that $|F|_\rho| \leq |F|(\frac{1}{n})^{1.5}$ and $F|_\rho$ is non trivial. So, the size of $F|_\rho$ is at least 1. So, we get $|F|(\frac{1}{n})^{1.5} \geq 1$. This implies $|F| \geq n^{1.5}$.

4 Andreev's method

In this section, we will use Subbotovskaya's method of random restrictions to obtain a stronger lower bound for formula's in De Morgan's basis. Let us first define the function for which we obtain the lower bound.

Definition 13. *Consider the function* $f : \{0,1\}^{2n} \to \{0,1\}$ *def as follows. First, partition the input* $2n$ *bits as follows:*

- *Let the first* n *bits be indicated by* ϕ*. These bits will be treated as the truth table of a boolean function on* $\log(n)$ *bits.*
- *The next* n *bits should be viewed as sitting in a matrix* X *with* $\log(n)$ *rows of size* $\frac{n}{\log n}$ *each.*

Then, $f(\phi, X) = \phi(\oplus_{j=1}^{\frac{n}{\log n}} X_{1j}, \oplus_{j=1}^{\frac{n}{\log n}} X_{2j}, \ldots, \oplus_{j=1}^{\frac{n}{\log n}} X_{(\log n)j})$.

The following observation follows from the definition.

Observation 14. *The function* f *defined as above is explicitely computable for any input* $x \in \{0,1\}^{2n}$.

Key Idea for the lower bound : The proof builds up on Subbotovskaya's idea. Consider a string ϕ which is the truth table of a hard function on $\log n$ bits. Now, if we randomly restrict some variables in the matrix X, such that each row of X has at least one live random variable, we can obtain a resulting formula which can be used to compute the function ϕ over all inputs. Since ϕ was choosen to be a hard function on $\log n$ bits, the resulting formula should have size at least $\frac{n}{\log\log(n)}$, and we will now use the inequalities obtained to get a lower bound.

We will now formalize this proof sketch to prove the following theorem.

Theorem 15. *Any formula computing the function* f *as defined in definition 13 in the De Morgan basis is of size* $\Omega(n^{2.5-\epsilon})$ *for every epsilon* > 0.

Proof. Consider the optimal sized formula F for f. Without loss of generality, we can assume that it is a nice formula. Now, let us restrict the first $\log(n)$ bits to the truth table of a hard function ϕ. By a hard function, we mean any function with formula size $\Omega(\frac{2^{\log n}}{\log\log(n)})$. We know from Shannon's theorem that such functions exist. Let us call the resulting function f_ϕ and the resulting formula F_ϕ. We can preprocess F_ϕ now to obtain a nice formula for f_ϕ. For the sake of simplicity, let us again call it F_ϕ. Now, set $k = 10\log(n)\log\log(n)$ and apply lemma 11 to F_ϕ.

Claim 16. *All the rows of the matrix* X *contain at least one live variable with probability at least* 0.9.

Proof. For a fixed row i of X, the probability that row i has no live variable is equal to $\frac{\binom{n(1-\frac{n}{\log n})}{k}}{\binom{n}{k}}$. Let us call this p. Now, expanding the binomial coefficients in the numerator and the denomninator, we get $p = \Pi_{i=0}^{k-1} \frac{n(1-\frac{n}{\log n})-i}{n-i}$. Now, using the inequality $\frac{n(1-\frac{n}{\log n})-i}{n-i} \leq \frac{n(1-\frac{n}{\log n})-(1-\frac{n}{\log n})i}{n-i}$, we get that $p \leq (1-\frac{1}{\log n})^k$. For, $k = 10\log(n)\log\log(n)$, we obtain $p \leq \frac{1}{(\log n)^{10}}$. Now, by the union bound,

there exists a row in X which contains no live variables with probability at most $\log n * \frac{1}{(\log n)^{10}} \leq 0.1$ for large enough n. □

Now, from lemma 11, the expected size of the formula after randomly restricting to k variables is at most $|F|(\frac{k}{n})^{1.5}$. Since formula size is a non negative random variable, Markov's inequality tells us that the formula size is larger than $20|F|(\frac{k}{n})^{1.5}$ with a probability at most 0.05. Hence, with probability at most 0.15 either of these bad events happen. So, with positive probability, the formula shrinks a lot and all of the rows in the matrix X have at least one live variable. Then, formula obtained at the end of the process can be used to compute the function ϕ, which we know is hard. So, we have $\frac{n}{\log\log(n)} \leq 20|F|(\frac{k}{n})^{1.5}$. This implies $|F| \geq \Omega(n^{2.5-\epsilon})$, for every positive ϵ.

□

Exercise: For the function f defined in definition 13, show that f can be computed by a formula of size $O(n^3)$.

5 Lower Bounds over Full Binary Basis

All the above methods give a lower bound on the formula size over *De Morgan's basis*. A natural question would be to ask for a lower bound on the formula size of some explicit function over *full binary basis*. Recall that in the formula over full binary basis we are allowed to use a gate which computes arbitrary binary boolean operation.

5.1 Nechiporuk's Method

Let us start by defining one notion.

Definition 17. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function on n variables and Y be a subset of variables. $N_f(Y)$ is the number of different functions $g : \{0,1\}^Y \to \{0,1\}$ we can get when we fix all input bits not in Y.*

Intuitively, larger the quantity $N_f(Y)$ for a function f, harder is the function f and hence formula computing f should require large size. Nechiporuk's Method uses the above notion of "hard" function and gives a lower bound. We are now ready to define a function for which Nechiporuk obtained a lower bound.

Definition 18. *$f : \{0,1\}^n \to \{0,1\}$ be a boolean function on n variables. We can group variables to form a block i.e.*

$$f(x_1, x_2, \cdots, x_n) = f(\underline{x_1, \cdots, x_c}, \underline{x_{c+1}, \cdots, x_{2c}}, \cdots\cdots, \underline{x_{(b-1)c+1}, \cdots, x_{bc}})$$

So there are b blocks $(Y_1, Y_2, \cdots, Y_b)$ each having c variables. The function f is given as:

$$f(Y_1, Y_2, \cdots, Y_b) = \begin{cases} 1 & \text{if all } Y_i's \text{ are distinct} \\ 0 & \text{otherwise} \end{cases}$$

Theorem 19. *Size of formula over full binary basis computing function f defined above is at least* $\Omega\left(\frac{n^2}{\log^2 n}\log\log n\right)$.

Proof. It is easy to get a bound on $N_f(Y_i)$ where f is defined as above. Consider fixing variables in the blocks $\{Y_1, Y_2, \cdots, Y_b\} \setminus \{Y_i\}$. If at least two blocks have the same assignment then the function defined in Y_i is unique which is a *zero* function. On the other hand, if all blocks have different assignments then it defines unique function in Y_i for every such assignment (upto permutation of these assignments between blocks). Since there are 2^c different binary strings of length c and and if we choose any $b-1$ different strings as an assignment for variables in blocks $\{Y_1, Y_2, \cdots, Y_b\} \setminus \{Y_i\}$, it gives followng simple relation:

$$N_f(Y_i) = \binom{2^c}{b-1} + 1 \tag{15}$$

Consider a formula F for computing f. Let $l(Y_i)$ be number of leaves labelled by variables in Y_i in F. When we fix the variables in $\{Y_1, Y_2, \cdots, Y_b\} \setminus \{Y_i\}$, we can assume without loss of generality that the formula induced by fixing these varaibles has leaves labelled by variables in Y_i only and no leaf is labelled by 0 or 1. If we get an upper bound on number of functions computable on variables in Y_i by fixing variables in other bolcks in F, then using equation 15 it gives us a lower bound on number of leaves labelled by Y_i in F. Since number of internal nodes roughly equals number of leaves in a formula, if we restrict F to variables in Y_i, then total number of internal nodes is roughly $l(Y_i)$. Since the formula F is over full binary basis, we have 16 different possibilities of function that a particular internal node computes.

$$\forall i, \quad \left.\begin{array}{l}\textit{Number of different functions}\\ \textit{computable on variables in } Y_i\end{array}\right\} \leq 16^{l(Y_i)} \tag{16}$$

From equation 15 and 16,

$$\forall i,\ l(Y_i) \geq \log\binom{2^c}{b-1}$$

Since the formula size is atleast the number of leaves labelled by its variables,

$$\begin{aligned}|F| &\geq \sum_{i=1}^{b} l(Y_i)\\ &\geq b\log\binom{2^c}{b-1}\\ &\geq b(b-1)\log\left(\frac{2^c}{b}\right)\end{aligned}$$

We know that $bc = n$, by setting $b = \frac{n}{\log n}$ and $c = \log n$, we get

$$|F| = \Omega\left(\frac{n^2}{\log^2 n}\log\log n\right)$$

□

By setting the values of b and c optimally in the above proof you can get a lower bound of $\Omega\left(\frac{n^2}{\log n}\right)$. Nechiporuk bound is the best known lower bound on the size of formulae over full binary basis.

Exercise : Show that by this method one cannot get a lower bound better than $\Omega\left(\frac{n^2}{\log n}\right)$.

6 Randomized Circuits

Use of randomization is found to be useful in many applications. We can ask the same question about circuit complexity. Let us first formally define what randomized circuit is:

Definition 20. Randomized circuits: *Let $C(x_1, \cdots, x_n, r_1, \cdots, r_m)$ be a circuit. We say C computes function $f : \{0,1\}^n \to \{0,1\}$ with error probability ϵ if*

$$\forall x, Pr_{r \in \{0,1\}^m}\left[C(x,r) \neq f(x)\right] \leq \epsilon$$

then C is a randomized circuit for computing f.

We can ask several questions about randomized circuits

1. Can we have a function whose randomized circuit complexity is exponential in the number of variables?
2. Are randomized circuits more powerful than deterministic circuits?
3. Given a randomized circuit for a function, can we derandomized it?

6.1 Hard functions for randomized circuits

Following theorem gives an affirmative answer for question 1.

Theorem 21. *There exists a function f on n variables such that for all randomized circuits C of size $<< \frac{2^n}{n}$, C cannot compute f.*

Proof. Shannon's counting argument for deterministic circuits works in this case also since any fixed randomized circuits cannot compute two different functions. □

6.2 Derandomization

From definition 20, we can make the following observation:

Observation 22. *If C is a randomized circuit computing f with some error probability $\epsilon < 1/2$ then*

$$\exists r s.t. Pr_{x \in \{0,1\}^n}\left[C(x,r) = f(x)\right] \geq (1 - \epsilon)$$

It means if a randomized circuit C computes a function f then there exists one fixed string r that works for most of the inputs. When we talk about derandomizing a given circuit, we only allow blow up in size of circuit which is polynomial in size of given circuit. So the problem of derandomizing a circuit is equivalent to asking following question

1. Given a randomized circuit C computing a function f, can we find a cricuit C' of size $O(|C|^k)$ for some constant k and a string r' such that $C'(x, r') = f(x)$ for all x?

Surprisingly, the answer to the above question is also yes, given by following theorem

Theorem 23. *For every randomized circuit of size s computing a function f, there exists a deterministic circuit of size at most polynomial in s computing the same function.*

Proof Idea : We will prove above theorem by showing the existence of a circuit C' that outputs correct answer for every input, given a randomized circuit computing f. We will use the method of amplifying the success probability of a circuit by *repetition*. It is enough to show existence of a circuit computing f with error probability less than 2^{-n}, since in this case by observation 22, there exists a string r' that works for more than $(1 - 2^{-n})$ fraction of inputs. It means it works for every input! So by fixing such string r' in circuit C' we get a deterministic circuit computing f.

Before proving the theorem we prove the following lemma, the proof of the theorem follows directly for the lemma

Lemma 24. *Suppose there exists a circuit $C(x_1, x_2, \cdots x_n, r_1, \cdots, r_m)$ computing a function $f : \{0,1\}^n \to \{0,1\}$ with error probability 0.1 (0.1 can be replaced with any positive constant less than 0.5) , then $\exists$ circuit $C'(x, r')$ such that C' computes f with error probability $\leq \delta$, where*

- $r' = (r_1^{'}, \cdots, r_{m'}^{'})$, $|r_i^{'}| = m$
- $m' = O\left(\log\left(\frac{1}{\delta}\right)\right)$
- $|C'| = poly\left(\log(\frac{1}{\delta})\right) + |C|.O\left(\log(\frac{1}{\delta})\right)$

Proof. We construct circuit C' as follows. C' consists of $k = O(\log(1/\delta))$ copies of circuit C, each of these circuits $C_1, C_2, \cdots, C_k$ has their own seperate random bits as an input. The k output bits is fed to MAJ and output of MAJ is the output of circuit C'. Clearly, circuit C' satisfies above *three* properties.

We will now show that the cricuit C' has desired error probability. Let Z_i be an indicator random variable which is 1 if $C_i(x, s_i) \neq f(x)$ and 0 otherwise.

$$Pr[Z_i = 1] = 0.1$$

Let

$$Z = \sum_{i=1}^{k} Z_i$$

Then the expected value of Z is:

$$\mathbb{E}[Z] = 0.1k$$

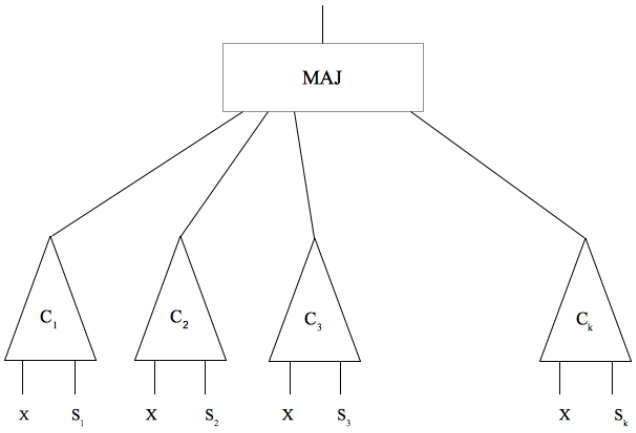

Figure 1: Construction of deterministic circuit from randomized circuit

Circuit C' outputs wrong answer iff more than half of the circuits $C_1, C_2, \cdots, C_k$ output wrong answer. So $Z > 0.5k$ is the event that circuit C' outputs wrong answer. Since the events $Z_1, Z_2, \cdots, Z_k$ are independent events, we can apply Chernoff bounds. The probability that C' outputs wrong answer is bounded by :

$$\begin{aligned} Pr[\text{ C' outputs wrong answer}] &= Pr[Z > 0.5k] \\ &\leq Pr\left[|Z - 0.1k| > 4 \times 0.1k\right] \\ &\leq e^{-O(k)} \\ &= e^{-O(\log(\frac{1}{\delta}))} \\ &< \delta \end{aligned} \tag{17}$$

□

From the above lemma we can now prove theorem 23

Proof of theorem 23 : Setting $\delta = 2^{-n}$, we get

$$m' = O\left(\log\left(\frac{1}{\delta}\right)\right) = O(n)$$
$$|C'| \leq poly(n) + O(|C|n)$$

So the size of deterministic circuit is at most poly in size of orignal randomized circuit and the error probability of circuit C' is less than 2^{-n}. Thus, the theorem follows. □

6.3 Pseudorandom Set

Motivation : Now that we know that every function being computed by a randomized circuit can also be computed by a deterministic circuit of about the same size, a natural question would be to find this circuit given the randomized circuit. From the proof discussed above, one way of

solving this problem would be to find the good string r, which can be substituted for the random string for every input, while keeping the output the same. The aim in the next section is to show the existence of a small set of strings such that a string choosen uniformly at random from this set behaves like a truly random string for the given circuit. Let us formalize the notion below.

Definition 25. *The set $\{r_1, r_2, \cdots, r_t\}$, where $r_i \in \{0,1\}^m$ is said to be pseudorandom set for a randomized circuits $C(x,r)$, $x \in \{0,1\}^n, r \in \{0,1\}^m$ computing some function $f : \{0,1\}^n \to \{0,1\}$ with error probability 0.1, if*

$$\forall x, Pr_{i\in[t]}\left[C(x,r_i) \neq f(x)\right] \leq 0.2$$

Following claim shows the existence of "small" pseudorandom set for family of circuits

Claim 26. *$\forall c$, $\forall n$, $m = n^c$, $\exists r_1, r_2, \cdots, r_t \in \{0,1\}^m$, $t \leq n^{2c}$, such that for all randomized circuits $C(x,r)$, $x \in \{0,1\}^n, r \in \{0,1\}^m$ computing some function $f : \{0,1\}^n \to \{0,1\}$ with error probability 0.1, and $|C| \leq n^c$*

$$\forall x, Pr_{i\in[t]}\left[C(x,r_i) \neq f(x)\right] \leq 0.2$$