U.C. Berkeley
CS294: Pseudorandomness and Combinatorial Constructions
Professor Luca Trevisan

Handout N5
September 13, 2005
Scribe: Gatis Midrijanis

Notes for Lecture 5

In the next few lectures we are going to look at generators that are good against circuits of feasible size. The constructions will be conditional by using complexity theoretical assumptions.

1 Circuits

A circuit $\mathcal{C}$ has n inputs, m outputs, and is constructed with AND gates, OR gates and NOT gates. Each gate has fan-in 2 except NOT gate which has fan-in 1. The fan-out can be any number. One can look at a circuit like a graph without cycles. See Figure 1 for an example.

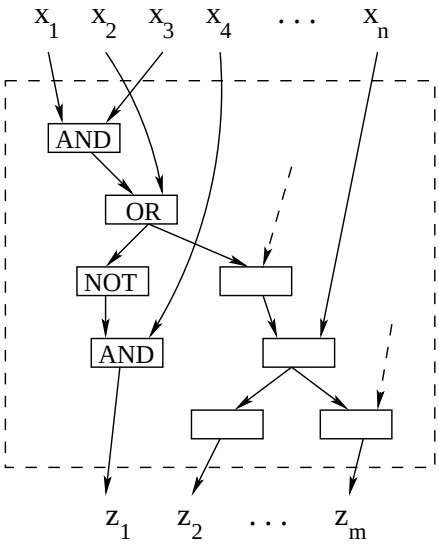

Figure 1: A Boolean circuit

A circuit $\mathcal{C}$ with n inputs gates and m output gates computes a function $f : \{0,1\}^n \to \{0,1\}^m$ if we fix an ordering of the input and output gates and the computation goes in intuitive way. The size of of a circuit $\mathcal{C}$ is defined as the number of AND and OR gates in it. By convention, we do not count the NOT gates. (Counting such gates would increase the size of circuit at most by factor of 2.)

If $m = 1$ then circuit $\mathcal{C}$ computes a predicate.

It is well known that any reasonable change in this definition is polynomially related with this one; if fan-in is bounded then even linearly related.

Unlike uniform complexity measures, like time and space, for which there are languages of arbitrarily high complexity, the size complexity of a problem is always at most exponential.

Theorem 1 *For every function $f : \{0,1\}^n \to \{0,1\}$ there is a circuit of size $O(2^n)$ that computes f.*

PROOF: We use the identity $f(x_1, ..., x_n) = (x_1 \wedge f(1, x_2, ..., x_n)) \vee (\overline{x_1} \wedge f(0, x_2, ..., x_n))$ to recursively construct a circuit for f, as shown in Figure 2. The recurrence relation for the size of the circuit is: $s(n) = 3 + 2s(n-1)$ with base case $s(1) = 1$, which solves to $s(n) = 2 \cdot 2^n - 3 = O(2^n)$. □

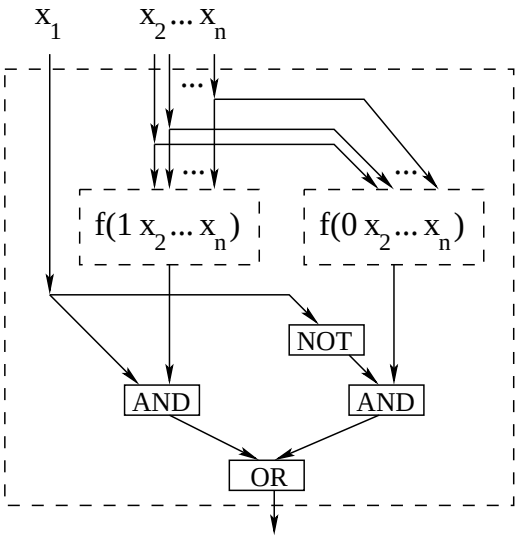

Figure 2: A circuit computing any function $f(x_1, x_2, ..., x_n)$ assuming circuits for functions $f(1, x_2, ..., x_n)$ and $f(0, x_2, ..., x_n)$

This bound is almost tight to the best possible $O(2^n/n)$ bound.

Another important fact is that everything that is computable with Turing machine is computable with a circuit similar complexity.

Theorem 2 *If* M *is a Turing machine s.t. on inputs on length n it always halts in* $\leq t(n)$ *steps (and accepts or rejects). Then for every n there is a circuit of size at most* $c_M \cdot (t(n))^2$ *with n inputs that simulates* M *on inputs on length n for some constant* c_M*.*

PROOF: Let Σ denote the alphabet of tape and Q denote the set of the states. Consider the $t(n) \times t(n)$ tableau of the computation of $M(x)$. See Figure 3.

Let $k \approx 2^{|\Sigma| \cdot (|Q|+1)}$ denote the number of bits necessary to encode each entry of the tableau. Each entry depends on three entries up. By Theorem 1, the transition function $f : \{0,1\}^{3k} \to \{0,1\}^k$ used by M can be implemented by "next state circuit" of size $k \cdot O(2^{3k})$, which is constant in n. Therefore the total size of a circuit that simulates complete tableau is $O((k2^{3k}t(n))^2) = O((t(n))^2)$. This is shown in Figure 4.

□

The best known result of this kind gives a $c_M \cdot t(n) \cdot (\log t(n))^2$ upper bound.

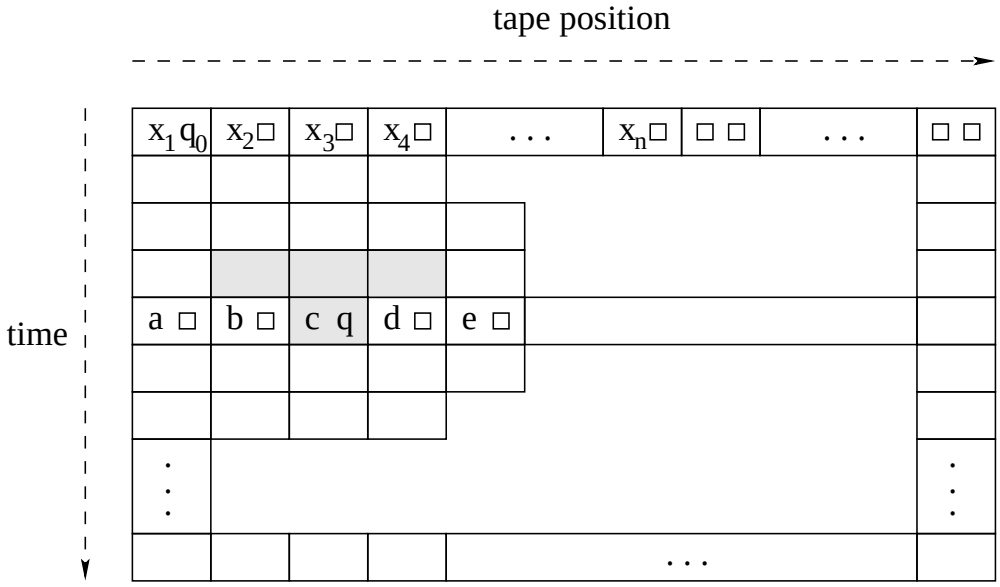

Figure 3: $t(n) \times t(n)$ tableau of the computation. The left entry of each cell is the tape symbol at that position and time. The right entry is the machine state or a blank symbol □, depending on the position of the machine head.

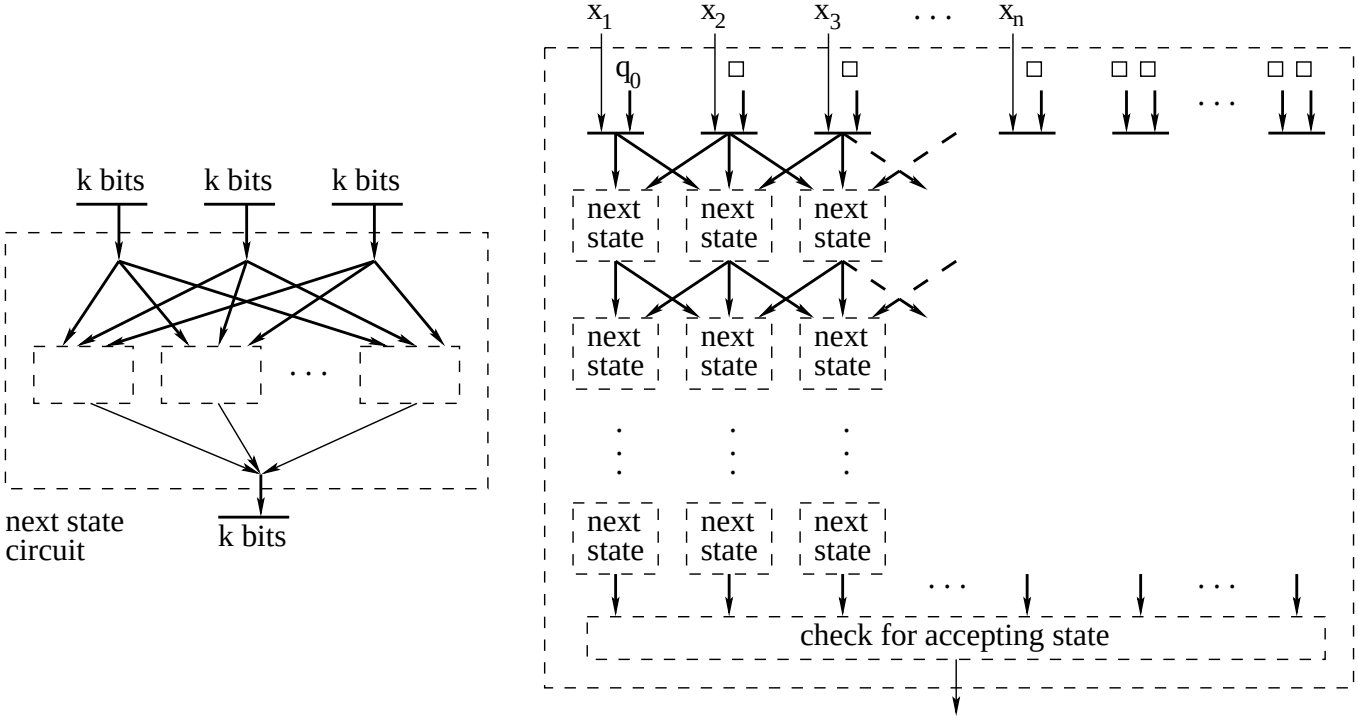

Figure 4: The circuit to simulate a Turing machine M computation by constructing the tableau.

2 Pseudorandom Generators

Definition 3 (Indistinguishability - finite definition) *Two random variables X and Y distributed over $\{0,1\}^n$ are called (S,ϵ)-indistinguishable if for every circuit $\mathcal{C}$ of size at most S,*

$$\left| \Pr_{x\sim X}[C(x)=1] - \Pr_{y\sim Y}[C(y)=1] \right| \leq \epsilon. \tag{1}$$

Definition 4 (Pseudorandomness - finite definition) *A random variable X is called (S,ϵ)-pseudorandom if it is (S,ϵ)-indistinguishable from U_n, the uniform distribution over $\{0,1\}^n$.*

This definition is much simpler than the definition using Turing machines because for TM we have to fix a universal TM to give a finite definition.

We can also give an asymptotic definition of pseudorandom generator as follows.

Definition 5 (BMY-type pseudorandom generator - asymptotic definition) *A function $G: \{0,1\}^n \to \{0,1\}^{l(n)}$, $l(n) > n$ is called a BMY-type (Blum-Micali-Yao) pseudorandom generator with stretch $l(n)$ if*

- *G on input of length n has output of length $l(n) > n$*
- *G is computable in polynomial time*
- *For every polynomials p, q and for every sufficiently large n, $G(U_n)$ is $(p(n), \frac{1}{q(n)})$-pseudorandom.*

For example, lets take $l(n) = n+1$ and $G: \{0,1\}^n \to \{0,1\}^{n+1}$. Let a circuit C be of size at most polynomial $p(n)$ with $n+1$ inputs. If C is BMY-type pseudorandom generator then

$$\left| \Pr_{x\sim\{0,1\}^n}[C(G(x))=1] - \Pr_{r\sim\{0,1\}^{n+1}}[C(r)=1] \right| \leq \frac{1}{q(n)}, \tag{2}$$

for any polynomial $q(n)$ and large enough n.

It is easy to see that if $\mathbf{P} = \mathbf{NP}$ (or even if $\mathbf{NP}$ is solvable with polynomial size circuits) then no BMY-type generators exist. Since $G(x)$ can be simulated by a Turing machine in a polynomial time, given output y we can just non-deterministically guess x and check whether $G(x) = y$. By Theorem 2 this algorithm can be simulated by a polynomial size circuit.

3 One-Way Functions

The main result of this and the next lecture is the following.

Theorem 6 *If one-way permutations exist then pseudorandom generators exist.*

Definition 7 (One-Way Permutation - finite definition) *A bijective function $f: \{0,1\}^n \to \{0,1\}^n$ is a (S,ϵ)-one-way permutation if for every circuit $\mathcal{C}$ of size at most S,*

$$\Pr_{x\sim\{0,1\}^n}[C(f(x)) = x] \leq \epsilon. \tag{3}$$

Definition 8 (One-Way Permutation - asymptotic definition) *Call $f = \{f_n\}_{n\geq 1}$ one-way, where $f_n : \{0,1\}^n \to \{0,1\}^n$ is bijective function computable in a polynomial time, if for every polynomials p and q and for every n large enough f_n is $(p(n), \frac{1}{q(n)})$-one-way.*

For example, let $g : \mathbb{N} \to \mathbb{N}$ be a function $g(x) := x^3 \mod N$, where $N = pq$ for randomly chosen prime numbers p and q such that 3 does not divide $(p-1)(q-1)$. Strictly speaking, g cannot be one-way permutation by previous definition even when we represent natural numbers by bit strings, but there are several ways how to extend the definition to include such functions. First, we can allow more structured input and output of one-way function, in this case, by allowing to give prime numbers p and q in the input. Second, we could speak about probability distribution over families of functions $\{f\}$. Both of those changes in the definition don't change the results in this lecture.

Definition 9 (Hard-Core predicate - finite definition) *A function $B : \{0,1\}^n \to \{0,1\}$ is a (S, ϵ)-hard-core predicate for a permutation $f : \{0,1\}^n \to \{0,1\}^n$ if for every circuit $\mathcal{C}$ of size at most S,*

$$\Pr_{x\sim\{0,1\}^n}[C(f(x)) = B(x)] \leq \frac{1}{2} + \epsilon. \tag{4}$$

Definition 10 (Hard-Core Predicate - asymptotic definition) *$B = \{B_n\}_{n\geq 1}$ is hard-core predicate for family of permutations $f = \{f_n\}_{n\geq 1}$, where $B_n : \{0,1\}^n \to \{0,1\}$ and $f_n : \{0,1\}^n \to \{0,1\}^n$ are polynomial time computable functions, if for every polynomial p and q and for every n large enough, B_n is $(p(n), \frac{1}{q(n)})$-hard-core for f_n.*

For above-mentioned function g, an equivalent of hard-core predicate is $x \mod 2$. It means that to distinguish ($x \mod 2$) from random bit is as hard as to compute x given only $g(x)$.

Exercise: Prove that if B is hard-core predicate for f then f is one-way permutation. Hint: notice that if f wouldn't be one-way then we could compute x from $f(x)$ and then $B(x)$ in polynomial time.

In the next lecture we will prove the following two important theorems.

Theorem 11 (Goldreich-Levin) *Let f be a one-way permutation and g such that $g_{2n}(x,r) = f(n), r$. Define $B_{2n}(x,r) = x \cdot r = \sum_i x_i r_i \mod 2$. Then g is one-way permutation and B is a hard-core predicate for g.*

Theorem 12 (Blum-Micali-Yao) *If B is a hard-core predicate for f then $x \to f(x), B(x)$ is a BMY-type pseudorandom generator.*