6.841 Advanced Complexity Theory Feb 11, 2009

Lecture 3

Lecturer: Madhu Sudan *Scribe: Debmalya Panigrahi*

In today's lecture, we will focus on *non-uniform models of computation.* In non-uniform computation, we have a *different gadget/program/machine for each input size* for a given problem. Specifically, in this lecture, we will focus on *circuit families* (which are equivalent to deterministic Turing Machines (DTMs) with *advice* strings), *branching programs* and *formulas*, and compare them. Finally, we will study *Neciporuk's lower bound* on the succinctness of branching programs.

1 Circuits and Circuit Families

A *circuit* is a *directed acyclic graph* (DAG), where the nodes are of three kinds: *input nodes*, *logic gates* and an *output node*. The goal of a circuit with n input nodes is to compute a boolean function $f : \{0,1\}^n \to \{0,1\}$[1] in the following way: if a boolean vector $\bar{x} = (x_1, x_2, \ldots, x_n)$ is fed to the n input nodes, then the logic gates compute the function $f(\bar{x})$ and output it at the output node. A *basis* is a set of logic gates that can be used to compute *any* boolean function.

A circuit C has two main complexity measures:

- The *size* of circuit C, denoted by $\mathrm{SIZE}(C)$ is the number of edges in C.[2] For a function f, $\mathrm{C-SIZE}(f) = \min_{C \text{computes} f} \mathrm{SIZE}(C)$. $\mathrm{C-SIZE}(f)$ depends on the basis of gates used in the circuit, but only upto a constant factor.
- The *depth* of circuit C, denoted by $\mathrm{DEPTH}(C)$ is the longest path in C.

The following is a set of simple facts about circuits:

- Given a circuit C of size S, evaluating $C(\bar{x})$ for an input vector $\bar{x}$ takes $O(S^2)$ time.
- To simulate a $DTIME(t(n))$ TM on an input of length n, we need a circuit of size $O(t(n)^2)$.
- For any function $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{C-SIZE}(f) = O(2^n)$ (compare this to the fact that some functions might take much longer to compute).
- Circuits of size at most S ($S \geq n$, where n is the input size) can compute roughly $S^{S \log S}$ different functions. (This can be used to show that there are functions (in fact, even a random function) which have circuit complexity of $\Omega(2^n/n)$. In fact, the worst-case complexity of an n-input function is also $(1 - o(1))2^n/n$.)

A *circuit family* $\{C_n\}$ is an infinite sequence of circuits, one for each input size. It is said to compute function $f : \{0,1\}^* \to \{0,1\}$ if C_n computes the function $f|_n : \{0,1\}^n \to \{0,1\}$, which is identical to the function f, except that the input is restricted to be of length n.

We now compare functions that can be computed by poly-sized circuits families and those that have poly-time DTMs (i.e. the complexity class **P**). Clearly, a poly-time DTM can be simulated by a poly-sized circuit (follows from the second fact we mentioned above). However, the converse is not true. Consider, for instance, the language

$$L = \{1^n : 1^n \text{ is the unary representation of a TM } M \text{ and a string } x \text{ such that } M \text{ halts on } x\}.$$

[1]In general, a circuit may have $m \geq 1$ output nodes and compute a function $f : \{0,1\}^n \to \{0,1\}^m$.

[2]Often, the size of a circuit is defined as the number of logic gates (i.e. internal nodes) in the circuit. The two definitions are identical upto polynomial factors.

Clearly, there is a poly-sized circuit family for L; the answer for a specific n can be hard-coded in the circuit. However, this is the halting problem which is known to be undecidable; therefore, in particular, $L \notin \mathbf{P}$.

This leads us to the definition of advice strings. A deterministic TM M with *advice sequence* $\bar{a} = (a_1, a_2, \ldots)$, where $a_i \in \{0,1\}^*$, accepts the following language:

$$L(M, \bar{a}) = \{x : M(x, a_{|x|})\text{accepts}\}. \tag{1}$$

If $\ell(n) = |a_n|$ and M runs in time $t(n)$, then

$$L(M, \bar{a}) \in DTIME(t(n))/\ell(n). \tag{2}$$

In particular,

$$\mathbf{P/poly} = \cup_{\text{polynomials}p,q} DTIME(p(n))/q(n) \tag{3}$$

is equivalent to poly-sized circuits in computation power. If a language is in $\mathbf{P/poly}$, then the advice string for input length n can be hard-coded into the poly-sized circuit simulating the poly-time DTM for the language. On the other hand, if a language has a poly-sized circuit family, then the corresponding circuits can be used as advice strings to the DTM, which simply evaluates the input for the circuit.

If one can show that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, then it would follow that $\mathbf{P} \neq \mathbf{NP}$[3]. However, for problems in $\mathbf{NP}$, the best lower bound known on the length of the advice string required is $4.5n$.

2 Branching Programs

A branching program (BP) is a DAG computing some function $f : \{0,1\}^n \to \{0,1\}$. The nodes of the DAG are of two types: *branching nodes*, each of which represents an input variable and has two outgoing edges labeled 0 and 1 respectively, and two *output nodes* labeled 0 and 1 and having no outgoing edges. To compute the function for a given input, we start at a designated input node and take the outgoing edge at each branching node corresponding to the value of the variable in the input. Finally, we reach the output node corresponding to the value of the function for the given input.

A BP is said to be *layered* if the nodes can be partitioned into layers $L_1, L_2, \ldots, L_k$ such that all edges are from nodes in layer L_i to those in layer L_{i+1}, for any i. The primary complexity measures of a layered BP B are the following:

- $WIDTH(B) = \max_i |L_i|$, and
- $\text{SIZE}(B) = \sum_i |L_i|$.

We mention the following facts abouts BPs:

- A language having a log-space DTM (i.e. is in the complexity class $\mathbf{L}$) has a poly-sized BP.
- A circuit can simulate a BP with roughly the same size, but not vice-versa.

3 Formulas

A *formula* is a circuit where each logic gate has maximum out-degree of 1. The following facts about formulas are worth mentioning:

- A change of basis of the logic gates affects the depth of a formula computing a function by only a constant factor, but its size can change by a polynomial factor.

[3]Minor typo in previous version corrected by Madhu.

- If for function f, $\mathrm{F}-\mathrm{SIZE}(f)$ and $F-\mathrm{DEPTH}(f)$ respectively represent the minimum size and depth of a formula computing f, then

$$F - \mathrm{DEPTH}(f) = O(\log \mathrm{F} - \mathrm{SIZE}(f)). \tag{4}$$

- For any formula, there is a BP computing the same function whose size at most a polynomial factor greater than that of the formula.

4 Neciporuk's Lower Bound

We construct an explicit family of boolean functions that require BPs of almost quadratic size. Consider any function f with a BP of size $\mathrm{BP}-\mathrm{SIZE}(f)$. Let S be a subset of input variables of f and $\mathrm{BP}-\mathrm{SIZE}_S(f)$ denote the number of nodes in the BP of f that read the variables in S. Now, if $S_1, S_2, \ldots, S_k$ is a partition of the input variables of f, then

$$\mathrm{BP} - \mathrm{SIZE}(f) = \sum_{i=1}^{k} \mathrm{BP} - \mathrm{SIZE}_{S_i}(f). \tag{5}$$

Now, consider the function f where all the variables other than those in S_i have been fixed by an assignment ρ. Let this function be denoted by $f|_\rho$. Then

$$\mathrm{BP} - \mathrm{SIZE}_{S_i}(f) \geq \mathrm{BP} - \mathrm{SIZE}(f|_\rho). \tag{6}$$

We define a function f (called *distinctness*) on an input of length kl, where the input is grouped into k groups of length l each. Let us denote the input by $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k)$, where $\bar{x}_i = (x_{i1}, x_{i2}, \ldots, x_{il})$. Then, f is defined as follows:

$$f(\bar{x}) = \begin{cases} 1 & , \text{ if } \bar{x}_i \neq \bar{x}_j,\ \forall i \neq j \\ 0 & , \text{ otherwise.} \end{cases} \tag{7}$$

Now, suppose we fix all inputs except $\bar{x}_i$, i.e. $\bar{x}_j = \bar{a}_j$ for $j \neq i$. This can be done in

$$\binom{2^l}{k-1} \geq \left(\frac{2^l}{k}\right)^k \tag{8}$$

ways. Thus, the number of different functions on $\bar{x}_i$ is at least $(\frac{2^l}{k})^k$. But, we know that a BP of size s can represent at most $2^{O(s)}$ functions. Thus,

$$\mathrm{BP} - \mathrm{SIZE}_{S_i}(f) = \Omega(lk - k\log k). \tag{9}$$

Therefore,

$$\mathrm{BP} - \mathrm{SIZE}(f) = \Omega(k(lk - k\log k)) = \Omega\left(\frac{n^2}{\log^2 n}\right), \tag{10}$$

by setting $k = n/\log n$ and $l = \log n$.