Recent advances in Complexity CIS 6930/CIS 4930 August 29th, 2002

Lecture 2

Lecturer: Dr. Meera Sitharam Scribe: Andrew Lomonosov

Boolean functions and Boolean circuits

The idea is that Turing Machines/RAM/usual computers that you have seen before are fairly closely related to Boolean circuits.

Recall the three themes we listed during previous lecture: P vs NP, NP vs Co-NP, P vs BPP. They concern respectively whether circuits are sufficient, whether short proofs are sufficient and whether randomization is necessary.

Recall the approach we mentioned that uses lower bounds for P vs NP theme. It concerns circuit depths and circuit sizes.

Now we define Boolean circuits (which are amenable to purely combinatorial analysis).

Definition of circuits

First we will be talking in terms of Boolean logic $\{\vee, \wedge, \neg\}$, i.e a complete basis or primitive logic functions.

Aside - Later we will talk about more general basis functions (+,*) leading to a more algebraic approach that extends to general arithmetic circuits and functions $f : N \to N$. For current Boolean basis $\{\vee, \wedge, \neg\}$ the approach is combinatorial, with unions and intersections of finite sets being employed to analyze and and or.

Our goal is to express a Boolean function $f \in B_n$ (set of boolean functions of n variables), that maps $f : \{0,1\}^n \to \{0,1\}$, using repeated applications of $\{\vee, \wedge, \neg\}s$. Function f *recognizes* a membership for a certain set, for example $S = \{x \in \Phi : (P_1(x) \wedge P_2(x) \vee P_3(x)\}$. Repeated applications - intuitively we break expression into 3, then represent $P_1(x)$ using $\{\vee, \wedge, \neg\}s$ etc, until they cannot be broken down any further. The *characteristic function* of S, X_S is the boolean function, computed using repeated applications of $\{\vee, \wedge, \neg\}s$.

Observation 1 *A characteristic function X_S of any set S of finite structures Φ can be broken down recursively into the computation of a boolean function using a composed application of $\{\vee, \wedge, \neg\}s$.*

The observation above illustrates *completeness* of boolean basis.

When we talk about circuits we think in terms of $\{\vee, \wedge, \neg\}s$ as gates. See Figure 1.

Recall that when we defined complexity measure we needed to define

- Input universe parameter. Here: $|X|$ $(X = x_1, \ldots, x_n, i_i \in \{0,1\})$.

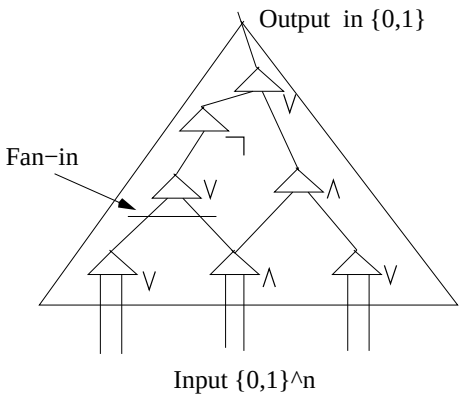

Figure 1: Boolean circuit

- Model of computation. Here: we specify circuit basis.
- Model parameters. Here: number of gates=size, longest path from output to input = depth.

In circuit model we also need to specify about whether fan-in is bounded or not. We can also consider fan-out, but it turns out that

Exercise 1 *All bounded fan-in circuits can be made to have bounded fan-out without increasing size or depth ([Hoover,Klawe,Pippenger, 1984, JACM 31]). Question: does it also holds for unbounded fan-in?*

Main things to say about circuit model:

- what the gates are, i.e what is the basis you are choosing
- whether fan-in is bounded or not
- whether you are bounding some other parameter such as depth

Why would you use circuits

1. First reason is that traditional, recursive-theoretic analysis does not work toward solving P vs NP problem. (Recursive-theoretic analysis is the one that studies what can be done by recursive functions). The one major lower bound result you have seen before was Halting problem, but that result was done as a proof by contradiction and it has no constructive algorithm. The problem with any method that does have an algorithm stems from the following result.

   BGS [1980] has showed that

   $$\exists A, P^A = NP^A$$

   for some oracle A

   but also

   $$\exists B, P^B \neq NP^B$$

   for some other oracle B.

Exercise 2 *Any method that is relativazable, i.e relative to an oracle cannot work to settle question of P=NP.*

And all traditional methods (e.g diagonalization that was used to show that $rec \subset r.e, rec \neq r.e$) are relativizable.

Thus we need to use alternative, more hands-on combinatorial/algebraic way of thinking and models of computation rather than from the recursion theory.

2. Second motivation is that circuits and circuit size/depth correspond strongly to RAM (sequential) time and space. It is easy to see that sequential time $\cong$ circuit size, since we need to spend constant time at every logic gate. It is also easy to see that parallel time $\cong$ circuit depth, since we cannot compute two gates simultaneously if one comes "after" another in the ordering of gates from leaves to root.

   See more details in "Connection of circuits to standard computational model" part of this lecture .

3. **THIS WAS NOT MENTIONED IN CLASS**

   Another advantage of circuits is that they easily lend themself to represent *superpositions* (or combinations) of the inputs, since each gate might be performing (simple) superposition of it inputs. Here is an example of a superposition. Let f and g be functions of two variables. Then $F(x, y, z) = f(x, g(y, z))$ is a function of the three variables x, y and z. This an example of a superposition constituted of the functions f and g.

   In general, a superposition of given functions means a function which is obtained by substitution of some of the functions in place of the arguments in other functions of the set.

   Important problem (so-called Hilbert's 13^{th} problem) is this

   Problem 1 *Is every analytic function of three variables a superposition of continuous functions of two variables? Is the root $x(a, b, c)$ of the equation*

$$x^7 + ax^3 + bx^2 + cx + 1 = 0$$

   a superposition of continuous functions of two variables?

   Initial attempts to solve this problem were using theory of multidimensional variation (Vitushkin 1955). Shannon (1948) suggested circuit size as a good parameter for capturing complexity of functions. Later these attempts were related to the concepts of Shannon's information theory (Kolmogorov 1955). Using these results, problem was eventually solved by Arnold in 1957.

Relationships between boolean formulae and boolean circuits

Generally boolean formulae and boolean circuits are considered equivalent. Every boolean formula, say $f(x) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge x_2 \wedge x_3$, corresponds to a boolean circuit and vice-versa. How is size of the circuit related to the size of the formula? If some gate has large $(=k)$ fan-out than the corresponding

expression (within some pair of ()s) will occur within formula k times, so size of the formula is greater or equal to the size of the circuit. If fan-out is bounded then size of the formula is the same as size of the circuit (within a constant multiple).

Relating circuits to standard computational models

By their nature circuits are a parallel model of computation. All gates on the same level can be executed simultaneously. Clearly depth of the circuit is equal to parallel execution time. Size of the circuit corresponds to sequential time, i.e, time in a standard sequential model of computation like a turing machine or the usual computer. . What about sequential space? If we look at the amount of space needed to compute result at gate A, we need 1 unit of space for leftmost child of A and than we can determine amount for right child of A recursively, so total amount of space is bounded by (max fan-in)*(depth of the circuit). If fan-in is assumed to bounded then sequential space corresponds to circuit depth.

Recall that we have shown that (sequential) polynomial time corresponds to polynomial circuit size. Hence we can talk about P vs NP in terms of circuit sizes. To clarify further, there is a notion

P/poly

which is a class of all boolean functions computed (*non-uniformly* - to be explained below) by polynomial size circuits.

The reason we talk about $P/poly$ and not P is because of the idea of *non-uniformity.* Notice that when we are talking about TM there is only one TM or computer that compute a desired function. This TM can take any input from $\{0,1\}^n$ and outputs 0 or 1. When we talk about circuits however, we need to have a different circuits for inputs of different size. There is no way to define circuits for variable values of n. So for circuits we have

$$f_n : \{0,1\}^n \rightarrow 0,1$$

and

$$f : \{f_n\}_{n \in N}$$

So the way we define a circuit is as a sequence of circuits

$$\{C_n\}_{n \in N}$$

or a *uniform sequence* (i.e we think of a way in which these circuits can be generated, or a program which generates these circuits).

Also there are *non-uniform sequences* where we actually write down (different) circuits, which may have no relation to smaller/larger ones what so ever. In $P/poly$ we allow poly-size non-uniform sequences of circuits (still of polynomial size though).

Note that given a TM it is easy to construct corresponding uniform sequence of circuits. However given a non-uniform sequence of circuits it is not clear how to construct a corresponding TM (unless this sequence is in fact uniform). So there is an important difference between P and P/poly, but for the purposes of this class this difference does not make a difference, since we will be talking about P/poly only. Note that $P \subset P/poly$, i.e set of non-uniform sequences of circuits is more powerful than P. Non-uniform circuits can solve Halting Problem for example, simply by taking an answer to Halting Problem and coding it into

a circuit, we don't need a computation for that. However note that this requires exponential circuit size and Halting Problem is not contained in P/poly.

What we are heading toward is showing that NP is not contained in P/poly i.e there are problems in NP that do not even have a non-uniform sequence of poly-size circuits, see Figure 2. Hence a superpolynomial (non-uniform) circuit size lower bound on an NP function (without loss of generality NP-complete function), i.e showing that $NP \not\subseteq P/poly$ is stronger than showing $P \neq NP$.

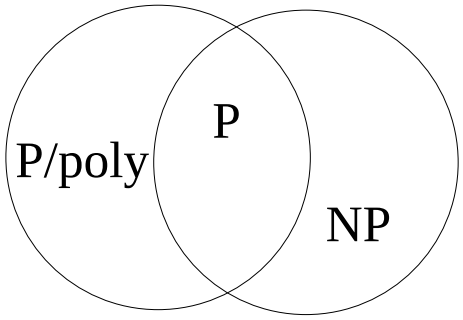

Figure 2: Map of NP and P/poly

Road map for next week

Tuesday 1^{st} period - circuit intuition - what circuits can and cannot do. What kind of circuit intuition will we need? First we will start with depth 2 circuits, i.e, plain CNF and DNF formulae and think about the sizes of these circuits for some Boolean functions. This leads to some basic concepts that we will use later. Next we talk about arbitrary depth circuits. It turns out that *symmetric functions* all have $O(\log n)$ depth and $O(n)$ size. Function $f(x_1, \ldots, x_n)$ is called symmetric function if

$$\forall \pi f(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$$

where π is a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$.

All the usual functions over booleans, such as addition, multiplication, subtraction, matrix multiplication, matrix determinant can be done in similar size and depth of circuits (Krapchenko adder, Lupanov (62)).

We will also talk about why almost all boolean functions need the large circuit size. Note that without bounding circuit depth some functions require linear size $O(n)$.

Tuesday 2^{nd} period - will show that *parity function* needs exponential size bounded depth circuits. Thursday will be spend on completing this proof (the proof was gradual progress by Furst-Saxe-Sipser, Ajtai, Yao, Hastad etc, we will directly jump into Hastad's version, using the so-called Hastad's switching lemma, which is a general combinatorial result that turns out to have plenty of other applications as well).