80240233: Complexity of Computation
ITCS, Tsinghua Univesity, Fall 2007

Lecture 4
19 October 2007

Instructor: Elad Verbin

Notes by: Decheng Dai

In this lecture, we discuss Circuit, which are a computational model stronger than Turing Machines. We prove the Karp-Lipton-Sipser Theorem, which states that if the Polynomial Hierarchy does not collapse, than polynomial-sized circuits cannot solve NP-hard problems. (Intuitively, this means that circuits are not a lot stronger than Turing machines as far as the P versus NP question is concerned).

1 Circuits

A *circuit* is a directed acyclic graph where nodes are called gates and edges are called wires. Input gates are gates with in-degree zero, and output gates are gates with out-degree zero. Input (resp. output) gates are labeled with bits of the input (resp. output) in a one-to-one fashion; each non-input gate is an AND, OR or NOT gate. AND and OR gates have a fan-in of 2, while NOT gates have a fan-in of 1. An important point is that gates may have unbounded fan-out, unless explicitly stated otherwise.

A circuit C with n inputs and m outputs computes a function $f_C : \{0,1\}^n \to \{0,1\}^m$. An important complexity measures for circuits is the size, where the size of C is the number of AND and OR gates in the circuit C. (We justified that the NOT gates are not counted by noting that any circuit C has a circuit C' that computes the same function, has the same number of AND and OR gates, and where the number of NOT gates is at most the number of other gates, plus n).

For simplicity we, from now on, consider only circuits with one output bit, i.e. $m = 1$.

2 How Strong are Circuits?

It is natural to wonder how many boolean functions can a circuit with at most n inputs and t AND and OR gates compute? Another interesting question is whether there is any relation between Turing machines and circuits? For the first question, there is an upper bound of $(n+t)^{O(n+t)}$ and a lower bound on $2^{\lfloor \frac{t-1}{n} \rfloor}$ which we show in this section. For the second question, the Karp-Lipton-Sipser Theorem, which we show in Section 4 gives some results.

Claim 1. *The set of all circuits on n bits of size t computes at most $(n+t)^{O(n+t)}$ distinct boolean functions.*

Proof. Note that the number of boolean functions with n inputs is at most 2^{2^n}. Therefore, when $t \geq 2^n$, the claim is trivially true. Suppose for now that there are no NOT gates. For any circuit with $t \in [0, 2^n)$, we consider the number of wires and possible entry-points in the circuit. Since

each gate has exactly two entry-points, there must be a total of $(2 \times t + 1)$ entry-points (the extra one is for the circuit's output). There is a total of $n + t$ gates (including the inputs), and each entry-point must be connected to exactly one of these. Therefore, there are at most $(2 \times t + 1)^{n+t}$ different connections. The effect of NOT gates is that each edge can be reversed by a NOT gate, so we need to multiply by 2^{n+t}. Furthermore, each two-input gate can be either AND or OR, so we need to multiply by 2^t. Overall, the number of circuits is at most

$$2^{n+t} \times 2^t \times (2t+1)^{n+t} \leq (n+t)^{O(n+t)}$$ □

Corollary 2. *if $n \geq 1000$, then there is a function f on n inputs that cannot be computed using only $2^{0.9n}$ gates.*

Proof. A counting argument. If $n \geq 1000$, $t = 2^{0.9n}$, then $(n + t) \approx 2^{0.9n}$, so $(n + t)^{O(n+t)} = 2^{0.9n \times 2^{0.9n}} < 2^{2^n}$. □

Claim 3. *The circuits with n inputs of size t can compute at least $2^{\lfloor \frac{t-n}{n} \rfloor}$ distinct functions.*

We leave this as an exercise and prove only an easier claim.

Claim 4. *Circuits of size $t = 2^n \times n + 1$ can compute all different functions of n variables.*

Proof. Let function $f : \{0,1\}^n \to \{0,1\}$, we use the CNF of f to construct a circuit. There are at most 2^n clauses in this formula and n variables in every clause. Every clause of the formula uses a n-AND gate(can be simulated with $n-1$ basic AND gates) and the formula is a 2^n-OR gate (can be simulated by $2^n - 1$ basic OR gates) taking all the clauses as its inputs.

$$(n-1) \times 2^n + (2^n - 1) \leq 2^n \times n - 1.$$ □

3 Asymptotic Circuit Complexity: Definitions

A circuit family $C = \{C_1, C_2, ..., C_n, ...\}$ is an infinite set of circuits such that C_i receives i input bits; a circuit family defines a function $f(x) = C_{|x|}(x)$ and a decision problem $L \subseteq \{0,1\}^*$, where

$$L = \{x \in \{0,1\}^* | C_{|x|}(x) = 1\}.$$

The class P/poly is the set of all decision problems accepted by circuit families $\{C_1, C_2, ..., C_n, ...\}$ s.t. $\text{SIZE}(C_n) \leq n^{O(1)}$. In general , for a function $m : \mathbb{N} \to \mathbb{N}$, $\text{SIZE}(m(n))$ is the set of languages accepted by circuit families $\{C_1, C_2, ..., C_n, ...\}$ where $\text{SIZE}(C_n) \leq m(n)$. So,

$$\text{P/poly} = \bigcup_{c \geq 1} \text{SIZE}(n^c)$$

4 The Karp-Lipton-Sipser Theorem

Theorem 5 (Karp-Lipton-Sipser)**.** *if* NP $\subseteq$ P/poly*, then* $\Pi_2 = \Sigma_2$*, so* PH $= \Sigma_2$.

According to the theorem, to prove that PH does not collapse to the second level, it is enough to prove that 3-SAT cannot be computed by poly-sized circuit. Before proving the above theorem, we first show a result that contains some of the ideas in the proof of the Karp-Lipton-Sipser theorem.

Lemma 6. *if* NP $\subseteq$ P/poly *then there exists a* poly*-size circuit family* $\{C_1, C_2, ..., C_n, ...\}$ *that gets as input a 3-CNF formula* Φ*, and returns a satisfying assignment if there exists one, or* 0*'s if* Φ *is not satisfiable.*

Proof sketch. Consider the following NP problem: given as input the formula Φ over n variables, a number k, and k bits $b_1, ..., b_k$, output 1 if and only if there is a satisfying assignment $x_1, ..., x_n$ for Φ where $x_i = b_i$, for all $1 \leq i \leq k$. By the assumption, there exists a polynomial size circuit family which realizes this NP computation. By composing $2n$ such circuits, we can build the witness from scratch. □

The lemma provides a way to guess the $0, 1$'s of some satisfying assignment for a 3-SAT formula one by one; we now prove the Karp-Lipton-Sipser theorem.

Proof of Theorem 5. We will show that if NP $\subseteq$ P/poly then $\Pi_2 \subseteq \Sigma_2$. By a result in lecture 3, this implies that PH $= \Sigma_2$. Let $L \in \Pi_2$, then there is a deterministic Turing Machine V such that

$$x \in L \iff \forall y_1 \exists y_2 . V(x, y_1, y_2) = 1.$$

(All quantifiers also restrict the variable to be a bit-string of polynomial size. To make the formulas easier to read, we do not write this explicitly in notation).

By adapting the proof of the lemma, we can show that for every n there is a circuit C_n of size poly(n) such that for every x and y_1,

$$\exists y_2 . V(x, y_1, y_2) = 1 \iff V(x, y_1, C_n(x, y_1)) = 1$$

Thus, we have that for inputs x of length n,

$$x \in L \iff \exists \langle C_n \rangle \forall y_1 . V(x, y_1, C_n(x, y_1)) = 1$$

where $\langle C_n \rangle$ stands for the binary string that represents C_n.

Thus, we have shown that $L \in \Sigma_2$. □

Note that in the proof we in fact show that if NP $\subseteq$ P/poly and if $L \in \Pi_2$ then there exists a machine V such that for every x,

$$x \in L \iff \langle C_n \rangle \forall y_1 . V(x, y_1, C_n(x, y_1)) = 1$$

where $\langle C_n \rangle$ is a string that *depends on* $|x|$ *but not on* x. Thus, we have shown that L is in a slightly smaller class than Σ_2, in which the existential variable depends on the length of the input but not on the input itself. This means that NP $\subseteq$ P/poly implies a collapsing result which is slightly stronger than $\Sigma_2 = \Pi_2$.