Instructor: Elad Verbin **Notes by:** Changcun Ma

In this lecture, we will talk about circuits and machines with advice.

1 Circuits

Recall the definition of P/poly from last time: This is the class of all decision problems that are decided by a family of circuits $\{C_1, C_2, \dots\}$, where C_n takes an n bit input and produces one bit of output, and the circuit sizes are polynomially bounded, that is there exists a polynomial p such that C_n has size at most $p(n)$.

We now show that circuits can simulate arbitrary deterministic computations:

Theorem 1. $\mathrm{P} \subseteq \mathrm{P/poly}$.

This theorem corresponds to the intuition that we can implement any computation in hardware — a circuit with AND, OR, and NOT gates — in a way that preserves the efficiency of the computation: Polynomial time computations give rise to polynomial size circuits.

This containment is in fact strict: There exist decision problems computable by polynomial-size circuit families but not by Turing Machines. This is related to the fact that circuit families are infinite objects – one for every input length, while Turing Machines are finite. So it is not surprising that circuits are more powerful than Turing Machines in this way. However, it is generally believed that circuits do not add much "interesting" power to Turing Machines. For instance there is no problem in NP (or EXP, or NEXP) that is believed to be decidable by circuit families but not by Turing Machines.

Proof sketch. Let $L \in \mathrm{P}$, so L is recognized by a TM M that runs in less than $p(n)$ time on inputs of length n for some polynomial p. Let Σ be the alphabet that M uses for its tape. (We will assume M has only one tape, as any machine can be simulated on a one-tape machine with polynomial slowdown.) Given an input length n, we look at the *computation tableau* of M on length n inputs: This is a table of dimensions $t(n) \times t(n)$ with entries in Σ that describes the computation history of M. The cell i, j of this tableau contains the contents of the jth cell of the tape at time i (including a special marker if the head of the tape happens to be there).

We associate a variable z_{ij} taking values in Σ with cell (i, j) of the tableau. The value in cell i at time j is then determined by a function

$$z_{i,j} = C_{i,j}(z_{i-1,j-1}, z_{i-1,j}, z_{i-1,j+1})$$

that determines the contents of cell i, j as a function of the contents of the three cells above it. Since $C_{i,j}$ is a function from Σ^3 to Σ, it can be represented by a circuit of size $O_\Sigma(1)$. We now

create a circuit C by "cascading" the circuits $C_{i,j}$ so that the input they receive comes from the top row $z_{1,j}$ and its output is the element $z_{t(n),0}$. We initialize the top row with the input x. The resulting circuit has size $O(t(n)^2)$. □

The idea in this proof is useful for establishing the NP-hardness of SAT. We will reduce from bounded halting by way of the following problem called CKTSAT:

INPUT: The description of a boolean circuit $C : \{0,1\}^n \to \{0,1\}$. PROBLEM: Is there an $x \in \{0,1\}^n$ such that $C(x) = 1$?

Clearly CKTSAT $\in$ NP. We now show CKTSAT is NP-hard by reducing from bounded halting (BH). The key observation we will use is that the way of building a circuit from a Turing Machine described in the proof of Theorem 1 is uniform: Given a description of a machine M an input length n, and a time bound t, we can construct in time poly$(\langle M \rangle, n, t)$ a circuit $C : \{0,1\}^n \to \{0,1\}$ such that for every x of length n, $C(x) = M(x)$.

Now we reduce BH to SAT as follows. Given an instance $(\langle N \rangle, x, 1^t)$ of BH, we think of the nondeterministic TM N as an NP-verifier that takes input x, $|x| = n$ and witness y, $|y| = m(n)$ and verifies that y is a witness for x. We then compute a circuit $C : \{0,1\}^{n+m(n)} \to \{0,1\}$ such that $C(z,y) = N(z,y)$ for every pair (z,y) of the appropriate length. We then hardwire $z = x$ in C to obtain a new circuit $C'(y)$. By construction $C'(y)$ accepts for some y if and only if $N(x,y)$ accepts in t steps for some y, so that $(\langle N \rangle, x, 1^t) \in$ BH if and only if $C' \in$ CKTSAT.

2 Machines with advice

One way to think of circuits is as *nonuniform* versions of Turing Machines. The Turing Machine has the same description on all input lengths, while circuits can act differently on different input lengths. But how different can circuits in the same circuit family be? There is a formal way to quantify this using the notion of *advice*.

A *Turing Machine with advice* is a special kind of machine that has, in addition to all its other tapes, an *advice tape*. When the machine receives its input, the advice tape is initialized by a string that depends only on the length of the input and not on the input itself. We say the advice has length $m(n)$ if on input x, only the first $m(|x|)$ cells of the advice tape are used. We will use $A(x,a)$ to denote the output of the machine A on input x and advice a. The running time of A on input length n is the maximum of the computation times of $A(x,a)$ for every input x of length n and advice string a of length $m(n)$.

We now give an acceptance condition for Turing Machines with advice. We say that A decides L if there exists a sequence of advice strings $a_1, \ldots$ such that

$$\begin{aligned} x \in L &\implies A(x, a_{|x|}) \text{ accepts} \\ x \notin L &\implies A(x, a_{|x|}) \text{ rejects.} \end{aligned}$$

Recall that SIZE$(s(n))$ is the class of decision problems recognized by circuit families of size $s(n)$. The two notions are closely related:

Theorem 2. *For every function $s(n) > n$, If $L \in \mathrm{SIZE}(s(n))$, then L can be decided by a Turing Machine that runs in time $O(s(n)^2)$ with advice of length $O(s(n) \log s(n))$.*

Proof. Suppose L is decided by a circuit family $\{C_1, \dots\}$ of size $s(n)$. Consider the following Turing Machine M with advice: On input x of length n, M interprets its advice a_n as the description of the circuit C_n and simulates C_n on input x. C_n can be described using $O(s(n) \log s(n))$ bits. Since C_n has size $s(n)$, the simulation can be done in time $O(s(n)^2)$. □

We have a containment in the other direction as well:

Theorem 3. *For every $t(n) > n$, if L can be decided by a Turing Machine running in time $t(n)$ with advice of length $m(n)$, then $L \in \mathrm{SIZE}(O(t(n + m(n))^2))$.*

This proof follows the same lines of the proof of Theorem 1. That argument gives a circuit $C(x, a)$ that simulates the corresponding Turing Machine for L. We hardwire the corresponding advice $a_{|x|}$ into this circuit in place of a to obtain a circuit C' for L on the given input length.

The notation $\mathrm{P}/m(n)$ is sometimes used for polynomial-time Turing Machines that take advice of length $m(n)$. The previous two theorems give the identity

$$\bigcup_{c>0} \mathrm{P}/n^c = \bigcup_{c>0} \mathrm{SIZE}(n^c).$$

This is consistent with our notation P/poly for the class of problems decided by polynomial-size circuit families.