Circuits

We're going to change our model of computation to the circuit model from Turing machines. There are several advantages of the circuit model:

- Enables discussion of parallelism.
- More manageable for lower bounds
- Introduces new ideas about uniformity & advice.

Circuit C

- directed acyclic graph
- nodes labeled with \( \wedge \lor \neg \times \text{ (variable) } \ 0/1 \)

\[ \text{in-degree 2} \quad \text{in-degree 1} \]

We don't actually need the 0/1 nodes since these can be removed from the CNet by changing the outcome.

\( n = \# \text{ input variables} \)
there is one sink (node w/ out-degree 0)

A circuit computes a function \( f: \mathbb{B}^n \to \mathbb{B}^n \).

\( C \leftrightarrow f \).

Complexity of a circuit is measured in terms of the gates.
We will also be interested in the depth of a circuit which is the longest path from an input to the output.

A formula is a net in which the graph is a tree
(fan-out of all gates = 1 - except the output).

Every function \( f: \mathbb{B}^n \to \mathbb{B}^n \) is computable by a circuit of size \( O(n \ 2^n) \).
Take the AND of n literals that encode each x s.t. f(x) = 1 then OR the ≤ 2^n such terms.

Note: circuits only work for a specific size input.

We've used to This which compute \( f: \Sigma^* \rightarrow \{0, 1\} \)

**Circuit Families** A circuit for each input length:\n
\[ C_1, C_2, C_3, \ldots \]

\( \exists C_n \text{ computes } f: \Sigma^* \rightarrow \{0, 1\} \text{ s.t. } \forall x \]

\[ C_n(x) = f(x) \]

We say that \( \exists C_n \text{ decides L where L is the language associated with } f. \)

How do the circuit to TM models compare?

**\( \exists \)** If there is a TM that decides L in time \( t(n) \) then there is a circuit family that decides L where the size of \( C_n \) is \( O(t(n)^2) \).

The proof of this is basically the same as A TM construction.

There is a tableaux corresponding to the TM computation on variable \( \pi \) input which can then be turned into a 

Tableaux (\( \pi \) bool) has size \( O(t(n)^2) \).

What about the other direction: if there is a circuit family that computes L, can it be computed by a TM?

Consider the following example:

\[ C_n = (x_1 \lor \neg x_1) \text{ if } \mu_n \text{ halts} \]

\[ C_n = (x_1 \land \neg x_1) \text{ if } \mu_n \text{ loops} \]
This circuit family decides a unary version of the halting problem. We can potentially encode uncomputable information into the specs of the Ck family.

**Solution: Uniformity.**

We require that the specification of a circuit is easy to compute.

**Definition:** A circuit family \( \{ C_n \} \) is *logspace uniform* if there is a TM which outputs \( C_n \) on input \( I_n \) and runs in logspace.

**Theorem:** \( P = \text{languages decidable by a logspace uniform poly-sized circuit families} \ \{ C_n \} \).

\[ P \Rightarrow \text{Un} \quad (\text{we did that above}) \]

\[ \text{Un} \Rightarrow \exists \text{poly-time TM} : M \quad \text{such that} \]

\[ C_{\text{inv}} \text{ rejects } C_k \text{ if } X \text{ is not accepted.} \]

**Turing Machines with Advice:**

A circuit family without the uniformity constraint is called "non-uniform." We can regard non-uniformity as another limited resource like space or time.

- add read-only advice tape to TM \( M \).
- advice only depends on \( n \), the size of the input. (\( A(n) = \text{advice} \))
- \( M \) decides \( L \) w/ advice \( A(n) \) iff \( M(x, A(1|x|)) \) accepts \( \iff x \in L \).
Complexity class: \( \text{DTM}(\log n) / \text{f}(n) \) = the set of languages for which:
\[
\exists A(n) \quad A : N \to \Sigma^* \quad |A(n)| \leq \text{f}(n)
\]
there is a TM \( M \) which decides \( L \) in time \( t(n) \) with advice \( A \).

The most important of such classes is \( \text{P/poly} = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)/n^k \)

Theorem: \( L \in \text{P/poly} \iff L \) decided by a family of (non-uniform) poly-sized \( A(n) \) is the description of \( C_n \)

On input \( x \), TM simulates \( C_{n+1}(x) \)

We believe that \( \text{NP} \neq \text{P} \).

Generally believed also that \( \text{NP} \neq \text{P/poly} \)
(i.e. SAT does not have poly-sized \( A(n) \)

even without the uniformity constraint).

Parallelism: Uniform circuits allow for a refinement of polynomial time:

\[
\text{depth} = \text{parallelism}, \quad \text{size} = \text{parallel size}
\]
NC ("Nick's Class") Hierarchy of log-space uniform circuits.

\[ NC_k = \text{languages computable by families of log-space uniform circuits w/poly # gates + depth } O(\log^k n) \]

\[ NC = \bigcup_{k \geq 1} NC_k. \quad \text{"Efficiently parallelizable problems."} \]

Example: Matrix Multiplication:

\[
\begin{bmatrix}
A \\
\end{bmatrix}
\begin{bmatrix}
B \\
\end{bmatrix} =
\begin{bmatrix}
AB \\
\end{bmatrix}
\]

What is the parallel complexity of this problem?

\begin{align*}
\text{Work} &= \text{poly}(n) \\
\text{Parallel Time} &= \log^k(n) \quad \text{-- for which } k? 
\end{align*}

Let's look at this problem where entries are boolean:

\[ (AB)_{ij} = \bigvee_k (a_{ik} \land b_{kj}) \]

To make the output a single bit, input \((A, B, i, j)\)

\[ \rightarrow (AB)_{ij} \]

Boolean Matrix Multiplication \(\leq NC_1\)

level 1: Compute \((a_{ik} \land b_{kj}) \land k\) in parallel.

Compute n-wise OR with a binary tree

(Height: \(\log n\),

(for single bit output, this is actually a formula).
$N_C \subseteq NC_2 \implies STCONN \in NC_2$

Let $A$ be the adjacency matrix for $G$ with self-loops added (1’s on diagonal).

$(A^k)_{ij} \iff \exists$ path from $i$ to $j$ of length $\leq k$.

We want: $(A^k)_{st} \implies A \to A^2 \to A^k \ldots \to A^{2^\log n}$

$\log n$ matrix multiplications $\implies \log^2 n$ depth.

Now we would like to establish that the notion of $P$-complete extends to the class $NC$ as well (i.e. if a $P$-complete problem is in $NC$, then $NC = P$).

In order to do this, we need to ensure that logspace reductions can be converted to logspace uniform $NC$ circuits. That is:

If $L \leq_L L'$ and $L' \in NC$ then $L \in NC$.

We need to show an $NC$ circuit that computes the reduction. Furthermore, the circuit needs to be from a logspace uniform $NC$ family. Consider a TM $T$ that takes in a pair $(x, i)$ st. $i \leq |R(x)|$. It accepts $x$ iff the $i$th bit of $R(x)$ is one. $T$ runs in logspace.

The idea is that the circuit will compute $STCONN$ on the configuration graph for $T$’s computation on $x$. Recall that we add a special node $t$ to this graph and connect all accepting
configurations to \( t \). The output of the whole circuit will be the \((c, i, t)\) entry of \((A^t)\)
where \(A^t\) is the configuration graph for \( t \)'s computation on input \( x \).

We already saw how to compute \( A^t \) for an arbitrary matrix \( A \). Now we need to discuss how to determine \( A^t \).

Consider two nodes in \( A^t \) corresponding to two configurations:
\[
C = (i, j, s_1, \ldots, s_r, q) \quad (i', j', s_1', \ldots, s_r', q') = C'
\]

The entry in \( A^t \) corresponding to these two nodes is either 0, 1, \( x_i \) or \( \overline{x_i} \).

- The only input bit on which this question \((c, c')\) an edge?
- can depend is \( x_i \). The rest of this decision is determined by the configurations themselves.

The log-space TM that computes the circuit can determine which of these four possibilities should be the bit \((A^t)_{c,c'}\).

\textbf{Circuit Lower Bounds:}

Major effort in complexity theory:

- prove lower-bounds for size of circuits that compute problems in \( \text{NP} \).

(a super-poly l.b. would actually show \( \text{NP} \neq \text{P/poly} \)).

The best known lower-bound is 4.5n

\(\text{w/o uniformity constraint} \).
There are, however, e.b.'s for restricted classes of circuits.

**Frustrating Fact:** almost all functions require huge circuits, just by a counting argument.

**Theorem (Shannon)** w.p. $\geq 1 - o(1)$ a random function $f: \{0,1\}^n \rightarrow \{0,1\}$ requires a circuit of size $\Omega(2^{n/\log(n)})$.

**Proof:** $B(n) = 2^n$ is the # of boolean functions with $n$ inputs.

\[ C(n, s) = \text{the # of circuits w/ } n \text{ inputs and size } s \]

\[ C(n, s) \leq \left( \frac{(n+3)2^s}{s^2} \right)^s \]

\[ C(n, c^{2^n/n}) < \left( 2n c^2 \cdot 2^{2^n/n^2} \right)^{c^{2^n/n}} < o(1) \cdot 2^{2c^2} < o(1) \cdot 2^n \quad \text{(if } c < \frac{1}{2} \text{)} \]

The probability that a randomly selected function has a circuit of size $s = c^{2^n/n}$ is

\[ \frac{C(n, s)}{B(n)} = o(1) \]

Naturally the best candidates for circuit lower bounds are NP complete problems.
Recent work has focused on special classes of circuits.

Monotone languages: \( L \subseteq \Sigma^0, \Sigma^1 \)
\[ x \in L \Rightarrow x' \notin L \quad \forall x' \quad x < x' \]

Saturating 0’s to 15.

Flipping an input bit from 0 to 1 either changes the output from 0 to 1 or not at all.

Some NP-complete problems are monotone:
( Hamiltonian cycle, clique, set cover)
but not others: SAT, graph coloring...

A monotone circuit has \( n + v \) gates, but no \( 7 \) gates.
Monotone circuits can only compute monotone functions.

\[ \text{Do all poly-time computeable monotone functions have poly-size monotone circuits?} \]
(this is true in the non-monotone case).

A monotone circuit for CLIQUE \( (n, k) \)
For each subset \( S \subseteq V \) \( |S| = k \)
\[ \land_{i,j \in S} x_{ij} \]
Take the OR of all the \( S \)'s.
Size = \( \binom{n}{k} \frac{1}{2} \)
For \( k = n^{1/4} \) size \( \approx n^{n^{1/4}} \)

Theorem (Rasborov '85)
Any monotone circuit for clique \( n, k \) \( \Rightarrow k = n^{1/4} \)
\( \text{has size} \geq 2^\Omega(n^{1/4}) \)
Note that a "yes" answer to (*) would imply \( P \neq NP \).

Unfortunately, Razborov also later proved:

Any monotone circuit for MATCHING also requires exponentially large circuits.