Outline

- Circuit Complexity
- Uniform Circuit Complexity
- P vs. Uniform Poly-size circuits
- P/poly
- BPP $\subseteq$ P/poly
- Monotone problems and circuits
Boolean Circuits and Languages

• Circuit $C_n$ with $n$ inputs $x_1, \ldots, x_n$, 1 output, gates NOT, OR, AND with fan-in 2. Two parameters of interest:
• **Size** of circuit = # gates
• **Depth** of circuit = maximum # gates in an input-output path
• $C$ **accepts** the set of binary strings $x = (x_1, \ldots, x_n)$ such that $C_n(x) = 1$

• If $L \subseteq \{0,1\}^*$ is a language then for each length $n$, we need a separate circuit $C_n$ to accept the subset of $L$ of strings of length $n$ (usually denoted as $L_n$)
• **Family of circuits** $C = (C_1, C_2, \ldots)$ where $C_n$ has $n$ inputs, accepts the language $L$ if for every $n$ and every string $x$ of length $n$, $C_n(x) = 1 \iff x \in L$
Circuit Complexity

- Size complexity of circuit family $C = (C_1, C_2, ...)$:
  \[ s(n) = \text{size}(C_n) \]

- Depth complexity of circuit family $C = (C_1, C_2, ...)$:
  \[ d(n) = \text{depth}(C_n) \]

- Circuit size complexity of a language $L$ over $\{0,1\}$ = minimum size complexity of a circuit family that accepts $L$ (i.e. for every $n$, pick a circuit $C_n$ of minimum size that accepts the strings of $L$ of length $n$).

- Circuit depth complexity of a language $L$ over $\{0,1\}$ = minimum depth complexity of a circuit family that accepts $L$.
Circuit complexity and uniformity

- There are undecidable languages that have linear circuit (size) complexity

  - Proof: Take undecidable language L over \{0,1\}. Corresponding unary language unary(L) over \{1\} is also undecidable. Every unary language has a trivial circuit family: $C_n$ outputs 1 (resp. 0) iff $1^n \in L$ (resp. $1^n \notin L$).

- Problem: We can pick a different circuit for every n, but definition of circuit complexity does not reflect the difficulty of finding the circuit for each n

- Uniform circuit family: There is a log-space Turing machine which on input $1^n$ outputs $C_n$.

- Uniform circuit size/depth complexity of a language
P vs. Polynomial Circuits

• A language $L$ over $\{0,1\}$ has uniformly polynomial size circuits iff $L \in P$.

• Proof: 1. Suppose $L \in P$. As in the proof that the Circuit-Value Problem is P-complete, for each $n$ we can construct a polynomial size circuit $C_n$ such that for every binary string $x$ of length $n$, $C_n(x) = 1 \iff x \in L$. The construction is done by a log-space TM.

• 2. Suppose $L$ is accepted by a uniform polynomial size circuit family $C = (C_1, C_2, \ldots)$. Given an input $x$ of length $|x| = n$, construct the circuit $C_n$ and evaluate it on input $x$. Accept iff $C_n(x) = 1$.

• More carefully, it can be shown that $TIME(f(n)) \subseteq CIRCUIT\text{-}\text{SIZE}(f(n)\log(f(n)))$ for proper $f(n)$.
P/poly

• **Notation:** P/poly = class of languages L that can be accepted by (in general, nonuniform) circuits of polynomial size, i.e., there is a constant c and a circuit family C=(C₁,C₂,…) with size(Cₙ) = O(nᶜ) that accepts L.

• Reason for notation: P/poly = languages that can be accepted by a polynomial time Turing Machine, which for inputs x of length n can access also an *advice string* aₙ of polynomial length (the advice string depends only on the length n, not on the input x).

  - advice = description of the circuit Cₙ.
BPP ⊆ P/poly

• All (binary) languages in BPP have polynomial size circuits.

• Proof:

Suppose \( L \in \text{BPP} \). Recall that we can make the error probability exponentially small, \( \leq 2^{-\text{poly}(n)} \), eg. \( \leq 2^{-(n+1)} \).

In terms of the certificate version of BPP, there is a polynomial \( p() \) and a two-input polynomial-time algorithm \( V(.,.) \) such that for every string \( x \)

\[
\begin{align*}
\square \quad & x \in L \implies | \{ y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ rejects} \} | \leq 2^{-(|x|+1)} \cdot 2^{p(|x|)} \\
\square \quad & x \notin L \implies | \{ y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ accepts} \} | \leq 2^{-(|x|+1)} \cdot 2^{p(|x|)}
\end{align*}
\]

We will show that for every \( n \), there exists a polynomial size circuit \( C_n \) that accepts a string \( x \) of length \( n \) iff \( x \in L \). The proof is not efficiently constructive.
BPP ⊆ P/poly proof

• **Claim:** There is a certificate $y$ of length $p(n)$ for which $V$ gives the correct answer for all strings $x$ of length $n$:
  \[ x \in L \iff V(x, y) \text{ accepts} \]

  Proof: There are $2^n$ strings $x$ of length $n$. For each one of them there are at most $2^{p(n)-(n+1)}$ bad certificates $y$ (i.e. certificates for which $V$ gives the wrong answer). So altogether there are at most $2^{p(n)-1} = 2^{p(n)}/2$ certificates that are bad for some $x \Rightarrow$ at least $\frac{1}{2}$ the certificates are good for all $x$

  • If we knew such a good certificate $y^*$, we could decide whether $x \in L$ by running $V$ on input $x, y^*$.

  • Map $V$ to a circuit and set the input bits corresponding to the certificate according to $y^* \rightarrow \text{ circuit } C_n$
Exponential Circuit Size

- Recall that every Boolean function has an exponential size circuit (in fact, formula) ⇒ every binary language has at most exponential circuit size complexity
  (in contrast to the fact that there are languages that have much higher time complexity)
- There are Boolean functions that require exponential size circuits, in fact almost all of them do.

Proof: There are $2^{2^n}$ Boolean functions with $n$ inputs and 1 output. A circuit with $s$ gates and wires can be specified by specifying for each gate its type (2 bits), and for each wire the endpoints (2log($n+s$) bits) ⇒ $s(2 + 2\log(n+s))$ bits total ⇒ there are $\leq s2^{s(2 + 2\log(n+s))}$ circuits of size at most $s$. This is less than $2^{2^n}$ if $s$ is less than $2^n/2n$. 
Conjecture: NP-complete problems cannot be decided by polynomial size circuits, probably require exponential size.

Although we know that most Boolean functions/languages require exponential size circuits, no such bound for any concrete, natural problem; in fact, nothing better than linear lower bounds.

Known: There is a language with exponential space complexity that requires exponential size circuits (Proof by diagonalization – HW exercise).
Monotone Problems

- **Monotone Boolean function** $f$: If $x \geq x'$ (bitwise comparison) then $f(x) \geq f(x')$

- **Monotone problem**: corresponds to family of monotone Boolean functions.

- **Examples:**
  - **Graph Reachability**, where graph is represented by its adjacency matrix: one input bit for each pair $(u,v)$ of nodes, bit =1 if there is edge, 0 otherwise.
  - **Hamiltonicity**, same representation.
  - **n/2-Clique**: Given graph, does it have a clique with $n/2$ nodes?
  - **Perfect matching in bipartite or in general graphs.**

Represent bipartite graphs with the two parts $L=(l_1,\ldots,l_n)$, $R=(r_1,\ldots,r_n)$ by $n^2$ input bits, one bit for each pair $(l_i,r_j)$.
Monotone Circuits

- No NOT gates: compute monotone functions
- Does this reduce the power?

- [Razborov] The n/2-Clique problem requires exponential size monotone circuits.
  (Proof: see the book, Chapter 14.4.)

  In fact,
- The bipartite perfect matching problem also requires exponential size monotone circuits.
- The bipartite perfect matching problem $\in P$ $\Rightarrow$ has polynomial size circuits
  $\Rightarrow$ exponential gap between the power of general and monotone circuits