Lecture 31: Circuit Complexity

Real computers are built from gates.

Circuit complexity is a low-level model of computation.

Circuits are directed acyclic graphs. Inputs are placed at the leaves. Signals proceed up toward the root, \( r \).

Straight-line code: gates are not reused.

Let \( S \subseteq \{0, 1\}^* \) be a decision problem.

Let, \( C_1, C_2, C_3, \ldots \) be a circuit family.

\( C_n \) has \( n \) input bits and one output bit \( r \).

**Definition 31.1** \( \{C_i\}_{i \in \mathbb{N}} \) computes \( S \) iff for all \( n \) and for all \( w \in \{0, 1\}^n \),

\[
(w \in S) \iff (C_{|w|}(w) = 1)
\]
or
and
or
and
or

\[ t(n) \]

not

\[ b_1 \]

\[ b_2 \]

\[ r \]

\[ n \]

\[ n \]

\[ b \]

\[ 1 \]

\[ 2 \]

Depth = parallel time = \( t(n) \)

Number of gates = computational work = sequential time

Width = max number of gates at any level = amount of hardware in corresponding parallel machine

“not” gates are pushed down to bottom
Circuit Complexity Classes

\[ S \subseteq \{0, 1\}^* \text{ is in } \text{NC}[t(n)], \ AC[t(n)], \ \text{ThC}[t(n)], \text{ iff exists uniform circuit family, } C_1, C_2, \ldots, \text{ s.t.} \]

1. For all \( w \in \{0, 1\}^* \), \( w \in S \iff C_{\lfloor w\rfloor}(w) = 1 \)
2. \( \text{depth}(C_n) = O(t(n)); \ |C_n| \leq n^{O(1)} \)
3. The gates of \( C_n \) consist of,

**NC**
- bounded fan–in
- and, or gates

**AC**
- unbounded fan–in
- and, or gates

**ThC**
- unbounded fan–in
- threshold gates
Notation: for $i = 0, 1, \ldots$, $\text{NC}^i = \text{NC}[(\log n)^i]$;

\[ \text{AC}^i = \text{AC}[(\log n)^i]; \quad \text{ThC}^i = \text{ThC}[(\log n)^i] \]

We will see that the following inclusions hold:

\[
\begin{align*}
\text{AC}^0 & \subseteq \text{ThC}^0 & \subseteq & \text{NC}^1 & \subseteq & L & \subseteq & \text{NL} & \subseteq & \text{AC}^1 \\
\text{AC}^1 & \subseteq \text{ThC}^1 & \subseteq & \text{NC}^2 & & & & & & \subseteq \text{AC}^2 \\
\text{AC}^2 & \subseteq \text{ThC}^2 & \subseteq & \text{NC}^3 & & & & & & \subseteq \text{AC}^3 \\
\vdots & \subseteq & \vdots & \subseteq & \vdots & \subseteq & \vdots & \subseteq & \vdots & \subseteq & \vdots
\end{align*}
\]

Thus:

\[
\text{NC} = \bigcup_{i=0}^{\infty} \text{NC}^i = \bigcup_{i=0}^{\infty} \text{AC}^i = \bigcup_{i=0}^{\infty} \text{ThC}^i
\]
Uniform means that the map, $f : 1^n \mapsto C_n$ is very easy. $f \in F(L); \ f \in F(FO)$

Each $C_n$ is an instance of the same program.
Proposition 31.2  Every regular language is in $\text{NC}^1$.

Proof: DFA $D = \langle \Sigma, Q, \delta, s, F \rangle$  

\[ f_i(q) \overset{\text{def}}{=} \delta(q, w_i) \quad f_{ij}(q) \overset{\text{def}}{=} \delta^*(q, w_i w_{i+1} \cdots w_j) \]

\[
\begin{align*}
\text{f}(s) \text{ in } F \quad &\overset{1n}{\iff} \quad f_{1n}(s) \in F \\
\end{align*}
\]

\[
\begin{align*}
&\text{f}_{12} \quad \text{f}_1 \quad \text{f}_2 \quad \text{f}_3 \\
&\delta \quad \delta \quad \delta \\
&w_1 \quad w_2 \quad w_3 \\

&\text{f}_{n-2} \quad \text{f}_{n-1} \quad \text{f}_n \\
&\delta \quad \delta \quad \delta \\
&w_{n-2} \quad w_{n-1} \quad w_n
\end{align*}
\]

\[ w \in \mathcal{L}(D) \iff f_{1n}(s) \in F \]
Theorem 31.3  \( \text{FO} = \text{AC}^0 \)

**Example:**  \[ \varphi \equiv \exists x \forall y \exists z (M(x, y, z)) \]
Proposition 31.4  For $i = 0, 1, \ldots,$

\[ NC^i \subseteq AC^i \subseteq \text{ThC}^i \subseteq NC^{i+1} \]

**Proof:** All inclusions except $\text{ThC}^i \subseteq NC^{i+1}$ are clear.

\[
\text{MAJ} \overset{\text{def}}{=} \{ w \in \{0, 1\}^\ast \mid w \text{ has more than } |w|/2 \text{ "1"s} \}
\]

\[
\text{MAJ} \in \text{ThC}^0
\]

Lemma 31.5  \text{MAJ} \in NC^1

(and the same for any other threshold gate).
Try to build an NC¹ circuit for majority by adding the $n$ input bits via a full binary tree of height $\log n$.

**Problem:** the sums being added have more and more bits; still want to add them in constant depth.
Solution: **Ambiguous Notation**

Binary representation; but with digits: 0, 1, 2, 3

\[
3213 = 3 \cdot 2^3 + 2 \cdot 2^2 + 1 \cdot 2^1 + 3 \cdot 2^0 = 37 \\
3221 = 3 \cdot 2^3 + 2 \cdot 2^2 + 2 \cdot 2^1 + 1 \cdot 2^0 = 37
\]

**Lemma 31.6** *Addition of pairs of numbers in Ambiguous Notation is computable by \( \text{NC}^0 \) circuits.*

**Example 31.7** *Example:*

```
  carries:  3  2  2  3
  3  2  1  3
+  3  2  1  3
  =  3  2  2  1  0
```

\[ \square \]

The carry from column \( i \) is determined by columns \( i \) and \( i + 1 \): use the largest carry we are sure to get.
Translating from ambiguous to binary, is just addition, thus first-order, thus $AC^0$, and thus $NC^1$. 

\[ > \frac{n}{2} \]

\begin{center}
\begin{tikzpicture}
\node[draw] (n) {back to unambiguous};
\node[draw] (m) at (n.south) [below] {
\begin{tikzpicture}
\node[draw] (p) at (0,0) {+};
\node[draw] (q) at (0,-1) [below] {+};
\node[draw] (r) at (0,-2) [below] {+};
\node[draw] (s) at (0,-3) [below] {+};
\node[draw] (t) at (0,-4) [below] {+};
\end{tikzpicture}};
\end{tikzpicture}
\end{center}

\[ \log n \]

\[ \log n > \frac{n}{2} \]

\[ \log n \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12} \]