Lecture 19: 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 \( r \).

**Def:** \( \{C_i\}_{i \in \mathbb{N}} \) 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

"not" gates are pushed down to bottom

Depth = parallel time

Number of gates = computational work = sequential time

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

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

1. For all $w \in \{0, 1\}^*$, $w \in S \iff C_{|w|}(w) = 1$
2. 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 $i = 0, 1, \ldots$, $\text{NC}^i = \text{NC}[(\log n)^i]$;

\[
\begin{align*}
\text{AC}^i &= \text{AC}[(\log n)^i]; \\
\text{ThC}^i &= \text{ThC}(\log n)^i
\end{align*}
\]

We will see that the following inclusions hold:

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

Thus:

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

Each \( C_i \) is an instance of the same program.
Prop: Every regular language is in NC¹.

Proof: DFA $D = (\Sigma, Q, \delta, s, F)$. Build circuits: $C_1, C_2, \ldots$,

$$f_i(q) = \delta(q, w_i); \quad w \in \mathcal{L}(D) \iff f_{1n}(s) \in F$$

□
**Thm:** \( \text{FO} = \text{AC}^0 \)

**Example:**

\[ \varphi \equiv \exists x \forall y \exists z (M(x,y,z)) \]
**Prop:** For $i = 0, 1, \ldots,$
\[
\text{NC}^i \subseteq \text{AC}^i \subseteq \text{ThC}^i \subseteq \text{NC}^{i+1}
\]

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

\[
\text{MAJ} = \left\{ w \in \{0, 1\}^* \mid w \text{ has more than } |w|/2 \text{ “1”s} \right\} \in \text{ThC}^0
\]

**Lemma:** $\text{MAJ} \in \text{NC}^1$

(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

\[
\begin{align*}
3213 &= 3 \cdot 2^3 + 2 \cdot 2^2 + 1 \cdot 2^1 + 3 \cdot 2^0 &= 37 \\
3221 &= 3 \cdot 2^3 + 2 \cdot 2^2 + 2 \cdot 2^1 + 1 \cdot 2^0 &= 37
\end{align*}
\]

**Lemma:** Ambiguous Notation Addition is in NC⁰

**Example:**

<table>
<thead>
<tr>
<th>carries: 3</th>
<th>2</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 2 1 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ 3 2 1 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $NC^1$. 