Lecture 33: Circuit Complexity Concluded, plus Barrington’s Theorem

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

\[
(w \in S) \iff (C_{|w|}(w) = 1)
\]
“not” gates are pushed down to bottom

\[ \text{Depth} = \text{parallel time} = t(n) \]

Number of gates = computational work = sequential time

\[ \text{Width} = \text{max number of gates at any level} = \text{amount of hardware in corresponding parallel machine} \]
Circuit Complexity Classes

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

1. For all \( w \in \{0, 1\}^\ast \), \( w \in S \iff C_{|w|}(w) = 1 \)

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

\[ \wedge \]
\[ \wedge \]
\[ k \]
**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:

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

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 33.2  Every regular language is in $\mathsf{NC}^1$.

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

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

\[ w \in \mathcal{L}(D) \iff f_{1n}(s) \in F \]
Theorem 33.3 \[ \text{FO} = \text{AC}^0 \]

Example: \[ \varphi \equiv \exists x \forall y \exists z (M(x, y, z)) \]
Proposition 33.4  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.

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

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

Lemma 33.5  $\text{MAJ} \in \text{NC}^1$

(and the same for any other threshold gate).
Try to build an $\text{NC}^1$ 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 33.6** *Addition of pairs of numbers in Ambiguous Notation is computable by NC⁰ circuits.*

**Example 33.7** *Example:*

<table>
<thead>
<tr>
<th>carries:</th>
<th>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>
<td></td>
</tr>
<tr>
<td>+ 3 2 1 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 2 1 0</td>
<td></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 $\text{AC}^0$, and thus $\text{NC}^1$.
Lecture 33: $\text{NC}^1$ and Barrington’s Theorem

CS501: Branching Programs  Lecture 33

\begin{center}
\begin{tikzpicture}
  \node[shape=circle,draw=black] (s) at (0,0) {s};
  \node[shape=circle,draw=black] (x1) at (1,1) {$x_1$};
  \node[shape=circle,draw=black] (x5) at (1,2) {$x_5$};
  \node[shape=circle,draw=black] (x2) at (2,2) {$x_2$};
  \node[shape=circle,draw=black] (t) at (3,3) {t};

  \draw[->] (s) -- (x1);
  \draw[->] (x1) -- (x5);
  \draw[->] (x5) -- (x2);
  \draw[->] (x2) -- (t);

  \node[shape=circle,draw=black] (s) at (0,0) {s};
  \node[shape=circle,draw=black] (x1) at (1,1) {$x_1$};
  \node[shape=circle,draw=black] (x5) at (1,2) {$x_5$};
  \node[shape=circle,draw=black] (x2) at (2,2) {$x_2$};
  \node[shape=circle,draw=black] (t) at (3,3) {t};

  \draw[->] (s) -- (x1);
  \draw[->] (x1) -- (x5);
  \draw[->] (x5) -- (x2);
  \draw[->] (x2) -- (t);
\end{tikzpicture}
\end{center}
**Theorem 33.8**  The set of problems accepted by uniform (polynomial size) branching programs is \( \text{DSPACE} \log n \).

\[
\text{BranchingPrograms} = \text{L}
\]

**Proof:**

BranchingPrograms \( \subseteq \text{L} \): just keep track of where you are!

\( \text{L} \subseteq \text{BranchingPrograms} \):

Let \( M \) be a \( \text{DSPACE} \log n \) Turing machine.

The computation graph of \( M \) on some variable input \( x_1 \cdots x_n \) is a branching program! \( \square \)
**Proposition 33.9** The set of problems accepted by uniform, bounded-width branching programs is contained in $\text{NC}^1$.

**Proof:** This is similar to the proof that $\text{REACH} \in \text{sAC}^1$. However, instead of $n$ choices to guess the middle point, there are only a bounded number of choices. \qed
Bounded Width Branching Programs look very much like finite automata.

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

Natural Conjecture:

\[
\text{MAJ} \notin \text{Bounded Width BPs}
\]
$S_5$ is the permutation group on 5 objects.

$$\alpha = (12345), \quad \beta = (13542) \in S_5$$

$$[\alpha, \beta] = \alpha \beta \alpha^{-1} \beta^{-1}$$

$$= (12345)(13542)(54321)(24531)$$

$$= (13254)$$
**Definition 33.10** A width 5 Branching Program, $B$, 5-cycle recognizes $S$ iff for some 5-cycle $\sigma$,

- For $x \in S$, $B(x) = \sigma$
- For $x \notin S$, $B(x) = e$

**Lemma 33.11** Let $S_i = \{x \in \{0, 1\}^n \mid x_i = 1\}$.

$S_i$ can be 5-cycle recognized.

**Lemma 33.12** If $S$ is 5-cycle recognized, then so is $\overline{S}$
Lemma 33.13 If $S$ is 5-cycle recognized using 5-cycle $\sigma$, then $S$ can be 5-cycle recognized using 5-cycle $\tau$.

Proof: Every two 5-cycles are conjugates, i.e.,

$$(\exists \theta \in S_5)(\tau = \theta^{-1}\sigma\theta)$$

Lemma 33.14 If $S$ and $T$ can be 5-cycle recognized by branching programs $B$ and $C$, then $S \cap T$ can be 5-cycle recognized by a branching program of size $2(|B| + |C|)$

Proof:

$B \quad C \quad B^{-1} \quad C^{-1}$

\[\square\]
Theorem 33.15 (Barrington’s Theorem)

\begin{align*}
\text{Bounded Width Branching Programs} & = \text{NC}^1
\end{align*}

Proof:

Given an NC$^1$ circuit, simulate it using the above lemmas.

We multiply the size of the branching programs by 4 as we go up one level.

Total size is $4^{O(\log n)} = n^{O(1)}$