Lecture 12: 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 $w \in \{0, 1\}^n$,

$$w \in S \iff C_{|w|}(w) = 1.$$
"not" gates are pushed down to bottom

\[
\text{Depth} = \text{parallel time}
\]

\[
\text{Number of gates} = \text{computational work} = \text{sequential time}
\]

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

\( S \subseteq \{0, 1\}^* \) is in \( \text{NC}[t(n)], \text{AC}[t(n)], \text{ThC}(n) \), 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. \( \text{depth}(C_n) = O(t(n)) \); \( |C_n| \leq n^{O(1)} \)
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 $i = 0, 1, \ldots$, \[ NC^i = \text{NC}[(\log n)^i]; \]
\[ AC^i = \text{AC}[(\log n)^i]; \quad \text{ThC}^i = \text{ThC}(\log n)^i \]

We will see that the following inclusions hold:
\[
\begin{align*}
\text{AC}^0 & \subseteq \text{ThC}^0 \subseteq \text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{AC}^1 \\
\text{AC}^1 & \subseteq \text{ThC}^1 \subseteq \text{NC}^2 \subseteq \text{AC}^2 \\
\text{AC}^2 & \subseteq \text{ThC}^2 \subseteq \text{NC}^3 \subseteq \text{AC}^3 \\
\vdots & \subseteq \vdots \subseteq \vdots \subseteq \vdots
\end{align*}
\]

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 $C_i$ is an instance of the same program.
**Prop:** Every regular language is in NC\(^1\).

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

\[
\begin{align*}
 f(s) \text{ in } F \quad & \overline{1_n} \\
 f_1 & \overline{n} \\
 f_1 & \overline{n/2} \\
 f_{n/2+1} & \overline{n} \\
 f_1 & \overline{n/2} \\
 f_1 & \overline{n} \\
\end{align*}
\]

\[
\begin{align*}
 f_1 & \overline{q_1} \\
 f_2 & \overline{q_2} \\
 f_3 & \overline{q_3} \\
 w_1 & \overline{w_1} \\
 w_2 & \overline{w_2} \\
 w_3 & \overline{w_3} \\
\end{align*}
\]

\[
\begin{align*}
 f_{n-2} & \overline{q_{n-2}} \\
 f_{n-1} & \overline{q_{n-1}} \\
 f_n & \overline{q_n} \\
 w_{n-2} & \overline{w_{n-2}} \\
 w_{n-1} & \overline{w_{n-1}} \\
 w_n & \overline{w_n} \\
\end{align*}
\]

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

\( \square \)
Thm: \[ \text{FO} = \text{AC}^0 \]

Example:

\[ \varphi \equiv \exists x \forall y \exists z \left( M(x, y, z) \right) \]
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.

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

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

(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

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

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

**Lemma:** Ambiguous Notation Addition is in $\text{NC}^0$

**Example:**

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

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 $NC^1$. 

\[
\begin{align*}
\text{back to unambiguous} \\
+ \\
+ \\
+ \\
+ \\
+ \\
+ \\
> n/2 \\
\log n
\end{align*}
\]
Arithmetic Hierarchy

FO(N)  r.e. complete

co-r.e.  FO(\forall(N))

r.e.  FO(\exists(N))

Recursive

Primitive Recursive

SO(LFP)  SO[2^{n^{O(1)}}]

EXPTIME

QSAT  PSPACE complete

FO[2^{n^{O(1)}}]  FO(PFP)

PSPACE

SO[2^{n^{O(1)}}]

PTIME Hierarchy

SO

NP  co-NP

NP \cap co-NP

CO-NP complete

SAT

SAT

NP complete

FO[n^{O(1)}]

P complete

P

FO(LFP)  SO(Horn)

Horn-SAT

\text{“truly feasible”}

NC

FO[(\log n)^{O(1)}]

AC^1

FO[\log n]

sAC^1

FO(CFL)

2SAT

NL complete

NL

FO(TC)  SO(Krom)

2COLOR

L complete

L

FO(DTC)

FO(REGULAR)

NC^1

FO(COUNT)

ThC^0

FO

LOGTIME Hierarchy

AC^0