

601 Lect 21: Recall From Last Time: Alternation

Examples:

1. $\text{MCVP} \in \text{ASPACE}[\log n]$
2. $\text{QSAT} \in \text{ATIME}[n]$

Thm: For $s(n) \geq \log n$,

$$\text{NSPACE}[s(n)] \subseteq \text{ATIME}[(s(n))^2] \subseteq \text{DSPACE}[(s(n))^2]$$

$$\text{ASPACE}[s(n)] = \text{DTIME}[2^{O(s(n))}]$$

Cor:

$$\begin{aligned} \text{ASPACE}[\log n] &= \mathbf{P} \\ \text{ATIME}[n^{O(1)}] &= \mathbf{PSPACE} \\ \text{ASPACE}[n^{O(1)}] &= \mathbf{EXPTIME} \end{aligned}$$

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.

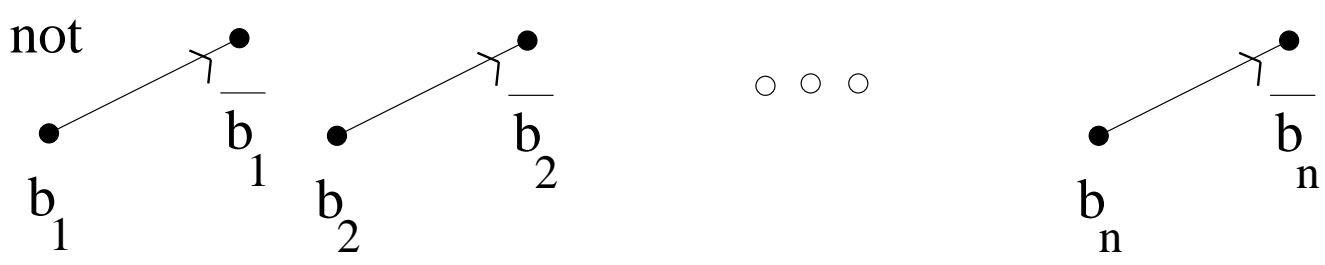
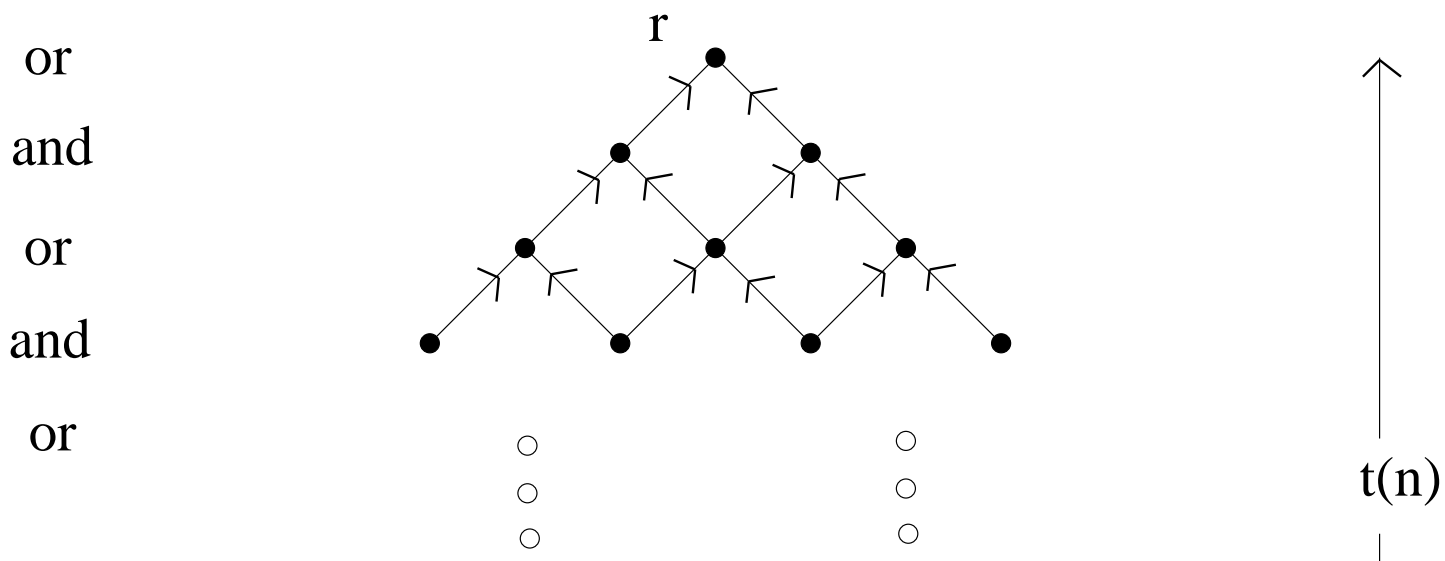
Let $S \subseteq \{0, 1\}^*$ be a decision problem.

Let, C_1, C_2, C_3, \dots be a circuit family.

C_n has n input bits and one output bit r .

Def: $\{C_i\}_{i \in \mathbf{N}}$ **computes** S iff for all n and for all $w \in \{0, 1\}^n$,

$$w \in S \quad \Leftrightarrow \quad C_{|w|}(w) = 1 .$$



“not” gates are pushed down to bottom

Depth = parallel time

Number of gates = computational work = sequential time

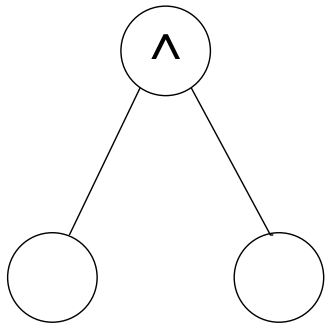
Circuit Complexity Classes

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

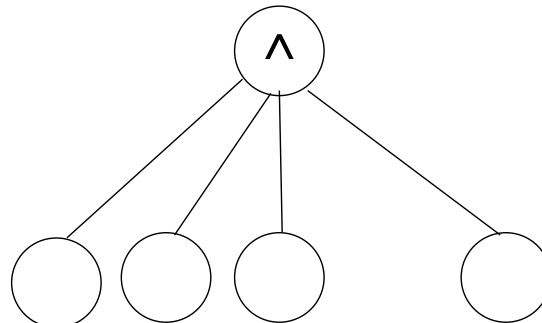
NC

bounded fan-in
and, or gates



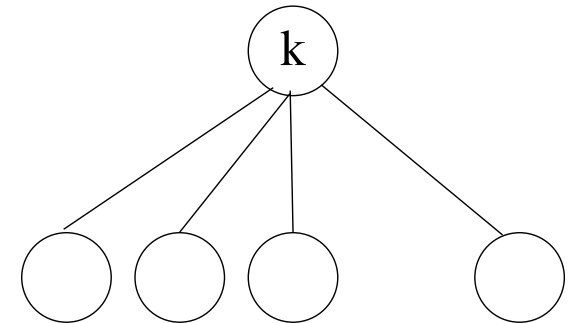
AC

unbounded fan-in
and, or gates



ThC

unbounded fan-in
threshold gates



Notation: for $i = 0, 1, \dots$, $\mathbf{NC}^i = \mathbf{NC}[(\log n)^i]$;

$$\mathbf{AC}^i = \mathbf{AC}[(\log n)^i];$$

$$\mathbf{ThC}^i = \mathbf{ThC}[(\log n)^i]$$

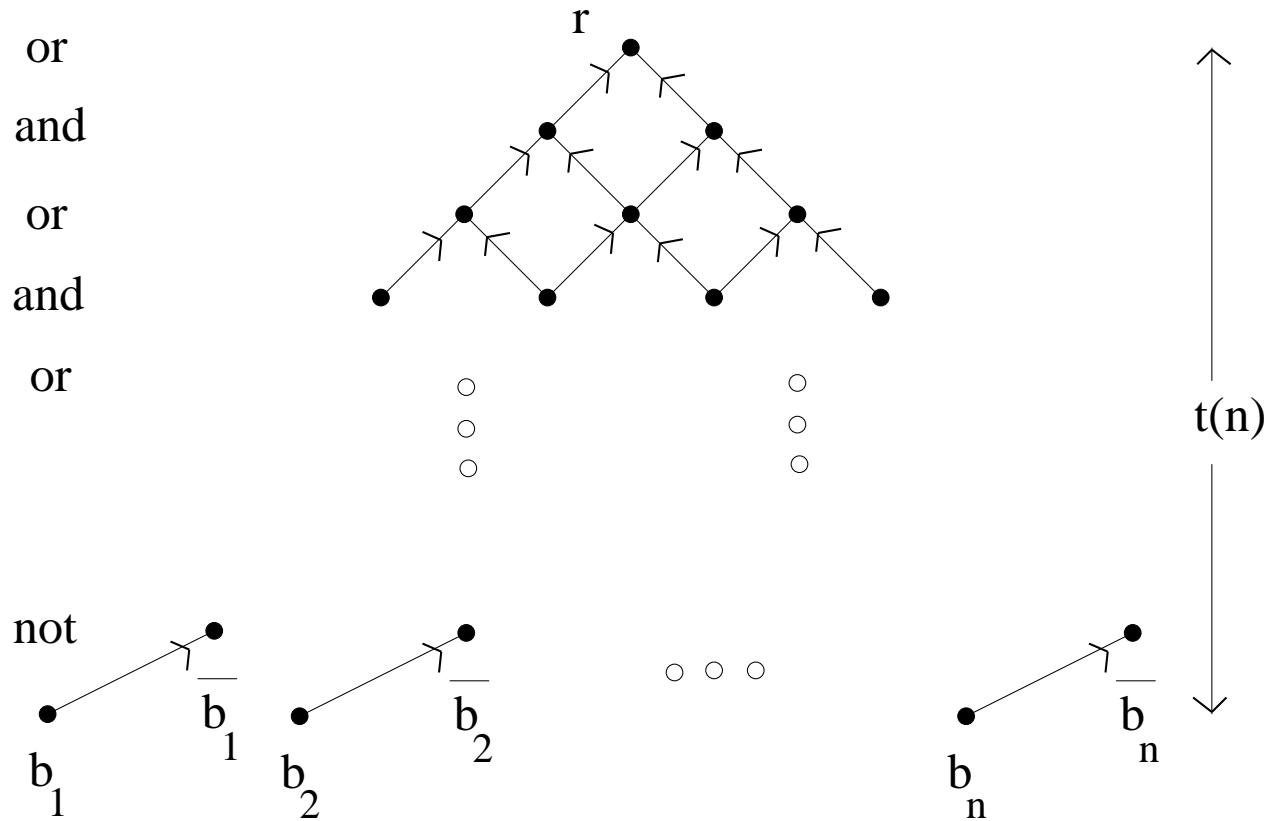
We will see that the following inclusions hold:

$$\begin{array}{ccccccccccc} \mathbf{AC}^0 & \subseteq & \mathbf{ThC}^0 & \subseteq & \mathbf{NC}^1 & \subseteq & \mathbf{L} & \subseteq & \mathbf{NL} & \subseteq & \mathbf{AC}^1 \\ \mathbf{AC}^1 & \subseteq & \mathbf{ThC}^1 & \subseteq & \mathbf{NC}^2 & & & & & \subseteq & \mathbf{AC}^2 \\ \mathbf{AC}^2 & \subseteq & \mathbf{ThC}^2 & \subseteq & \mathbf{NC}^3 & & & & & \subseteq & \mathbf{AC}^3 \\ \vdots & \subseteq & \vdots & \subseteq & \vdots & & & & & \subseteq & \vdots \end{array}$$

Thus:

$$\mathbf{NC} = \bigcup_{i=0}^{\infty} \mathbf{NC}^i = \bigcup_{i=0}^{\infty} \mathbf{AC}^i = \bigcup_{i=0}^{\infty} \mathbf{ThC}^i$$

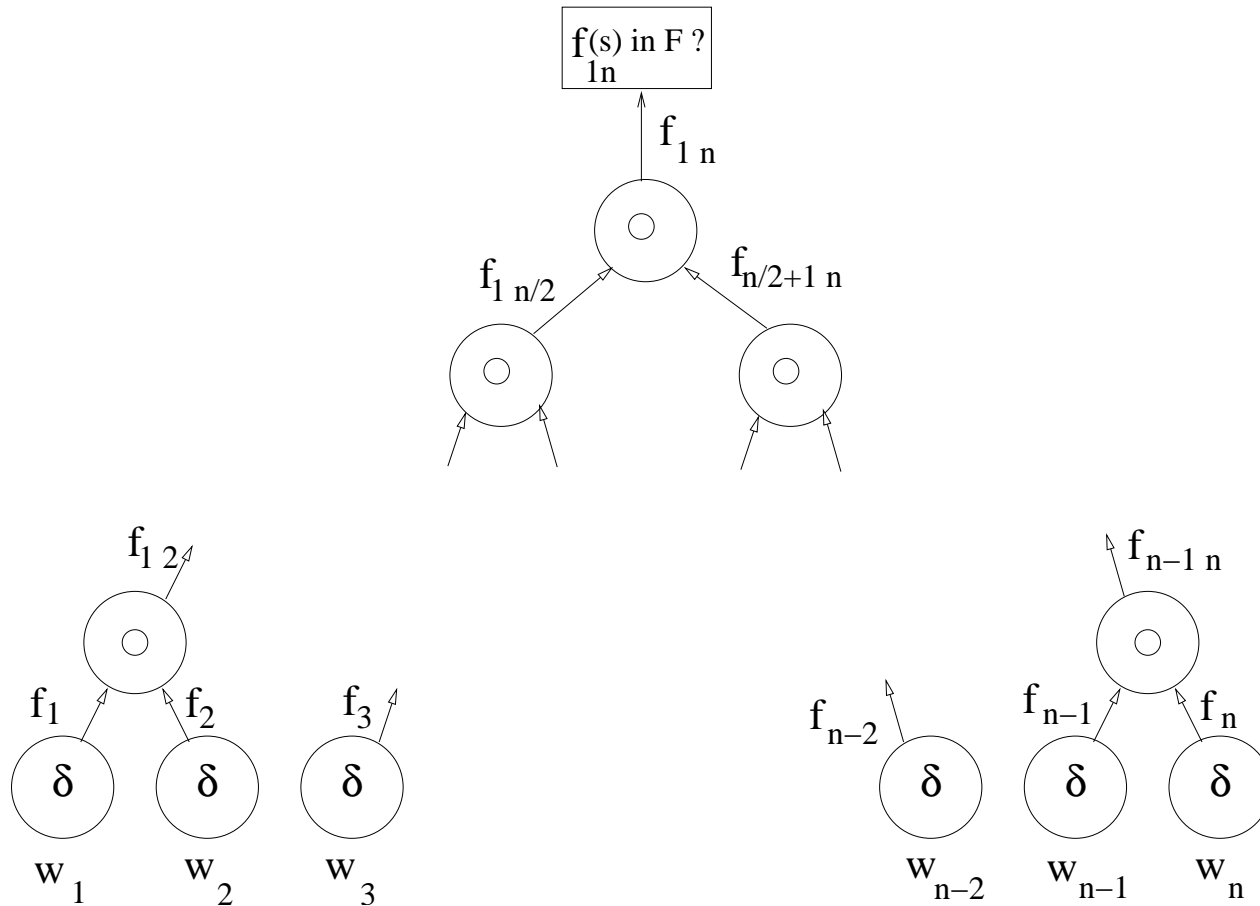
Uniform means that the map, $f : 1^n \mapsto C_n$ is **very easy**.
 $f \in F(\mathbf{L})$; $f \in F(\mathbf{FO})$



Each C_i is an instance of the same program.

Prop: Every regular language is in \mathbf{NC}^1 .

Proof: DFA $D = \langle \Sigma, Q, \delta, s, F \rangle$. Build circuits: $C_1, C_2, \dots,$

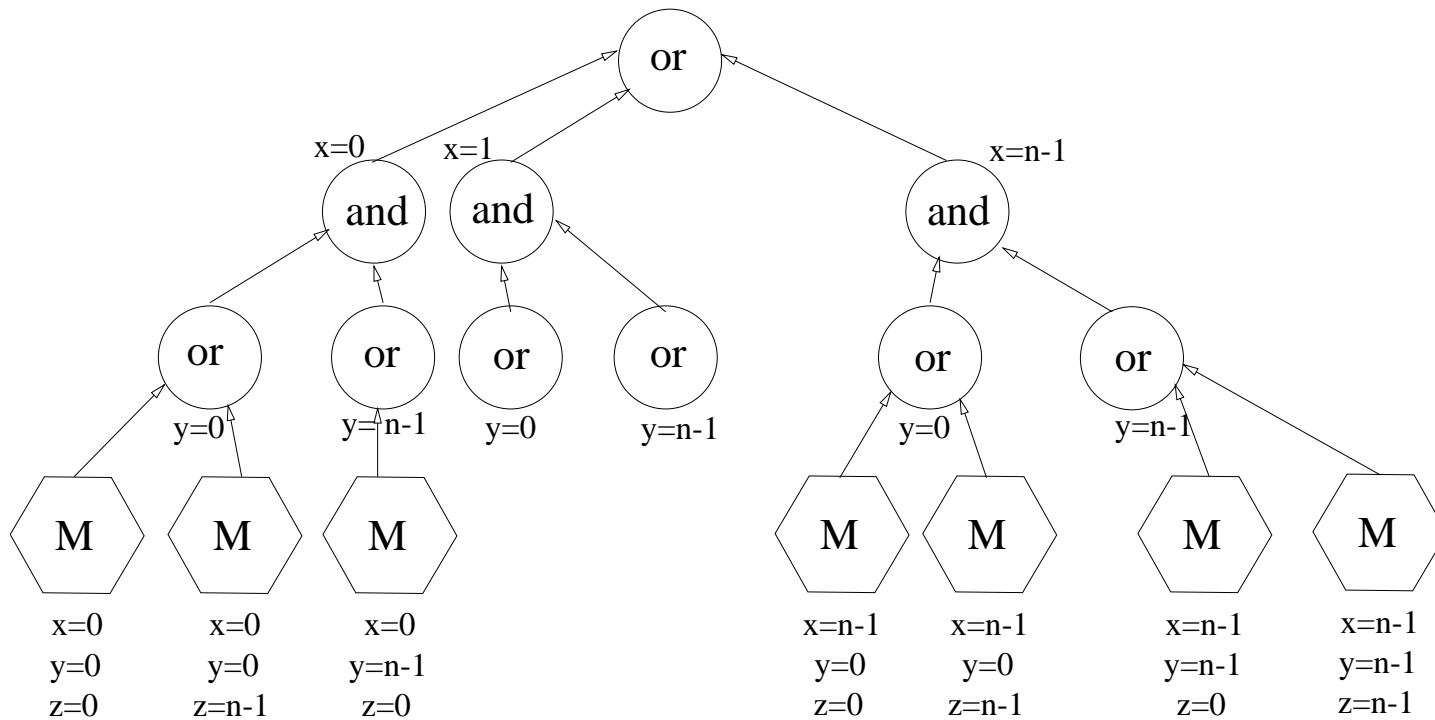


$$f_i(q) = \delta(q, w_i); \quad w \in \mathcal{L}(D) \quad \Leftrightarrow \quad f_{1n}(s) \in F$$

□

Thm: **FO** = **AC⁰**

Example: $\varphi \equiv \exists x \forall y \exists z (M(x, y, z))$



Prop: For $i = 0, 1, \dots$,

$$\mathbf{NC}^i \subseteq \mathbf{AC}^i \subseteq \mathbf{ThC}^i \subseteq \mathbf{NC}^{i+1}$$

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

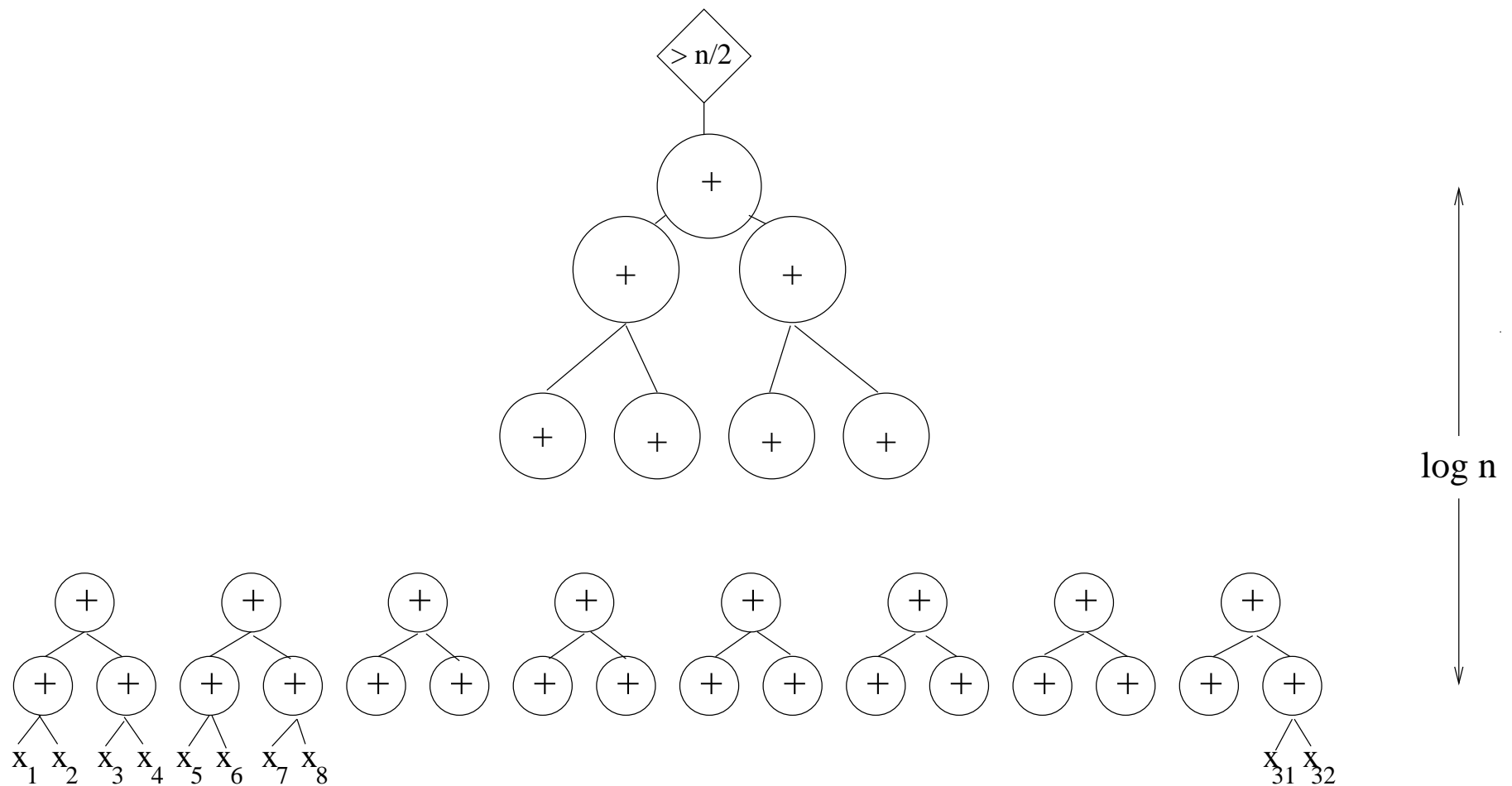
$$\text{MAJ} = \{w \in \{0, 1\}^* \mid w \text{ has more than } |w|/2 \text{ "1"s}\} \in \mathbf{ThC}^0$$

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

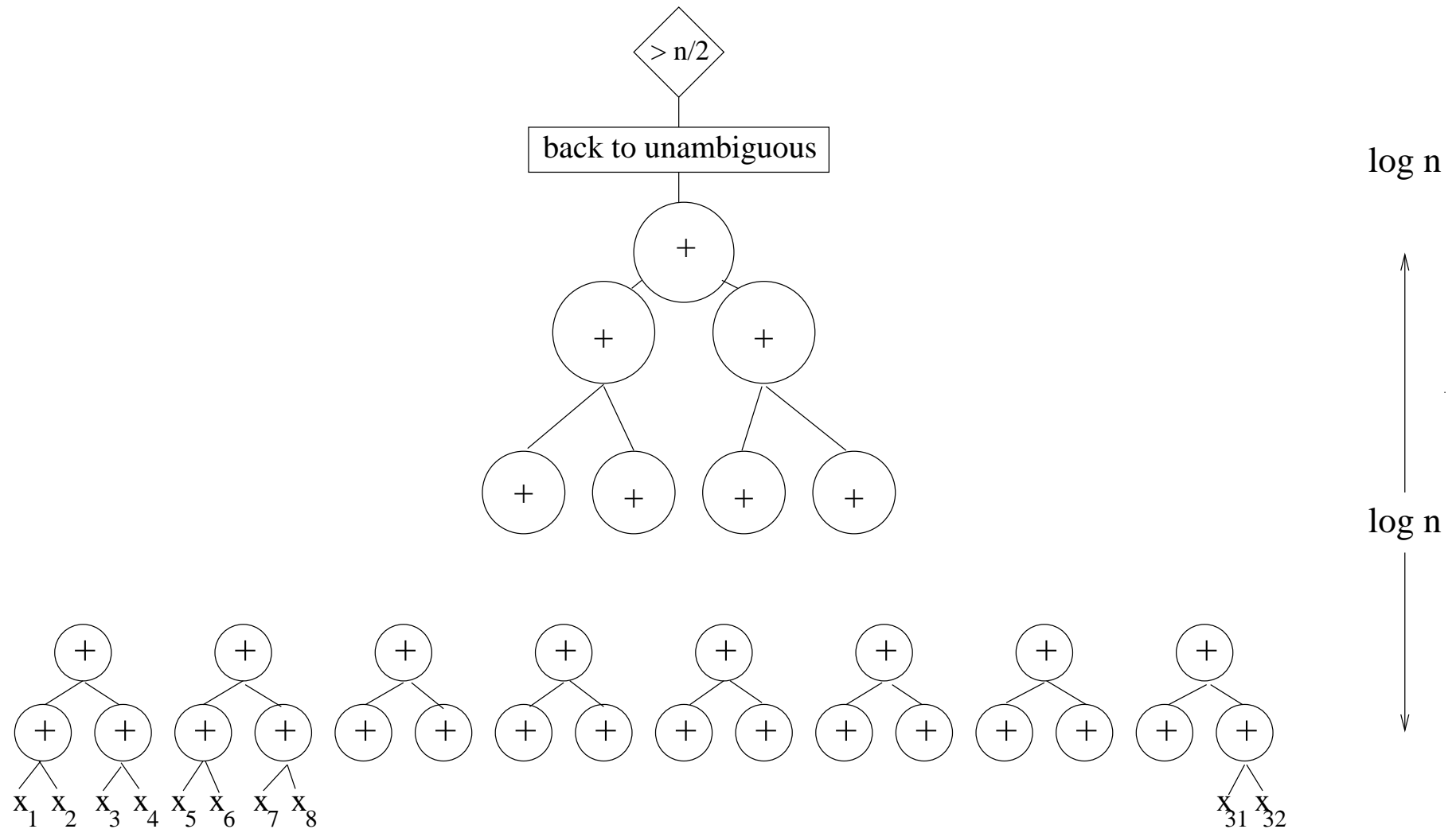
(and the same for any other threshold gate).

Try to build an 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.



Translating from ambiguous to binary, is just addition, thus first-order, thus \mathbf{AC}^0 , and thus \mathbf{NC}^1 .



Simultaneous Resource Classes

Def: $\text{ATIME-ALT}[t(n), a(n)]$ is the set of problems solved by alternating Turing machines in time $O(t(n))$, while making at most $a(n)$ alternations between existential and universal states, starting with existential.

Example: $\text{ATIME-ALT}[t(n), 0] = \text{NTIME}[t(n)]$

Similarly, define the simultaneous classes:

$\text{ASPACE-TIME}[s(n), t(n)], \quad \text{ASPACE-ALT}[s(n), a(n)]$

LH and PH

Define the LOGTIME hierarchy (**LH**) and the PTIME hierarchy (**PH**) as follows:

$$\mathbf{LH} = \mathbf{ATIME-ALT}[\log n, O(1)]$$

$$\mathbf{PH} = \mathbf{ATIME-ALT}[n^{O(1)}, O(1)]$$

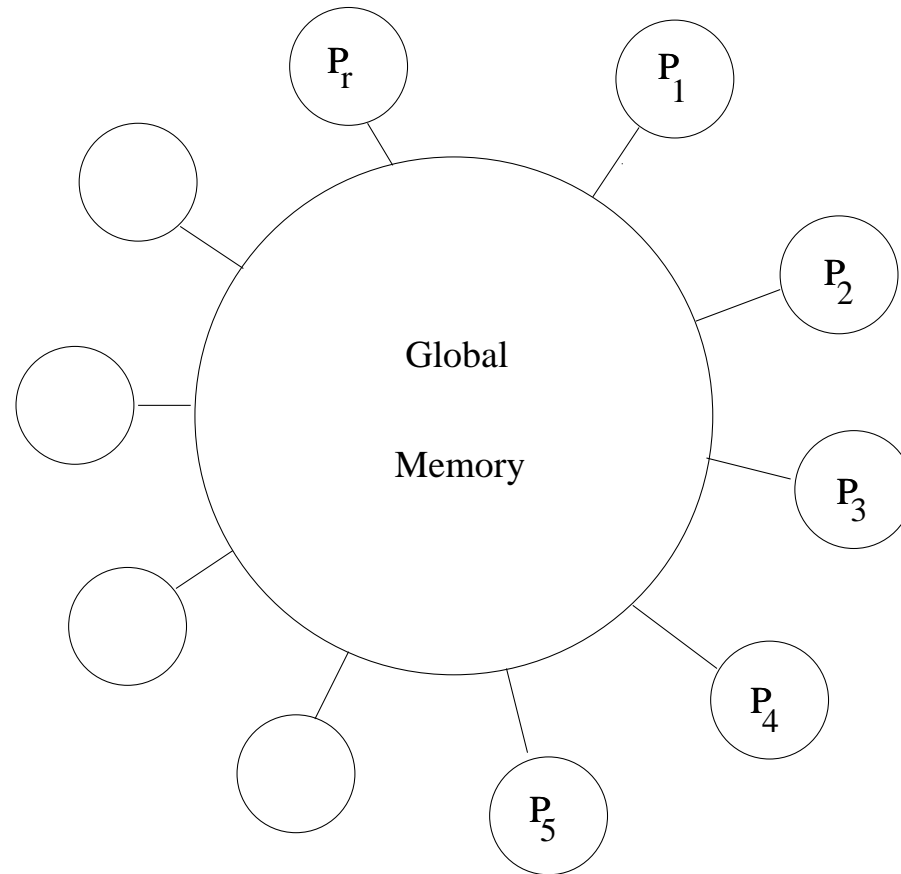
Thm: **PH** = **SO**

Proof: [idea] Follows from Fagin's Thm: **NP** = **SO**∃. □

Fact: **LH** = **FO**

$$\mathbf{CRAM}[t(n)] = \mathbf{CRCW-PRAM-TIME}[t(n)]\text{-HARD}[n^{O(1)}]$$

synchronous, concurrent, $n^{O(1)}$ processors and memory



priority write: lowest number processor wins conflict

common write: no conflicts allowed

Thm: For all $t(n)$, $\mathbf{CRAM}[t(n)] = \mathbf{AC}[t(n)]$.

Proof: (Sketch)

$\mathbf{CRAM}[1] \subseteq \mathbf{AC}[1] = \mathbf{AC}^0 = \mathbf{FO}$:

Only the global memory is tricky: all else is sequential.

$B(i, w, b)$ means that bit i of global memory word w is b .

$$B'(i, w, b) \equiv B(i, w, b) \wedge \forall p (\text{"}p \text{ didn't just write into word } w\text{"}) \\ \vee \exists p (\text{"}p \text{ just wrote into word } w\text{"} \wedge R(p, i, b))$$

CRAM[1] \supseteq FO:

$\forall x(\alpha(x))$

1. $w := 1$
2. Each processor P_i in parallel, **do** {
3. **if** $(\neg\alpha(i))$ **then Write**(0,w)
4. }

$\exists x(\alpha(x))$

1. $w := 0$
2. Each processor P_i in parallel, **do** {
3. **if** $(\alpha(i))$ **then Write**(1,w)
4. }



Def: sAC^1 (semi-unbounded sAC^1) is the subset of AC^1 where the and-gates are binary and only the or-gates are unbounded.

Fact: [Ruzzo] $sAC^1 = \log(CFL)$.

Thm: $NC^1 \subseteq L \subseteq NL \subseteq sAC^1 \subseteq AC^1$

Proof: Some of this may be in HW 11 – I'll fill in the rest later.

Alternation as Parallelism

Fact: [Ruzzo and Tompa] For $t(n) \geq \log n$,

$$\mathbf{ASPACE-TIME}[\log n, t(n)] = \mathbf{NC}[t(n)]$$

$$\mathbf{ASPACE-ALT}[\log n, t(n)] = \mathbf{AC}[t(n)]$$

Cor: $\mathbf{ATIME}[\log n] = \mathbf{NC}^1$

$$\mathbf{ASPACE-TIME}[\log n, t(n)] = \mathbf{NC}[t(n)]$$

$$\mathbf{ASPACE-ALT}[\log n, t(n)] = \mathbf{AC}[t(n)]$$

Proof: (sketches)

The computation graph of an $\mathbf{ASPACE-TIME}[\log n, t(n)]$ machine on an input of size n is an $\mathbf{NC}[t(n)]$ circuit.

The computation graph of an $\mathbf{ASPACE-ALT}[\log n, t(n)]$ machine is a series of $t(n)$ computation graphs of \mathbf{NL} or $\mathbf{co-NL}$ machines.

We can modify the $\mathbf{ASPACE-ALT}[\log n, t(n)]$ machine to make a series of alternating guesses: $\mathbf{ID}_1, \mathbf{ID}_2, \dots, \mathbf{ID}_{t(n)}$ of the endpoints of each of these \mathbf{NL} or $\mathbf{co-NL}$ computations, and **check only once** that \mathbf{ID}_{i+1} follows correctly from \mathbf{ID}_i .

Since $\mathbf{NL} \subseteq \mathbf{AC}^1$, the whole thing is an $\mathbf{AC}[t(n) + \log n] = \mathbf{AC}[t(n)]$ circuit. □

