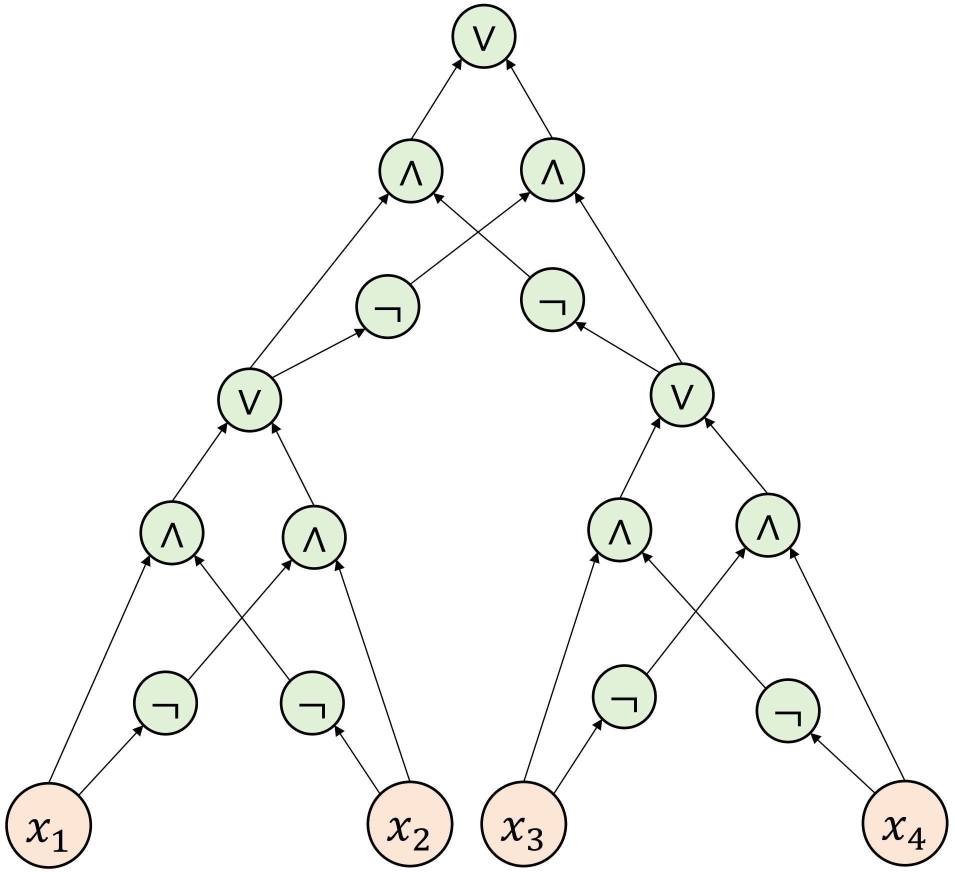
CMSC 28100

Introduction to Complexity Theory

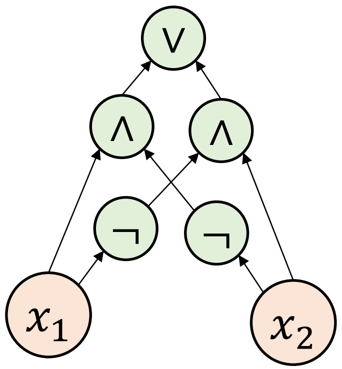
Spring 2025
Instructor: William Hoza

Boolean circuits

- A Boolean circuit is a network of logic gates (AND, OR, NOT) applied to Boolean variables $(x_1, \dots, x_n)$
- Boolean formula: Special case in which graph is a tree



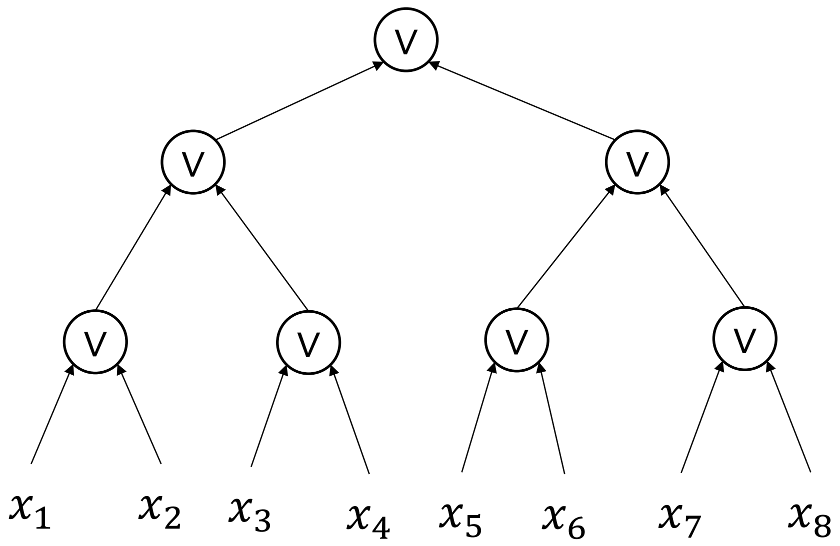
Circuit complexity



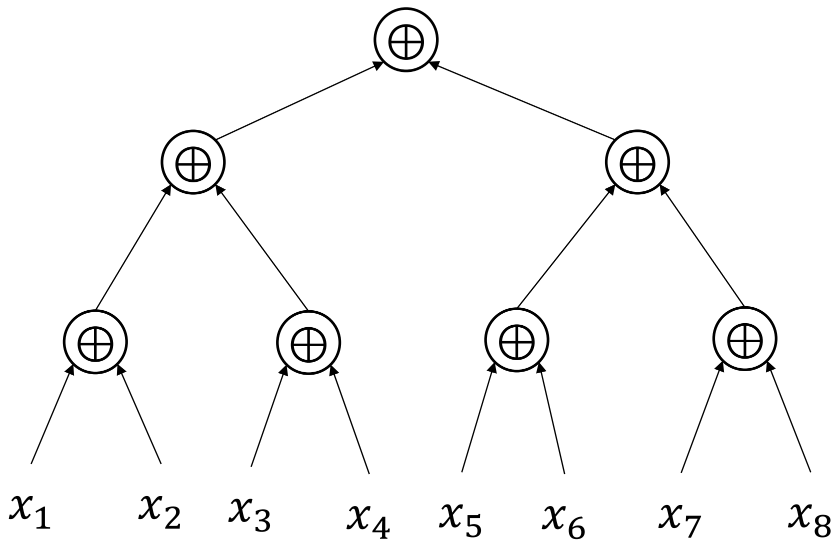
- **Definition:** The size of a circuit is the total number of AND/OR/NOT gates
- **Definition:** The circuit complexity of $f: \{0,1\}^n \to \{0,1\}^m$ is the size of the smallest circuit that computes f

Circuit complexity example 1

- Let $f(x) = x_1 \vee x_2 \vee \cdots \vee x_n$
- Circuit complexity: $\Theta(n)$



Circuit complexity example 2



- Let $f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$
- Circuit con
- Each "$\oplus$ g gates

What is the circuit complexity of f?

A: $\Theta(n^2)$

B: $O(1)$

C: $\Theta(n)$

D: $\Theta(2^n)$

Respond at PollEv.com/whoza or text "whoza" to 22333

The power of Boolean circuits

- Recall: Some languages cannot be decided by algorithms
- Are there functions that cannot be computed by circuits?

Theorem: For every $f: \{0,1\}^n \to \{0,1\}$, there exists a Boolean formula that computes f.

Theorem: For every $f: \{0,1\}^n \to \{0,1\}$, there exists a Boolean formula that computes f.

- **Proof (1 slide):** For each $z \in \{0,1\}^n$, construct T_z that is satisfied only by z
  - E.g., $T_{010} = \bar{x}_1 \wedge x_2 \wedge \bar{x}_3$

$$\text{Then } f(x) = \bigvee_{z \in f^{-1}(1)} T_z(x)$$

DNF formulas

- **Definition:** A literal is a variable or its negation (x_i or $\bar{x}_i$)
- **Definition:** A term is a conjunction of literals (AND of literals). Example:

$$T_{010} = \bar{x}_1 \wedge x_2 \wedge \bar{x}_3$$

- **Definition:** A disjunctive normal form (DNF) formula is a disjunction of terms (OR of ANDs of literals). Example:

$$f(x) = (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3)$$

Every function has a DNF formula

- Let $f: \{0,1\}^n \to \{0,1\}$ be any function

Theorem: There is a DNF formula that computes f, with at most 2^n terms and n literals per term

- **Proof:** For each $z \in \{0,1\}^n$, construct a term T_z that is satisfied only by z

$$\text{Then } f(x) = \bigvee_{z \in f^{-1}(1)} T_z(x)$$

CNF formulas

- **Definition:** A clause is a disjunction of literals (OR of literals). Example:

$$C = \bar{x}_1 \vee x_2 \vee \bar{x}_3$$

- **Definition:** A conjunctive normal form (CNF) formula is a conjunction of clauses (AND of ORs of literals). Example:

$$f(x) = (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

Every function has a CNF formula

- Let $f : \{0,1\}^n \to \{0,1\}$ be any function

Theorem: There is a CNF formula that computes f,

with at most 2^n clauses and n literals per clause

- **Proof:** For each $z \in \{0,1\}^n$, construct a clause C_z that is violated only by z
  - E.g., $T_{010} = x_1 \vee \bar{x}_2 \vee x_3$

$$\text{Then } f(x) = \bigwedge_{z \in f^{-1}(0)} C_z(x)$$

Multi-output functions

Corollary: For every $f: \{0,1\}^n \to \{0,1\}^m$, there exists a circuit of size $O(m \cdot n \cdot 2^n)$ that computes f

- **Proof:** Write $f(x) = \big(f_1(x), \dots, f_m(x)\big)$
- Each f_i can be computed by a circuit of size $O(n \cdot 2^n)$ (DNF/CNF)
- Combine those m circuits into one

Polynomial-size circuits

- Every function has a circuit
- But the circuit we constructed has exponential size
- Which functions have polynomial circuit complexity?
- Note: The circuit complexity of $f: \{0,1\}^n \to \{0,1\}$ is just a number
- Let's define the circuit complexity of a language $Y \subseteq \{0,1\}^*$

Circuit complexity of a binary language

- Let $Y \subseteq \{0,1\}^*$
- For each $n \in \mathbb{N}$, we define $Y_n : \{0,1\}^n \to \{0,1\}$ by the rule

$$Y_n(w) = \begin{cases} 1 & \text{if } w \in Y \\ 0 & \text{if } w \notin Y \end{cases}$$

- **Definition:** The circuit complexity of Y is the function $S : \mathbb{N} \to \mathbb{N}$ defined by

  $S(n) =$ the size of the smallest circuit that computes Y_n

- Note: Each circuit only handles a single input length! Different from TMs

The complexity class PSIZE

- Let $S: \mathbb{N} \to \mathbb{N}$ be a function
- **Definition:**

$$\mathrm{SIZE}(S) = \{Y \subseteq \{0,1\}^* : \text{the circuit complexity of } Y \text{ is } O(S)\}$$

- **Definition:**

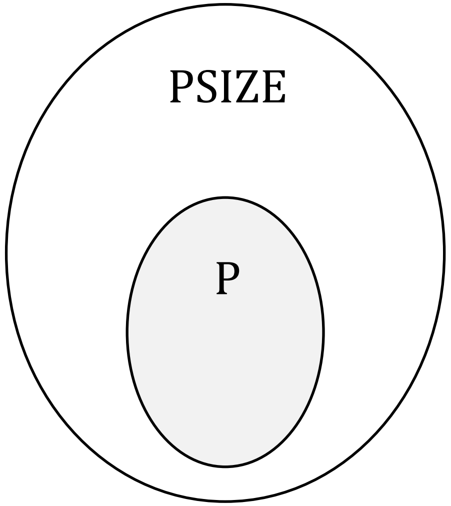
$$\mathrm{PSIZE} = \{Y \subseteq \{0,1\}^* : \text{the circuit complexity of } Y \text{ is } \mathrm{poly}(n)\} = \bigcup_{k=1}^{\infty} \mathrm{SIZE}(n^k)$$

Turing machines vs. circuits

- Let M be a Turing machine that decides a language Y
- Let $T(n)$ be M's time complexity; let $S(n)$ be M's space complexity

Theorem: $Y \in \mathrm{SIZE}\big(S(n) \cdot T(n)\big)$.

In particular, $\mathrm{P} \subseteq \mathrm{PSIZE}$.



- Proof (next 6 slides) is based on computation histories

To figure out c'_{206}, which symbols of C do we need to inspect?

A: All of them $(c_1, c_2, \dots, c_\ell)$

B: Only c_{206}

C: c_{205}, c_{206}, and c_{207}

D: c_{205}, c_{206}, c_{207}, and c_{208}

Respond at PollEv.com/whoza or text “whoza” to 22333

Locality of compu

- Let C be a configuration of t
- We can write $C = c_1 c_2 \dots c_\ell$ for some $c_1, \dots, c_\ell \in \Sigma \cup Q$
- Then $\text{NEXT}(C) = c'_1 c'_2 \dots c'_\ell$ for some $c'_1, \dots, c'_\ell \in \Sigma \cup Q$
- **Fact:** If $2 \leq i \leq \ell - 2$, then

$$c'_i = \begin{cases} \text{the third symbol of NEXT}(\sqcup\, c_{i-1} c_i c_{i+1} c_{i+2}) & \text{if } c_{i-1} \in Q \text{ or } c_i \in Q \text{ or } c_{i+1} \in Q \\ c_i & \text{otherwise} \end{cases}$$

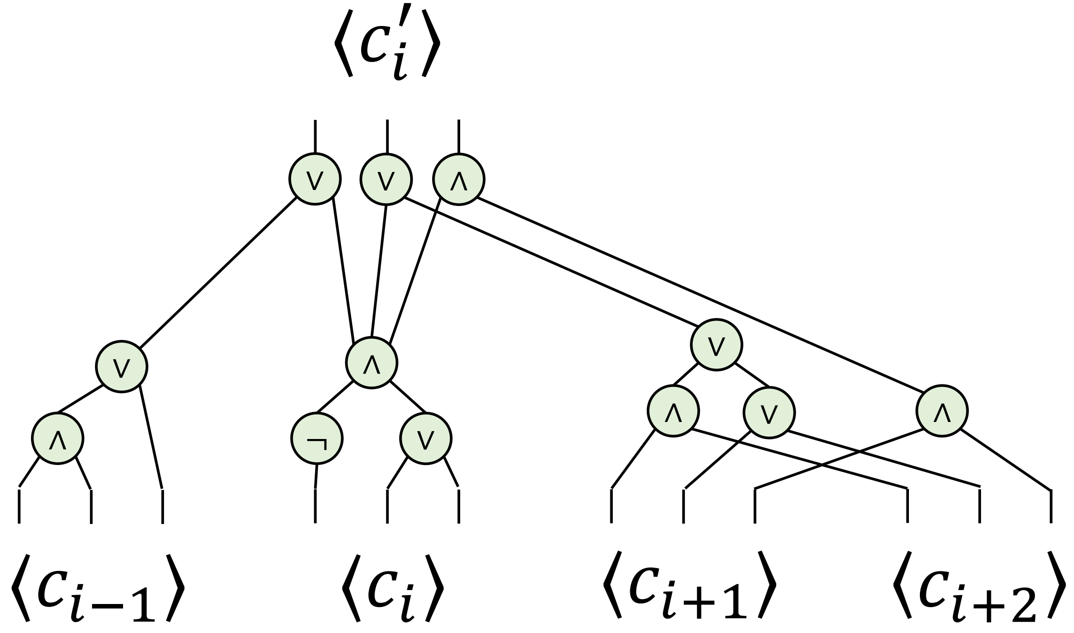
For simplicity, assume the head is not at beginning/end

Encoding configurations in binary

- Let C be a configuration of a TM M, say $C = u_1u_2 \ldots u_kqv_1v_2 \ldots v_m$
- Each symbol/state $b \in \Sigma \cup Q$ can be encoded in binary as $\langle b \rangle \in \{0,1\}^r$ for some $r = O(1)$
- We define $\langle C \rangle = \langle u_1 \rangle \langle u_2 \rangle \cdots \langle u_k \rangle \langle q \rangle \langle v_1 \rangle \cdots \langle v_m \rangle$

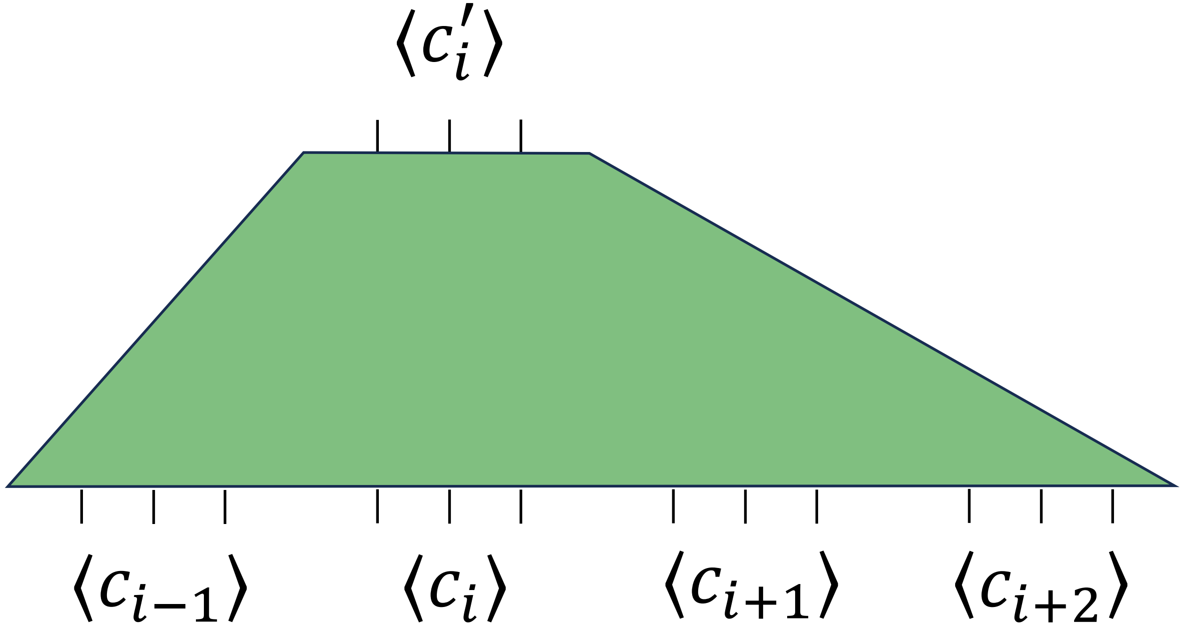
TM ⇒ Circuit

- There is a circuit C_M that computes $\langle c_i' \rangle$ given $\langle c_{i-1} \rangle, \langle c_i \rangle, \langle c_{i+1} \rangle, \langle c_{i+2} \rangle$

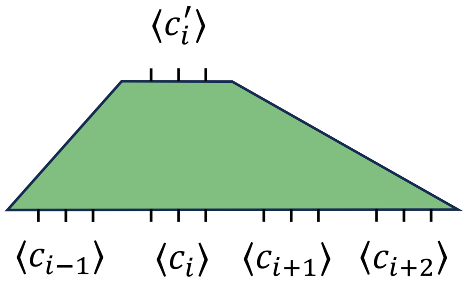


TM $\Rightarrow$ Circuit

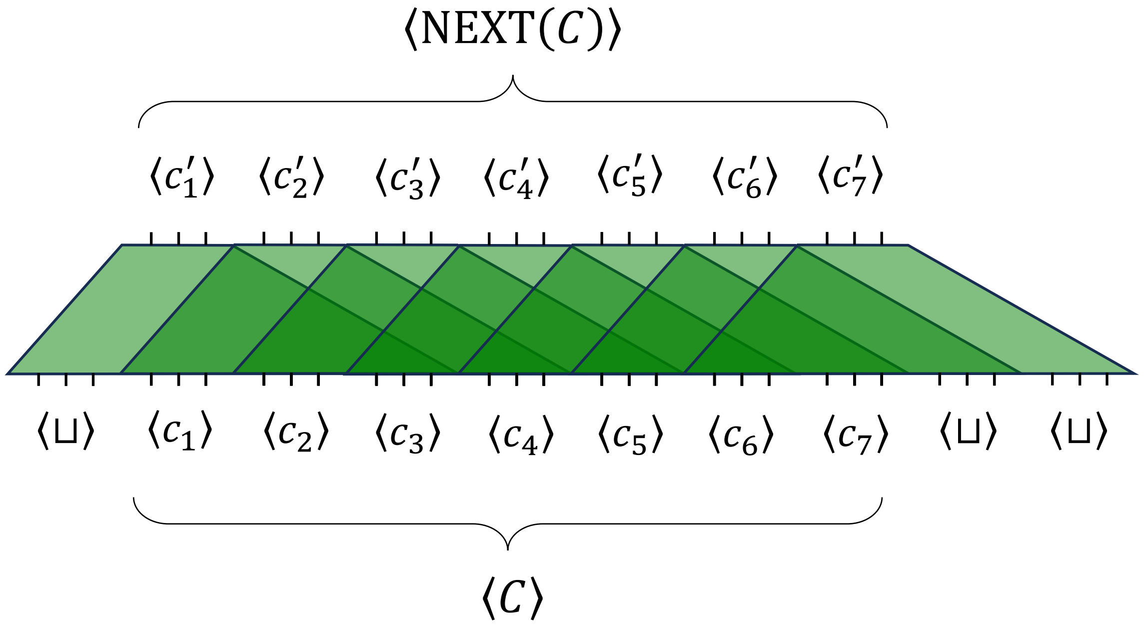
- There is a circuit C_M that computes $\langle c_i' \rangle$ given $\langle c_{i-1} \rangle, \langle c_i \rangle, \langle c_{i+1} \rangle, \langle c_{i+2} \rangle$



TM ⇒ Circuit

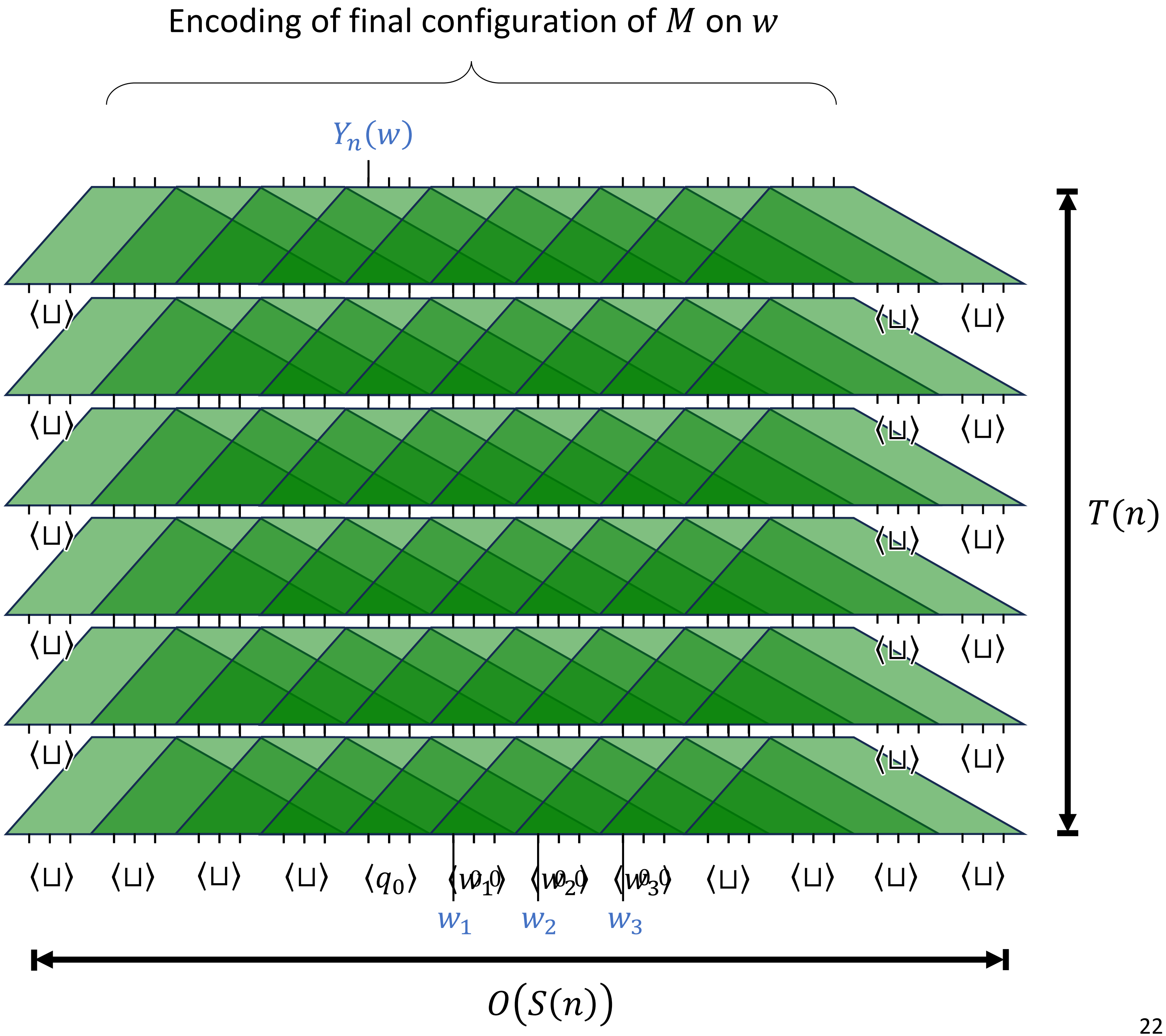


- There is a circuit C_M that computes $\langle c_i' \rangle$ given $\langle c_{i-1} \rangle, \langle c_i \rangle, \langle c_{i+1} \rangle, \langle c_{i+2} \rangle$
- Now let's combine many copies of C_M in parallel:



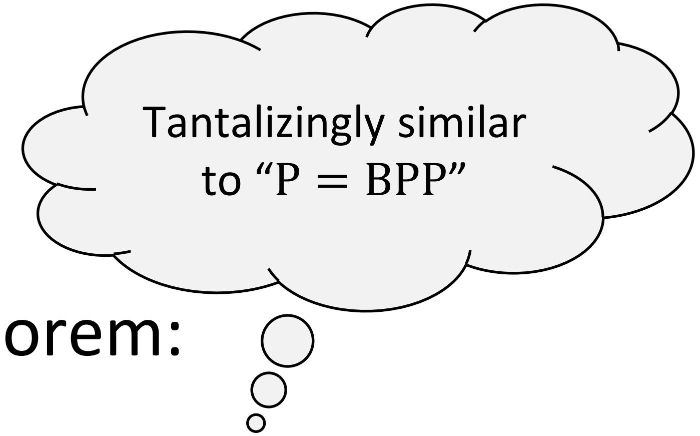
TM ⇒ Circuit

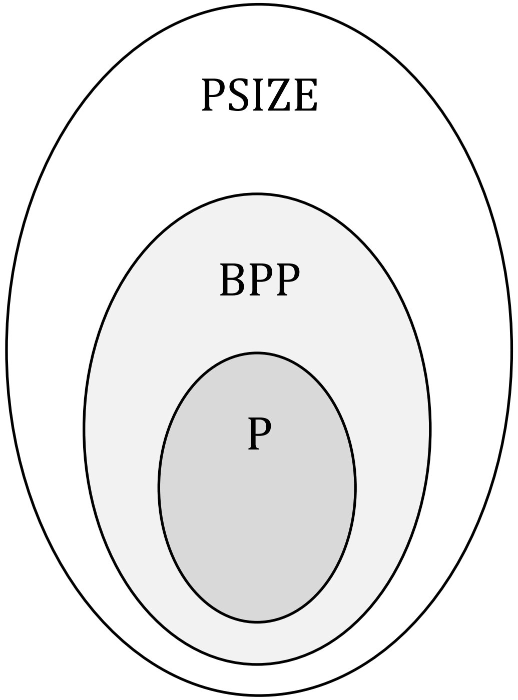
- Size: $O\big(S(n) \cdot T(n)\big)$
- Assume WLOG:
  - $\langle 0 \rangle = 0^r$ and $\langle 1 \rangle = 10^{r-1}$
  - M halts in starting cell
  - $\mathrm{NEXT}(C) = C$ if C is a halting configuration
  - $\langle q_{\mathrm{accept}} \rangle = 1^r$
  - $\langle q_{\mathrm{reject}} \rangle = 01^{r-1}$



Adleman's theorem

- We just showed that $\mathrm{P} \subseteq \mathrm{PSIZE}$
- Next, we will prove a stronger theorem:





Adleman's Theorem: $\mathrm{BPP} \subseteq \mathrm{PSIZE}$

- Note: The circuit model is a deterministic model of computation!
- Proof of Adleman's theorem: Next 8 slides