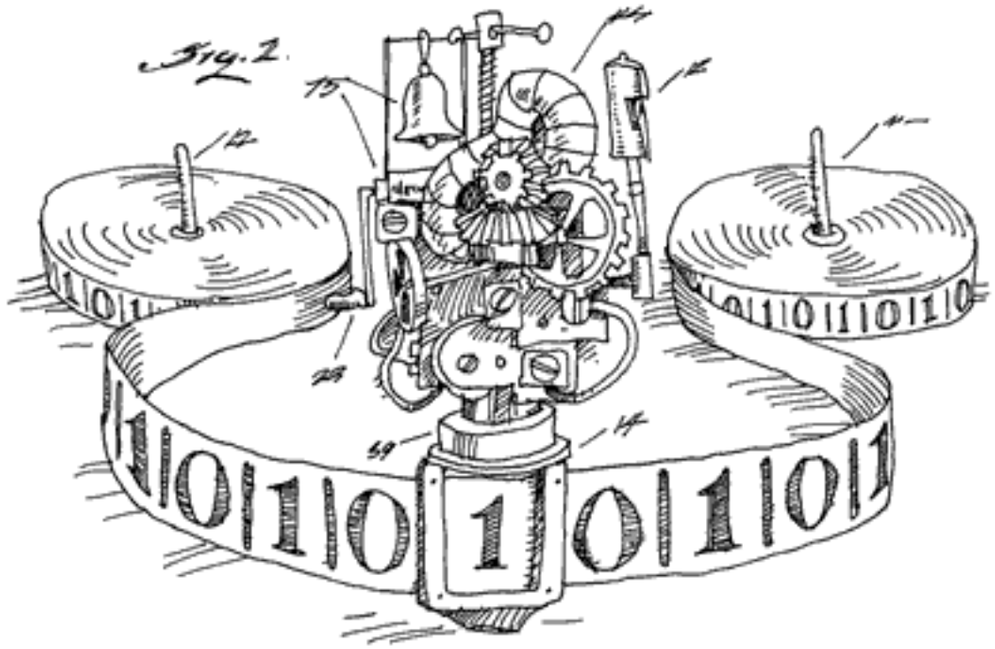
CSC4010: Introduction to Theoretical Computer Science



Lecture 16

Circuit Complexity of a Binary Language

Let $Y \subseteq \{0,1\}^*$. Let $Y_n = Y \cap \{0,1\}^n$. (strings in Y of length n)

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.

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

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

Definition: PSIZE $= \{Y \subseteq \{0,1\}^* :$ circuit complexity of Y is poly$(n)\}$

$$\text{PSIZE} = \cup_{k=1}^{\infty} \text{SIZE}(n^k)$$

Circuit Complexity of a Binary Language

a) Circuit is a model of computation

b) $\langle C \rangle$: encoding of C in $\{0,1\}^*$.

c) $\langle C \rangle$ can be input to a TM.

Let $Y \subseteq \{0,1\}^*$. Let $Y_n = Y \cap \{0,1\}^n$.

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.

Circuit Value Problem: $\{\langle C, x \rangle \mid C(x) = 1\}$.

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

CVP $\in$ P

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

Definition: PSIZE $= \{Y \subseteq \{0,1\}^* :$ circuit complexity of Y is poly$(n)\}$

$$\text{PSIZE} = \bigcup_{k=1}^{\infty} \text{SIZE}(n^k)$$

Cook-Levin: reducing every problem in NP to CIRCUIT-SAT.

CIRCUIT-SAT: $\{\langle C \rangle \mid \exists x$ for which $C(x) = 1\}$

CIRCUIT-SAT is NP-hard.

Circuit Complexity vs. time complexity

$P = \bigcup_{c=1}^{\infty} TIME(n^c) \subseteq PSIZE = \bigcup_{c=1}^{\infty} SIZE(n^c)$

Theorem: Let $T : \mathbb{N} \to \mathbb{N}$ be any time constructible function (time bound). Then, TIME(T) $\subseteq$ SIZE(T^2) . In particular, $P \subseteq$ PSIZE .

↳ {Y | s.t. Y can be decided in time $O(T)$} ⇒ then the circuit complexity of Y is $O(T^2)$.

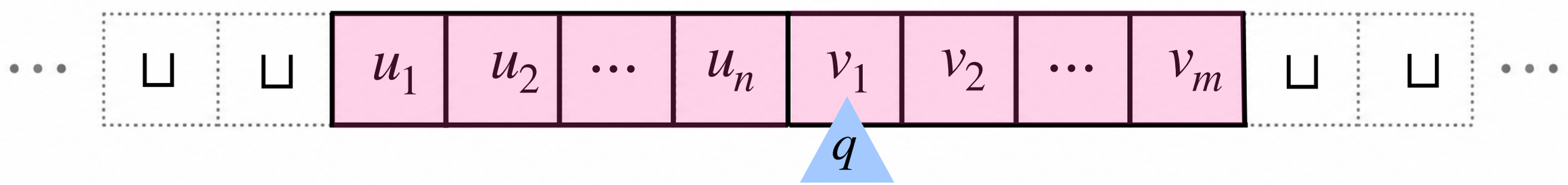
- Polynomial Time Algorithms imply a family of polynomial size circuits for a problem.
- The proof is based on computational histories.

a) Given any TM M, we will circuit that simulates it.

b) This will be a general proof strategy that we will also use to show CIRCUIT-SAT is NP-hard.

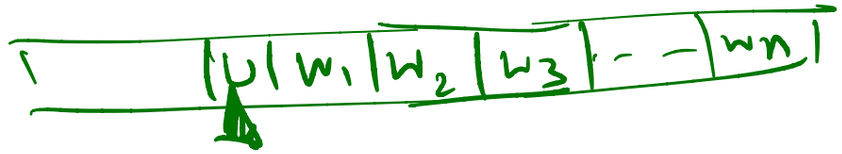
Recap: Configurations of a Turing Machine

- Let $M = (Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta, \sqcup, \Sigma)$ be a Turing Machine.
- A configuration uqv of M is a string over $\Sigma \cup Q$
- The tape currently contains uv .
  - The machine is in state q and the "head" is pointing at the first symbol of v .



Recap: The Initial Configuration

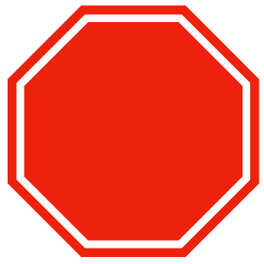
- Let $w \in \{0,1\}^*$ be the input.
- The initial configuration of M on w is $q_0 \sqcup w$.



Recap: The "next" configuration

- For any configuration uqv, we define NEXT(uqv) as follows:
  - Break uqv into individual symbols: $uqv = u_1, u_2, \cdots, u_n q v_1, v_2, \cdots, v_m$.
  - If $\delta(q, v_1) = (q', b, R)$, then NEXT($uqv$) $= u_1 u_2 \cdots u_n b q' v_2, v_3 \cdots v_m$.
    - **Edge Case:** If $m = 1$, then NEXT(uqv) $= u_1 u_2 \cdots u_n b q' \sqcup$.
  - If $\delta(q, v_1) = (q', b, L)$, then NEXT($uqv$) $= u_1 u_2 \cdots u_{n-1} q' u_n b v_2, v_3 \cdots v_m$.
    - **Edge Case:** If $n = 0$, then NEXT(uqv) $= q' \sqcup b v_2 v_3 \cdots v_m$.
  - We write $\text{NEXT}_M(uqv)$ if M is not clear from context.

Recap: Halting Configurations



- An accepting configuration is of the form $uq_{\text{accept}}v$.
- An rejecting configuration is of the form $uq_{\text{reject}}v$.
- A halting configuration is either an accepting or a rejecting configuration.

Recap: Computational History

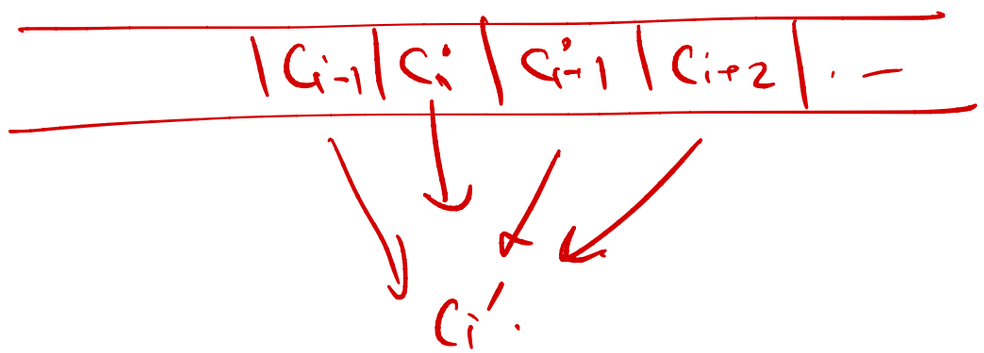
- Let $w \in \{0,1\}^*$ be an input string.
- Let C_0 be the initial configuration of M on w i.e. $C_0 = q_0 \sqcup w$.
- Inductively, for each $i \in \mathbb{N}$, let NEXT$(C_i) = C_{i+1}$.
- The computational history of M on w is the sequence $C_1, C_2, \cdots, C_T$, where C_T is the first halting configuration in the sequence.
- If there is no such C_T, then the computational history is infinite: $C_1, C_2, C_3, \cdots$.

Locality of Computation

- Let C be a configuration of a Turing Machine M.
- We can write $C = c_1 c_2 \cdots c_l$ for some $c_i \in \Sigma \cup Q$.
- Then, NEXT$(C) = c_1' c_2' \cdots c_l'$ for some $c_i' \in \Sigma \cup Q$.

How does any c_i change (its symbol) when we apply NEXT to C.

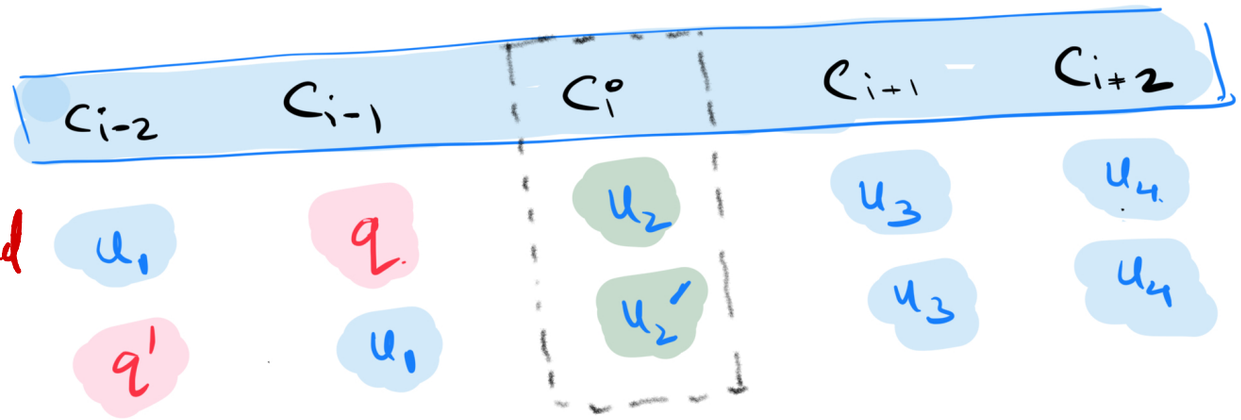
$c_i' \neq c_i \Rightarrow$ $c_i \in Q$ or $c_{i-1} \in Q$ or $c_{i+1} \in Q$. c_{i+2}.

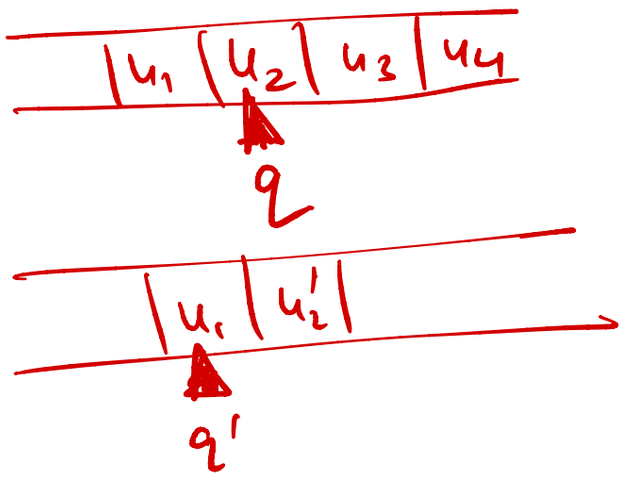


Locality of Computation

(How does M's tape symbol of C change when NEXT is applied)

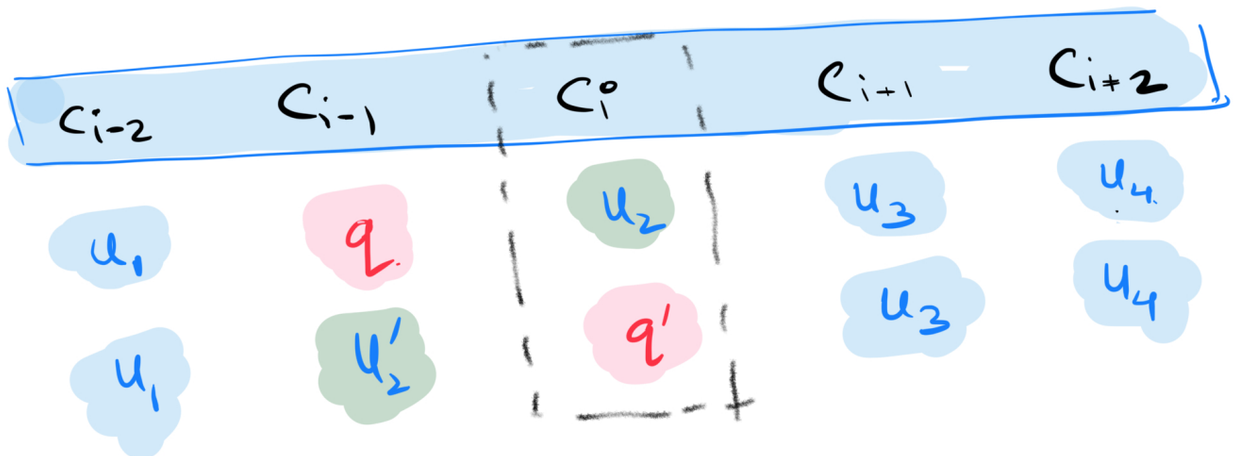
$\delta(q, u_2) = q', u_2', L$

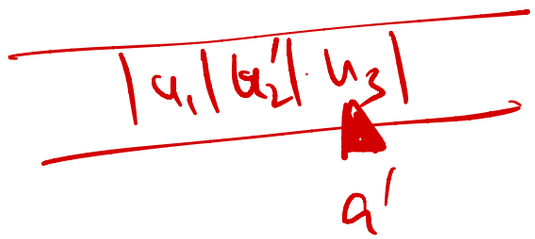




c_i' is determined by the tape contents of (c_i, c_{i-1})

$\delta(q, u_2) = q', u_2', R$

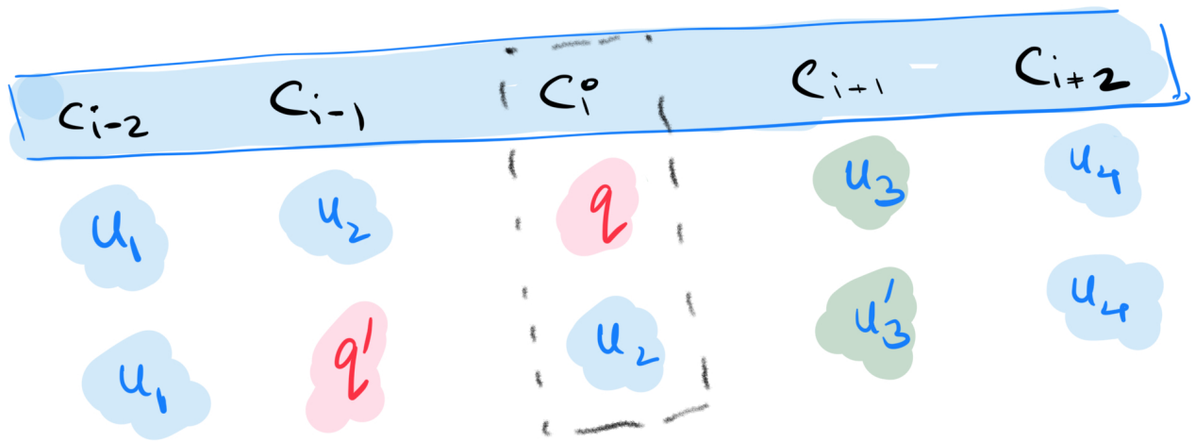


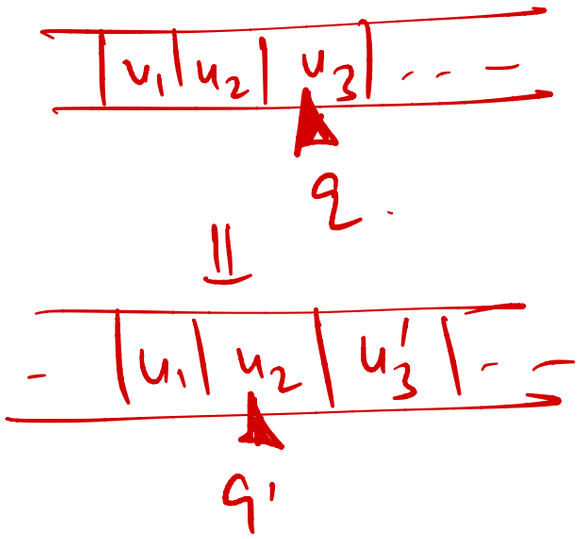


c_i' depends on c_{i-1}, c_i

Locality of Computation

$\delta(q, u_3) = q', u_3', L$

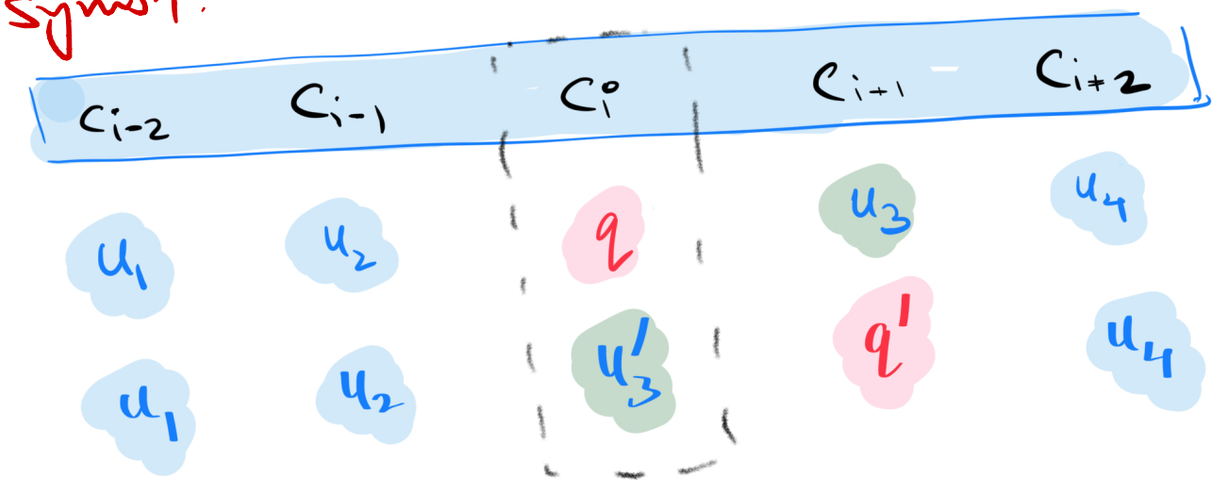




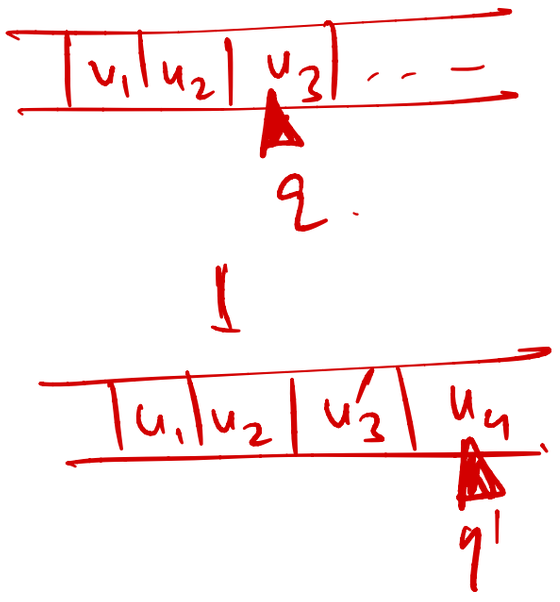
c_i' depends on c_i, c_{i+1}

The tape head was on c_{i+1} and that determines the contents of the ith symbol.

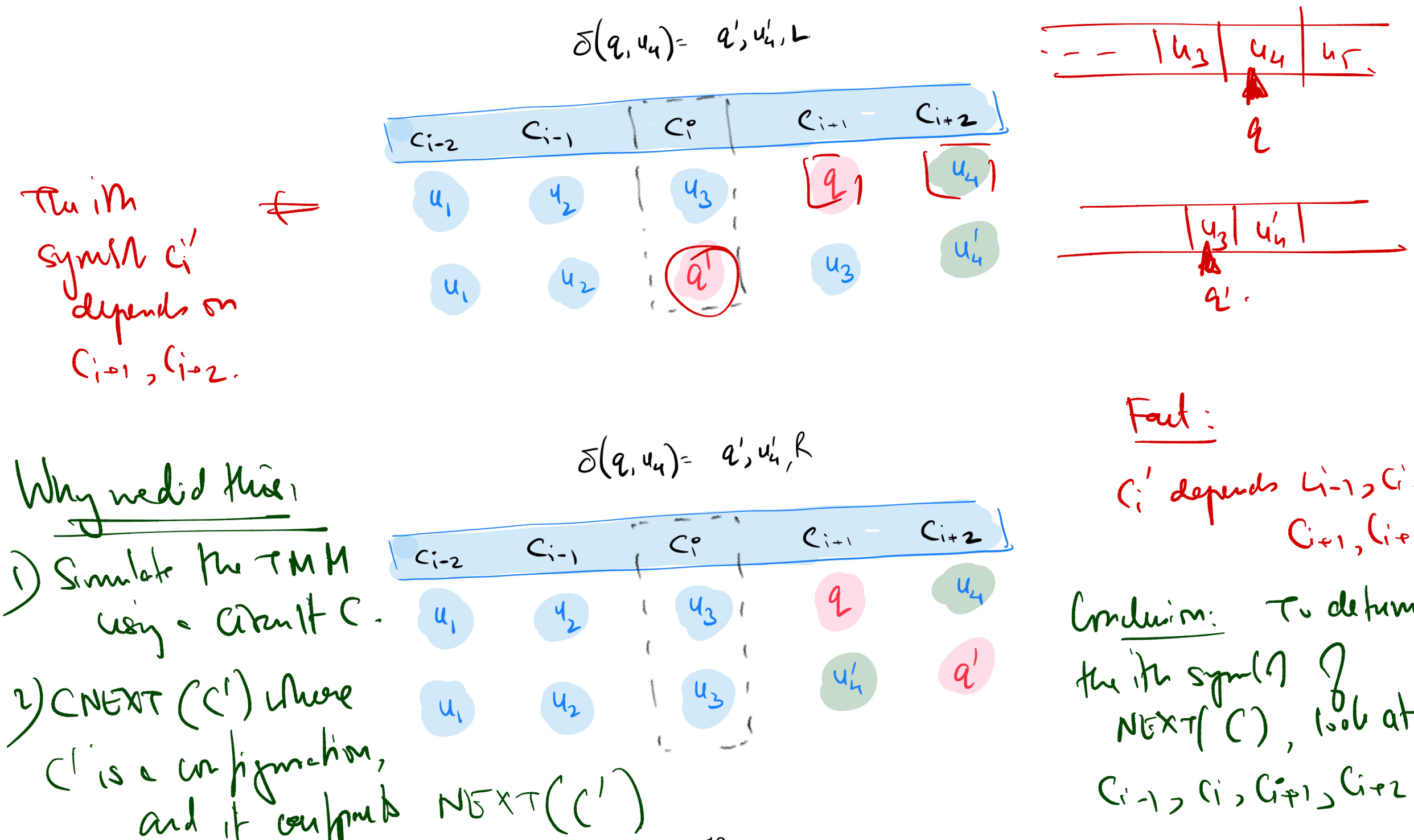
$\delta(q, u_3) = q', u_3', R$



c_i' depends on c_i, c_{i+1}



Locality of Computation



The ith symbol c_i' depends on $c_{i-1}, c_i, c_{i+1}, c_{i+2}$.

Why needed this:

1) Simulate the TM M using a circuit C.
2) C NEXT(C') where C' is a configuration, and it computes NEXT(C')

Fact:

c_i' depends $c_{i-1}, c_i, c_{i+1}, c_{i+2}$.

Conclusion: To determine the ith symbol of NEXT(C), look at $c_{i-1}, c_i, c_{i+1}, c_{i+2}$.

Encoding Configurations in Binary

- Let C' be a configuration of a Turing Machine M, say $C' = u_1 u_2 \cdots u_k q v_1 v_2 \cdots v_m$.
- 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 $\langle C' \rangle = \overline{\langle u_1 \rangle \langle u_2 \rangle \cdots \langle u_k \rangle \langle q \rangle \langle v_1 \rangle \cdots \langle v_m \rangle}$.

From one configuration to the next

- Suppose $\langle C \rangle = B_1 B_2 \cdots B_l$ where $B_i \in \{0,1\}^r$
- Suppose $\langle \text{NEXT}(C) \rangle = B'_1 B'_2 \cdots B'_l$ where $B'_i \in \{0,1\}^r$
- If we want to know B'_i, we do not need to look at all of $\langle C \rangle$.
- For B'_i sufficient to consider $B_{i-1}, B_i, B_{i+1}, B_{i+2}$.

From one configuration to the next

B_i' depends on $B_i, B_{i+1}, B_{i+2}, B_{i-1}$

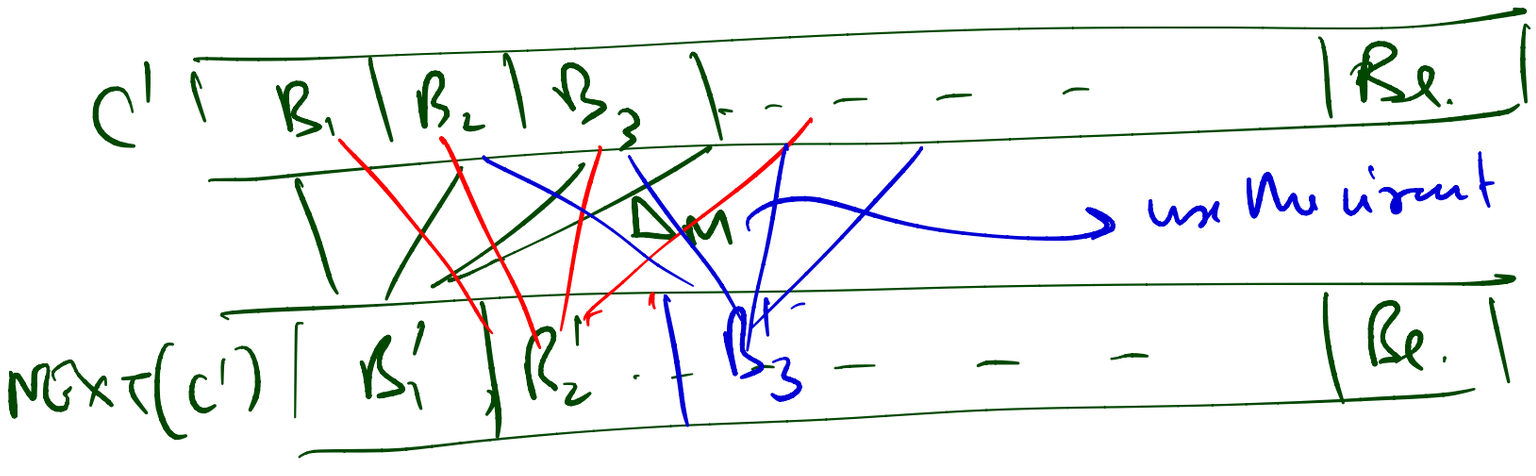
- There is a function $\Delta_M : \{0,1\}^{4r} \to \{0,1\}^r$ such that for every configuration C, if $\langle C \rangle = B_1 B_2 \cdots B_l$ and $\langle \mathrm{NEXT}(C) \rangle = B'_1 B'_2 \cdots B'_l$ then for every i, we have,

$$\Delta_M(B_{i-1} B_i B_{i+1} B_{i+2}) = B'_i$$

⇒ the size of the circuit

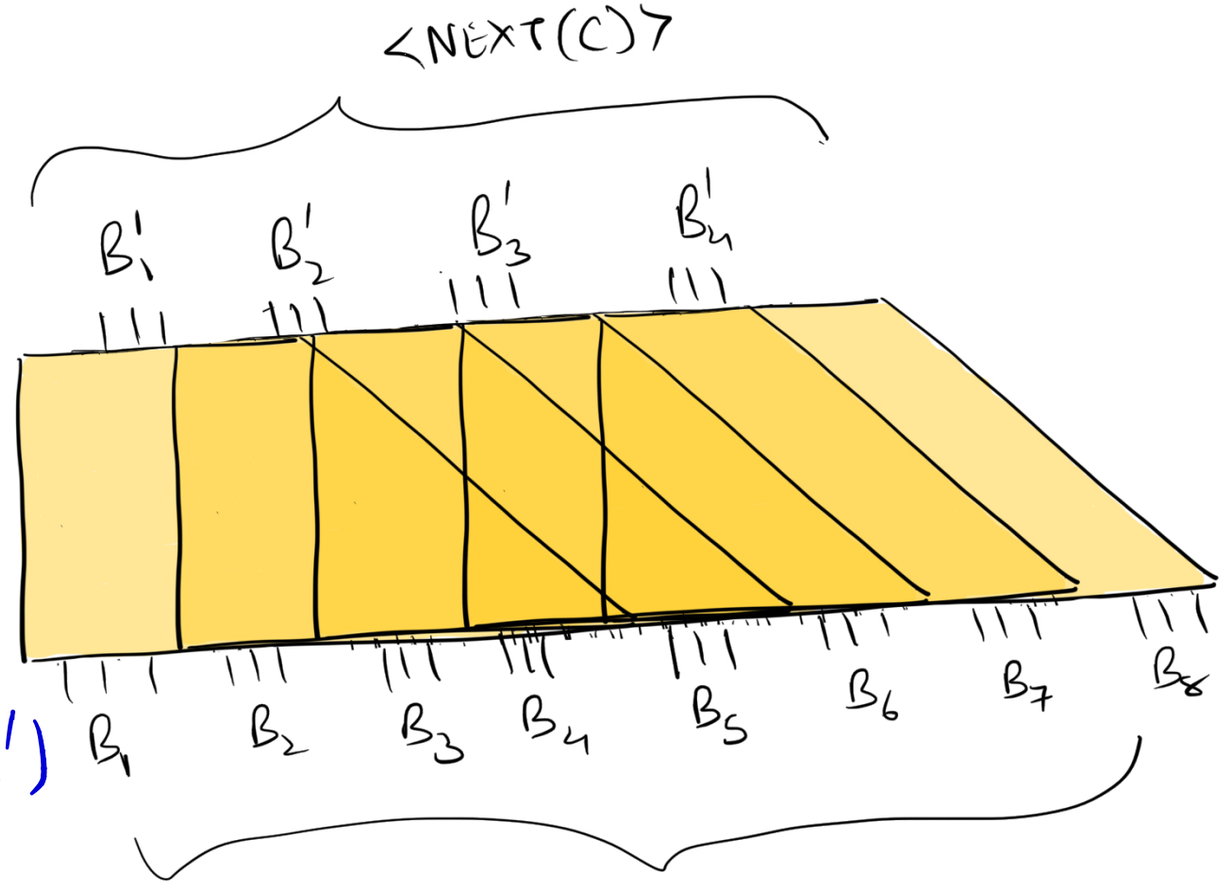
- Intuitively, Δ_M is a version of the transition function of M.

computing this function is also a function of n.
⇒ it is a constant.



From one configuration to the next

- There is a circuit of size $O(1)$ that computes Δ_M
- Combine the copies in parallel:



A circuit of size $\ell(\cdot)$ which takes as i/p a configuration $\langle C' \rangle$ and outputs $\langle NEXT(C') \rangle$. ℓ = the length of the configuration $\leq \tau(n)$.

Δ_M's circuit is called C_M. for each B'_i ⇒ a copy of C_M.

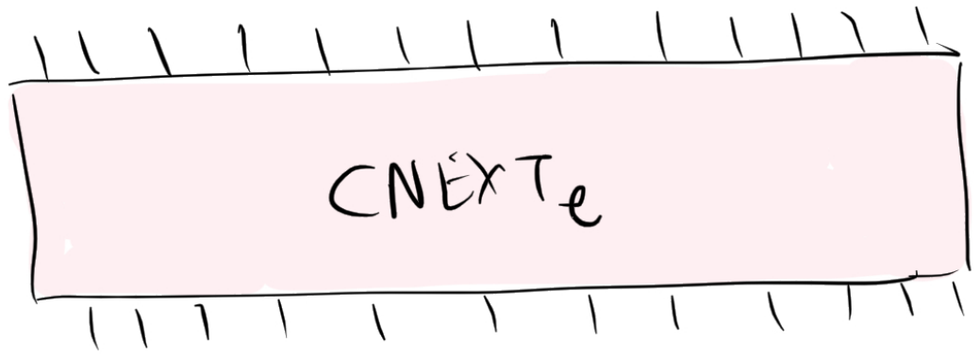
From one configuration to the next

- The construction on the previous slide shows that for every $\ell \in \mathbb{N}$, there is a circuit $\text{CNEXT}_\ell : \{0,1\}^{r\ell} \to \{0,1\}^{r\ell}$ satisfying:

  - If C is a configuration such that $|C| = |\text{NEXT}(C)| = l$, then

$$\text{CNEXT}_\ell(\langle C \rangle) = \langle \text{NEXT}(C) \rangle$$

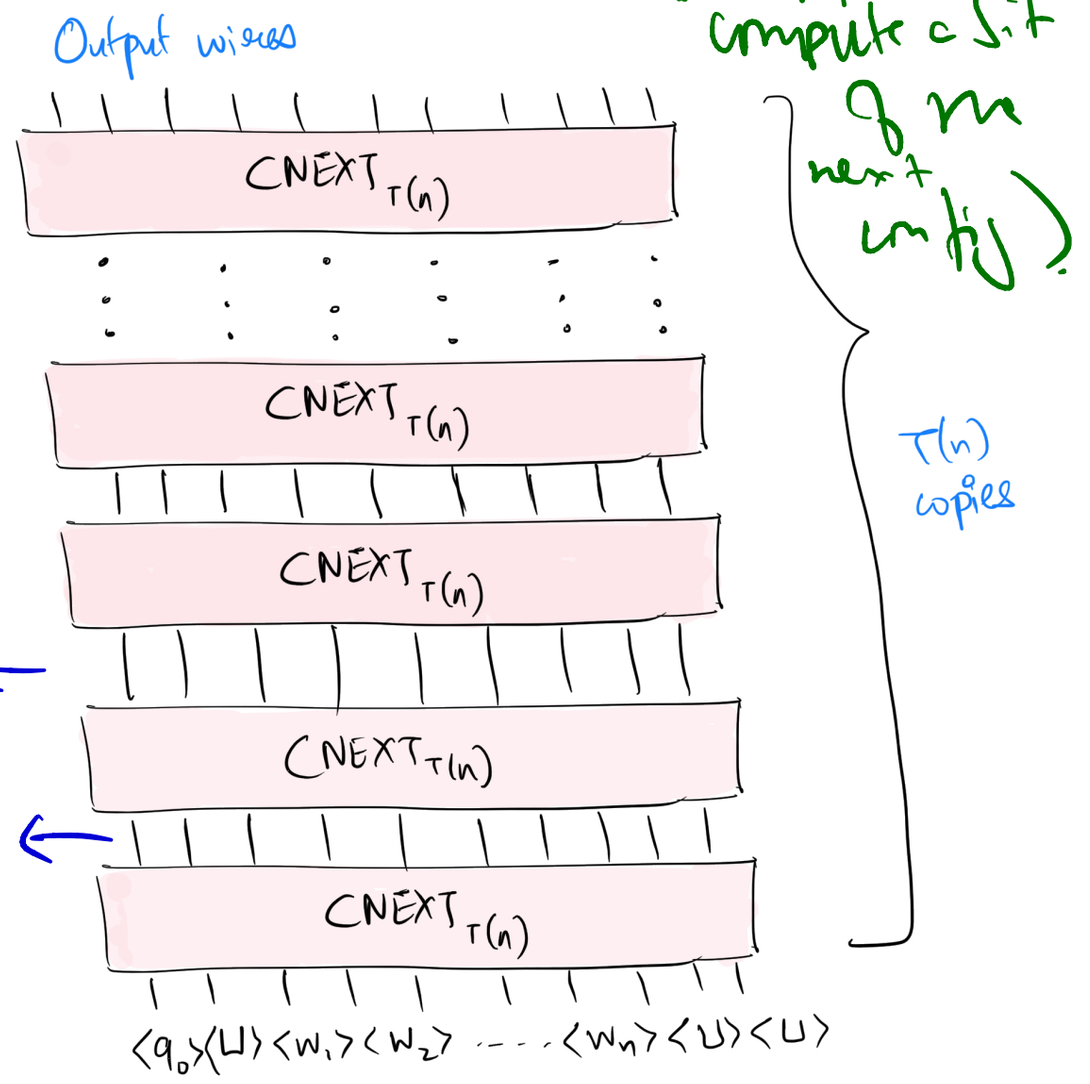
- The size of CNEXT_ℓ is $O(\ell)$.



Converting TM to Circuit

- Let M be a Turing Machine that decides L with time complexity $T(n)$.
- We get a circuit of size $O(T(n)^2)$ computing $L_n = \{0,1\}^n \cap L$.

Main idea: locality of computation. (Allows for small circuits to compute a bit of the next config)



Suppose the i/p to the TM is a string $w_1 w_2 \cdots w_n$

$C_0 = q_0 \sqcup w_1 w_2 \cdots w_n$

C_1 ← C_2 ← ...

On C_i's corresponds to halting configuration.

$O(\ell)$ ℓ = length of config

a) Have some auxiliary gates to check for acceptance/rejection. $\leq T(n)$

Circuit Complexity vs. time complexity

Theorem: Let $T : \mathbb{N} \to \mathbb{N}$ be any time constructible function (time bound). Then, TIME(T) $\subseteq$ SIZE(T^2) . In particular, $P \subseteq$ PSIZE .

given any $L \in$ TIME(T)
$\Rightarrow \exists M_L$ which decides in time T.
$\Rightarrow \exists$ a circuit β size $O(T^2)$ which simulates M_L on fixed length i/ps.

Next time: 1) CIRCUIT-SAT is NP-hard.
$L \in NP \Rightarrow M_L$ is an decider for it.
given $M_L \Rightarrow$ convert to a circuit in polynomial time.

2) SAT is NP-complete.
↳ $\{ \langle F \rangle \mid F$ is a boolean formula that is satisfiable $\}$.