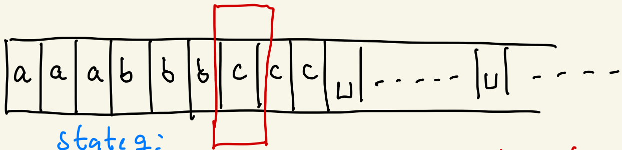
Sipser 3.1 Turing machines

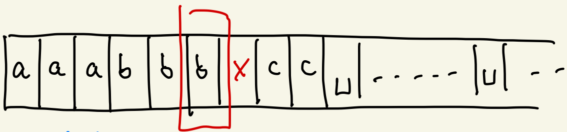
reading head

left end → 

infinite tape

state q_i

$$\delta(q_i, c) = (q_j, x, L)$$



state q_j

when reading 'c' in state q_i: replace 'c' by 'x', move head one step to the left and go to state q_j

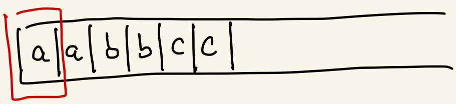
3 special states

q_0 initial state

q_{acc} accepting state

q_{reject} rejecting state

example $L = \{a^n b^n c^n \mid n \geq 0\}$ not a CFL

on input aabbcc: 

q_0

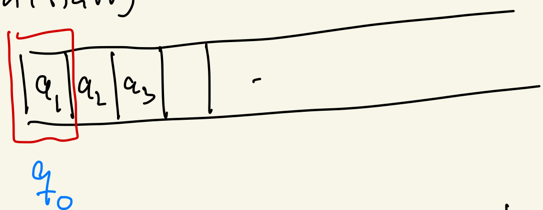
We saw 'a' next step look for a 'b' and then a 'c'

Formal definition of a TM

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$$

assumptions:

- $\text{'}\sqcup\text{'} \in \Gamma \setminus \Sigma$
- $\Sigma \subset \Gamma$
- initially



- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

head must move either right or left

<u>configuration</u> $\quad u q v \quad q \in Q, u, v \in \Gamma^*$

$$\Updownarrow$$

$$u_1 u_2 \cdots u_r \boxed{v_1} v_2 \cdots v_s$$

$$q$$

Configuration C_1 <u>yields</u> configuration C_2

$\Updownarrow$ def

M can go from C_1 to C_2 in one step

$$u\,a\,q_i\,b\,v \xrightarrow{\text{yields}} u\,q_j\,a\,c\,v$$

deterministic

$$\Updownarrow$$

$$\delta(q_i, b) = (q_j, c, L)$$

Important: head cannot move past left boundary!

$$q_i\,b\,v \longrightarrow q_j\,c\,v \quad \text{if } \delta(q_i, b) = (q_j, c, L)$$

start configuration $\quad q_0\,w \qquad w = \text{input}$

accepting configuration $\quad u\,q_{acc}\,v$

rejecting configuration $\quad u'\,q_{rej}\,v'$

Language accepted by M:

$$L(M) = \{ w \mid q_0 w \xrightarrow{*} u\,q_{acc}\,v \text{ for some } u, v \in \Gamma^* \}$$

$$q_0 w = C_0 \to C_1 \to C_2 \cdots \to C_n = u\,q_{acc}\,v$$

Example with $L = \{a^n b^n c^n \mid n \geq 0\}$

input aabbcc (states not shown)

$\underline{a}abbcc \to A\underline{a}bbcc \to Aa\underline{b}bcc \to AaB\underline{b}cc$

$\to AaBb\underline{c}c \to AaB\underline{b}Cc \to \cdots \to \underline{A}aBbCc$

$\to A\underline{a}BbCc \to AA\underline{B}bCc \to AAB\underline{b}Cc \to AABB\underline{C}c$

$\to AABBC\underline{c} \to AABB\underline{C}C \to \cdots \to A\underline{A}BBCC$

$\to \cdots \to AABBCC\underline{\sqcup}$ accept

Definition 3.5

L is Turing-recognizable (also called recursively enumerable)

$\Updownarrow$ def

$L = L(M)$ for some Turing machine M

3 possible outcomes when a TM M is started on a string w

- M accepts w $\quad q_0 w \rightsquigarrow u\, q_{acc}\, v$
- M rejects w $\quad q_0 w \rightsquigarrow u'\, q_{rej}\, v'$
- M runs forever

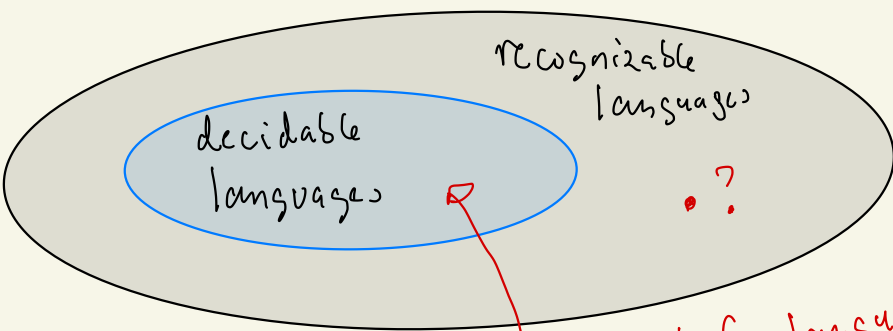
$\boxed{\text{A \textbf{decider} is a TM that always stops}}$

<u>Definition 3.6</u>

L is decidable

$\Updownarrow$

$L = L(M)$ for some decider M

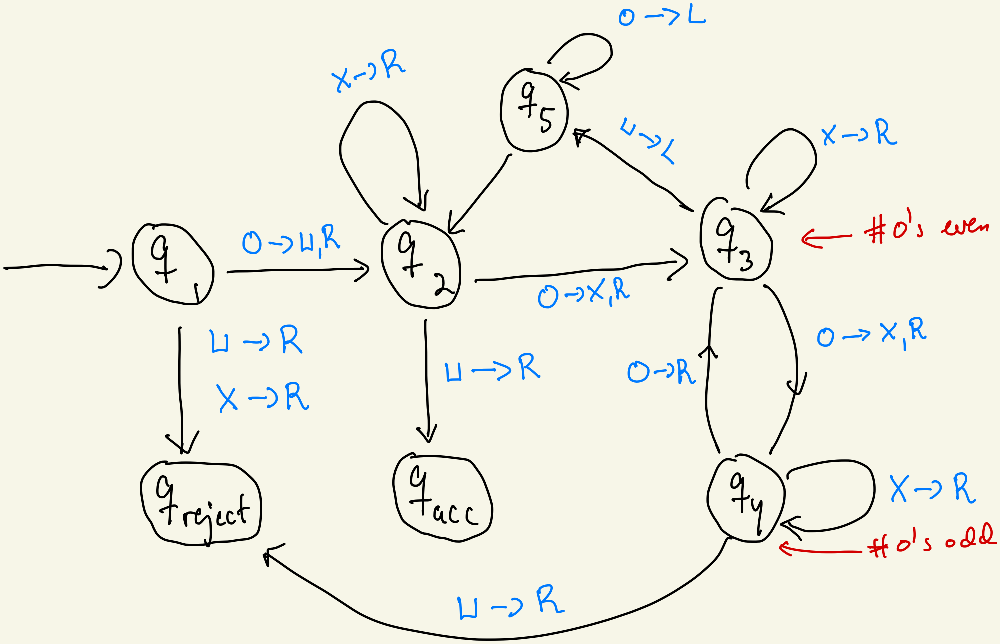


context-free languages are decidable

Example 3.7 Decider for $A = \{0^{2^n} \mid n \geq 0\}$

Idea:

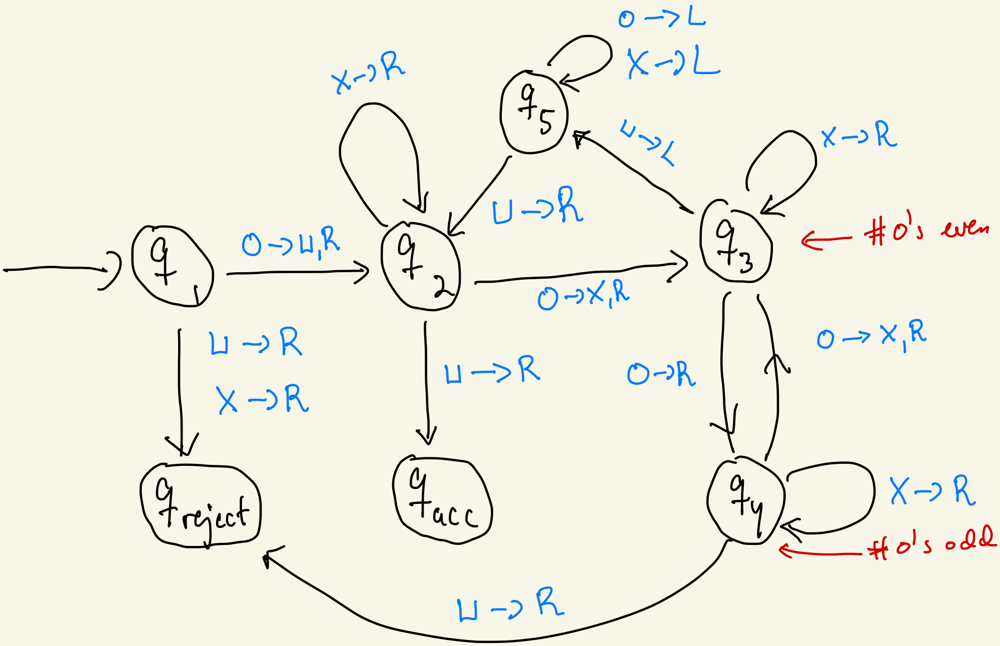
1. while moving right till the end of the string cross out every second '0'
2. If there was precisely one '0' accept
3. If there was $2k+1$ '0''s for some $k \geq 1$ reject
4. Return reading head to left end of the tape.
5. Goto 1.



Notation

$0 \to \sqcup, R$: when reading '0' write '$\sqcup$' and move head right

$0 \to R$: when reading '0' move head right without changing tape (print '0')



$q_1 0 0 0 0$

$\sqcup q_2 0 0 0$

$\sqcup x q_3 0 0$

$\sqcup x 0 q_4 0$

$\sqcup x 0 x q_3 \sqcup$

$\sqcup x 0 q_5 x \sqcup$

$\sqcup x q_5 0 x \sqcup$

$\sqcup q_5 x 0 x \sqcup$

$q_5 \sqcup x 0 x \sqcup$

$\sqcup q_2 x 0 x \sqcup$

$\sqcup x q_2 0 x \sqcup$

$\sqcup x x q_3 x \sqcup$

$\sqcup x x x q_3 \sqcup$

$\sqcup x x q_5 x \sqcup$

$\sqcup x q_5 x x \sqcup$

$\sqcup q_5 x x x \sqcup$

$q_5 \sqcup x x x \sqcup$

$\sqcup q_2 x x x \sqcup$

$\sqcup x q_2 x x \sqcup$

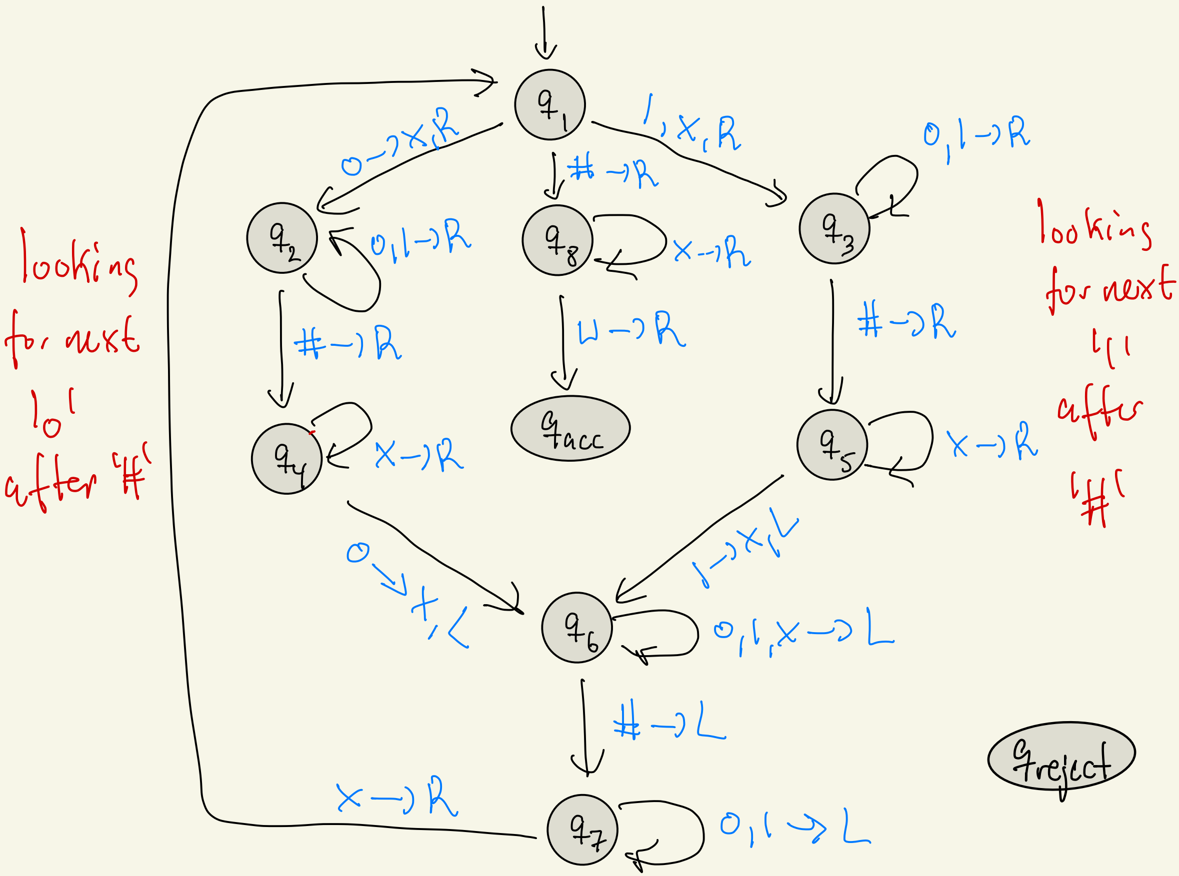
$\sqcup x x q_2 x \sqcup$

$\sqcup x x x q_2 \sqcup$

$\sqcup x x x x \sqcup q_{acc}$

Example 3.9

$B = \{ w\#w \mid w \in \{0,1\}^* \}$



0 1 # 0 1

1 0 # 1

Example 3.11 $C = \{a^i b^j c^k \mid i \cdot j = k \wedge i, j, k \geq 1\}$

Idea: 1. check wheter $w \in a^+ b^+ c^+$ while scanning left toright
reject if $w \notin a^+ b^+ c^+$ (simulating a DFA here)

2. a a .. a b .. b c ... c

3. x a .. a b .. b c ... c (cross out one a)

4. → ... → x a .. a b .. b c ... c (move to 'b''s)

5. cross of 'b''s with 'y' and one 'c'
until no more 'b''s. If too few 'c''s reject

6. Move to leftmost y and restore 'b''s

7. Move to leftmost 'a'

a. If there is an 'a' cross it out with x
move right and goto 3

b. if no more 'a''s go left to check if
no more 'c''s.
If no more 'c''s accept

else reject

Some usefull TMs (not in Sipser)

Rightshift: $q_0 a_1 a_2 \cdots a_n \xrightarrow{R_{shift}} \triangleright q_{acc} a_1 a_2 \cdots a_n$

new symbol to mark lefthand side of tape

$a_1 a_2 \cdots a_n \to \cdots \to \mathring{a}_1 a_2 \cdots a_n \underline{\sqcup}$

$\to \mathring{a}_1 a_2 \cdots \underline{a_n} \sqcup \to \mathring{a}_1 a_2 \cdots a_n \underline{a_n}$

$\to \cdots \to \mathring{a}_1 a_2 \cdot \underline{a_{n-1}} a_n a_n$

$\to \cdots \to a_1^{\circ} a_2 \cdots \underline{a_{n-1}} a_{n-1} a_n$

$\to \cdots \to \underline{a_1^{\circ}} a_1 a_2 \cdots a_n \to \triangleright \underline{a_1} \cdots a_n$

Notes 1. R_{shift} never touched the lefthand side of the tape

2. Could also have done e.g.

$q_0 a_1 a_2 \cdots a_n \longrightarrow \sqcup q_{acc} a_1 \cdots a_n$

and $w q_0 w' \longrightarrow w \sqcup q_{acc} w'$

Left shifting machine

$$\triangleright q_0 a_1 a_2 \cdots a_n \xrightarrow{\text{Lshift}} a_1 a_2 \cdots a_n q_{accept}$$

1. Move right and read character x (remember it via a state)

2. If $x = '\sqcup'$
   Move left
   write $'\sqcup'$
   stop

3. If $x \neq '\sqcup'$
   Move left
   write x
   move right
   goto 1

Turing machine which copies a string

$$q_0 a_1 a_2 \cdots a_n \xrightarrow{\text{Copy}} q_{acc} a_1 a_2 \cdots a_n a_1 \cdots a_n$$

$$\underline{a_1} a_2 \cdots a_n \rightsquigarrow \mathring{a}_1 a_2 \cdots a_n \underline{\sqcup} \rightsquigarrow \underline{\mathring{a}_1} a_2 \cdots a_n \#$$

$$\rightsquigarrow \mathring{a}_1 \underline{a_2} \cdots a_n \# a_1$$

$$\rightsquigarrow \mathring{a}_1 a_2 \cdots a_n \underline{\#} a_1 a_2 \cdots a_n$$

$\downarrow$ leftshift

$$\mathring{a}_1 a_2 \cdots a_n \underline{a_1} a_2 \cdots a_n \rightsquigarrow \underline{a_1} a_2 \cdots a_n a_1 \cdots a_n$$

operations $O(|w|^2)$