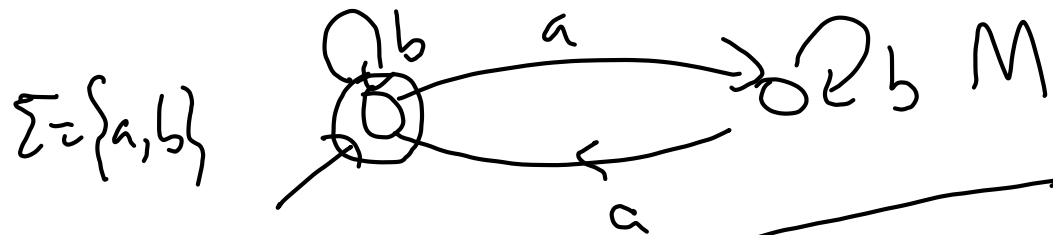


More on DFAs

- Regular languages
- Ops on languages
- Egs.

Σ - finite alphabet
 Deterministic finite automaton over Σ .



$L = \{ w \mid w \text{ has an even \# of } a\text{'s} \}$

$\delta: Q \times \Sigma \rightarrow Q$

M accepts L .

$L' = \{ w \mid \# \text{ of } a\text{'s in } w \text{ is a multiple of } 4 \}$

M does not accept L'

the L M accepts w — Not every w
 $\forall w$ acc by M — $w \in L$

L_1 , L_2 - reg. languages
over Σ

$L_1 \cup L_2$ - reg. lang over Σ .

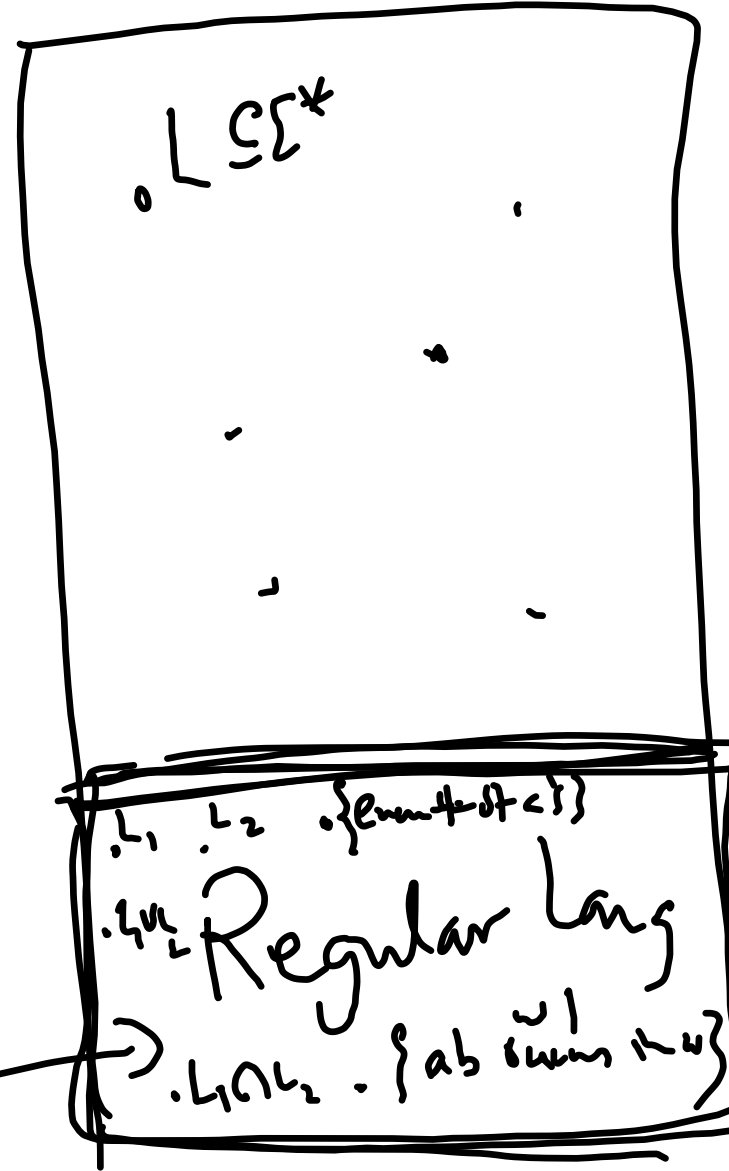
\hat{M} .

Fix $\Sigma = \{a, b\}$

$L_1, L_2 \in \text{Reg } L(\Sigma)$

Then $L_1 \cup L_2 \in \text{Reg.}$

All
Lang.
over Σ



Different characterizations of reg lang

Reg lang \longleftrightarrow Reg Exp.

DFA

$$L = \{ x = ww \mid w \in \Sigma^* \}$$

$abab \in L$

$aabbaabba$

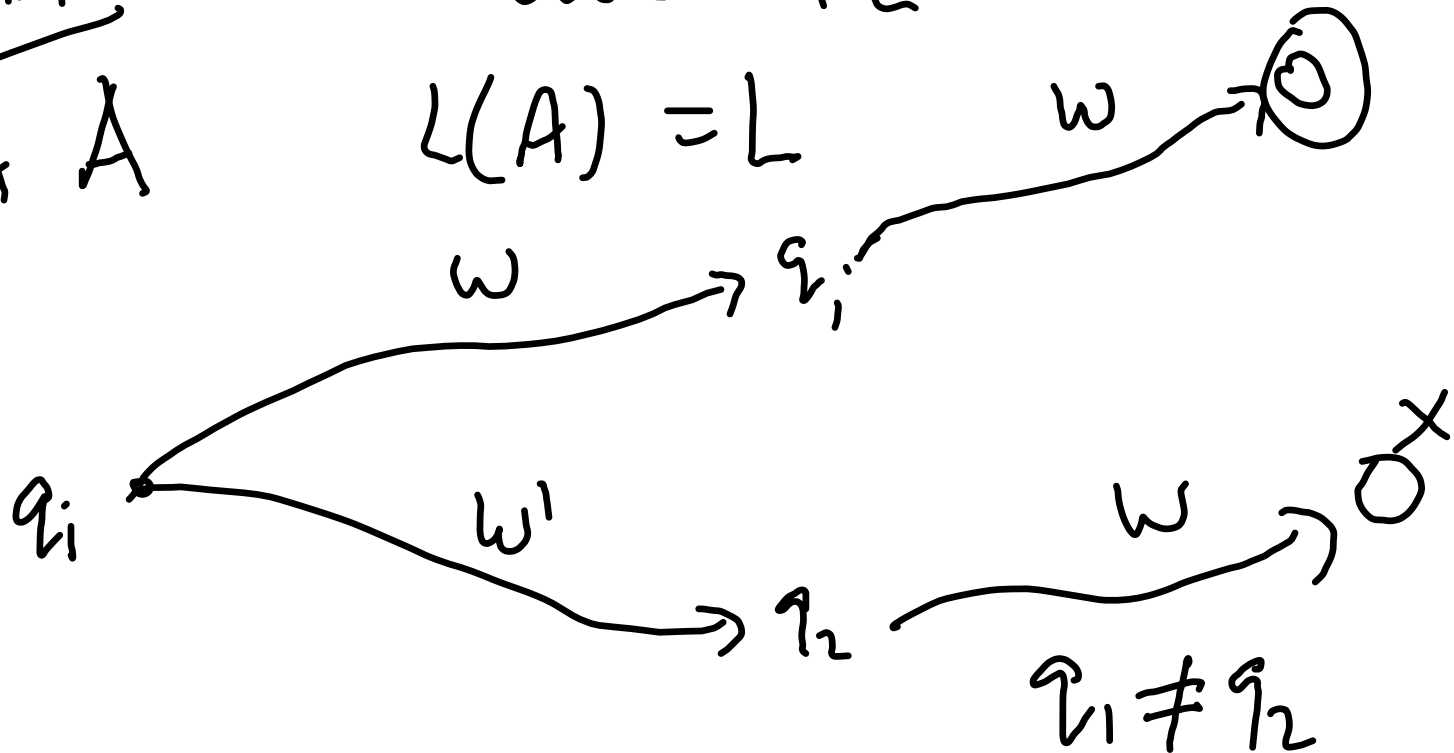
$abc \notin L$

$abba \notin L$

Proof by
Contradiction
DFA A

$L(A) = L$

$w \neq w'$



Operations on languages.

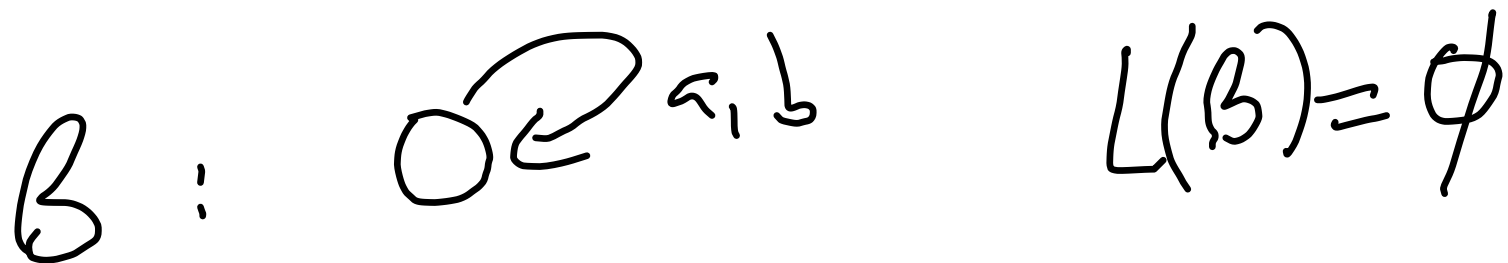
$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$$

$$L_1 \cap L_2 = \{ w \mid w \in L_1 \text{ and } w \in L_2 \}$$

$$\overline{L} = \{ w \mid w \notin L \}$$

$$L_1 = \emptyset$$
$$L_2 = \{ \epsilon \}$$

$$\Sigma = \{ a, b \}$$



$$L_1 \circ L_2 = \{ w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$$

$$L_1 = \{ a, ab, abb, abbb, \dots \}$$

$$L_2 = \{ ba, baba, babba, \dots \}$$

$$L_1 \circ L_2 = \{ \text{aba}, \underline{\text{abba}}, \dots \}$$

Concatenation.

$$L^* = \left\{ w_1 w_2 \dots w_n \mid \begin{array}{l} n \in \mathbb{N}_0 \\ w_i \in L \end{array} \right\}$$

$$L = \{ab\}$$

$$L^* = \left\{ \begin{array}{l} ab, abab, \dots \\ \epsilon \end{array} \right\}$$

$$L' = \{ab, bab\}$$

$$L'^* = \left\{ \begin{array}{l} ab, bab, abbab, babab, \dots \\ \epsilon \end{array} \right\}$$

$$\Sigma^* = \{a, b\}^*$$

Reg Lang \equiv Reg Exp
DFA • *

Several other ops.

Replace every 'a' by a 'b' in L

Replace every 'a' by a word w

Example construction of DFA.

L - a regular lang over Σ

$w \in \Sigma^*$

$$\{w\} \circ L = \{wx \mid x \in L\}$$

Ex. $L = \{a, aa, aaa, \dots\}$

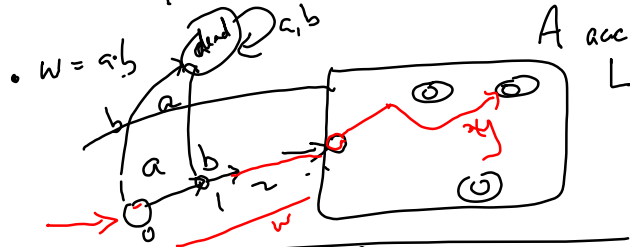
$w = bab$

$$\{w\} \circ L = \{baba, babaa, \dots\}$$

Given L, w .

Construct an automaton accepting $\{w\} \circ L$

• $w = \epsilon \quad \{w\} \circ L = L$



• $w, L \quad \{w\} \circ L$

State A accepts $L, A = (Q, \Sigma, q_0, \delta, F)$

$B = (Q', \Sigma, q'_0, \delta', F')$

$Q' = Q \cup [0, |w|-1] \cup \{dead\}$

$q'_0 = 0$

$F' = F$



$\delta' : \delta'(q, a) = \delta(q, a) \quad \forall q \in Q$

$\forall i \in [0, |w|-1] \quad \delta'(i, a) = i+1 \quad \text{if } w[i]=a$
 $\quad \quad \quad = \text{dead} \quad \text{if } w[i] \neq a$

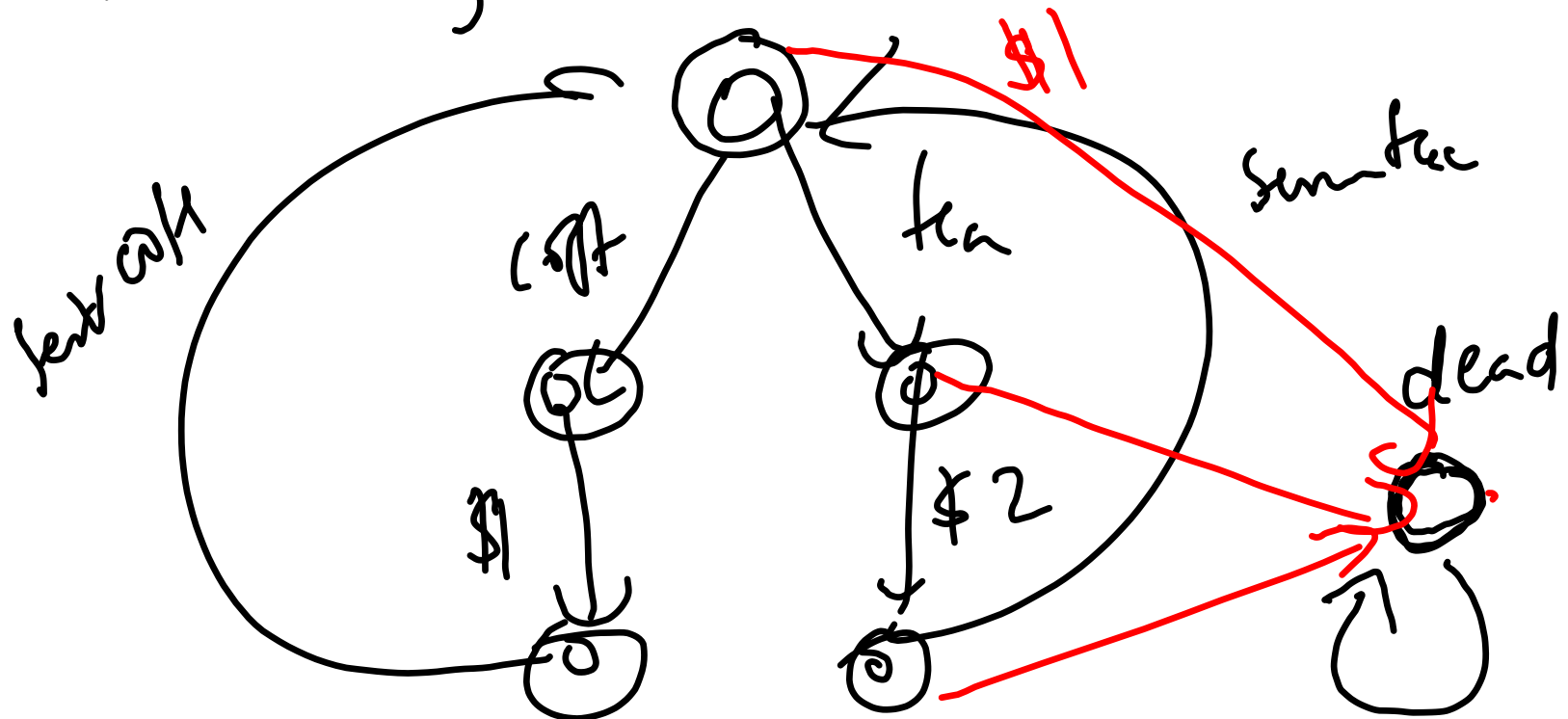
$\delta'(dead, a) = \text{dead}$

$L(B) = \underline{\{w\} \circ L}$

$\checkmark (\subseteq) \quad \forall x \text{ acc by } B, \quad x = wy \text{ where } y \in L$

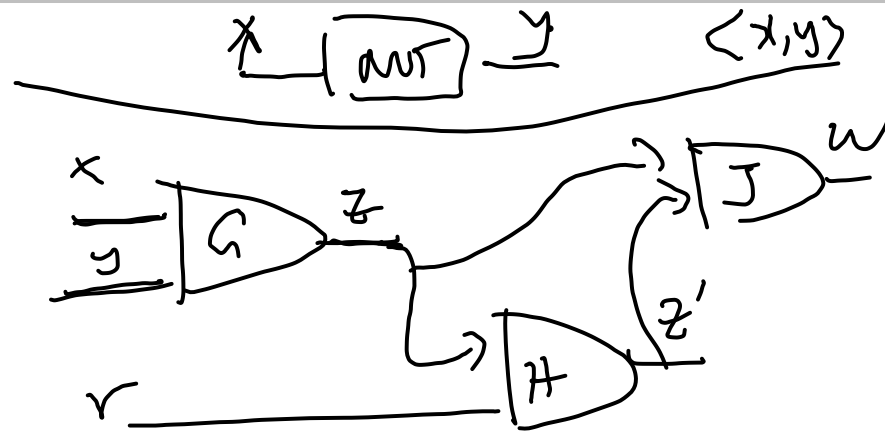
$(\supseteq) \quad x \in \{w\} \circ L \quad x = wy \quad y \in L$

Modeling of a system



$L(A)$ = behaviors of system.

cash,
tea,
\$1, \$2
send, st



$$\begin{aligned}
 \mathcal{I}W &= \{x, y, r, \dots\} \\
 \mathcal{N}IW &= \{z_1, z_2, \dots\} \\
 \mathcal{G} &= \{G_1, \dots, G_k\}
 \end{aligned}
 \left. \vphantom{\begin{aligned} \mathcal{I}W \\ \mathcal{N}IW \\ \mathcal{G} \end{aligned}} \right\} W$$

$$\forall G \in \mathcal{G} \quad f_G : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$$

$$C = \left\{ (x, y, G, z) \mid \begin{array}{l} x, y \in W \\ G \in \mathcal{G} \\ z \in W \end{array} \right\}$$

s.t. $\forall z$. there is at most one
 tuple of the form (x, y, G, z)
 is a circuit.

$$\Sigma = \{ \langle v_1, \dots, v_n \rangle \mid v_i \in \{0,1\} \}$$

L_C = behavior of the circuit.

A_C - Lang of A_C is L_C

$$A_C = (Q, \Sigma, q_1, \delta, F)$$

$$Q = \{ \langle v_1, \dots, v_n \rangle \mid \forall i, v_i \in \{0,1\} \}$$

$$q_1 = \langle 0, \dots, 0 \rangle$$

$$\langle v_1, \dots, v_n \rangle \xrightarrow{\langle v_1, \dots, v_n \rangle} \langle v_1', \dots, v_n' \rangle$$

- x_i is a non-input wire
& (x, y, G, x_i)

$$v_i = f_G(v_x, v_y) \quad y \xrightarrow{x} G \rightarrow x_i$$

$$\overline{v_i} = \overline{v_i'}$$

- $\forall i, v_i = v_i'$

Computer - FSM.

State: Memory + Regs. + HD

