Deterministic Finite Automaton (DFA)

$M = (Q, \Sigma, \delta, q_0, F)$

Q: states (memory)

Σ: alphabet

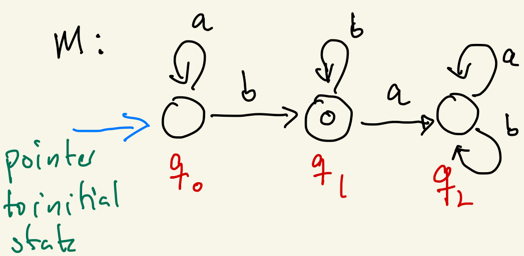
δ: transition function $\delta : Q \times \Sigma \to Q$

q_0: initial state

F: set of final (accepting) states (marked ◎)

example

M:



$Q = \{q_0, q_1, q_2\}$ $F = \{q_1\}$

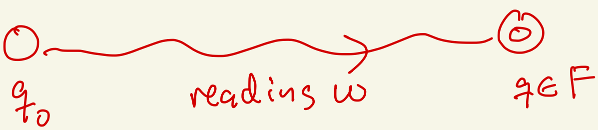
$L(M) = \{a^n b^m \mid n \geq 0, m \geq 1\}$

A DFA $M=(Q,\Sigma,\delta,q_0,F)$ accepts a string $w\in\Sigma^*$

$\Updownarrow$ definition

If M reads w, starting from q_0 and following transition indicated by δ on each symbol of w, M will end in a state $q\in F$.

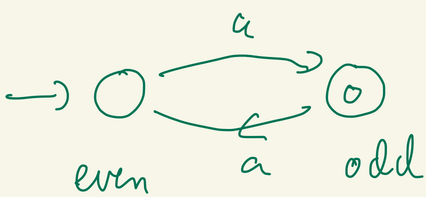
graphically:

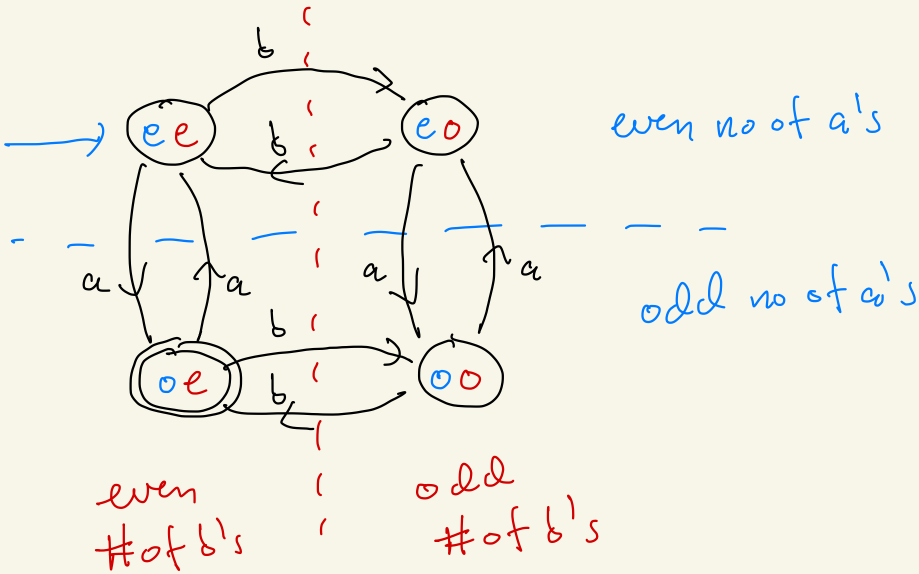


The language $L(M)$ of a DFA is the set of strings accepted by M

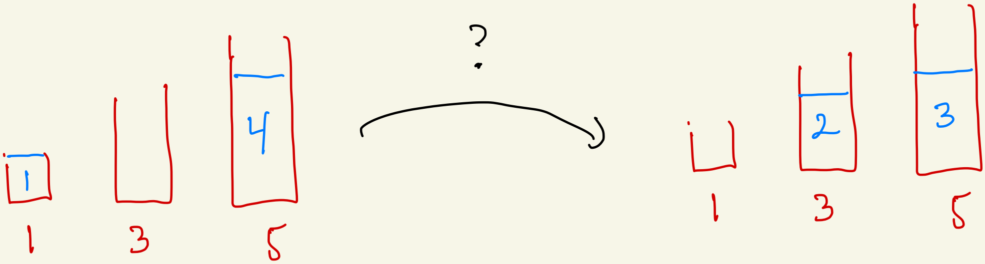
Constructing a DFA for

$$L=\{w\in\{a,b\} \mid \#_a(w) \text{ is odd and } \#_b(w) \text{ is even}\}$$

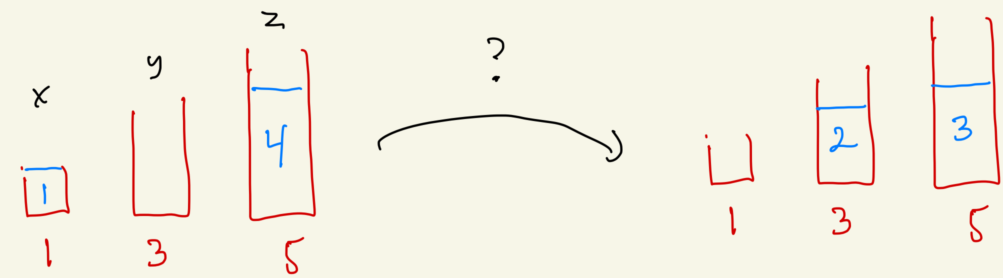




Water bottle example:



Rule: in each step either fill up a bottle completely or empty one completely

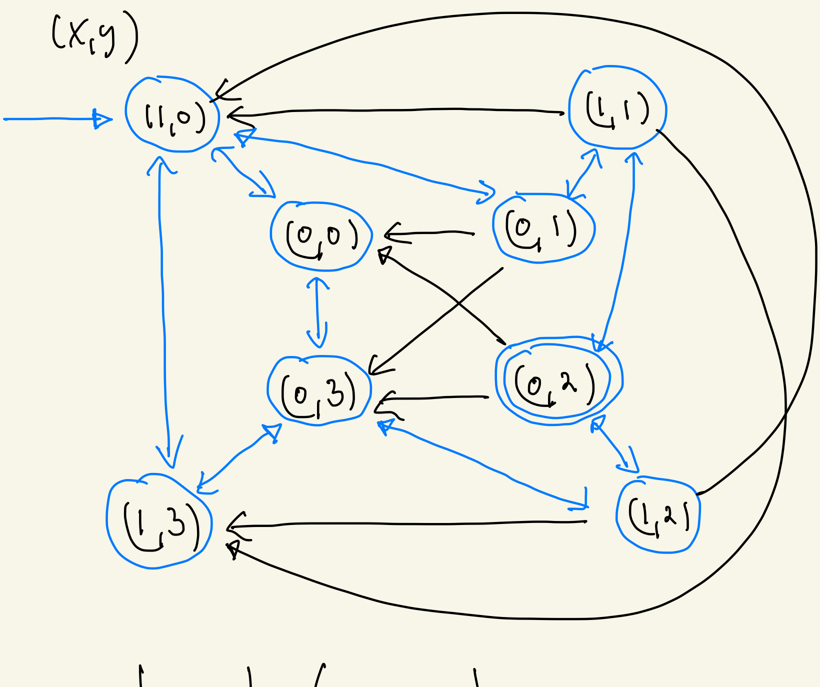


initially
$x=1,\ y=0,\ z=4$

goal
$x=0,\ y=2,\ z=3$

invariant $z = 5-x-y$ so enough to keep track of x, y

(x, y)

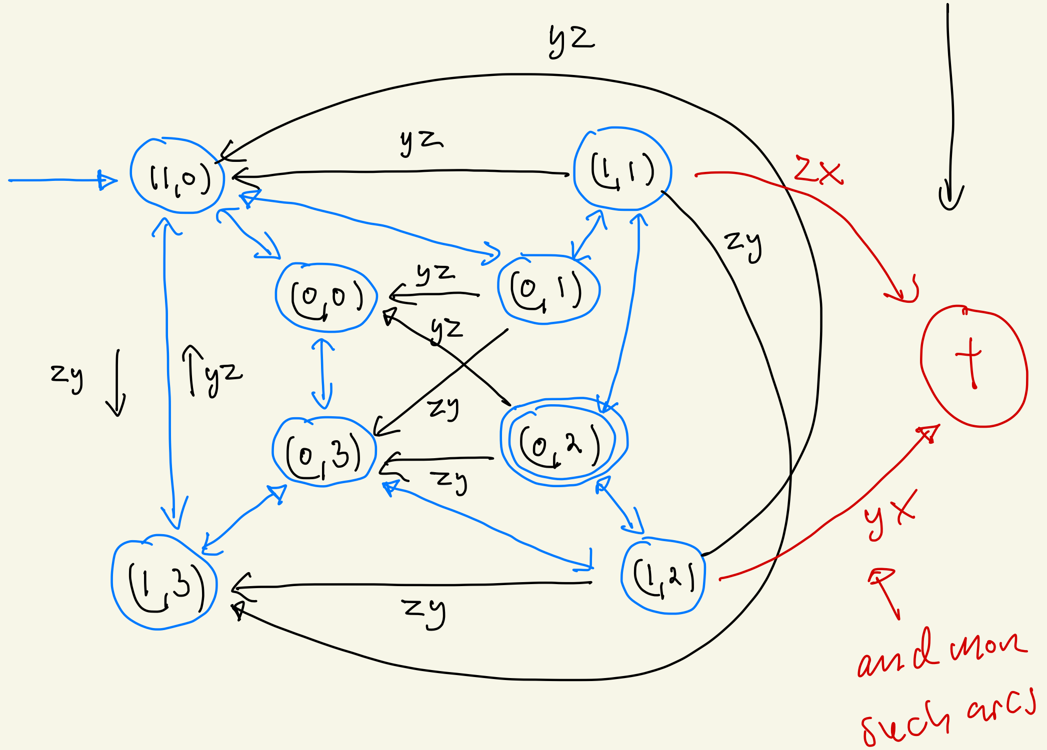


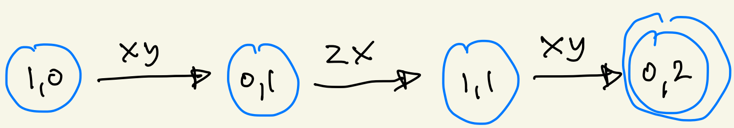
legal transitions

DFA? $\Sigma = \{xy, yx, xz, zx, yz, zy\}$

xy = water from bottle 1 to bottle 2

not all 6 transitions legal ⇒ add a dead state





Regular operations

- Union $\quad A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- concatenation $\quad A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- star $\quad A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_i \in A\}$

  ($k=0 \rightarrow \varepsilon$ the empty string)

Theorem 1.25 + 1.26 + 1.45

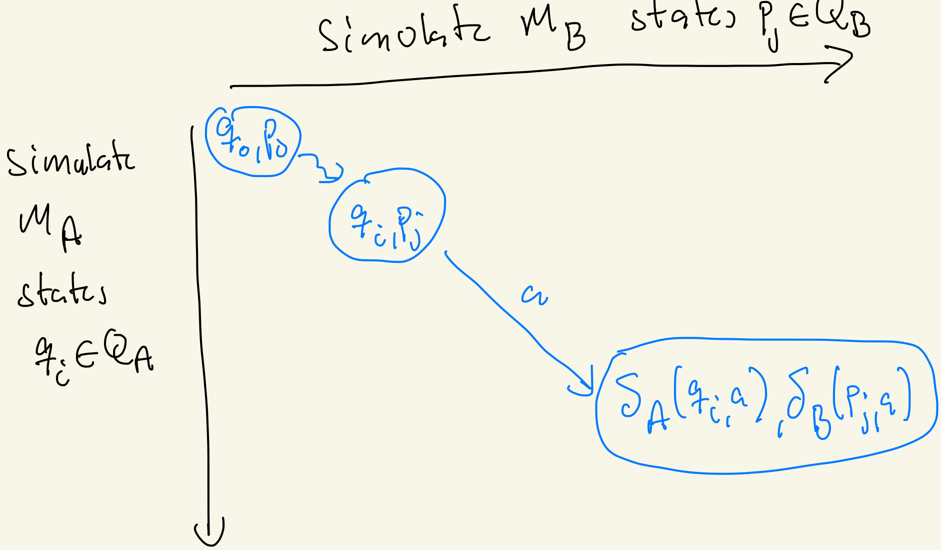
The class of regular languages is closed under each of the operations above

Union (and intersection)

idea: track positions of M_A and M_B while reading w

Here $L(M_A) = A$, $L(M_B) = B$

states of the form (q_i, p_j) where $q_i \in Q_A$, $p_j \in Q_B$



accept $w \iff$ reach state (q, p) where $q \in F_A$ or $p \in F_B$

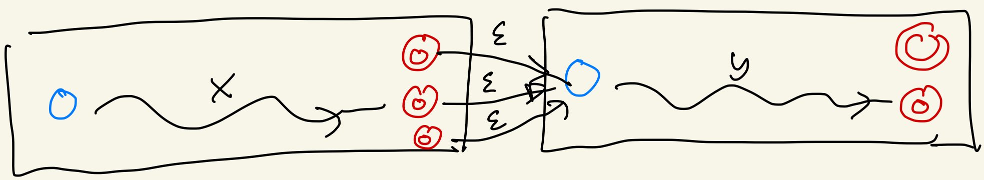
'and' then

Concatenation and star

much more complicated to show directly

easier with non-determinism

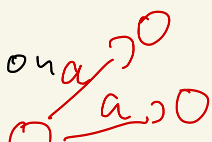
here we can 'guess'



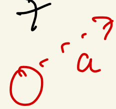
M_A M_B

<u>NFA</u> non-deterministic FA

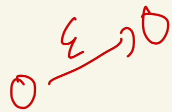
- may have many transitions on a symbol from a state



- may have no transitions from q on symbol a



- may take ε-moves

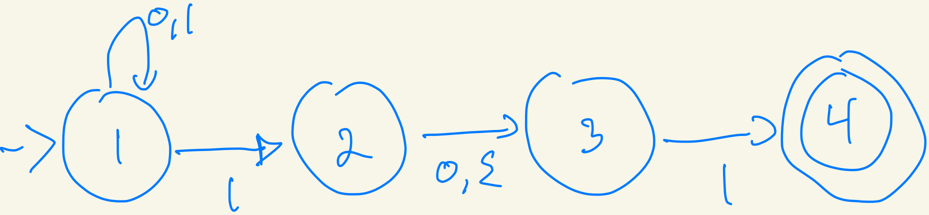


An NFA M is a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

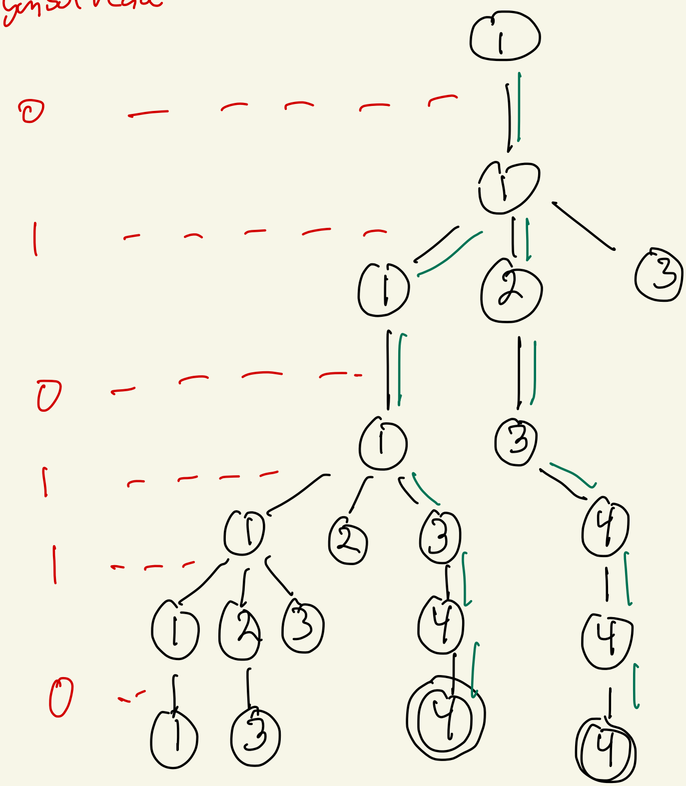
same as for DFAs, except that

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$$ ← set of all subsets of Q



symbol read

reading 010110



accepting computation

$L(M)$ is the set of strings w such that there exists a path from q_0 to some $p \in F$ which spells w.

A DFA is clearly also an NFA

more interesting:

<u>Theorem 1.39</u>

Every NFA has an equivalent DFA

proof idea: Keep track on the set of states that an NFA M can be in after reading a string $x \in \Sigma^*$

ε-closure:

a state p is ε-reachable from a state q

$\Updownarrow$

$$q \xrightarrow{\varepsilon} \bigcirc \xrightarrow{\varepsilon} \bigcirc \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} p$$

For a given subset $K \subseteq Q$ the ε-closure of K is the set

$$E(K) = \{ p \in Q \mid p \text{ is } \varepsilon\text{-reachable from some } q \in K \}$$

$$K \subseteq E(K)$$

use ε-closure to track where M could be after reading any prefix of w

Possible accepting path

$$q_0 \overset{\varepsilon}{\rightsquigarrow} q_{i_1} \xrightarrow{a_1} q_{i_2} \overset{\varepsilon}{\rightsquigarrow} q_{i_3} \xrightarrow{a_2} q_{i_4} \overset{\varepsilon}{\rightsquigarrow} \qquad \overset{\varepsilon}{\rightsquigarrow} q \in F$$

Given M we can construct a DFA M' with at most 2^{Q_M} states such that

M accepts $w \iff M'$ accepts w

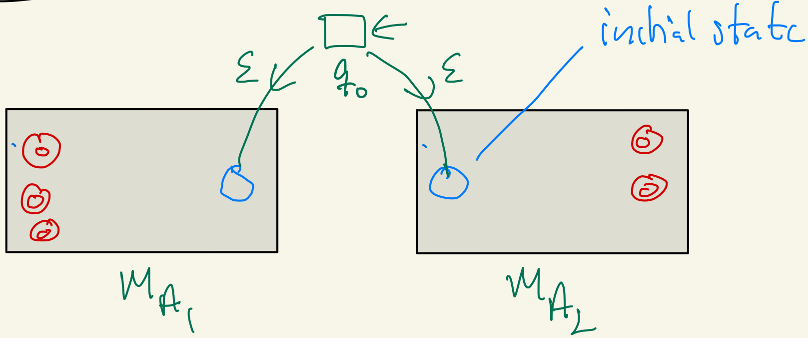
Important: This construction may take exponential time!

Hence it is not so useful algorithmically

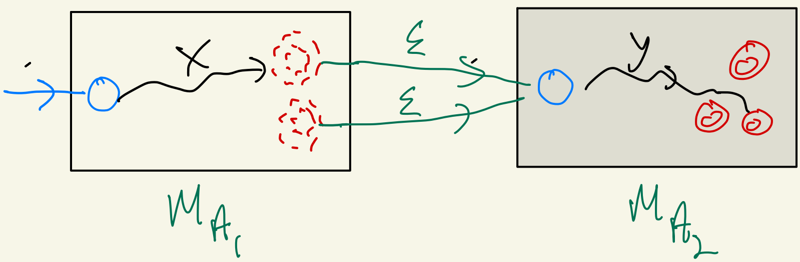
But very useful for closure properties

see next page.

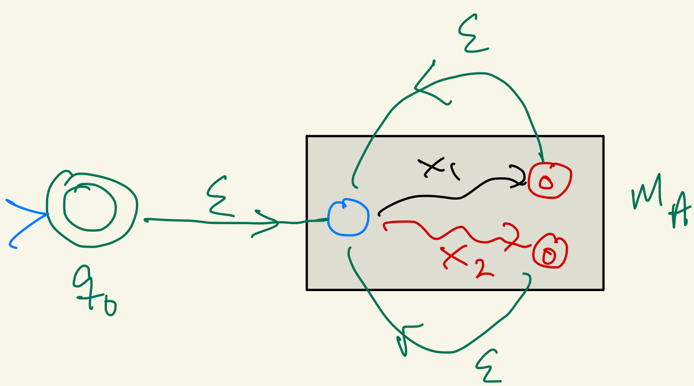
Thm 1.45 A_1, A_2 regular $\Rightarrow$ $A_1 \circ A_2$ regular



Thm 1.47 A_1, A_2 regular $\Rightarrow$ $A_1 \cdot A_2$ regular



Thm 1.49 A regular $\Rightarrow$ A^* regular



Theorem

Let L and L' be regular languages

Then each of the following are also regular languages

1. $L \cup L'$
2. $\overline{L} = \{ w \in \Sigma^* \mid w \notin L \}$
3. $L \cap L' = \overline{\overline{L} \cup \overline{L'}}$
4. $L - L' = L \cap \overline{L'}$
5. $L \cdot L'$
6. L^*

1.3 Regular expressions

The value of a regular expression over Σ is a subset of Σ^* $\quad a^*(a \cup b)b^* \circ aba$

Ex: $(0 \cup 1)^* = \{0,1\}^*$ set of all binary strings

- $\Sigma^* 1$ = all strings over Σ which end in 1

- Precedence order of regular operations

$$* > \circ > \cup$$

- Except when ( ) change this

Definition 1.52

R is a regular expression over Σ if R is

1. a for some $a \in \Sigma$,
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ when R_1 and R_2 are reg. expr over Σ
5. $(R_1 \circ R_2)$ — — — — — —
6. (R_1^*) when R_1 is a reg. expr over Σ

Notation

$$R^+ = R \circ R^* \iff R^* = R^+ \cup \{\varepsilon\}$$

$L(R)$ = language generated by R

Theorem 1.54

L is regular $\iff$ $L = L(R)$ for some regular expression

Corollary

Let Σ be an alphabet and $L \subseteq \Sigma^*$

Then the followings are equivalent

(1) $L = L(M)$ for some DFA M

(2) $L = L(M')$ for some NFA M'

(3) $L = L(R)$ for some regular expression R