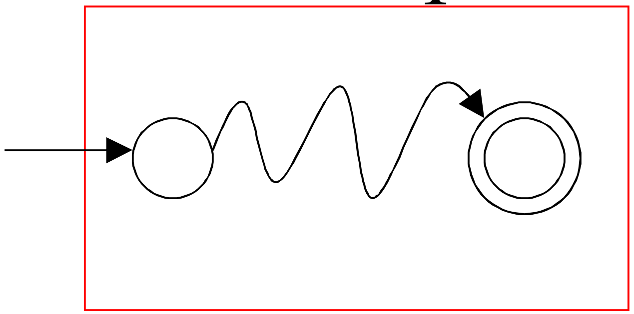
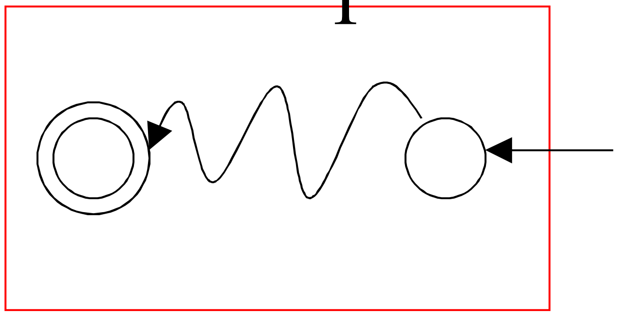
Reverse

DFA for L_1^R

L_1 M_1



M_1'



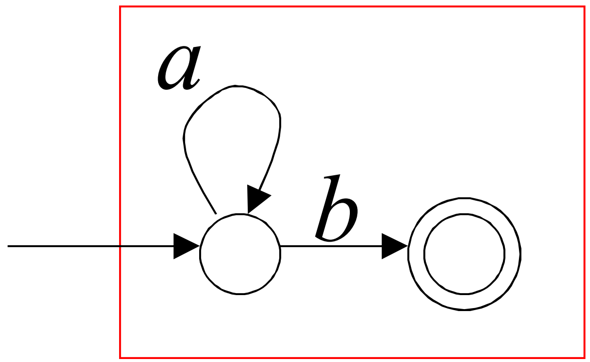
1. Reverse all transitions

2. Make initial state accepting state
   and vice versa

Example

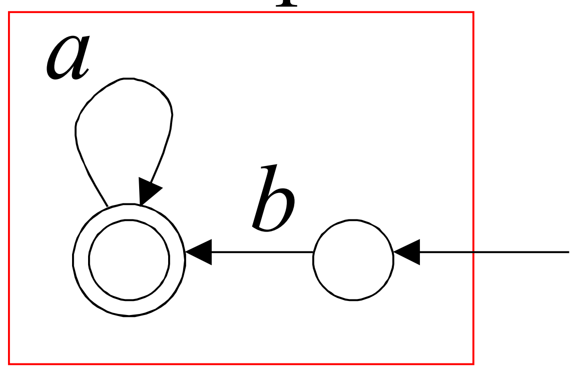
M_1

$$L_1 = \{a^n b\}$$



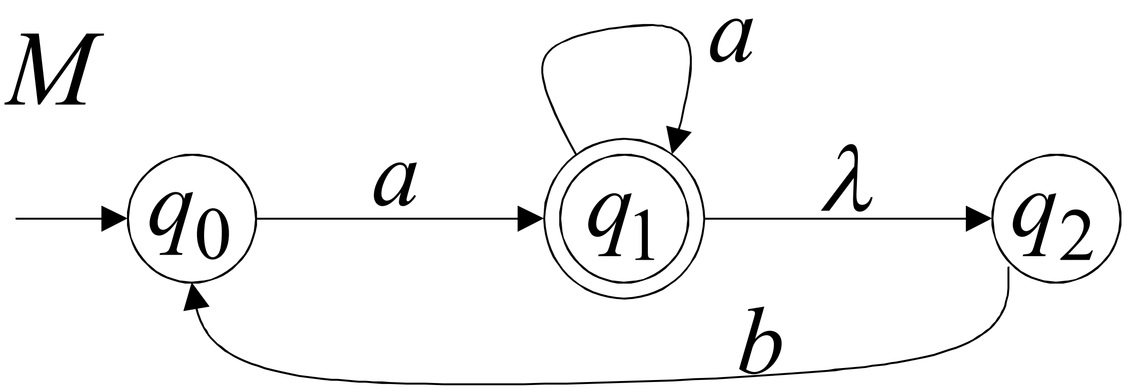
M_1'

$$L_1^R = \{ba^n\}$$

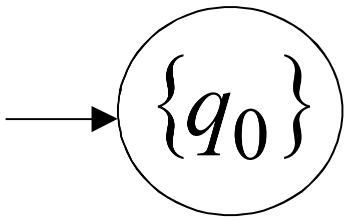


Convert NFA to FA(DFA)

NFA M

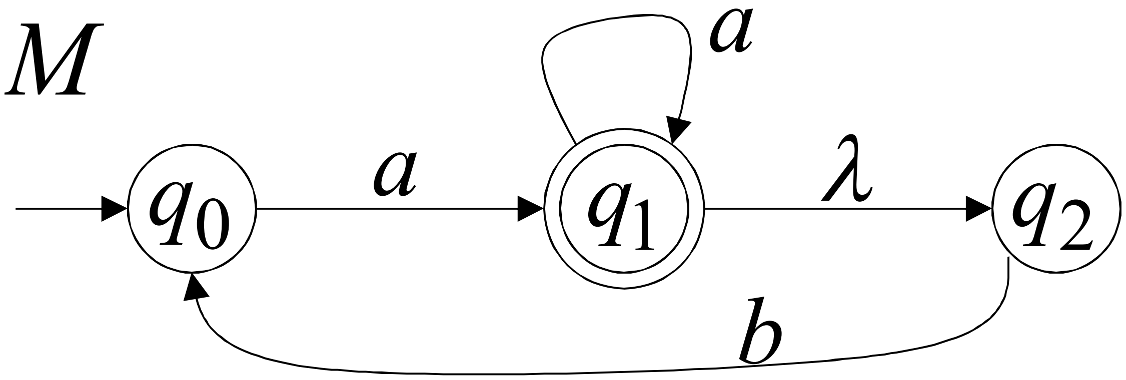


FA M'

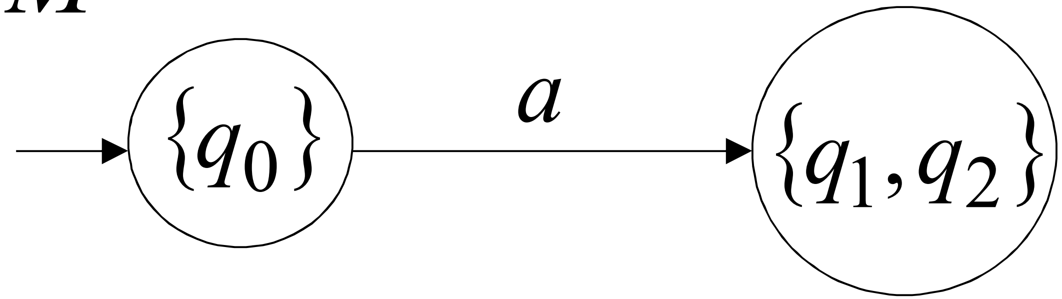


Convert NFA to FA

NFA M

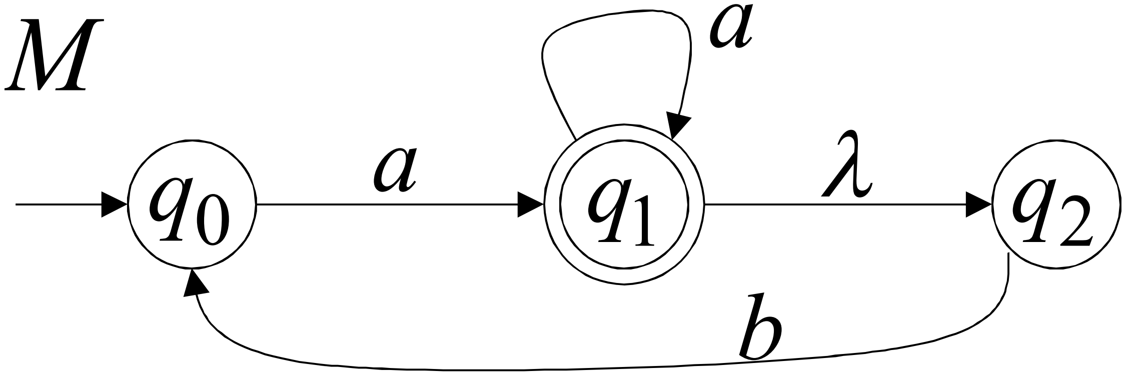


FA M'

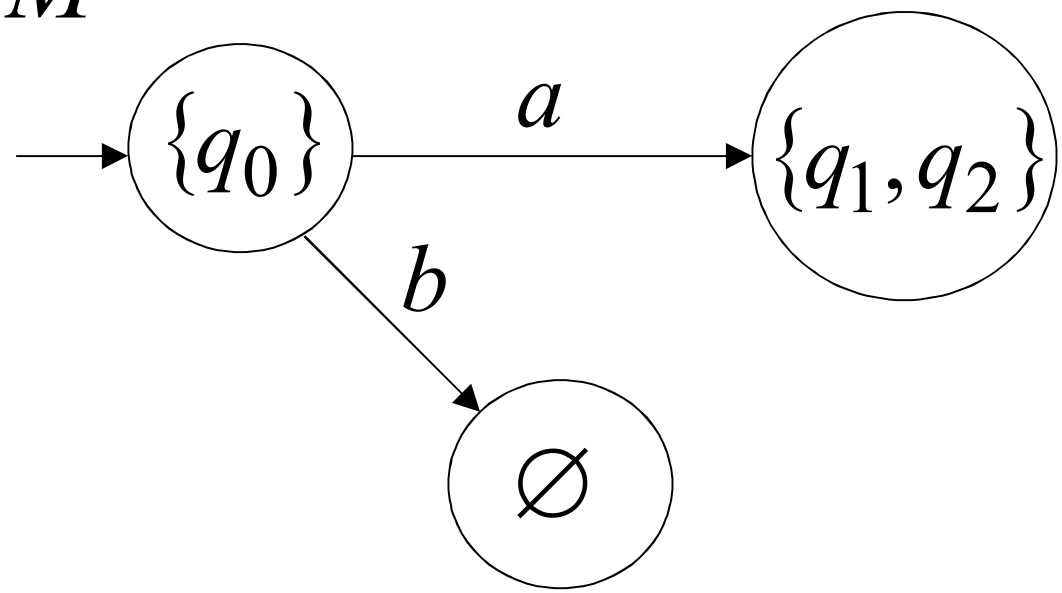


Convert NFA to FA

NFA M

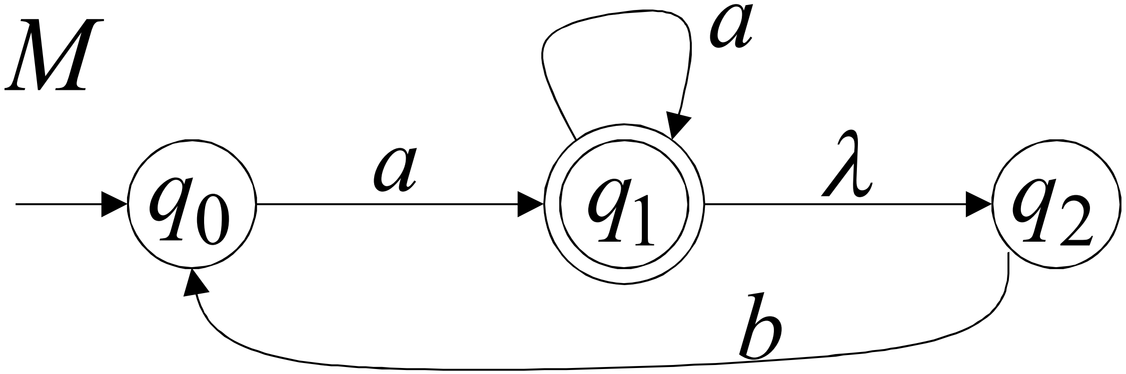


FA M'

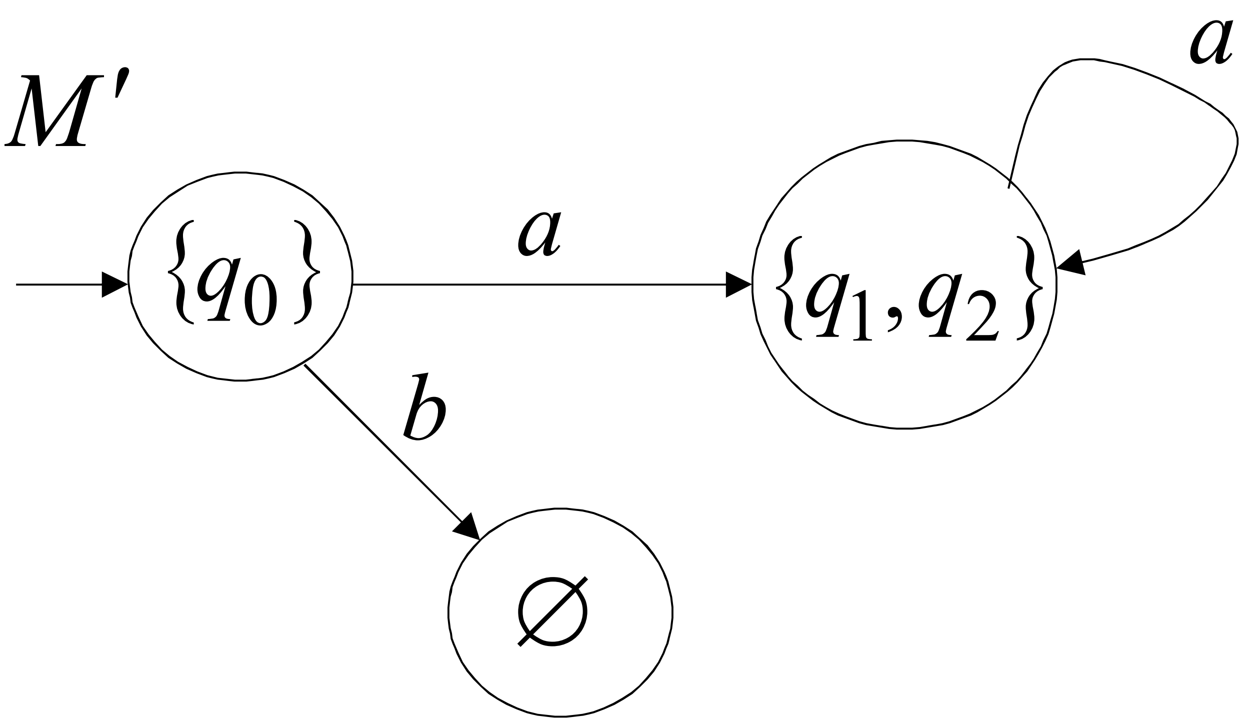


Convert NFA to FA

NFA M

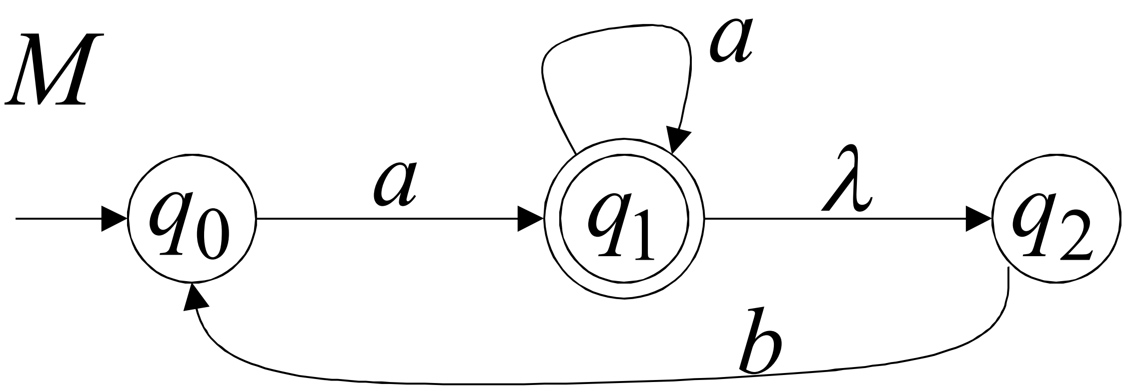


FA M'

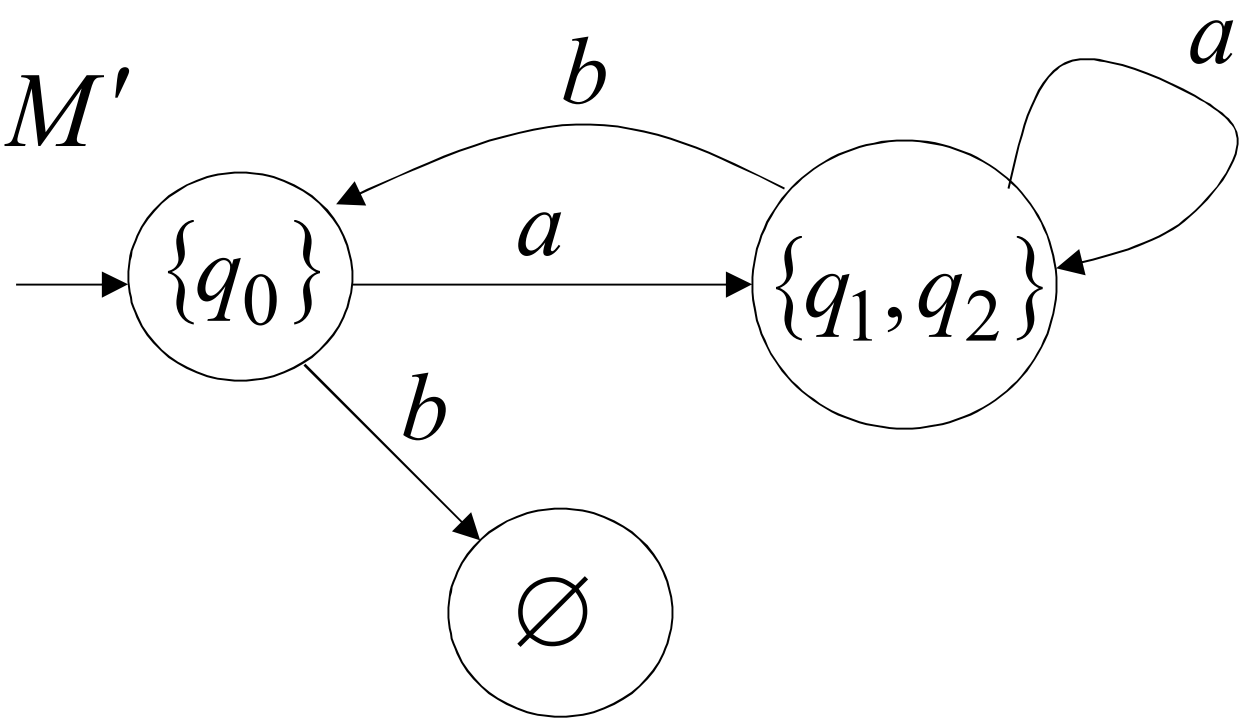


Convert NFA to FA

NFA M

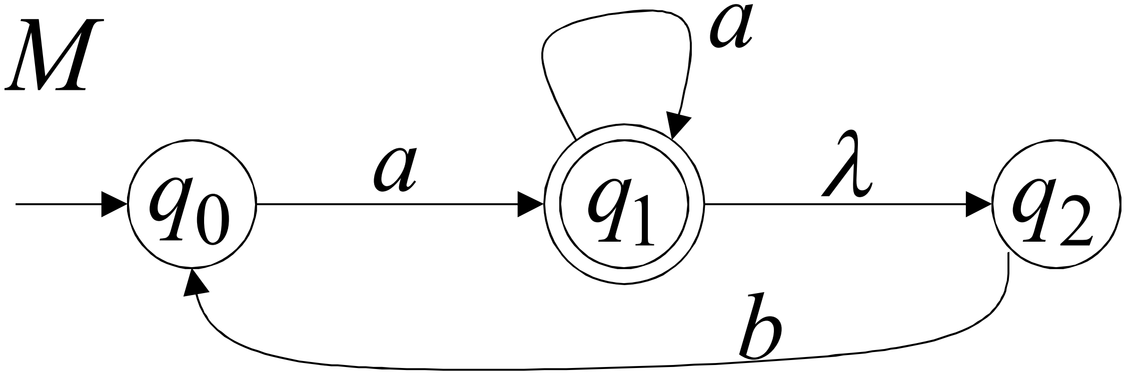


FA M'

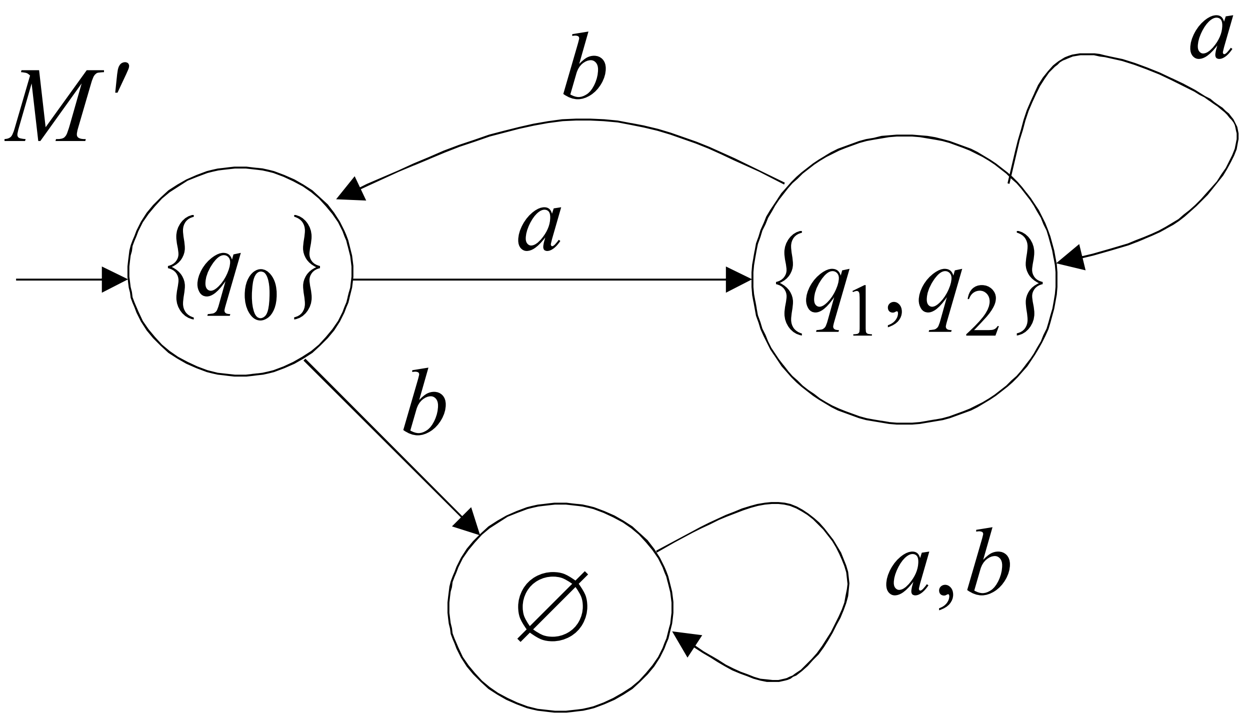


Convert NFA to FA

NFA M

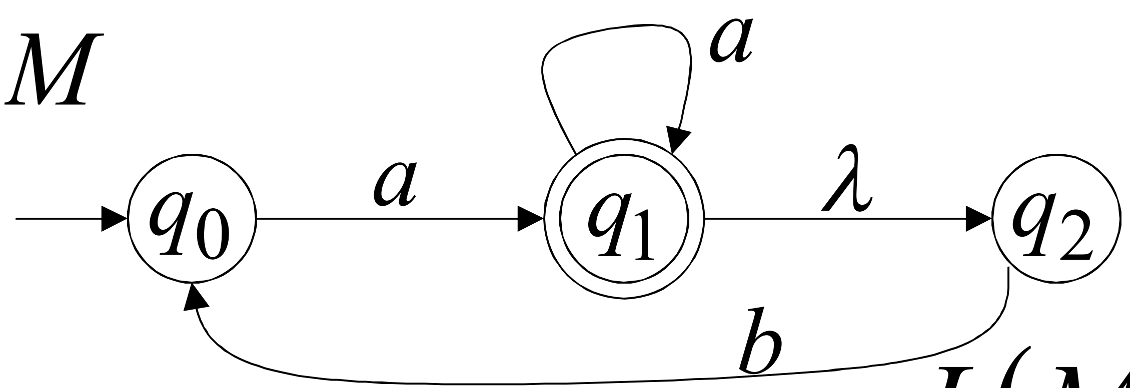


FA M'



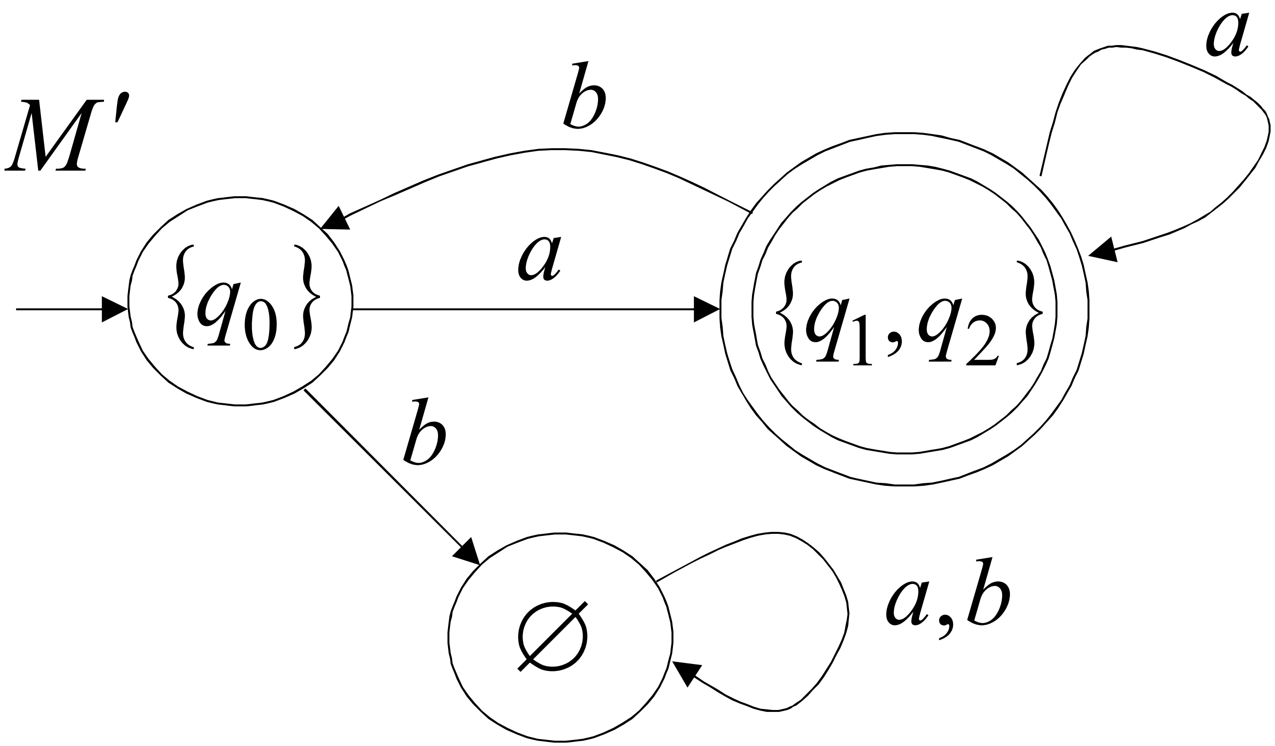
Convert NFA to FA

NFA M



$$L(M) = L(M')$$

FA M'



NFA to FA Conversion

We are given an NFA M

We want to convert it
to an equivalent FA M'

With $L(M) = L(M')$

What we need to construct

Finite Automaton (FA)

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

Q : set of states

Σ : input alphabet

δ : transition function

q_0 : initial state

F : set of accepting states

If the NFA has states

$$q_0, q_1, q_2, \ldots$$

the FA has states in the power set

$$\emptyset, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots.$$

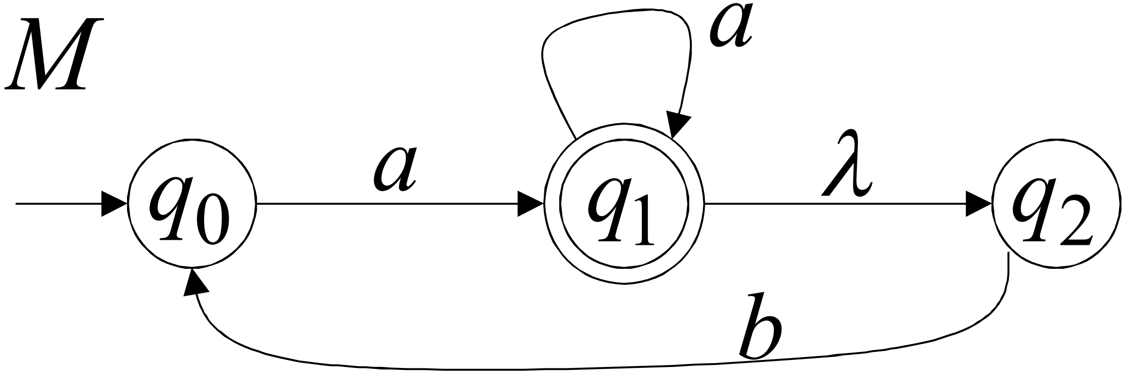
Procedure NFA to FA

1. Initial state of NFA: q_0

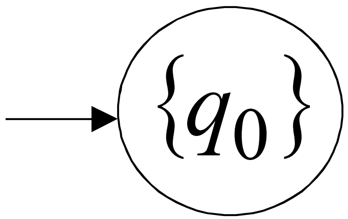
Initial state of FA: $\{q_0\}$

Example

NFA M



FA M'



Procedure NFA to FA

2. For every FA's state $\{q_i, q_j, \ldots, q_m\}$

Compute in the NFA

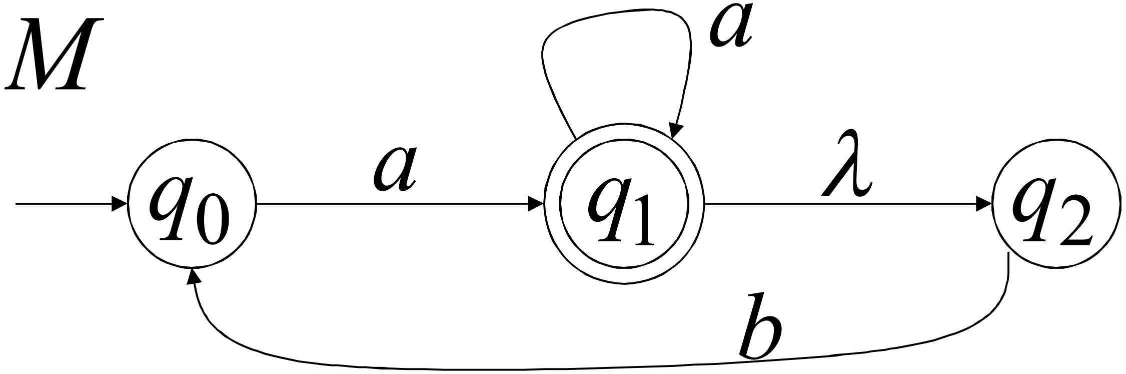
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to FA

$$\delta(\{q_i, q_j, \ldots, q_m\}, \ a) = \{q'_i, q'_j, \ldots, q'_m\}$$

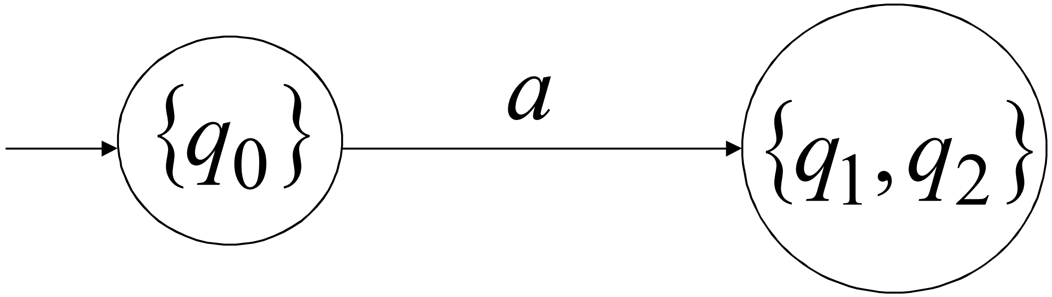
Example

NFA M



$$\delta^*(q_0, a) = \{q_1, q_2\}$$

FA M'



$$\delta(\{q_0\}, a) = \{q_1, q_2\}$$

Procedure NFA to FA

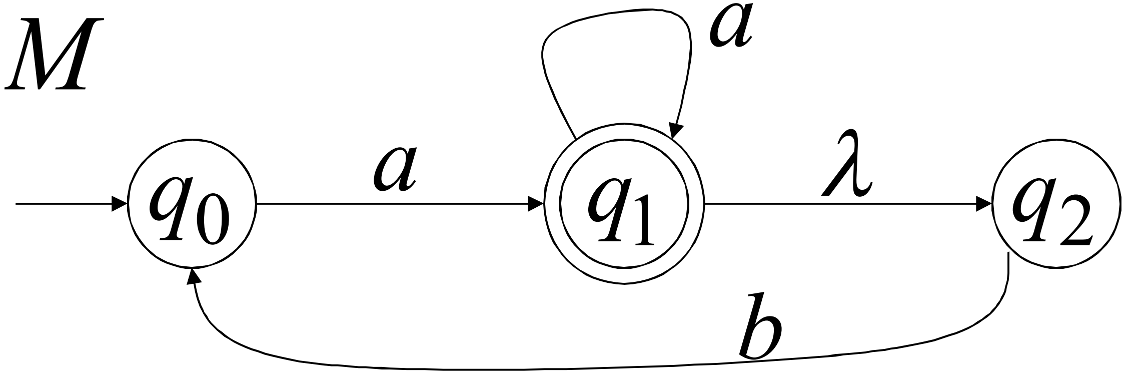
Repeat Step 2 for all letters in alphabet,

until

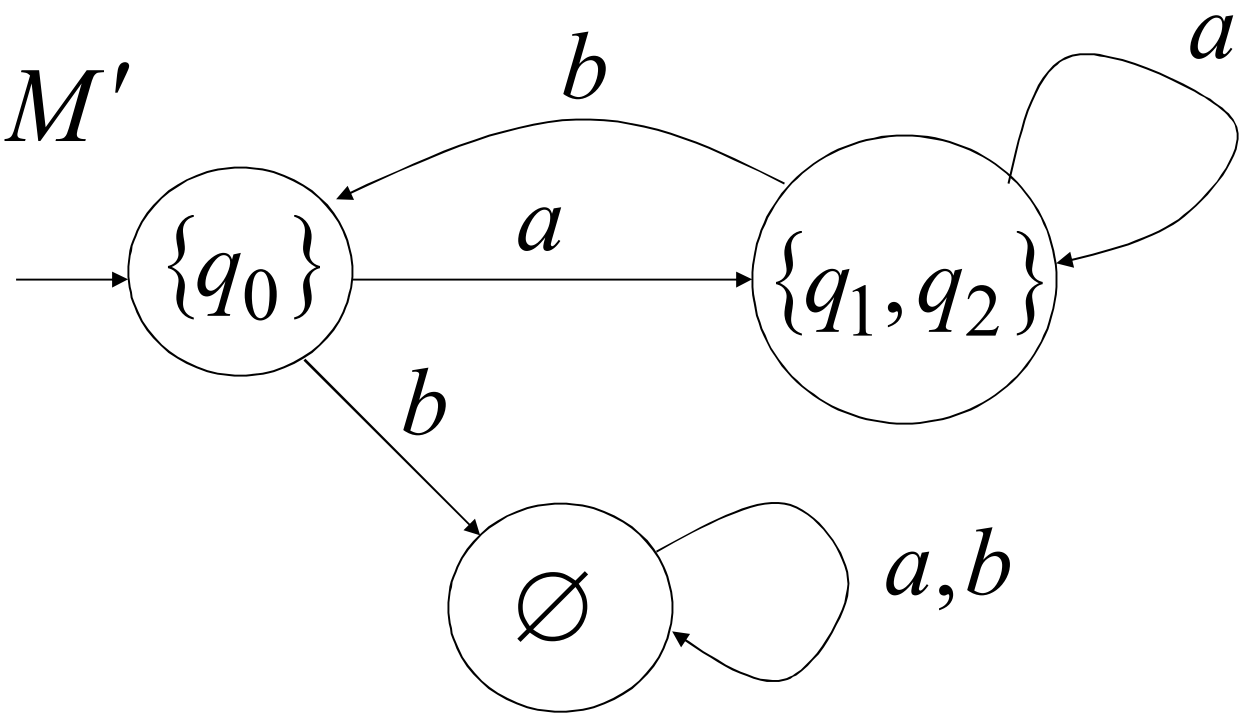
no more transitions can be added.

Example

NFA M



FA M'



Done?

Procedure NFA to FA

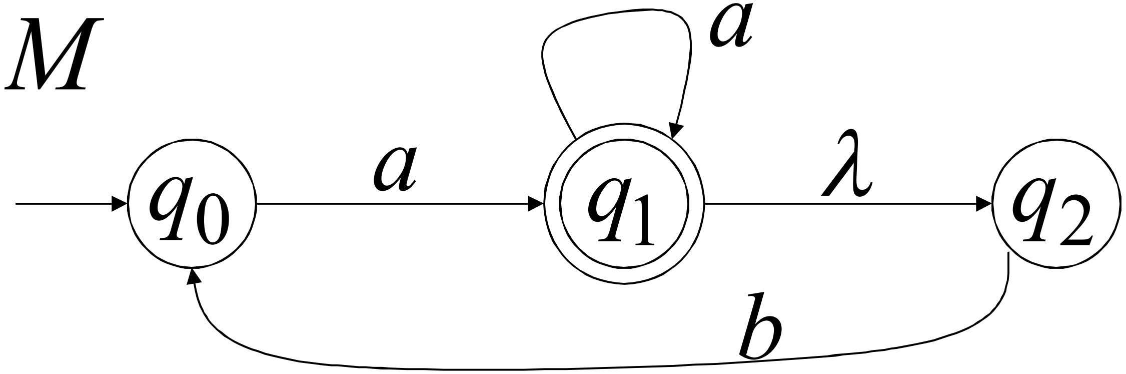
3. For any FA state $\{q_i, q_j, \ldots, q_m\}$

If q_j is accepting state in NFA

Then, $\{q_i, q_j, \ldots, q_m\}$
is accepting state in FA

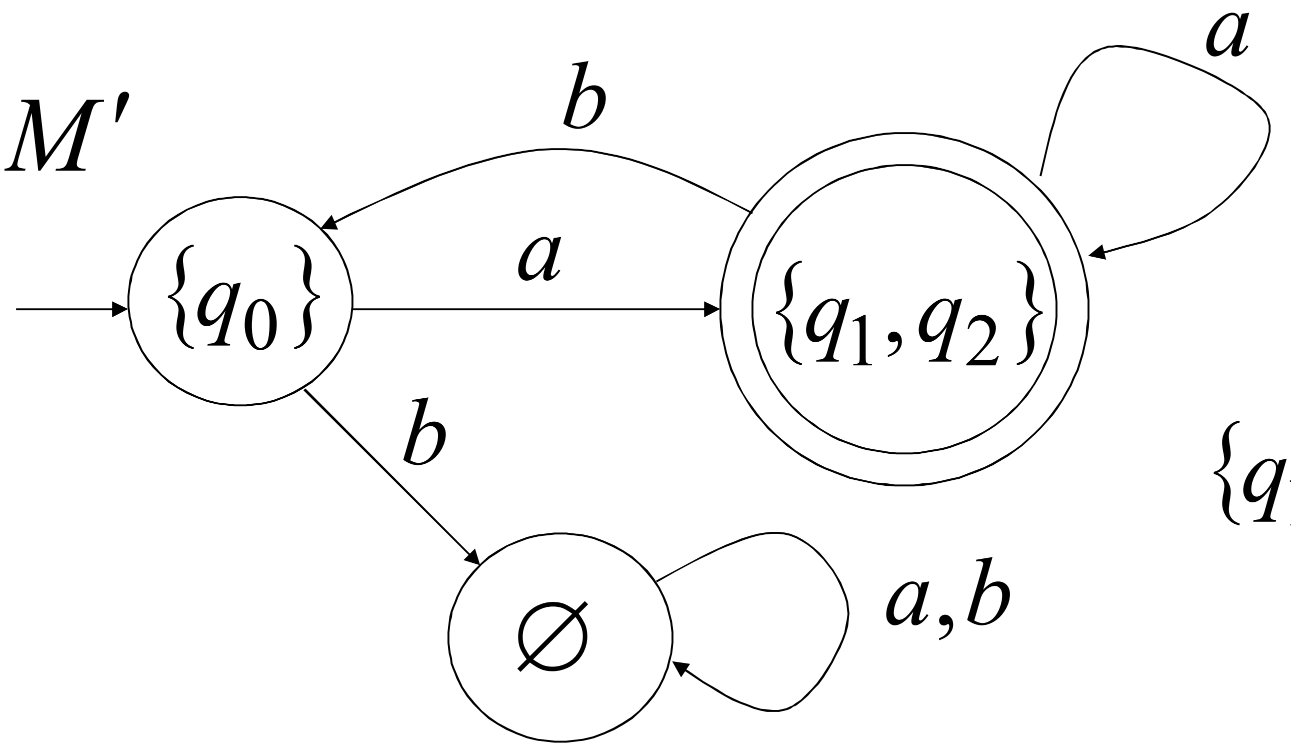
Example

NFA M



$q_1 \in F$

FA M'



$\{q_1, q_2\} \in F'$

Regular Expressions

- Another type of **language-defining** notation
- Algebraic description
- Declarative way to express desired strings
- RE: Input language for many string processing systems
- $01^* + 10^*$
- Lang={x: x is 0y or x is 1z, y=1* and z=0*}

Operators of REs

- Given two languages,

  $L=\{001, 10, 111\}$ and $M= \{\varepsilon, 001\}$
- union: $L \cup M$
- concatenation: LM or $L.M$
- Closure (star or Kleene closure) of L: L^*

  $L=\{0, 11\}$ then L^0, $L^1=L$, L^2

Building REs

- Algebra
  - Some basic expr; usually consts &/ vars
  - More exprs by applying operators to basic expr
  - Grouping, like paranthesis ( )
- REs : same
  - Basis: 3 parts
    - Constants; empty string and empty set
    - a any symbol, then a is re, {a}
    - Var L representing any lang
  - Induction: 4 parts (If E, F: REs)
    - E + F is RE, denoting the union of L(E) and L(F)
    - EF is RE, denoting the concatenation of L(E) and L(F)
    - E^* is RE, denoting the closure of L(E)
    - (E) is RE, denoting the same language as E

Examples

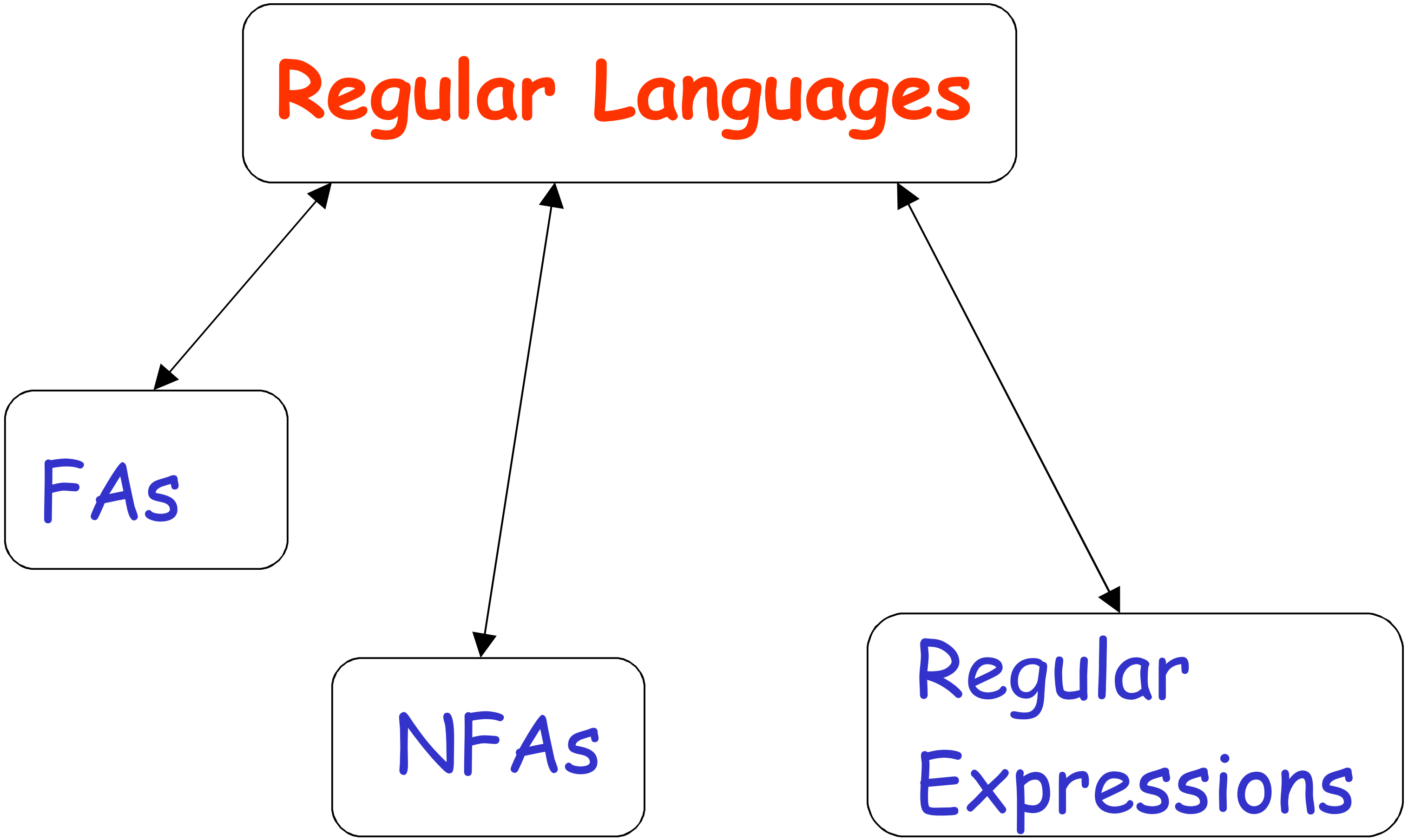
- Strings of alternating 0s and 1s
  - 0, 1, 01, (01)*
  - 10, (10)*
  - 1(01)*
  - 0(10)*

RE1: (01)* + (10)* + 1(01)* + 0(10)*

RE2: (ε + 1) (01)* (ε + 0)

- Precedence (of RE operators)
  - Star, *
  - Concatenation or dot, .
  - Union, U
- $01^* + 1 \equiv (0\ (1^*)\ ) + 1$
- String 1 or all strings consisting of a 0 followed by any number of 1's (including none)

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

Elementary Questions

about

Regular Languages

Membership Question

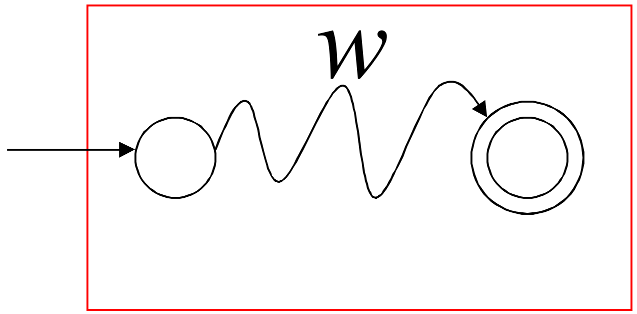
Question: Given regular language L
and string w
how can we check if $w \in L$?

Membership Question

Question: Given regular language L
and string w
how can we check if $w \in L$?

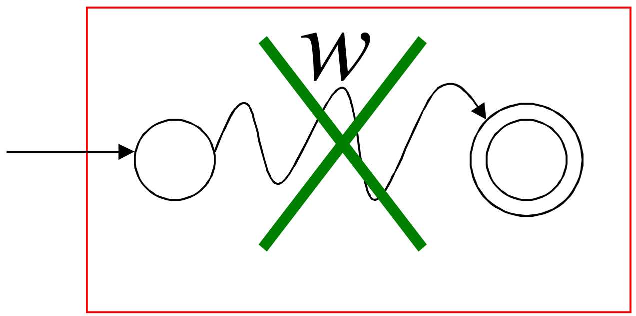
Answer: Take the DFA that accepts L
and check if w is accepted

DFA



$w \in L$

DFA



$w \notin L$

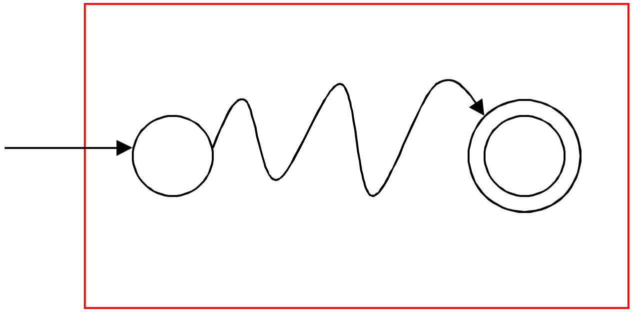
Question: Given regular language L
how can we check
if L is empty: $(L = \varnothing)$?

Question: Given regular language L how can we check if L is empty: $(L = \varnothing)$?

Answer: Take the DFA that accepts L

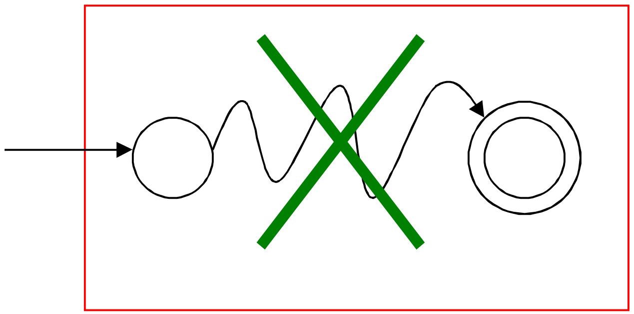
Check if there is any path from the initial state to a final state

DFA



$$L \neq \emptyset$$

DFA



$$L = \emptyset$$

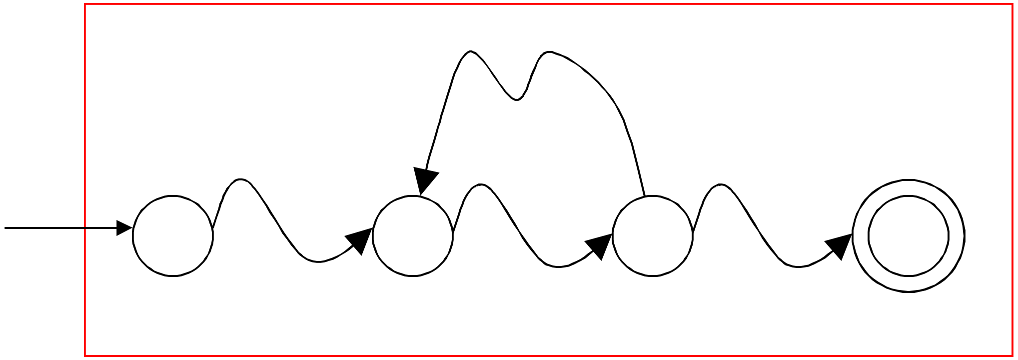
Question: Given regular language L
how can we check
if L is finite?

Question: Given regular language L
how can we check
if L is finite?

Answer: Take the DFA that accepts L

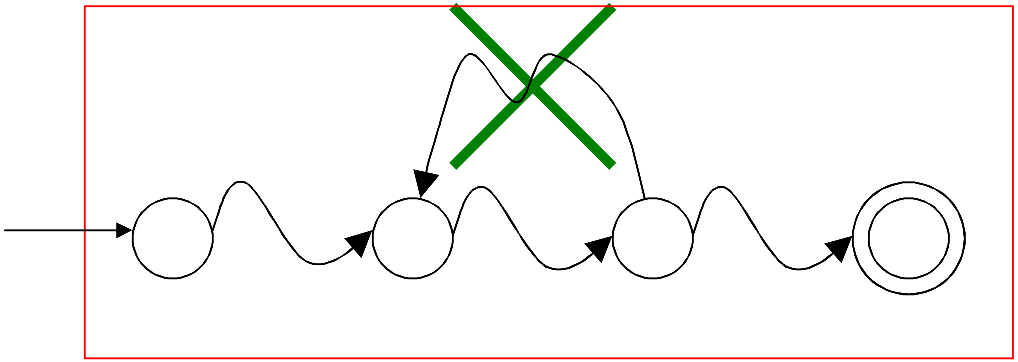
Check if there is a walk with cycle
from the initial state to a final state

DFA



L is infinite

DFA



L is finite

Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$?

Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$?

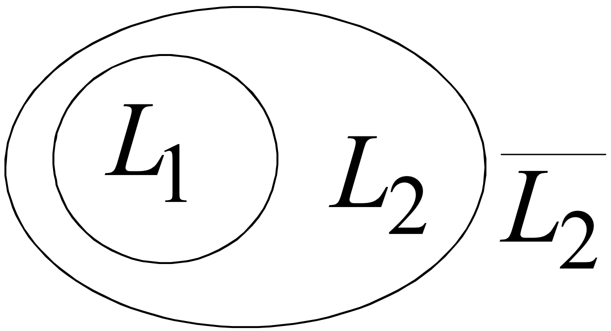
Answer: Find if $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

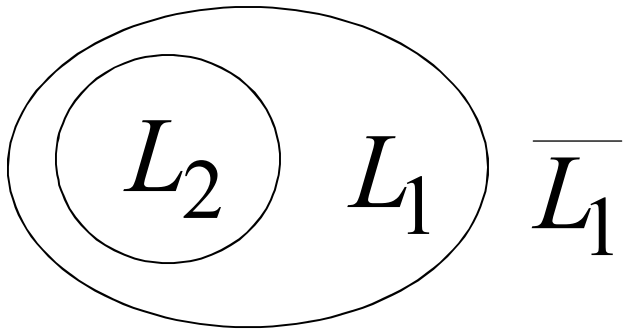
$$L_1 \cap \overline{L_2} = \varnothing$$

and

$$\overline{L_1} \cap L_2 = \varnothing$$



$$L_1 \subseteq L_2$$



$$L_2 \subseteq L_1$$

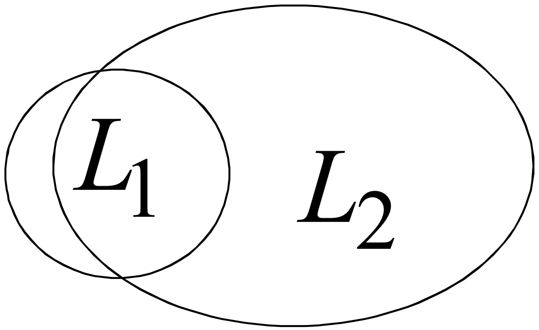
$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \varnothing$$

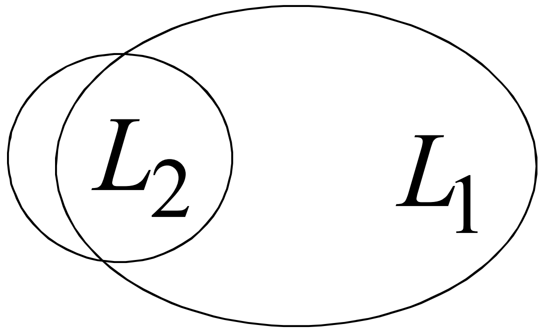
$$L_1 \cap \overline{L_2} \neq \varnothing$$

or

$$\overline{L_1} \cap L_2 \neq \varnothing$$



$$L_1 \not\subset L_2$$



$$L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

Regular Langs

- Languages accepted by DFA’s
- Languages accepted by NFA’s
- Languages accepted by λ-NFA
- Languages defined by REs

- **Note:**
  - Not every language is a regular language.