Formal Languages

NFAs Accept the Regular Languages

Equivalence of Machines

Definition:

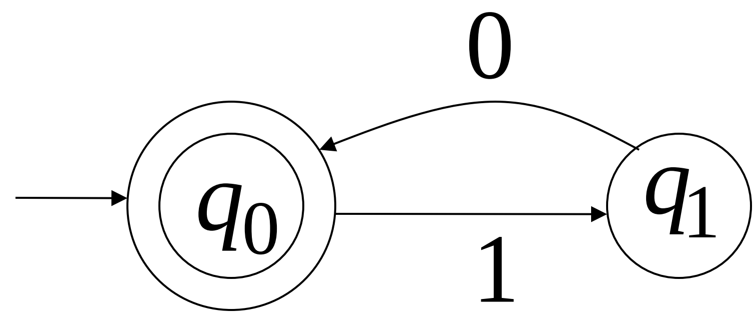
Machine M_1 is equivalent to machine M_2

if $L(M_1) = L(M_2)$

Example of equivalent machines

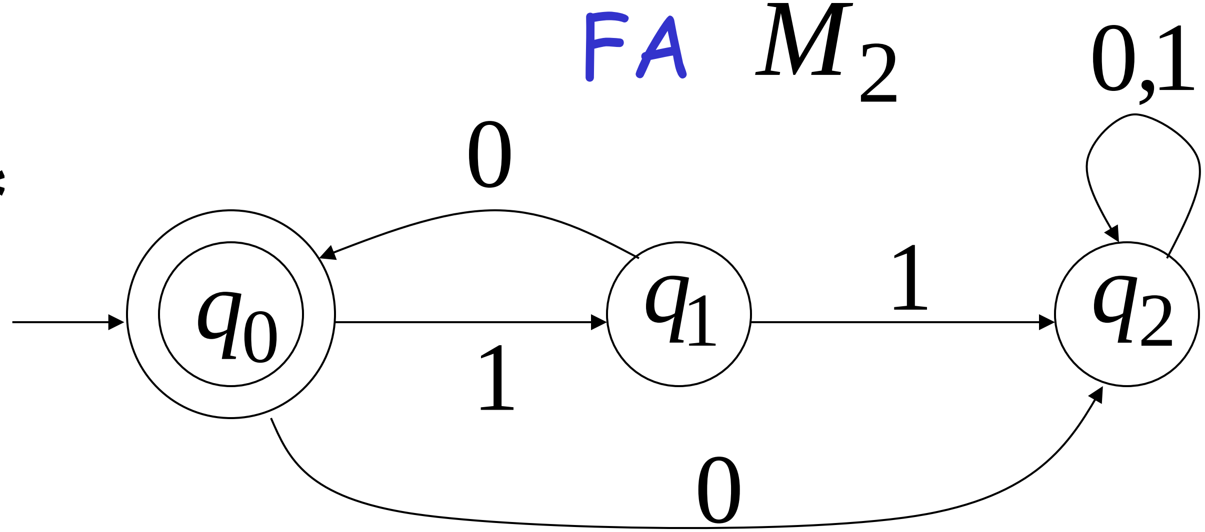
NFA M_1

$$L(M_1) = \{10\}^*$$



FA M_2

$$L(M_2) = \{10\}^*$$



We will prove:

$$\left\{\begin{array}{l}\text{Languages}\\\text{accepted}\\\text{by NFAs}\end{array}\right\} = \left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\}$$

Languages
accepted
by FAs

NFAs and FAs have the same computation power

We will show:

$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Proof-Step 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Proof?

Proof-Step 1

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Proof: Every FA is trivially an NFA

$$\Downarrow$$

Any language L accepted by a FA is also accepted by an NFA

Proof-Step 2

$$\left\{\begin{array}{l}\text{Languages}\\\text{accepted}\\\text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\}$$

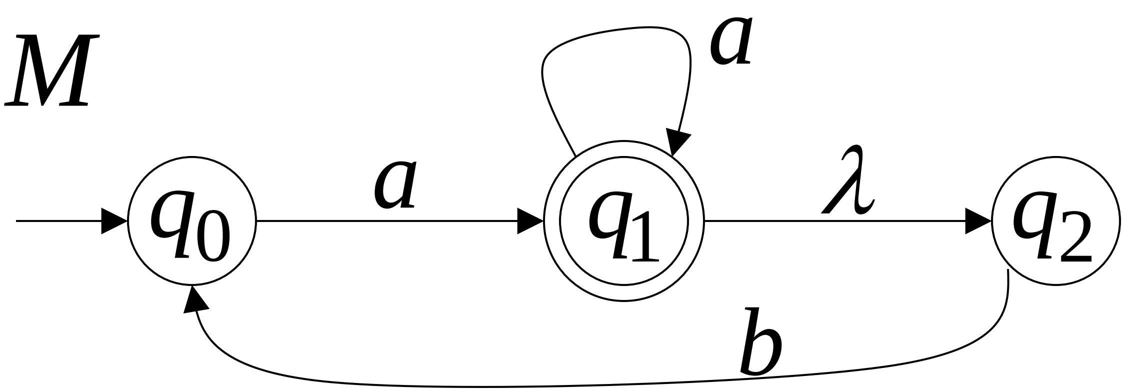
Proof: Any NFA can be converted to an equivalent FA

$\Downarrow$

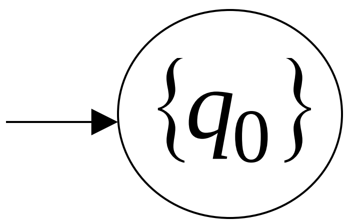
Any language L accepted by an NFA is also accepted by a FA

Convert NFA to FA

NFA M

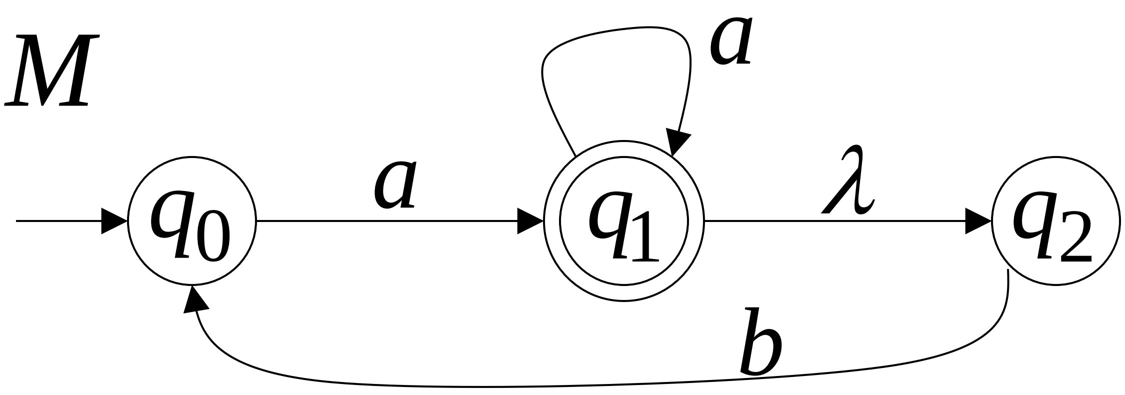


FA M'

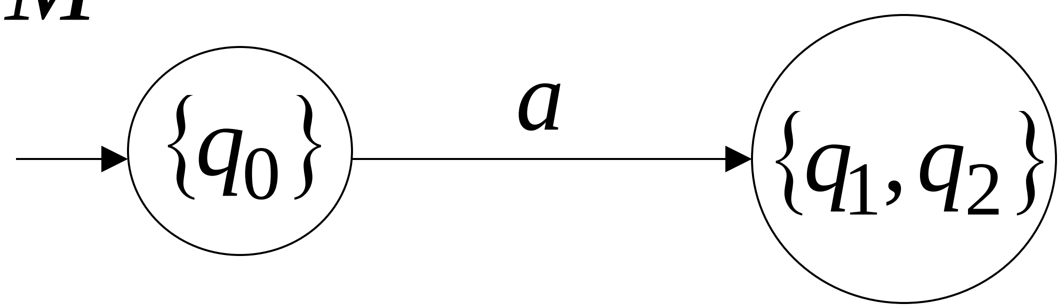


Convert NFA to FA

NFA M

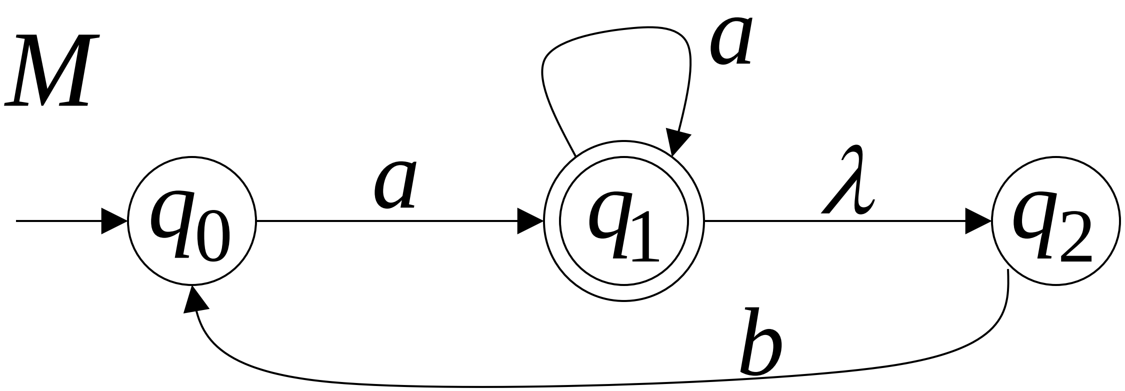


FA M'

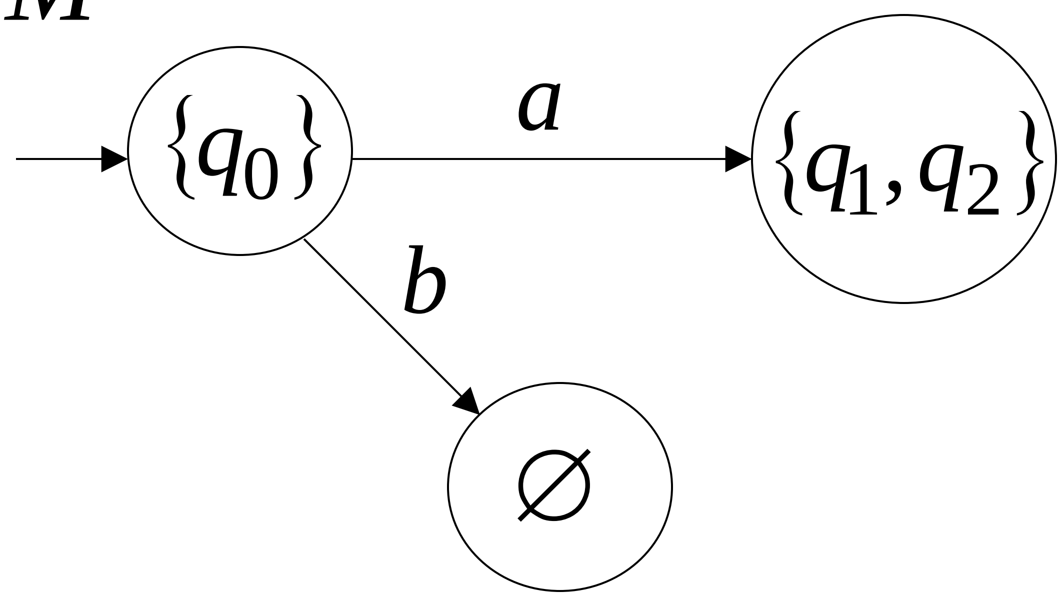


Convert NFA to FA

NFA M

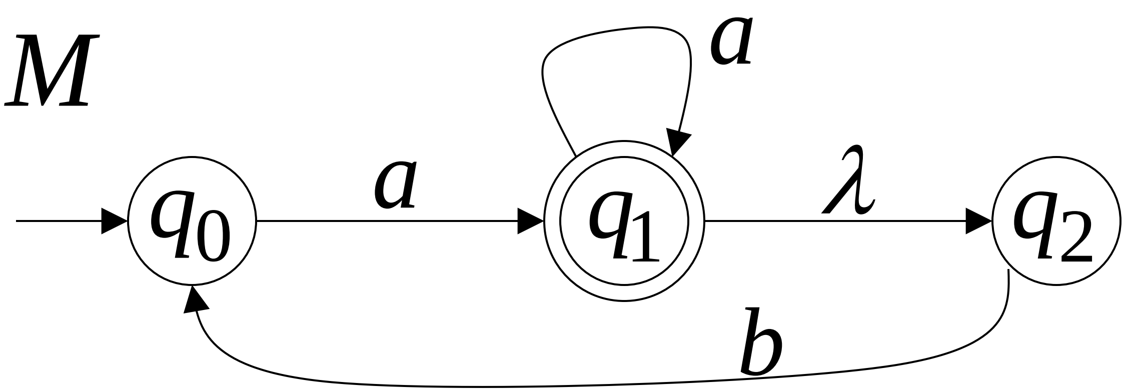


FA M'

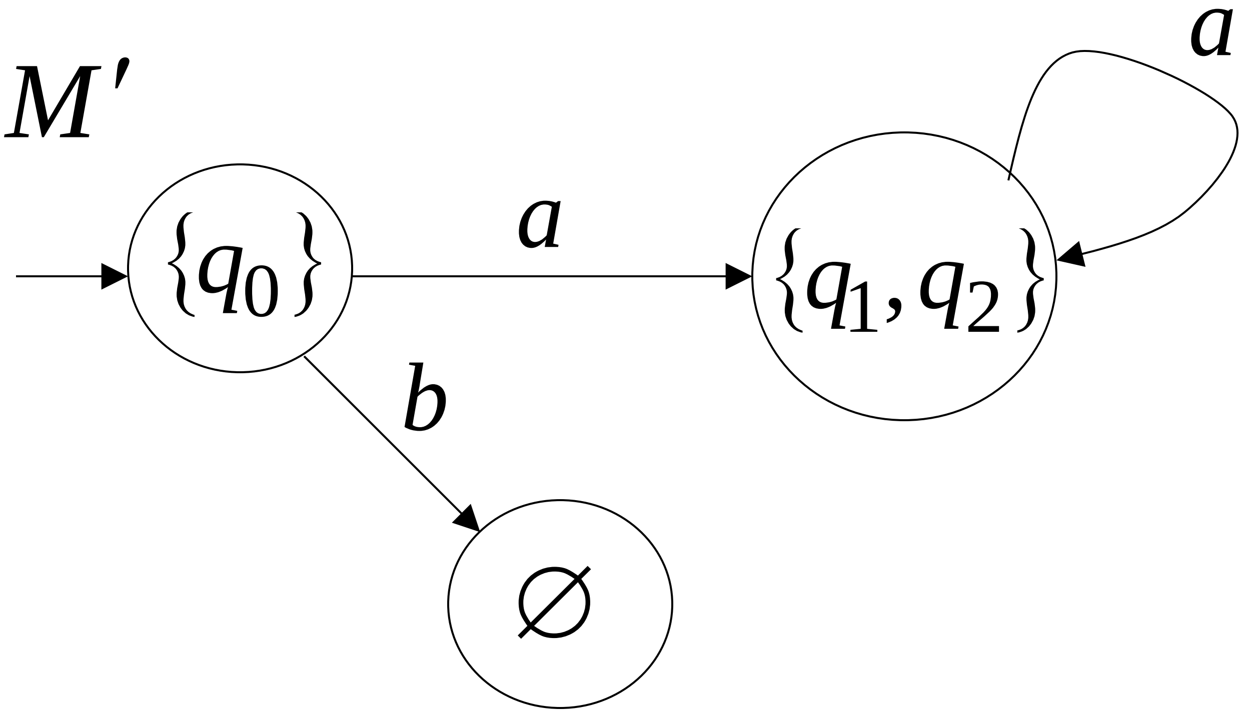


Convert NFA to FA

NFA M

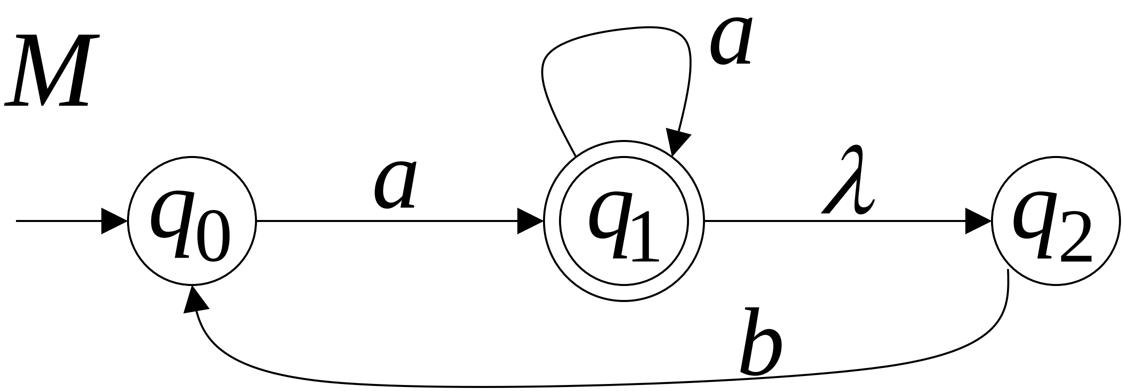


FA M'

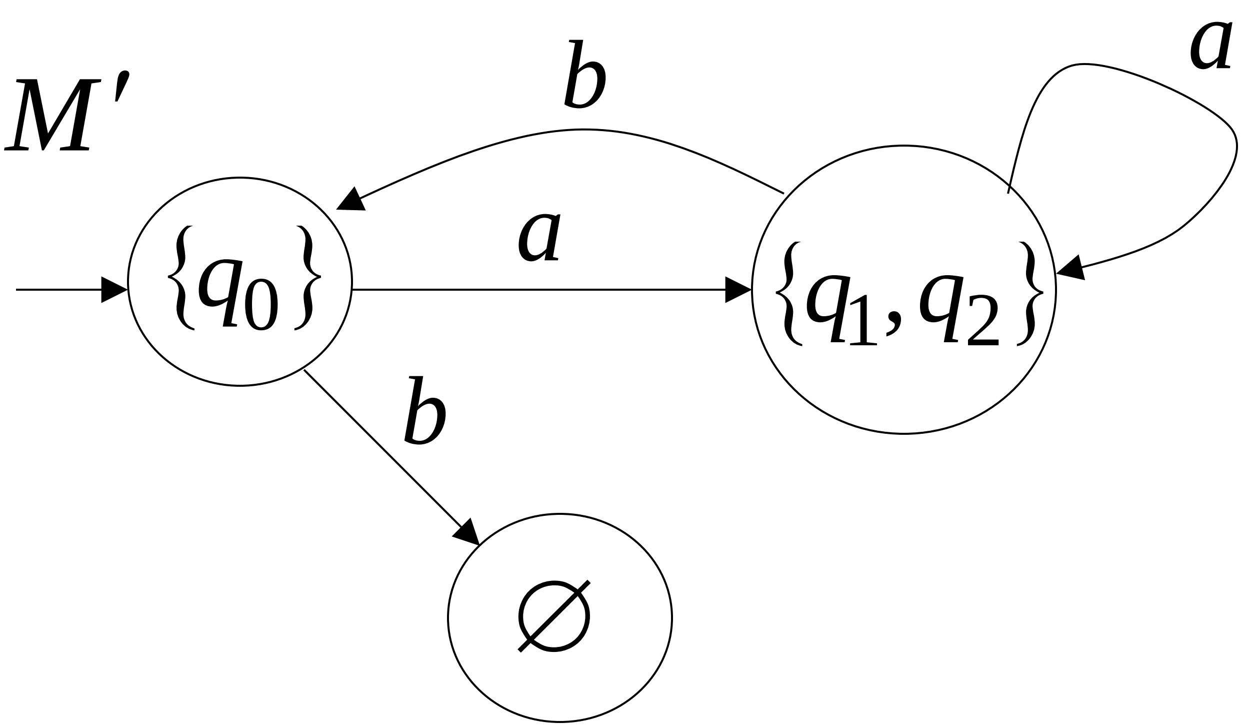


Convert NFA to FA

NFA M

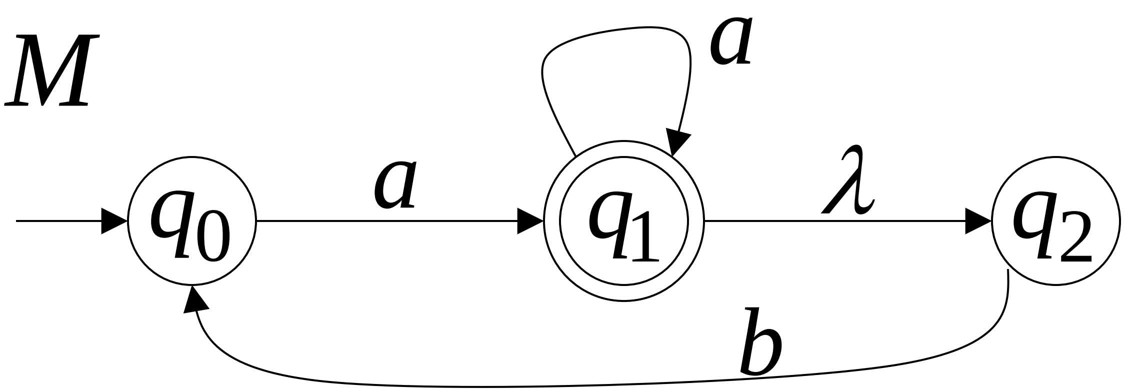


FA M'

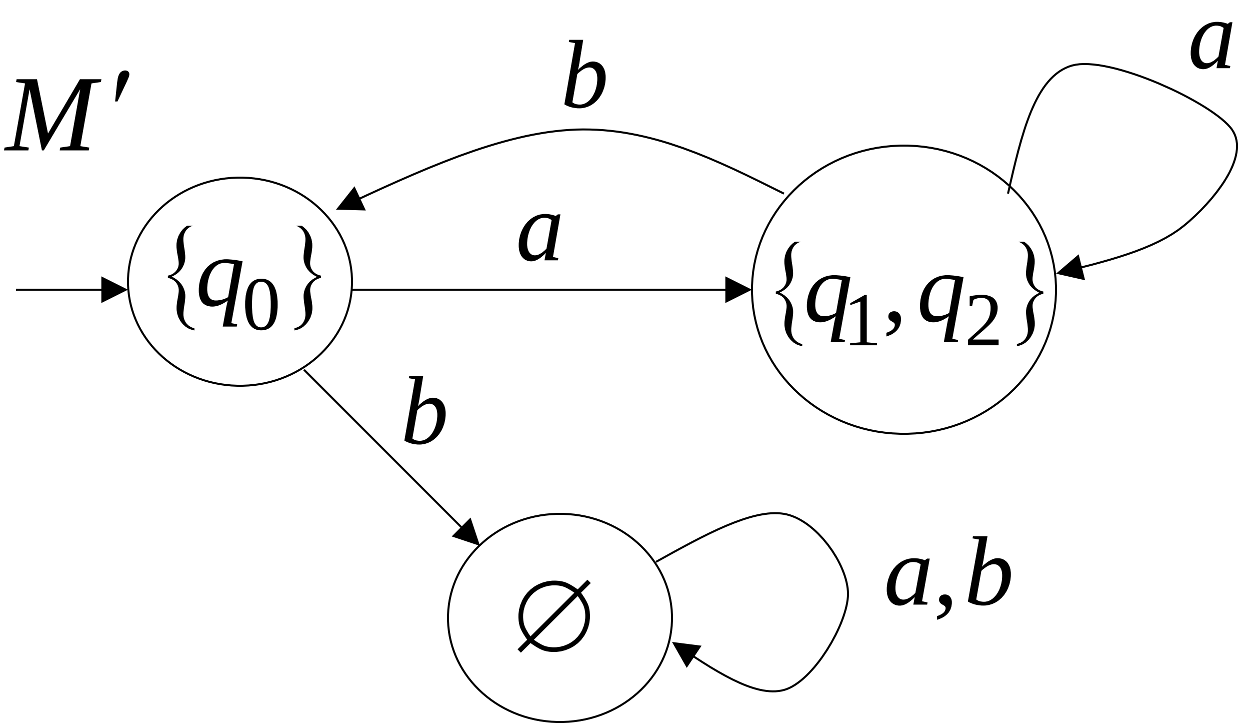


Convert NFA to FA

NFA M

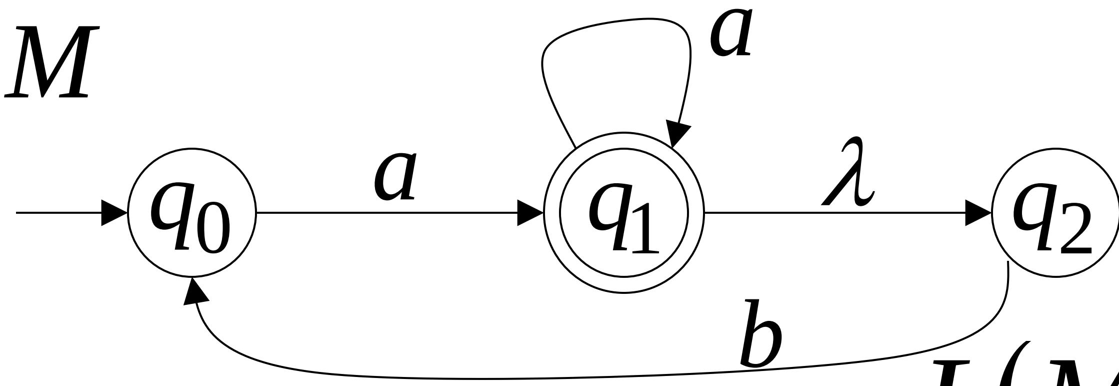


FA M'



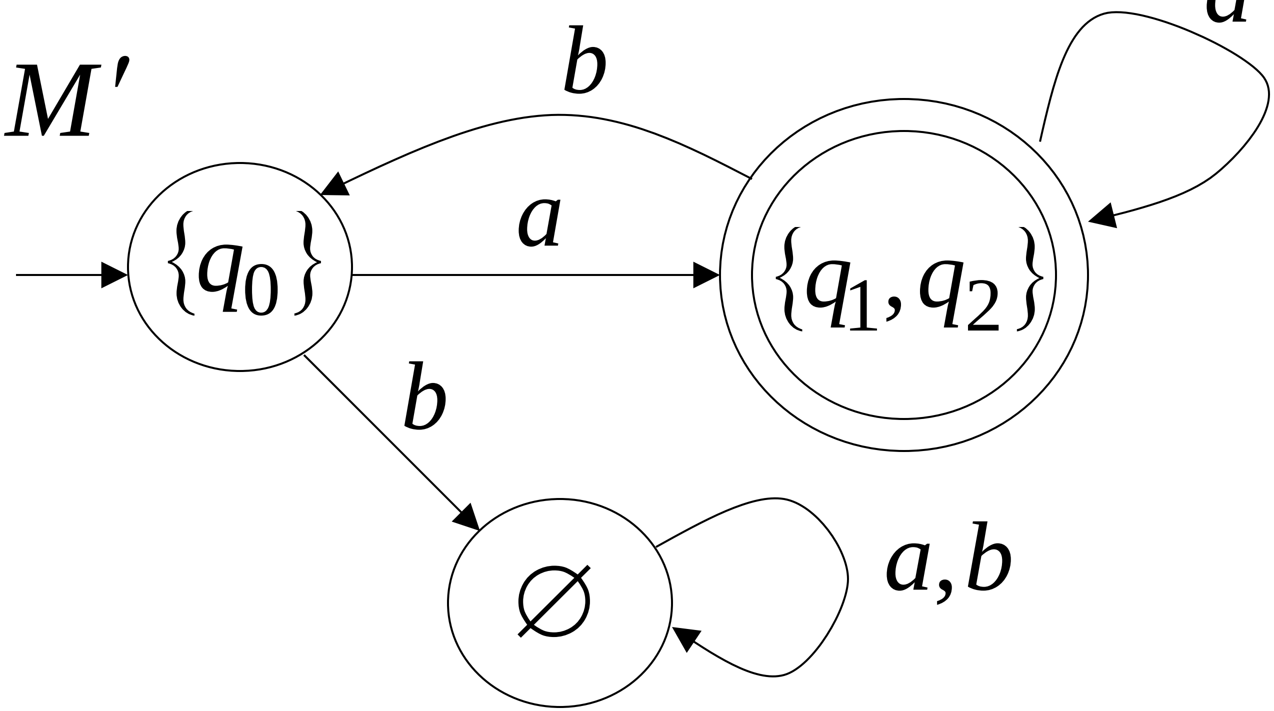
Convert NFA to FA

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

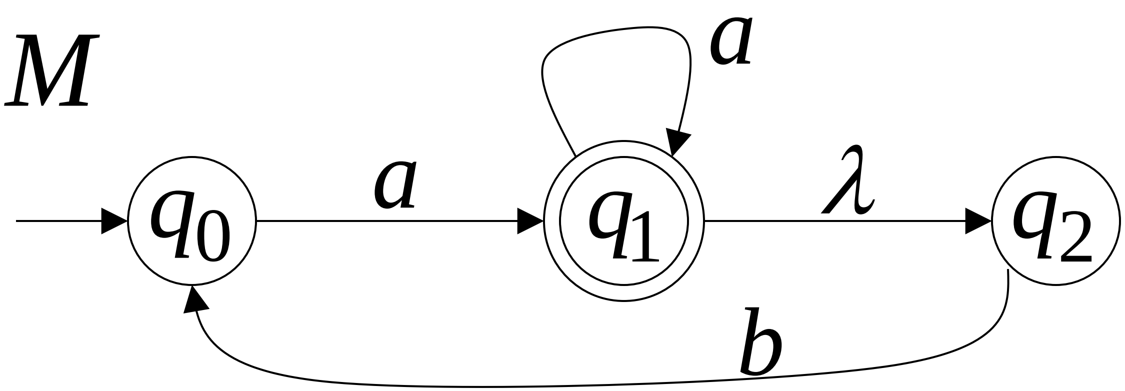
1. Initial state of NFA: q_0

$\Downarrow$

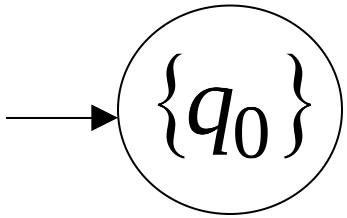
Initial state of FA: $\{q_0\}$

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

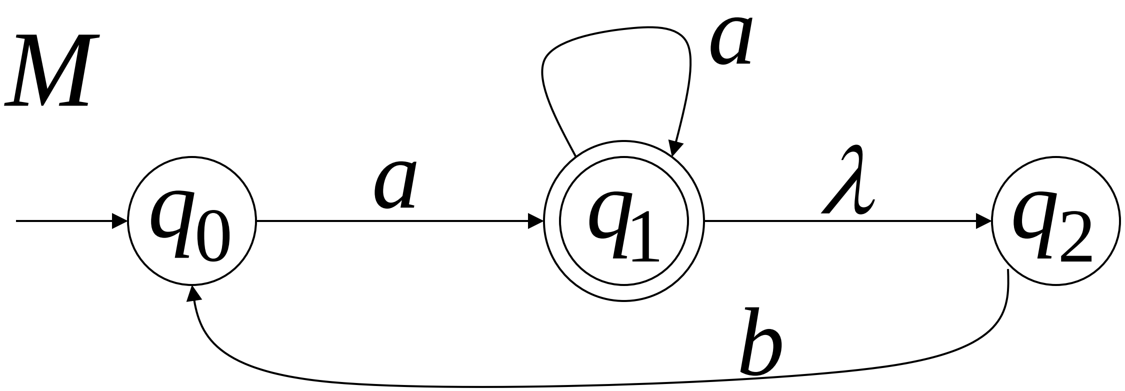
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to FA

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

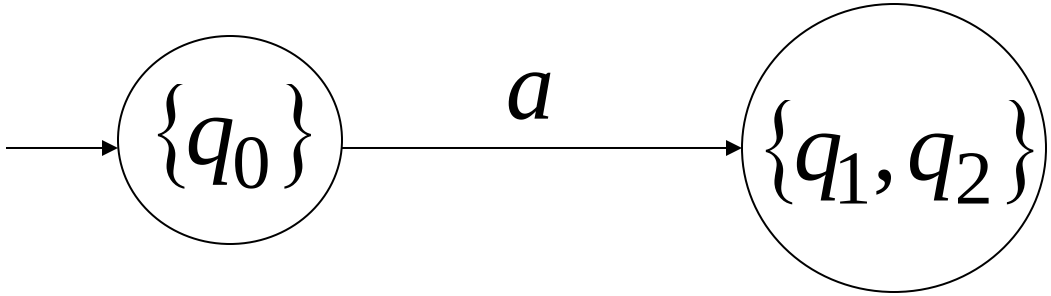
Example

NFA M



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

FA M'



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

Procedure NFA to FA

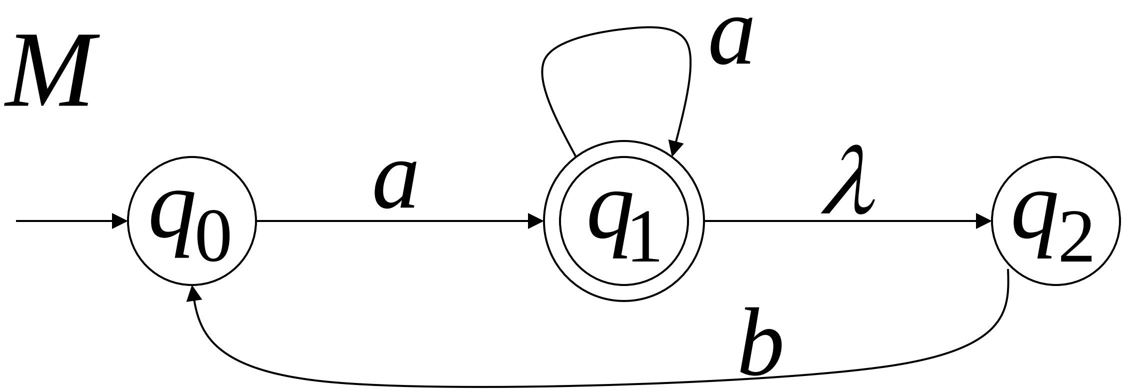
Repeat Step 2 for all letters in alphabet,

until

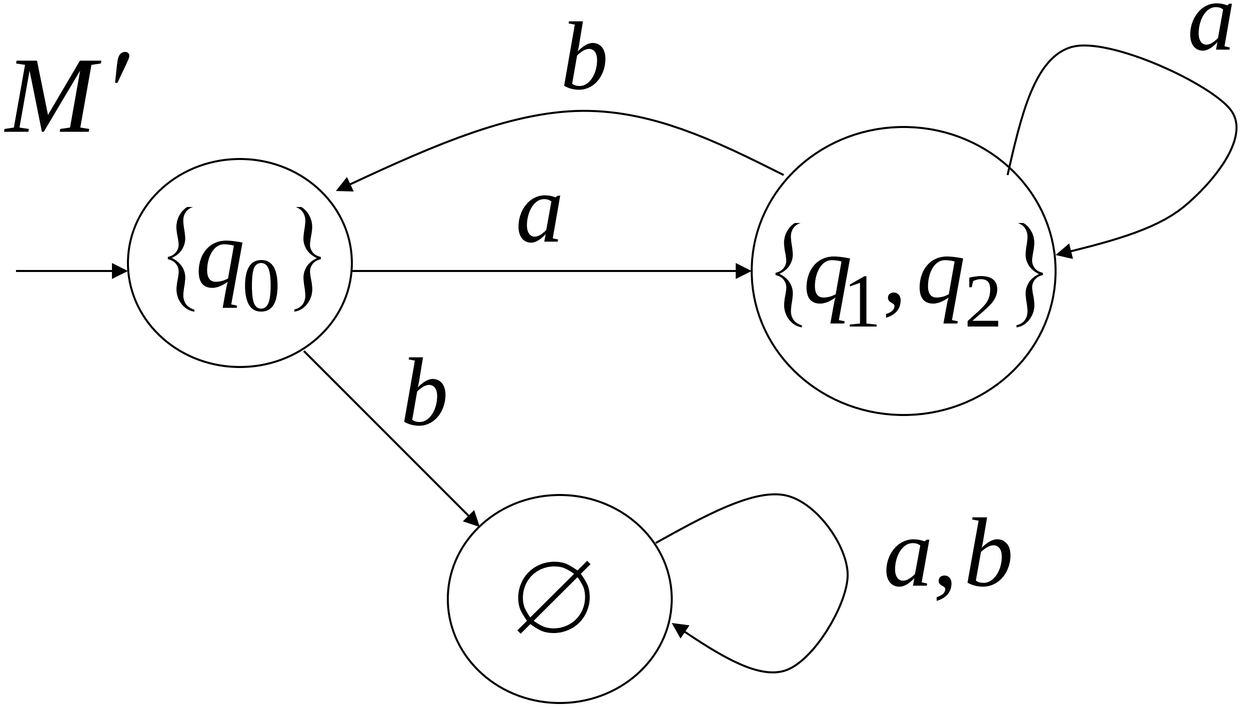
no more transitions can be added.

Example

NFA M



FA M'



Done?

Procedure NFA to FA

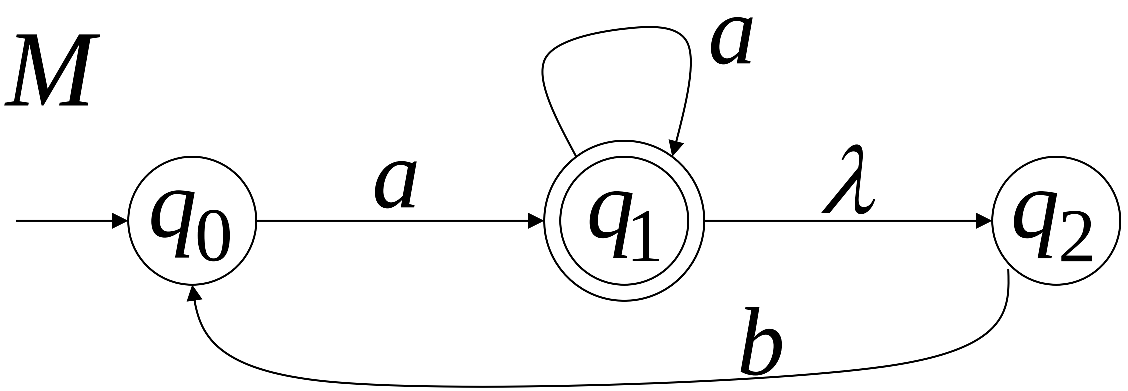
3. For any FA state $\{q_i, q_j, \ldots, q_m\}$

If q_j is accepting state in NFA

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

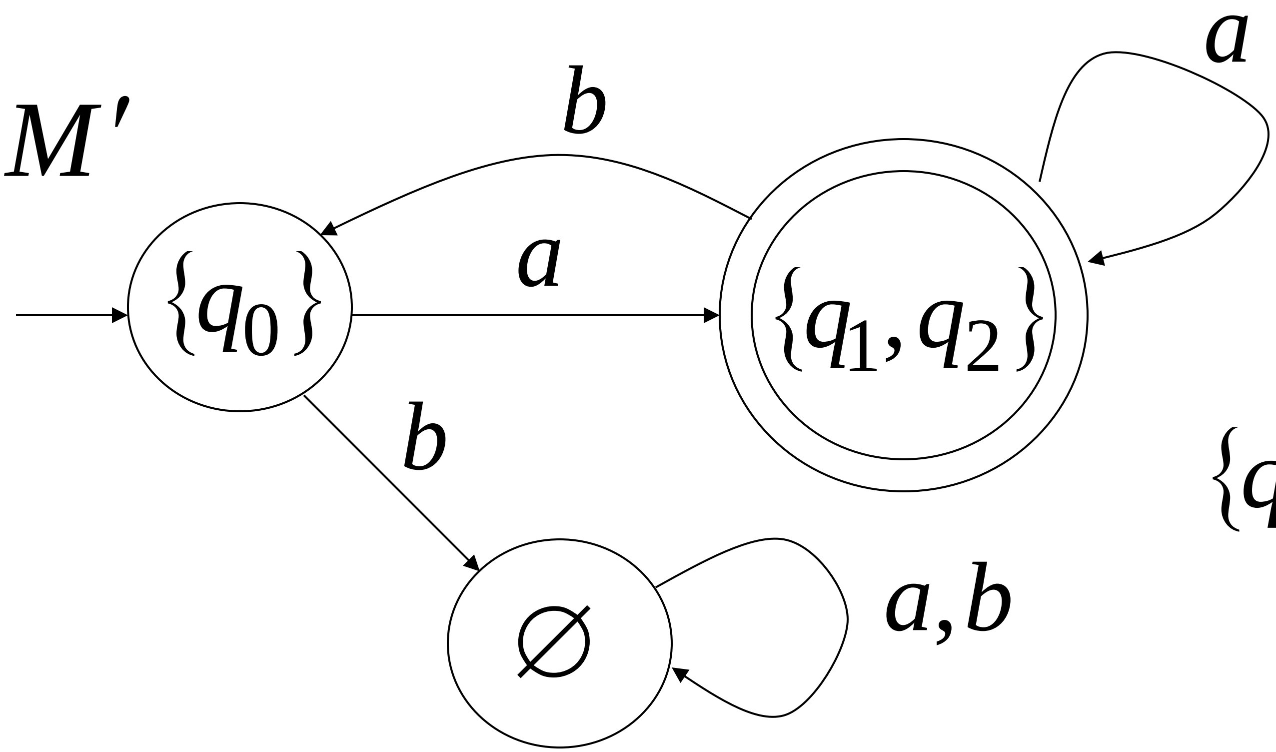
Example

NFA M



$q_1 \in F$

FA M'



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

Theorem

Take NFA M

Apply procedure to obtain FA M'

Then M and M' are equivalent :

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

Proof

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

$$\Updownarrow$$

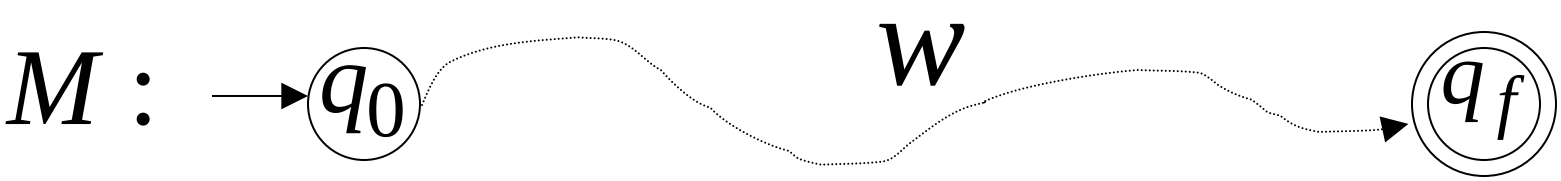
$$L(M) \subseteq L(M') \quad \text{AND} \quad L(M) \supseteq L(M')$$

First we show: $L(M) \subseteq L(M')$

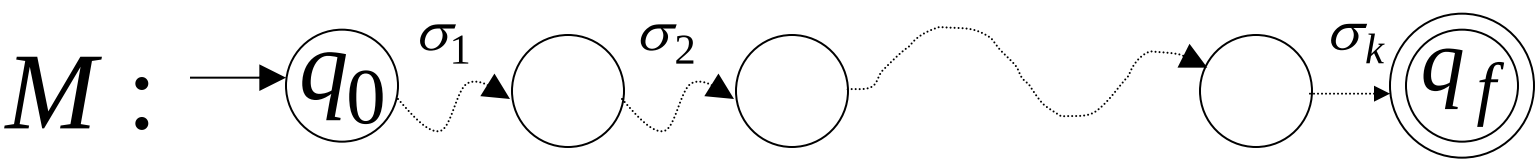
Take arbitrary: $w \in L(M)$

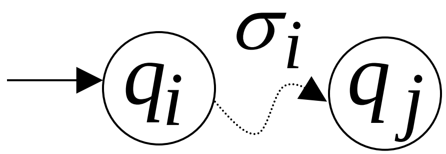
We will prove: $w \in L(M')$

$$w \in L(M)$$

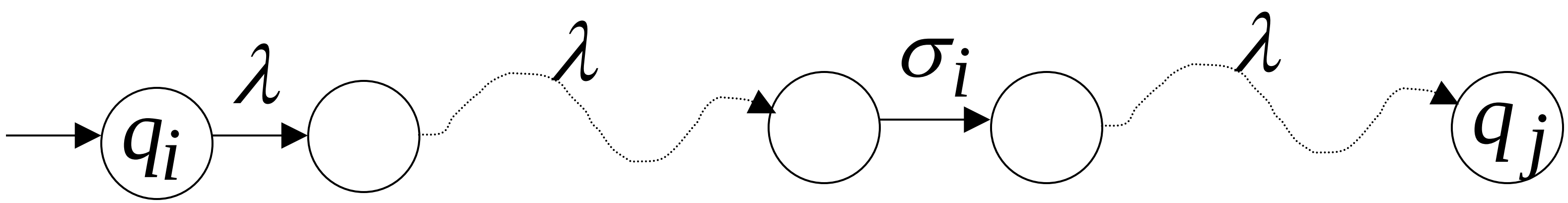


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$





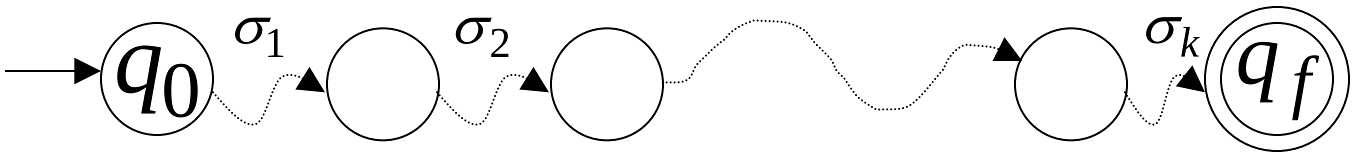
denotes



We will show that if $w \in L(M)$

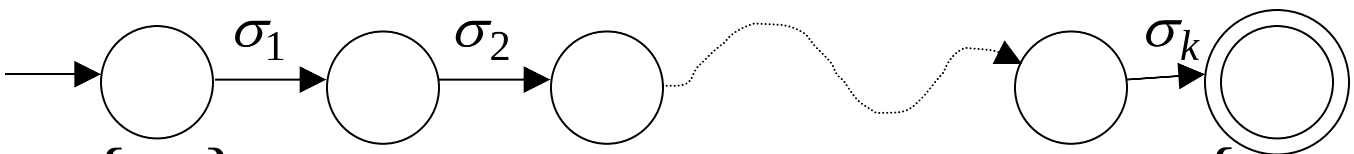
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$

$M:$



then

$M':$



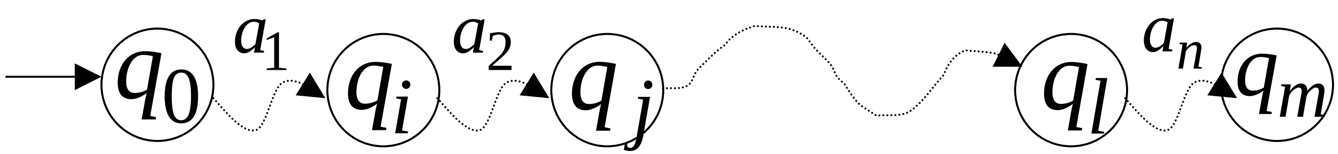
$\{q_0\}$ $\{q_f, \ldots\}$

$$w \in L(M')$$

More generally, we will show that if in M:

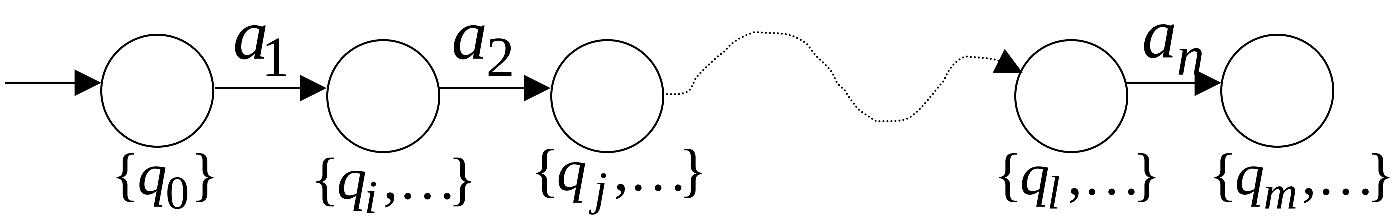
(arbitrary string) $v = a_1 a_2 \cdots a_n$

$M:$



then

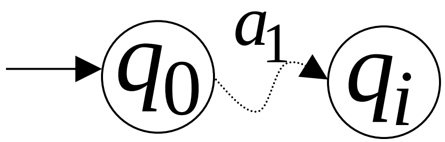
$M':$



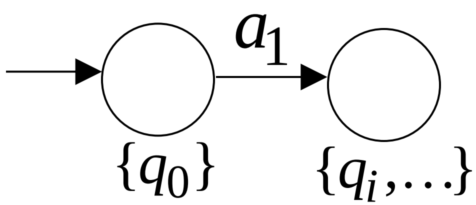
Proof by induction on $|v|$

Induction Basis: $v = a_1$

$M:$



$M':$

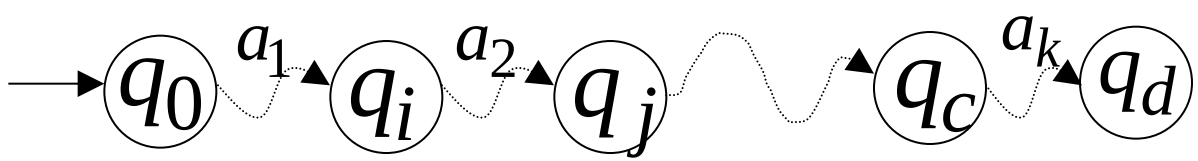


Is true by construction of M':

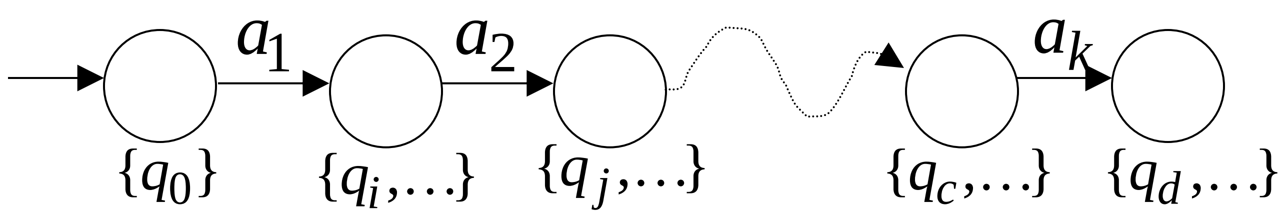
Induction hypothesis: $1 \leq |v| \leq k$

$$v = a_1 a_2 \cdots a_k$$

$M:$

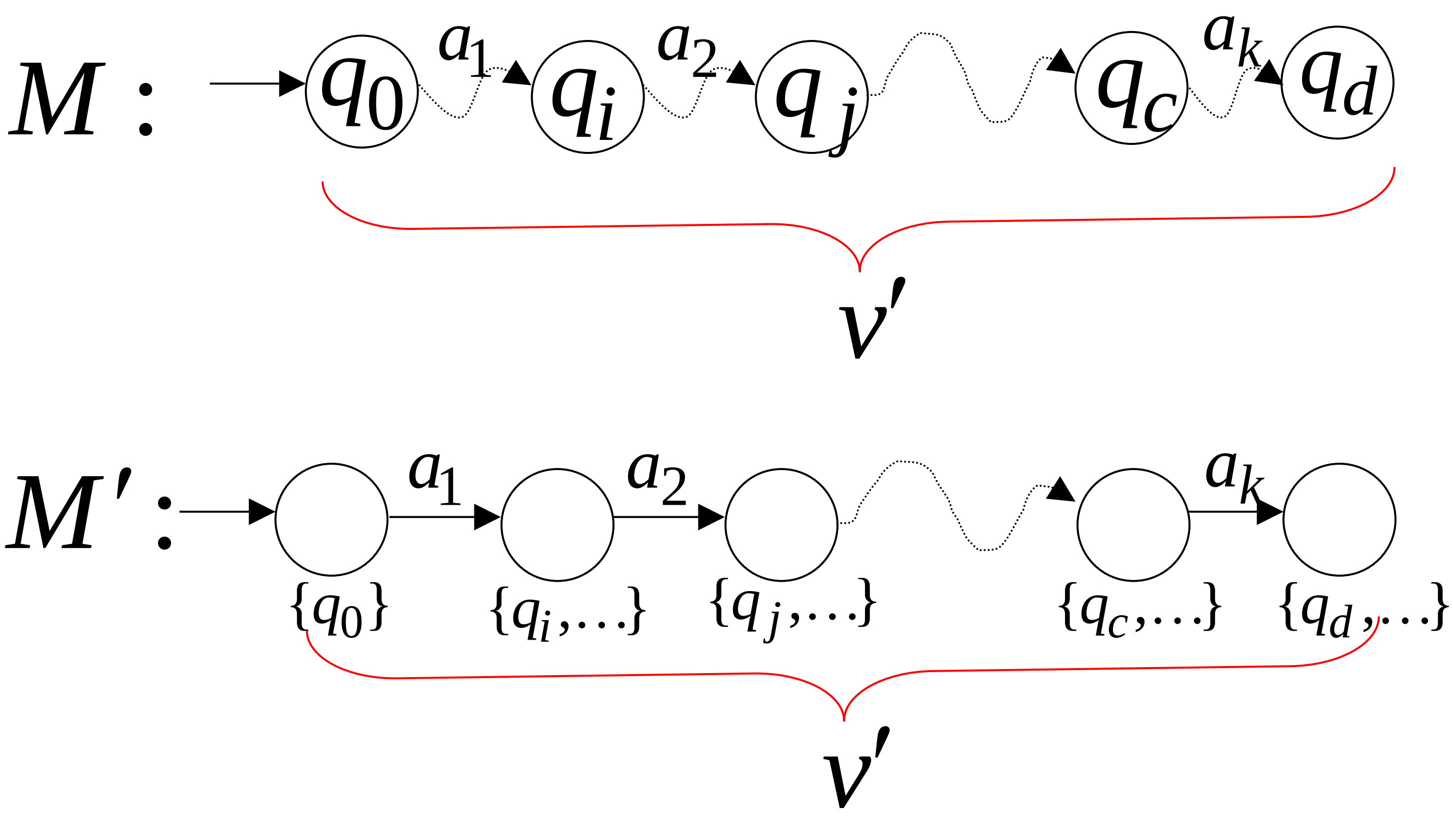


$M':$



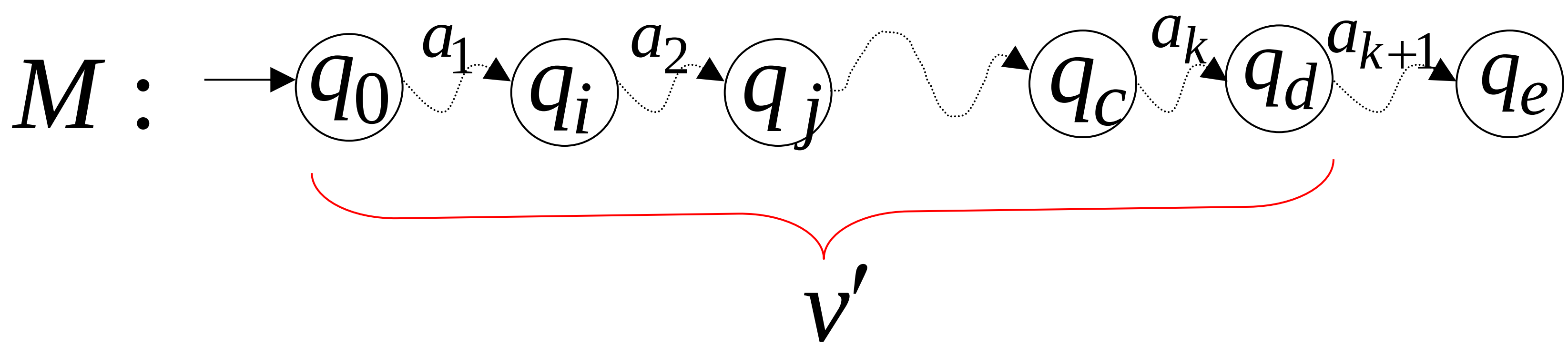
Induction Step: $|v| = k+1$

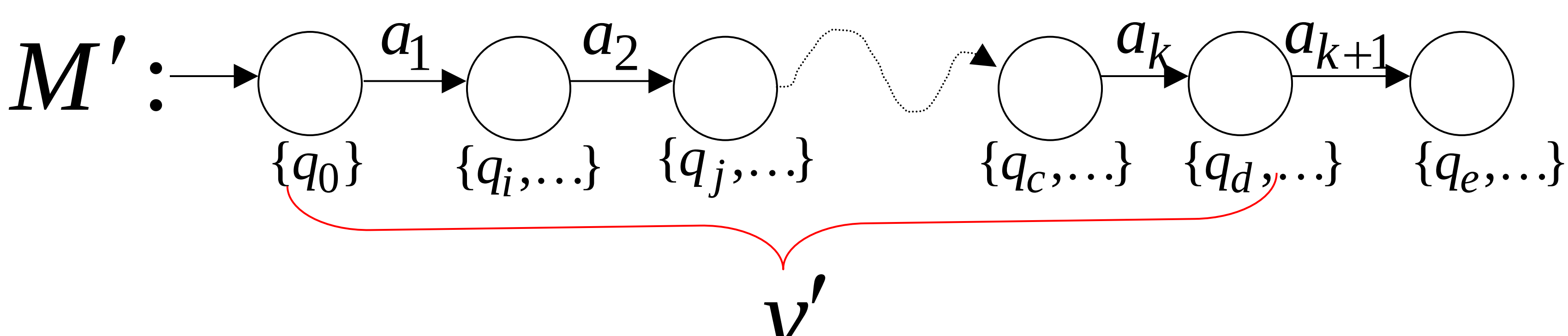
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$



Induction Step: $|v| = k+1$

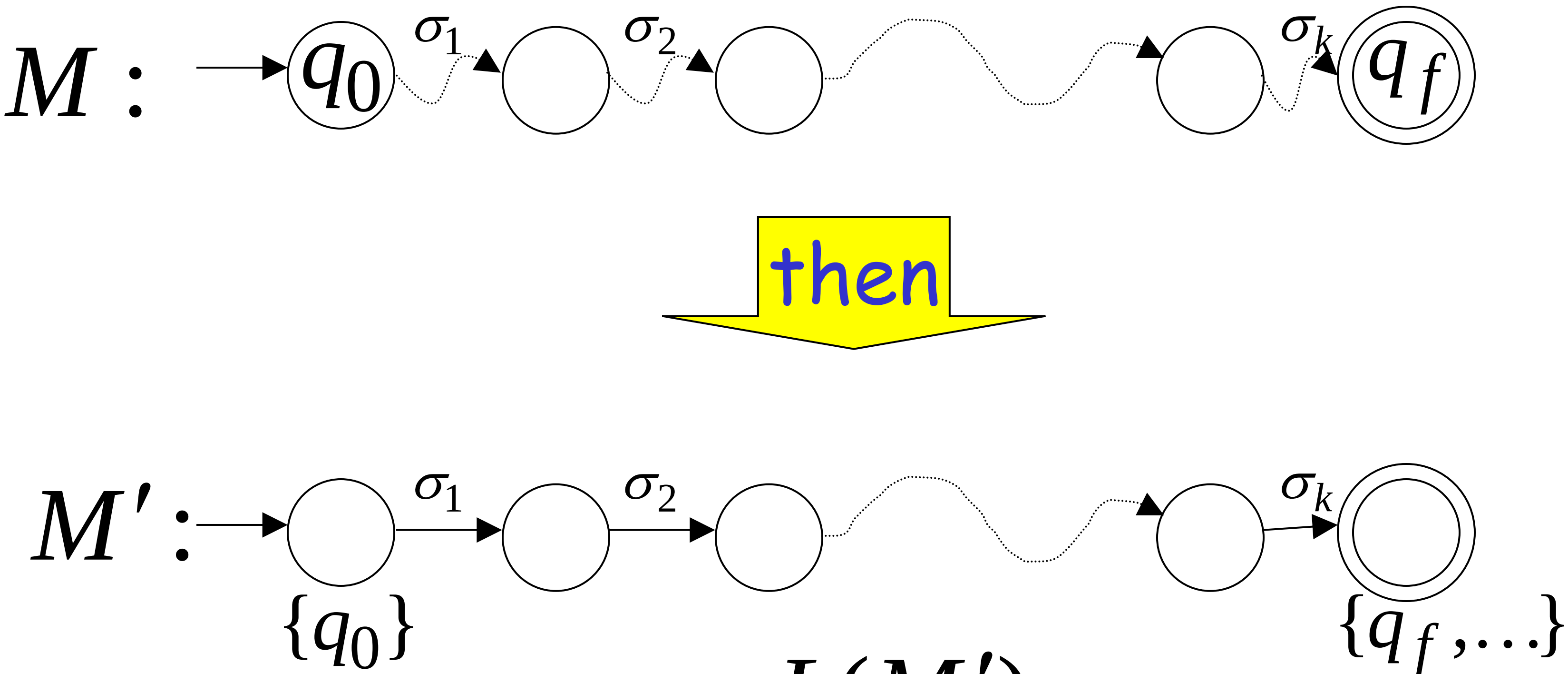
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$





Therefore if $w \in L(M)$

$w = \sigma_1 \sigma_2 \cdots \sigma_k$



$w \in L(M')$

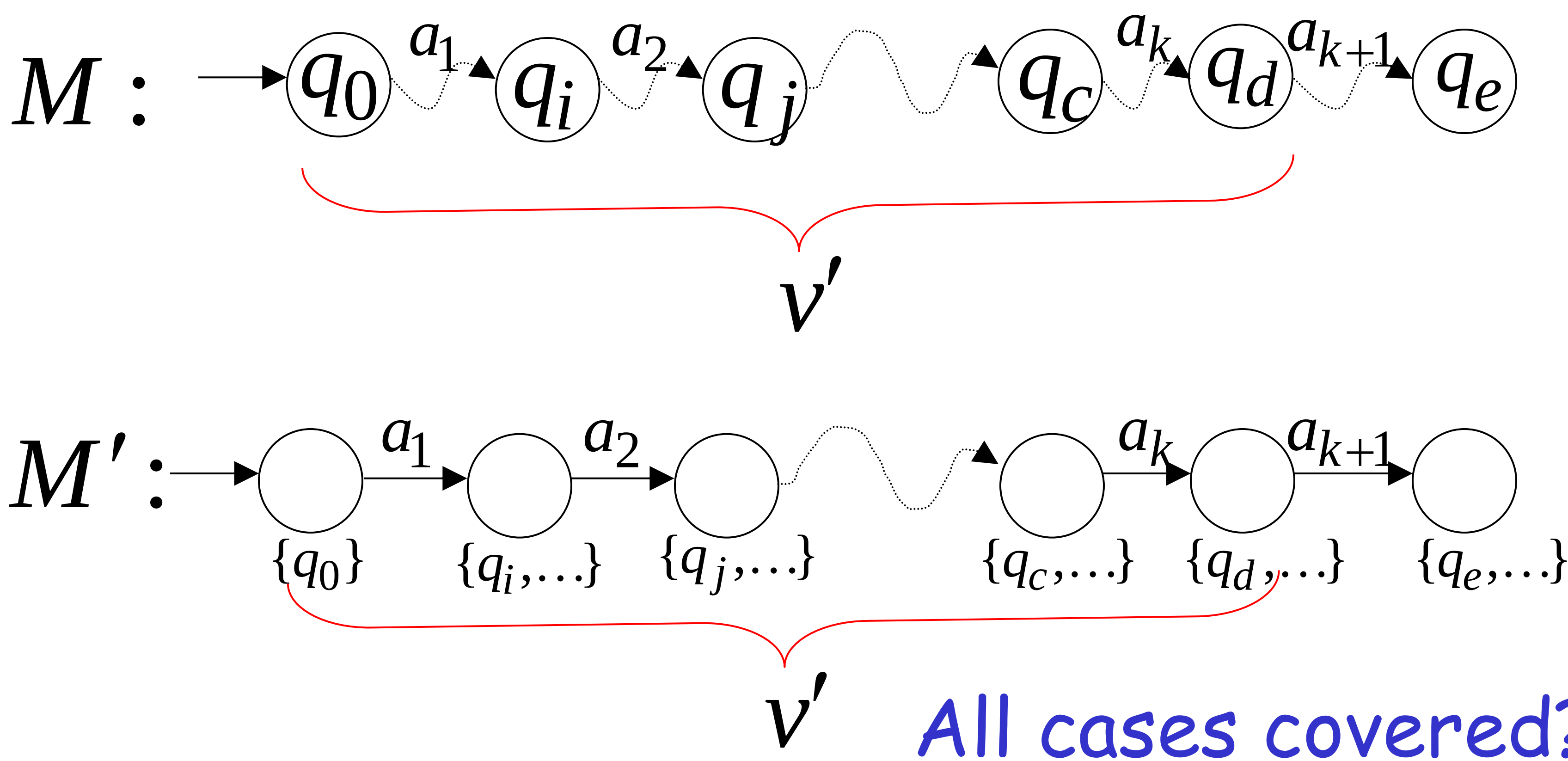
We have shown: $L(M) \subseteq L(M')$

We also need to show: $L(M) \supseteq L(M')$

(proof is similar)

Induction Step: $|v| = k+1$

$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$



All cases covered?

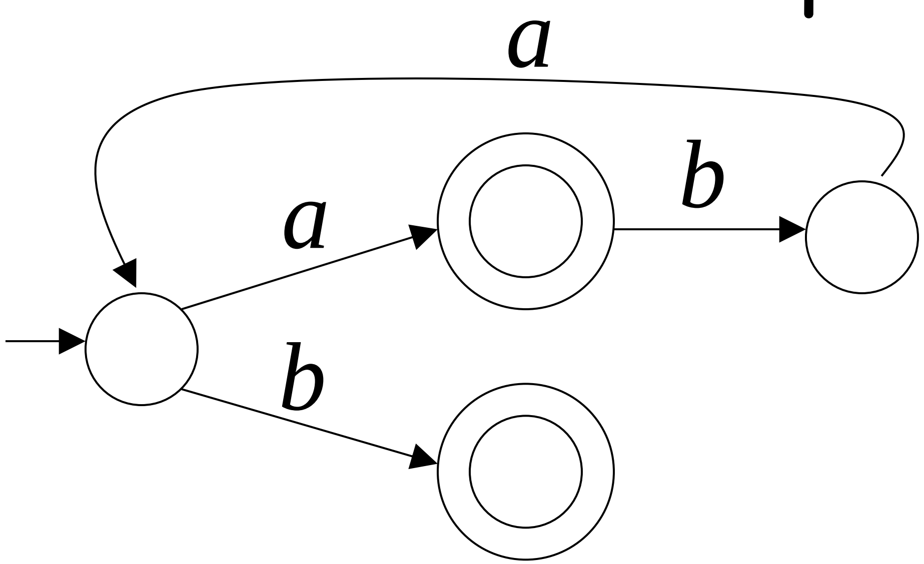
Single Accepting State for NFAs

Any NFA can be converted

to an equivalent NFA

with a single accepting state

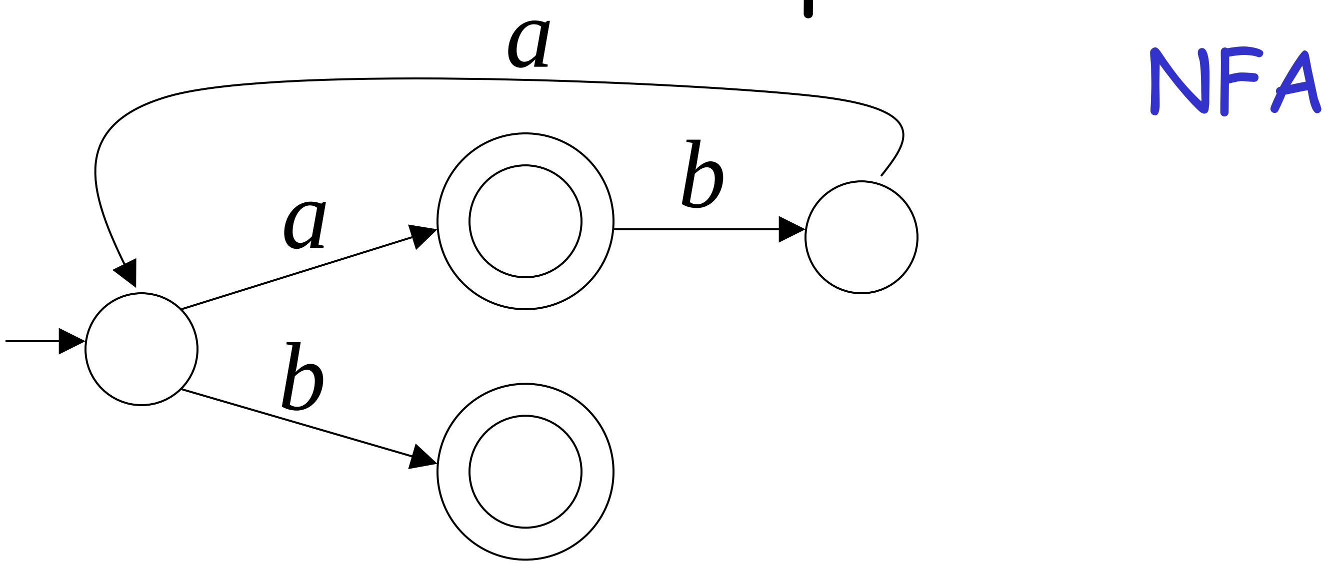
Example

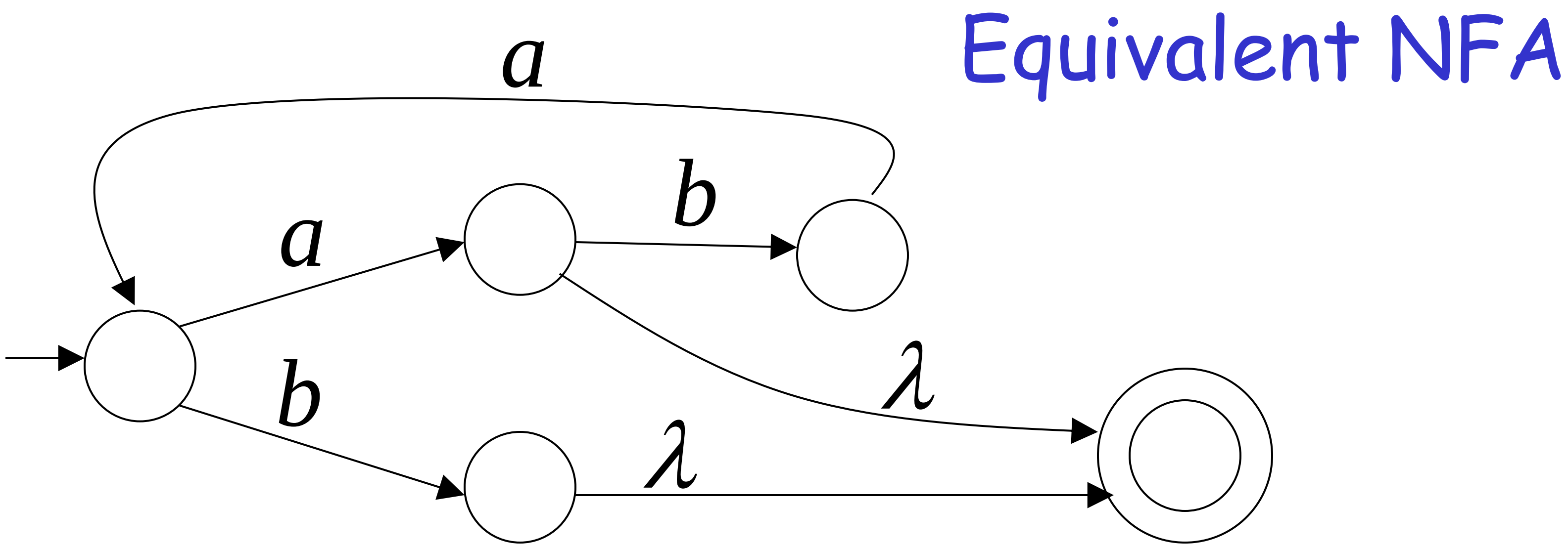


NFA

Equivalent NFA
with single
accepting state?

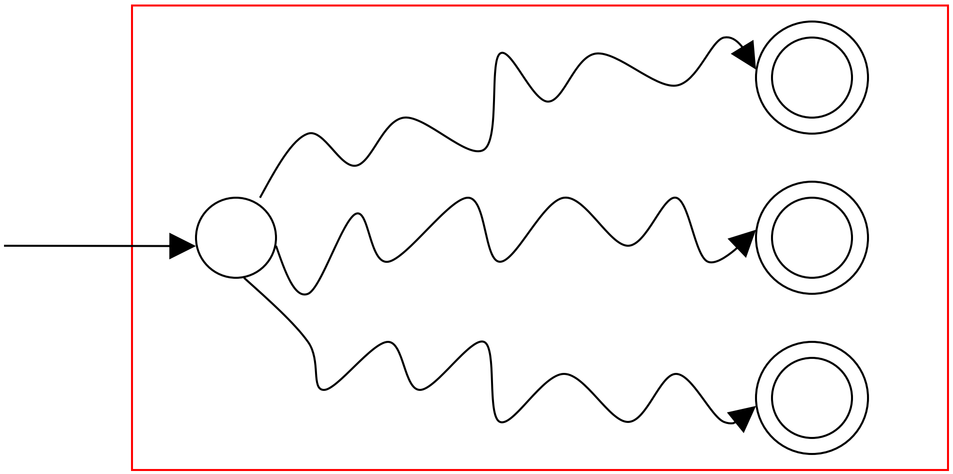
Example



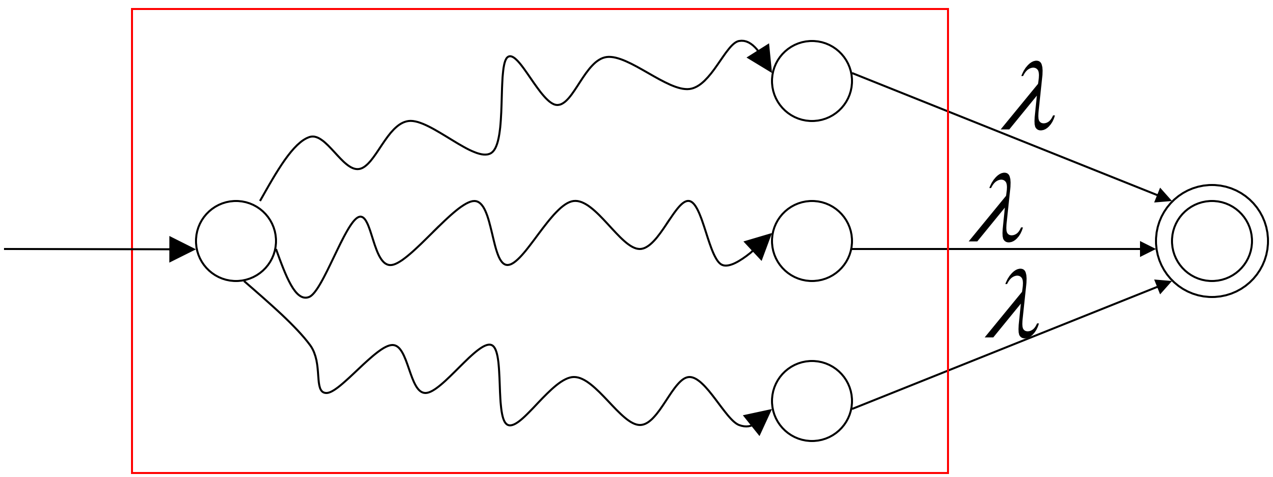


In General

NFA



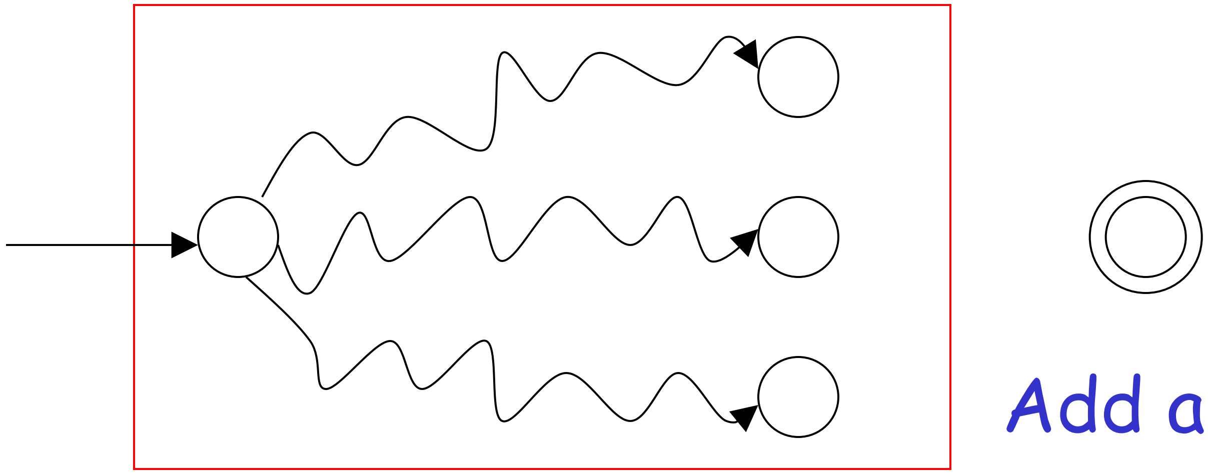
Equivalent NFA



Single accepting state

Extreme Case

NFA without accepting state



Add an accepting state without transitions

Properties of Regular Languages

For regular languages L_1 and L_2 we will prove that:

$$\left.\begin{array}{rl} \text{Union:} & L_1 \cup L_2 \\ \text{Concatenation:} & L_1 L_2 \\ \text{Star:} & L_1{}^* \\ \text{Reversal:} & L_1{}^R \\ \text{Complement:} & \overline{L_1} \\ \text{Intersection:} & L_1 \cap L_2 \end{array}\right\} \text{Are regular Languages}$$

We say: Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

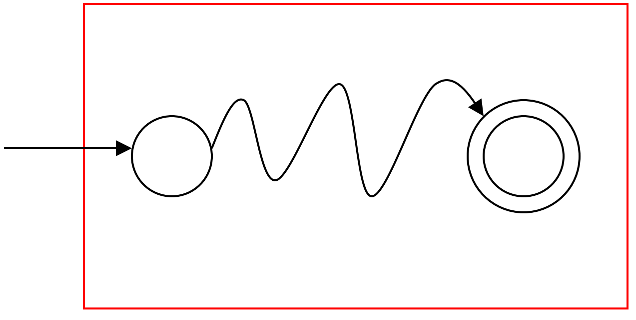
Regular language L_1

Regular language L_2

$$L(M_1) = L_1$$

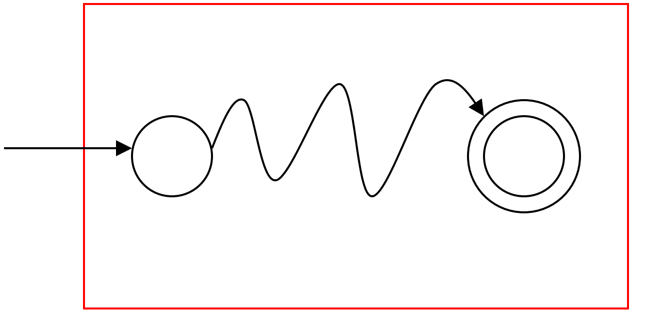
$$L(M_2) = L_2$$

NFA M_1



Single accepting state

NFA M_2



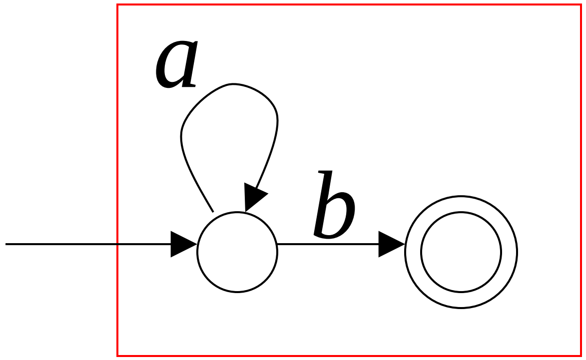
Single accepting state

Example

$n \geq 0$

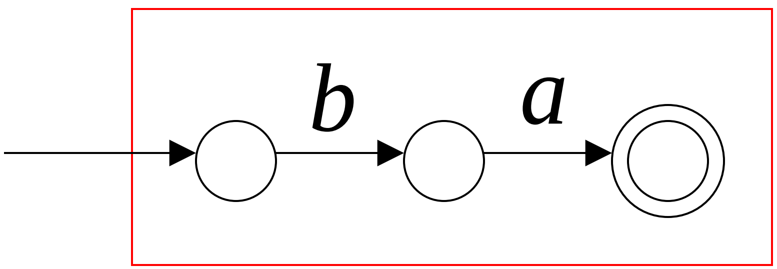
$L_1 = \{a^n b\}$

M_1



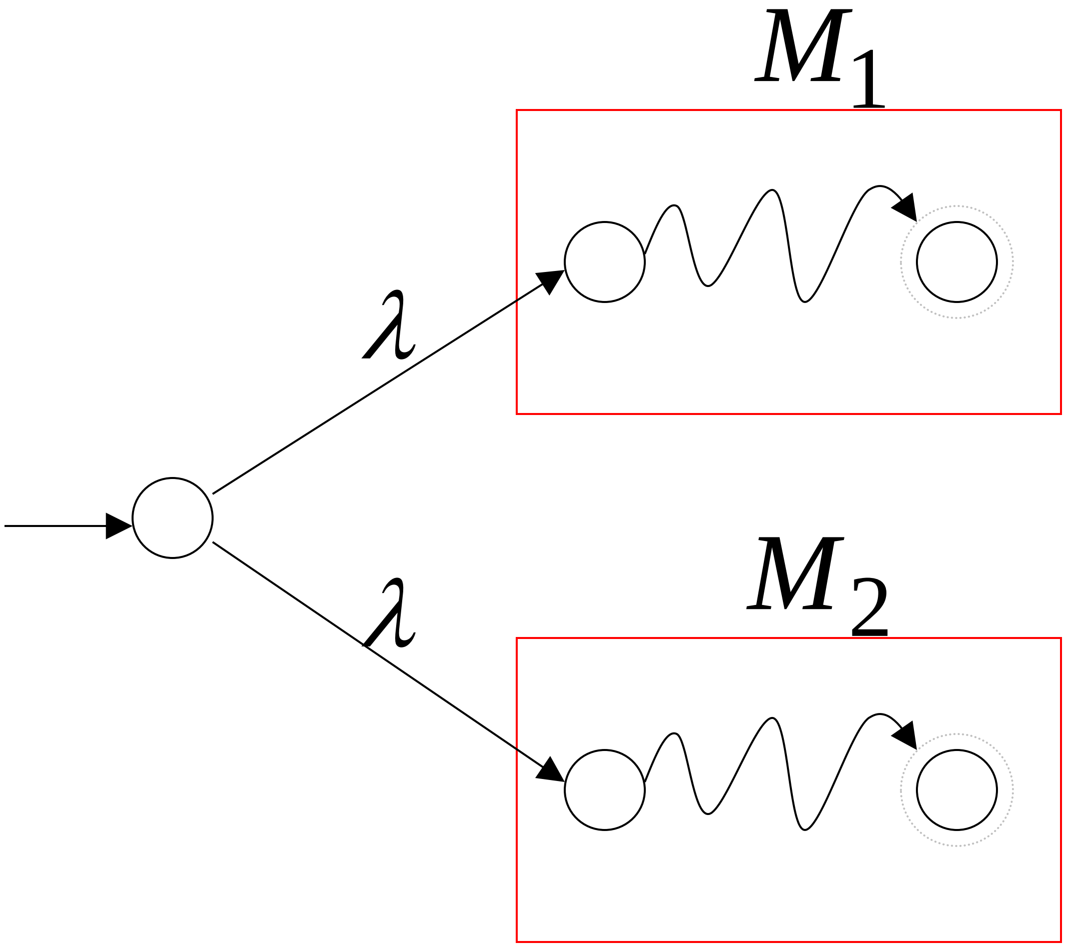
$L_2 = \{ba\}$

M_2



Union

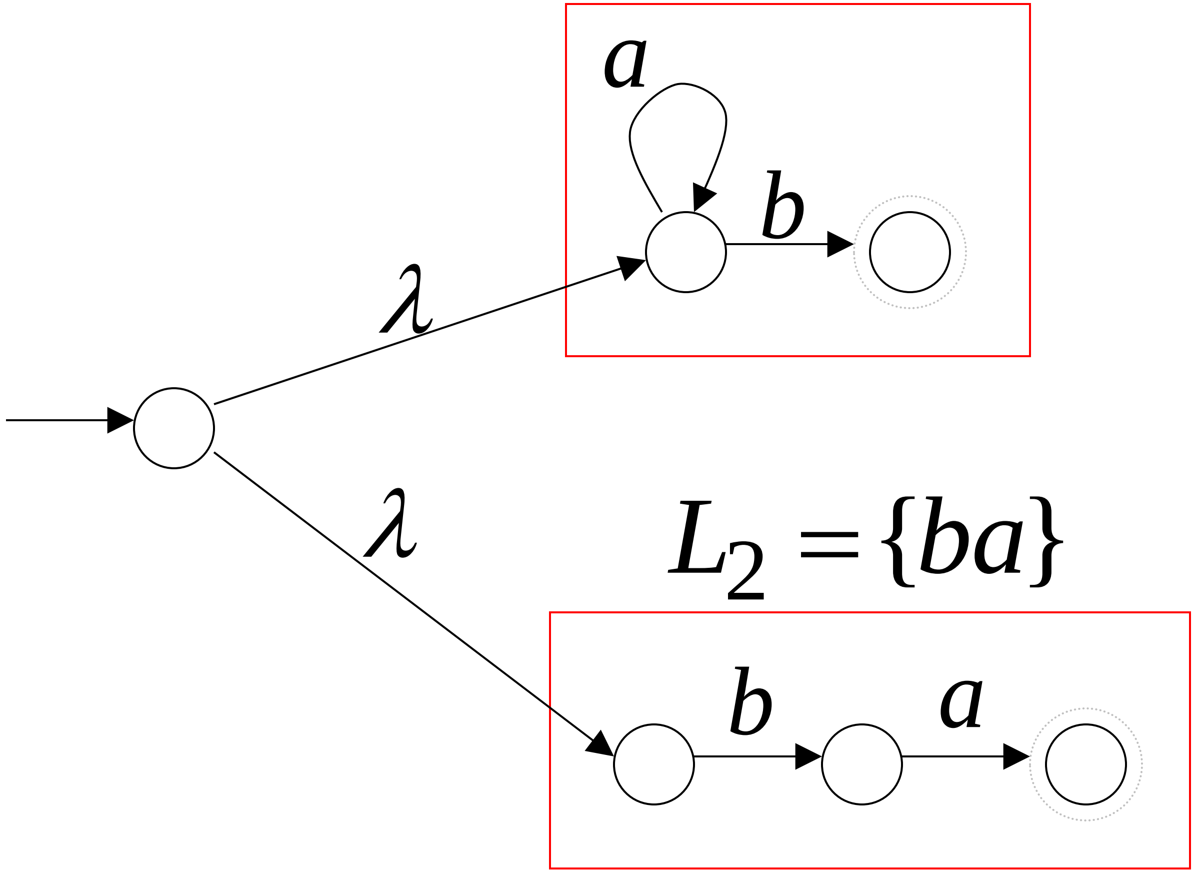
NFA for $L_1 \cup L_2$



Example

NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

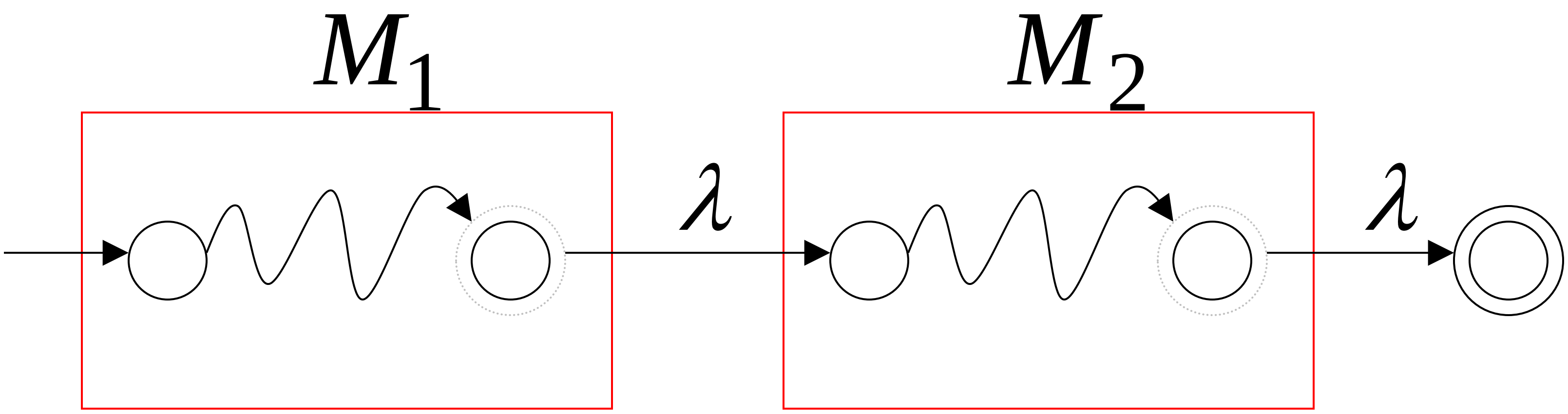
$L_1 = \{a^n b\}$



$L_2 = \{ba\}$

Concatenation

NFA for $L_1 L_2$

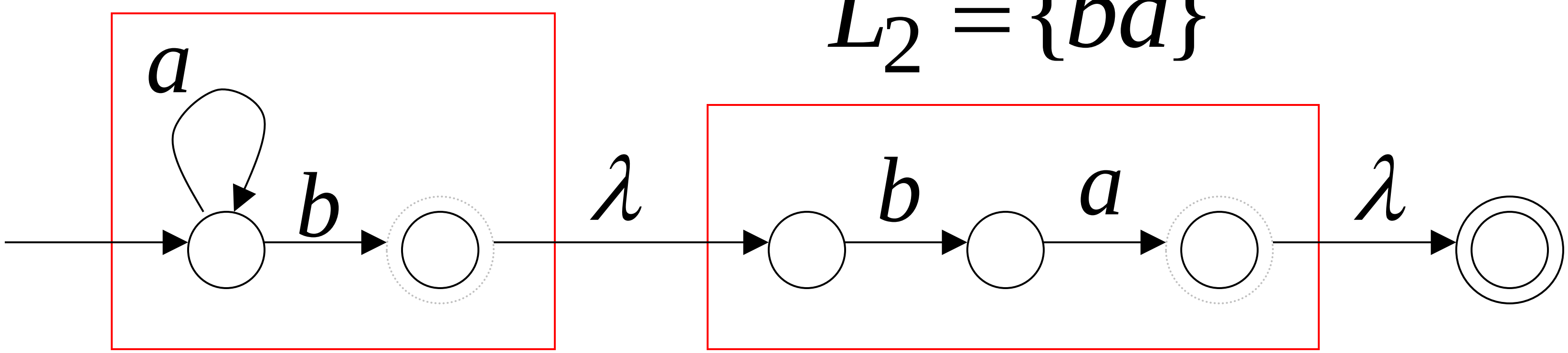


Example

NFA for $L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$

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

$L_2 = \{ba\}$



How do we construct automata for the remaining operations?

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

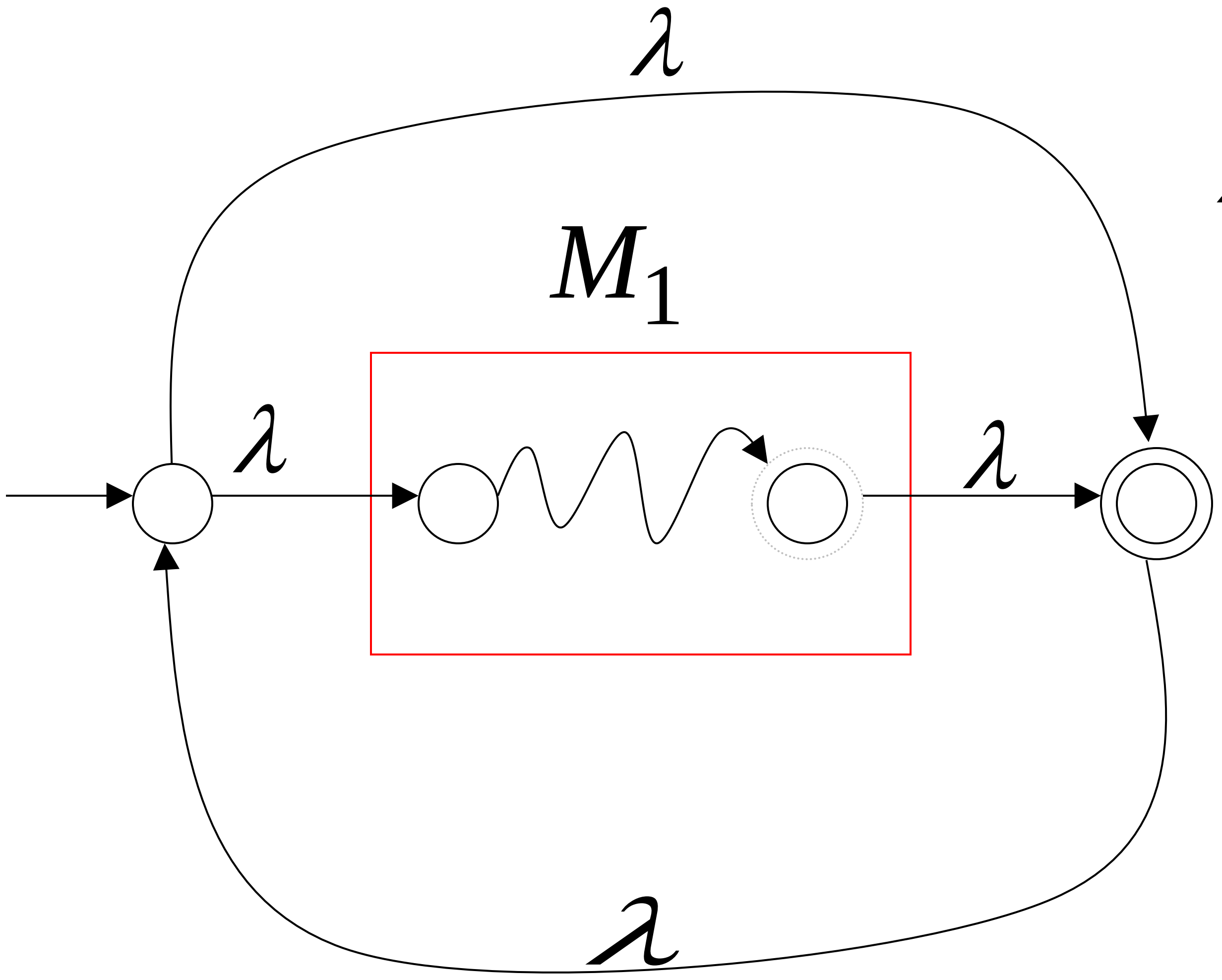
Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Star Operation

NFA for $L_1{}^*$



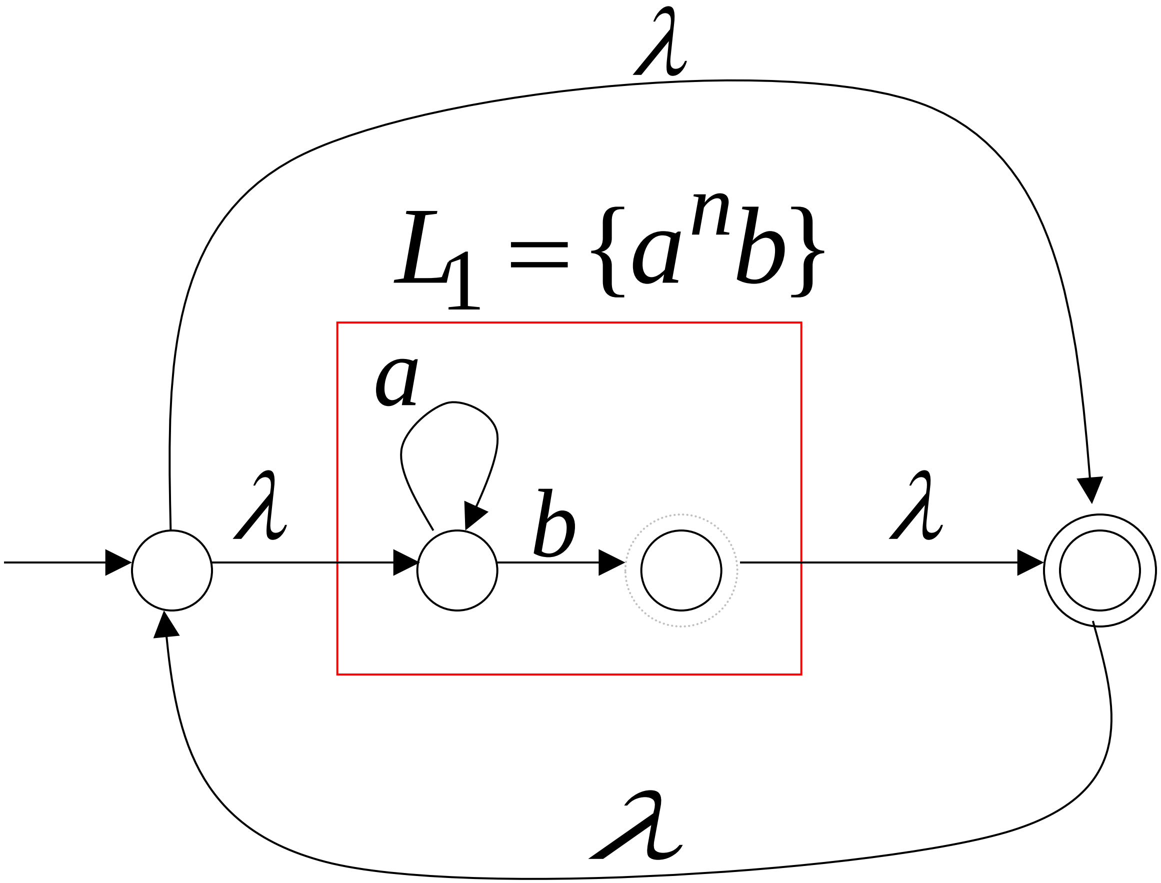
$\lambda \in L_1{}^*$

Example

NFA for $L_1{}^* = \{a^n b\}^*$

$w = w_1 w_2 \cdots w_k$

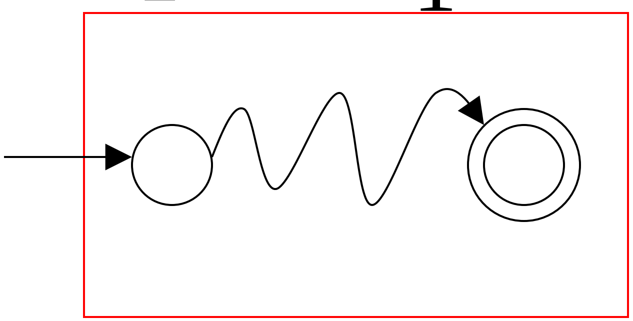
$w_i \in L_1$



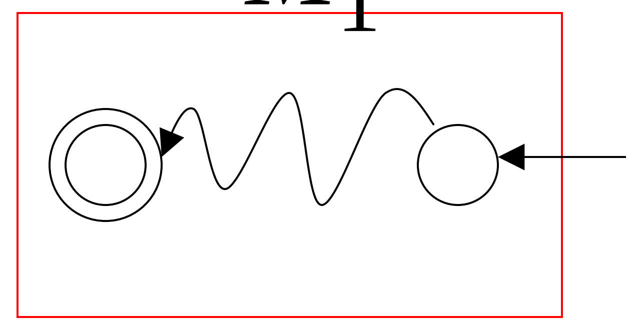
Reverse

NFA for L_1^R

L_1 M_1



M_1'



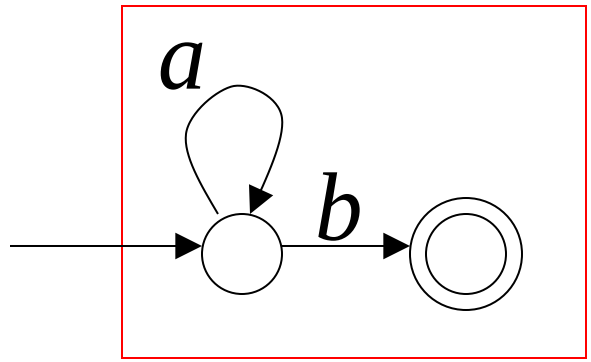
1. Reverse all transitions

2. Make initial state accepting state
   and vice versa

Example

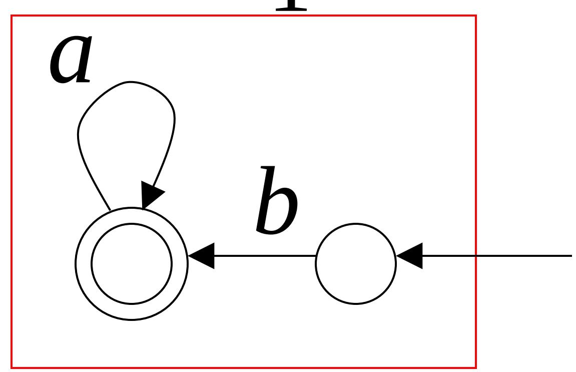
M_1

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

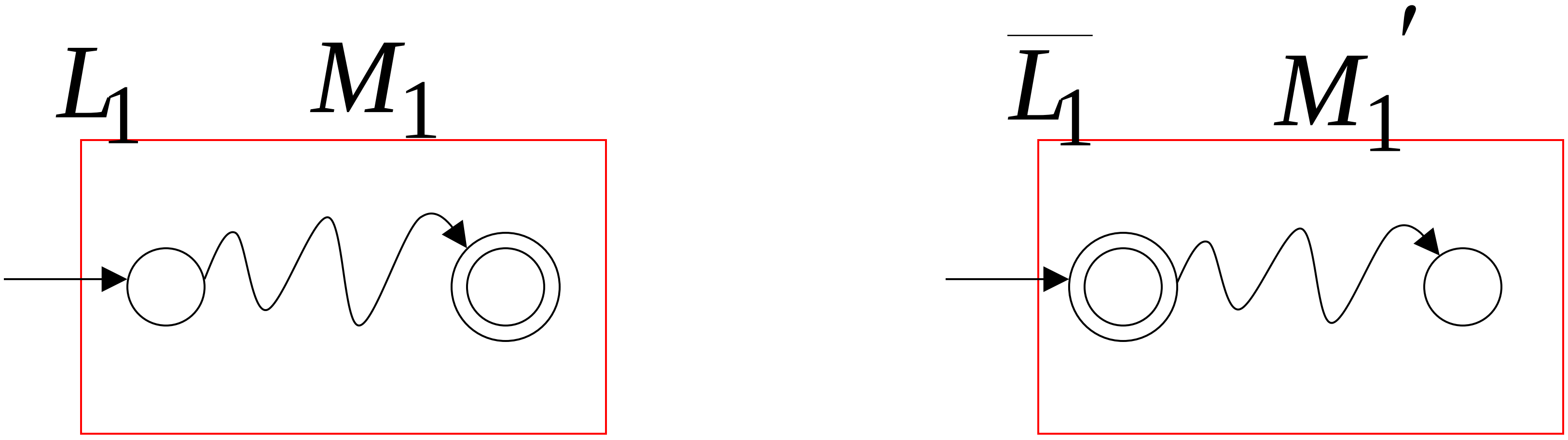


M_1'

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



Complement



1. Take the **FA** that accepts L_1

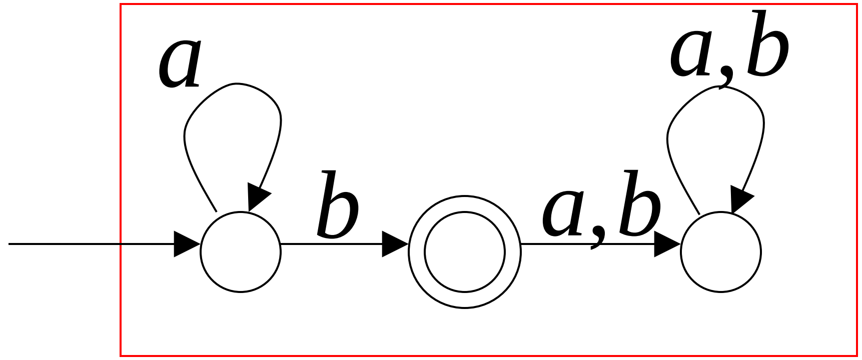
2. Make final states non-final, and vice-versa

Why not NFA?

Example

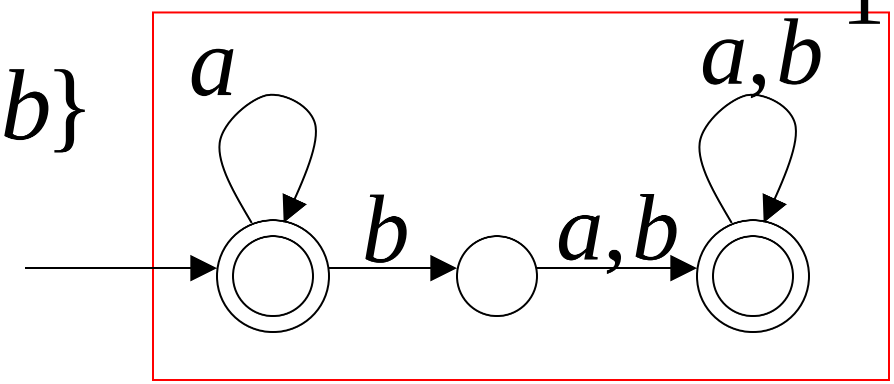
M_1

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



M_1'

$$\overline{L_1} = \{a,b\}^* - \{a^n b\}$$



Intersection

L_1 regular

L_2 regular

We show

$L_1 \cap L_2$ regular

DeMorgan's Law: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

$L_1 \, , \; L_2$ regular

$\Rightarrow$ $\overline{L_1} \, , \; \overline{L_2}$ regular

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regular

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$\Rightarrow$ $L_1 \cap L_2$ regular

Example

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

$L_2 = \{ab, ba\}$ regular

$\Rightarrow$ $L_1 \cap L_2 = \{ab\}$

regular

Another Proof for Intersection Closure

Machine M_1

FA for L_1

Machine M_2

FA for L_2

Construct a new FA M that accepts $L_1 \cap L_2$

M simulates in parallel M_1 and M_2

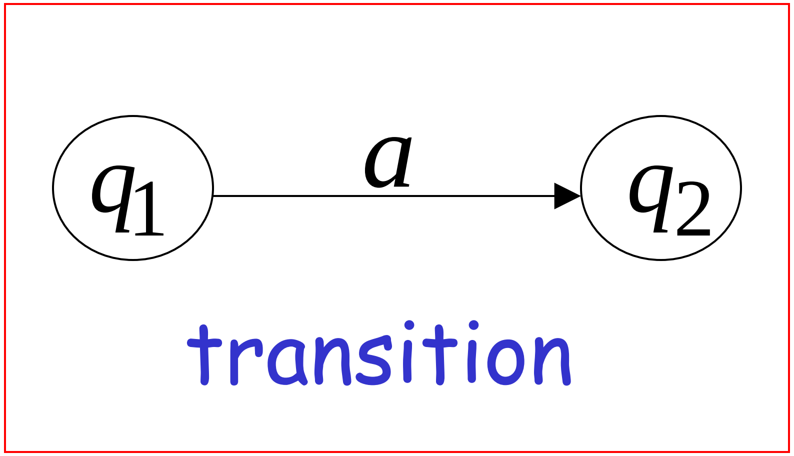
States in M

q_i, p_j

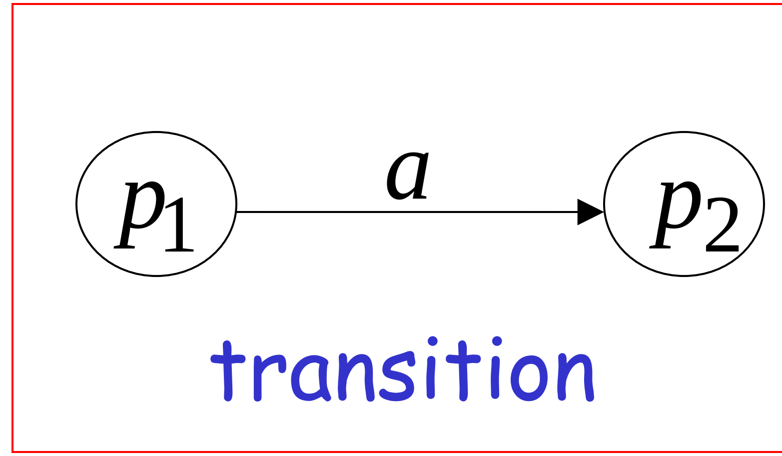
State in M_1

State in M_2

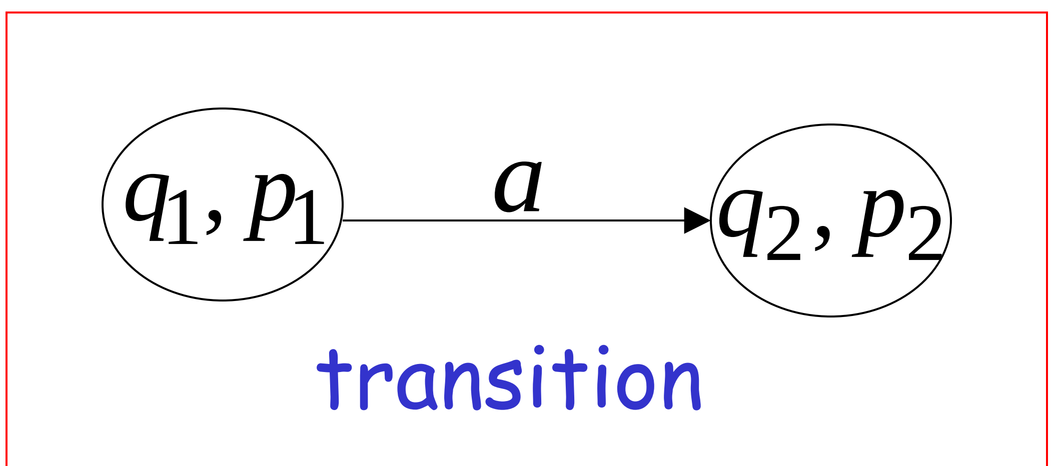
FA M_1



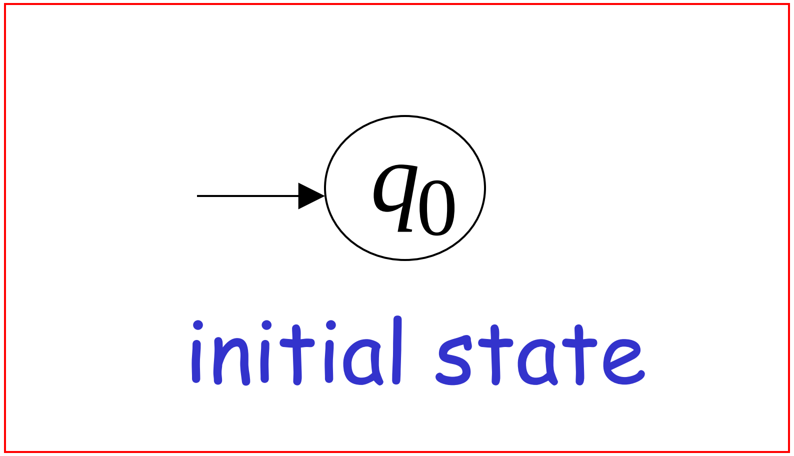
FA M_2



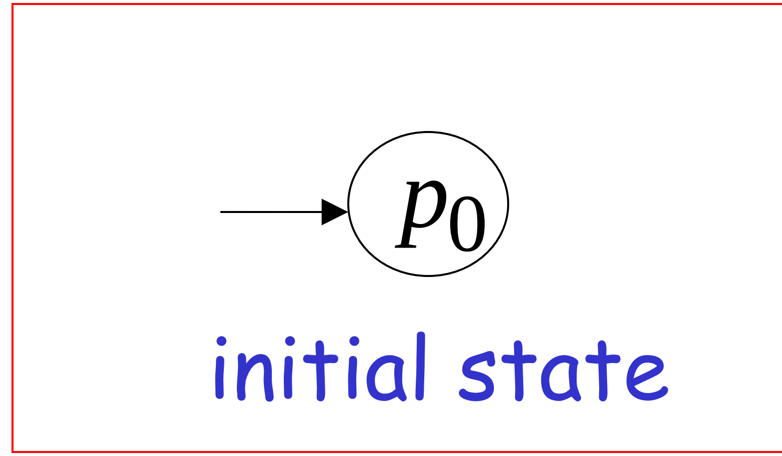
FA M



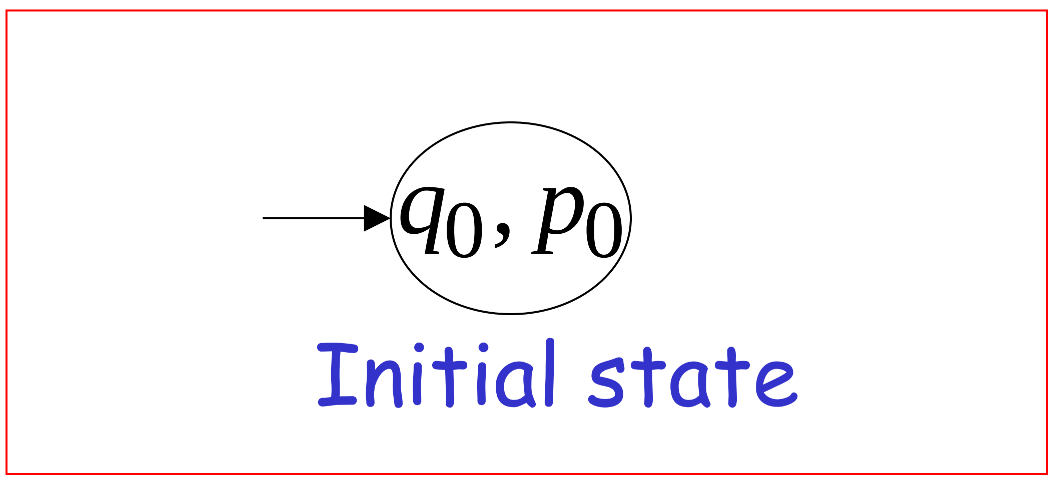
FA M_1

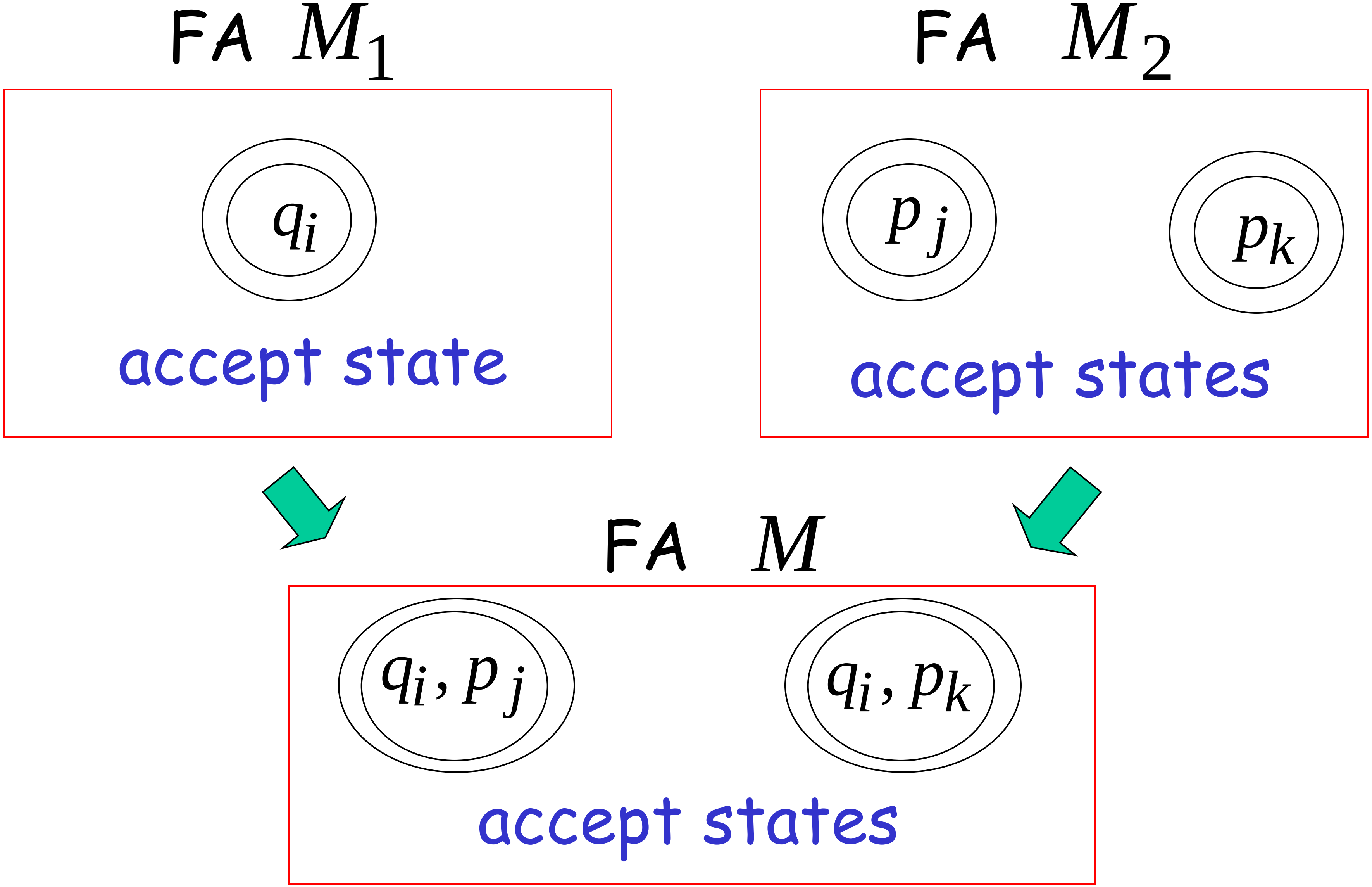


FA M_2



FA M





Both constituents must be accepting states

M simulates in parallel M_1 and M_2

M accepts string w if and only if

M_1 accepts string w and

M_2 accepts string w

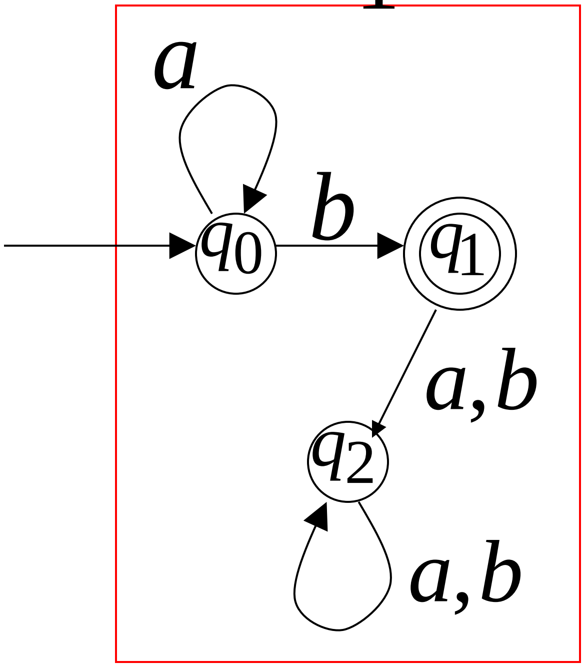
$$L(M) = L(M_1) \cap L(M_2)$$

Example:

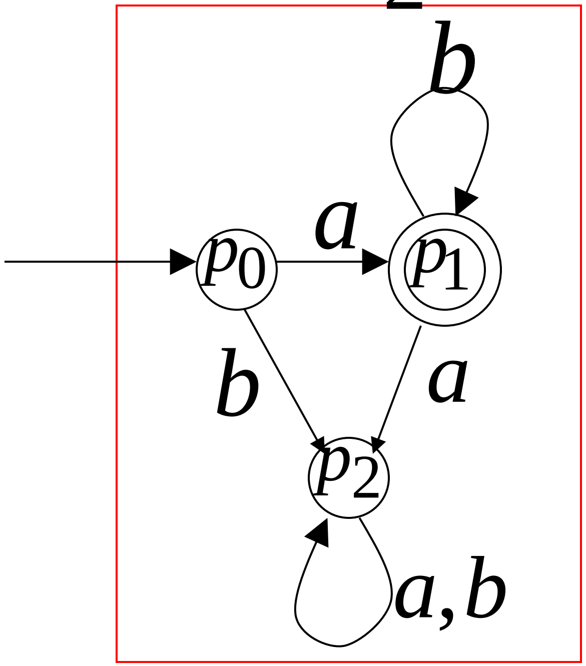
$$L_1 = \{a^n b\} \quad n \geq 0$$

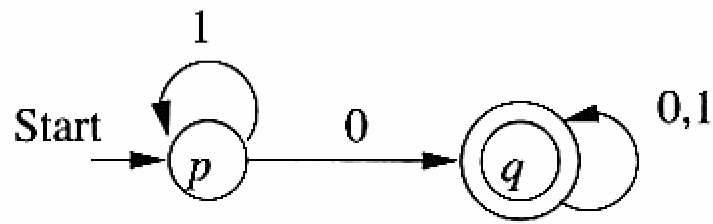
$$L_2 = \{ab^m\} \quad m \geq 0$$

M_1

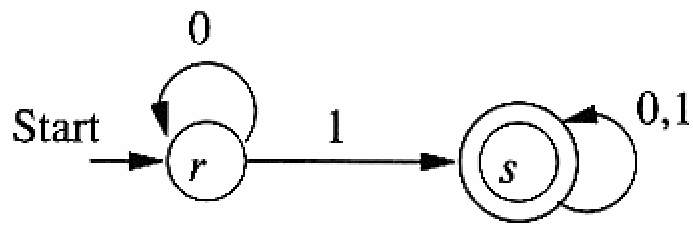


M_2

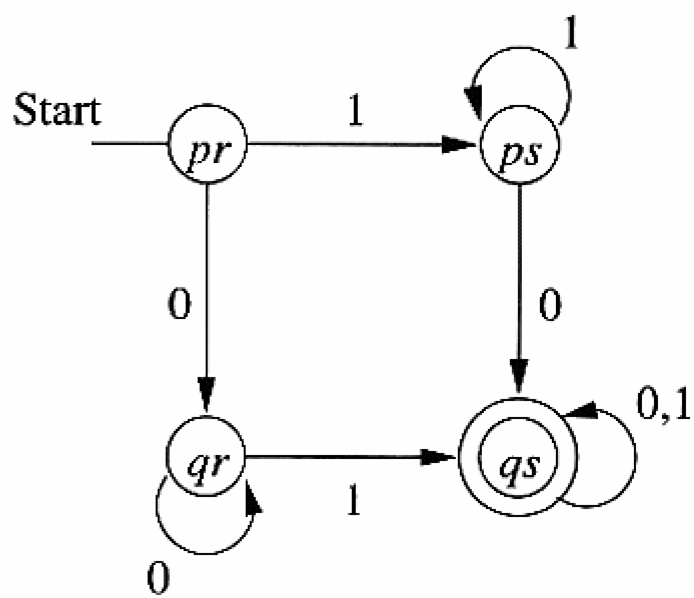




(a)



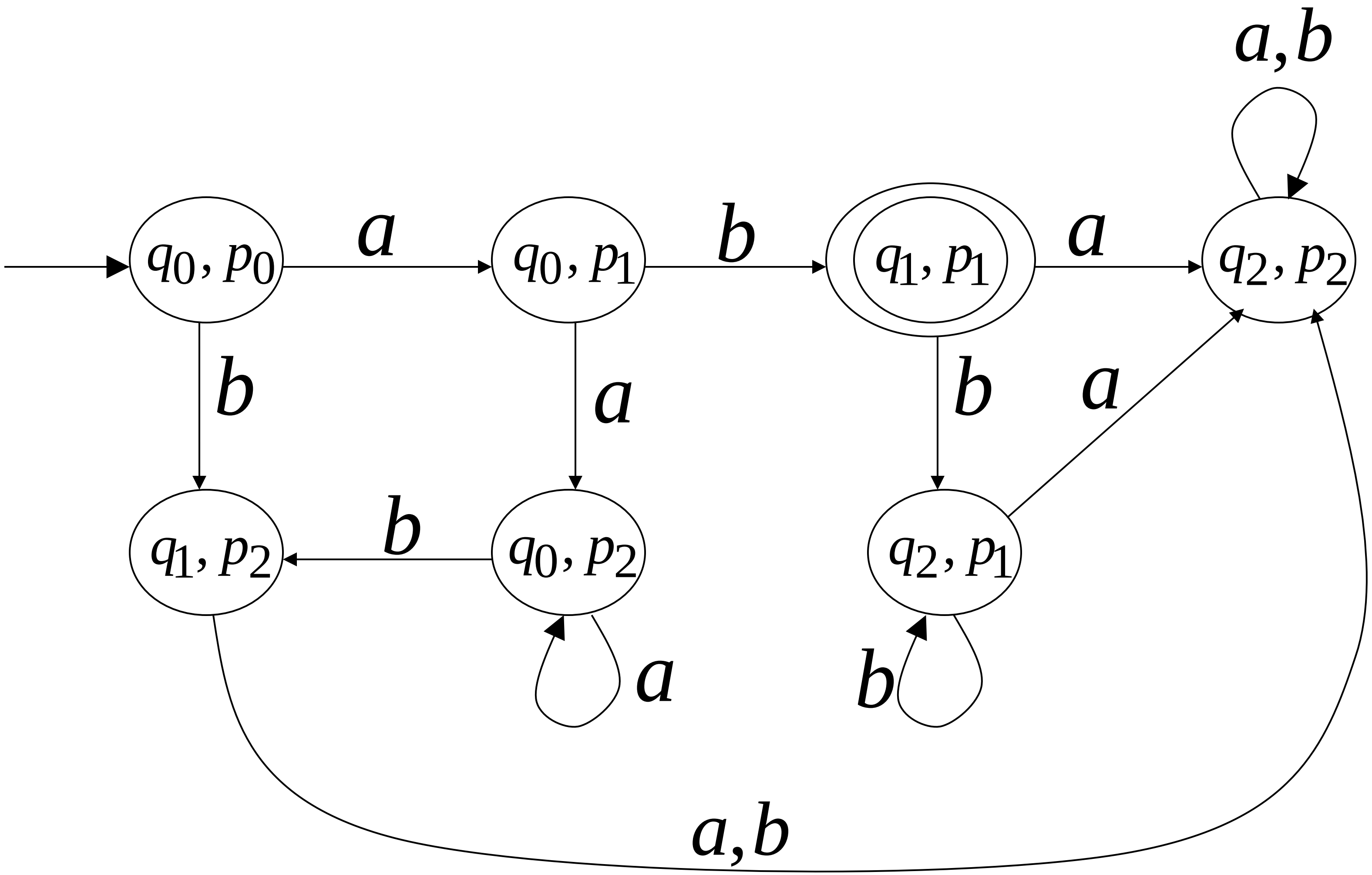
(b)



Construct machine for intersection

Automaton for intersection

$$L = \{a^n b\} \cap \{ab^n\} = \{ab\}$$



Note how easy it was to prove closure under union, star, concatenation with NFAs. Would be much harder with DFAs.