Properties of Regular Languages

For regular languages L_1 and L_2 we will prove that:

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: L_1^*

Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

are also regular languages

We say Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: L_1^*

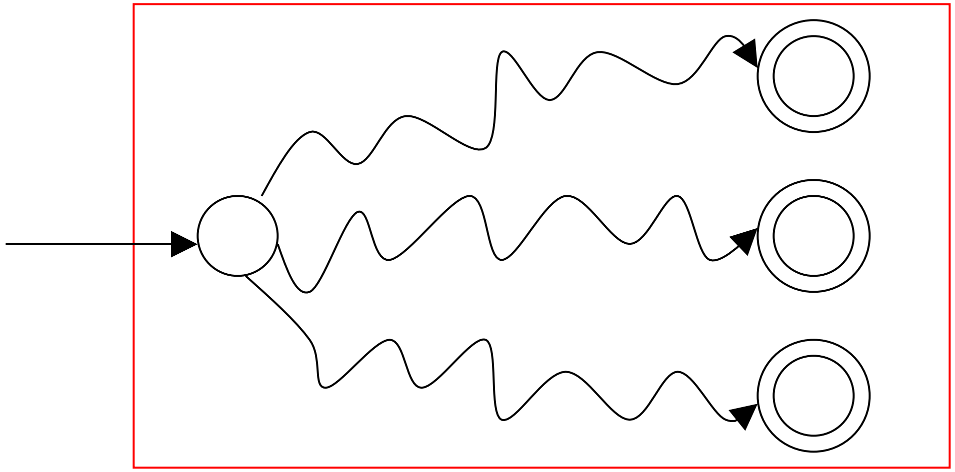
Reversal: L_1^R

Complement: $\overline{L_1}$

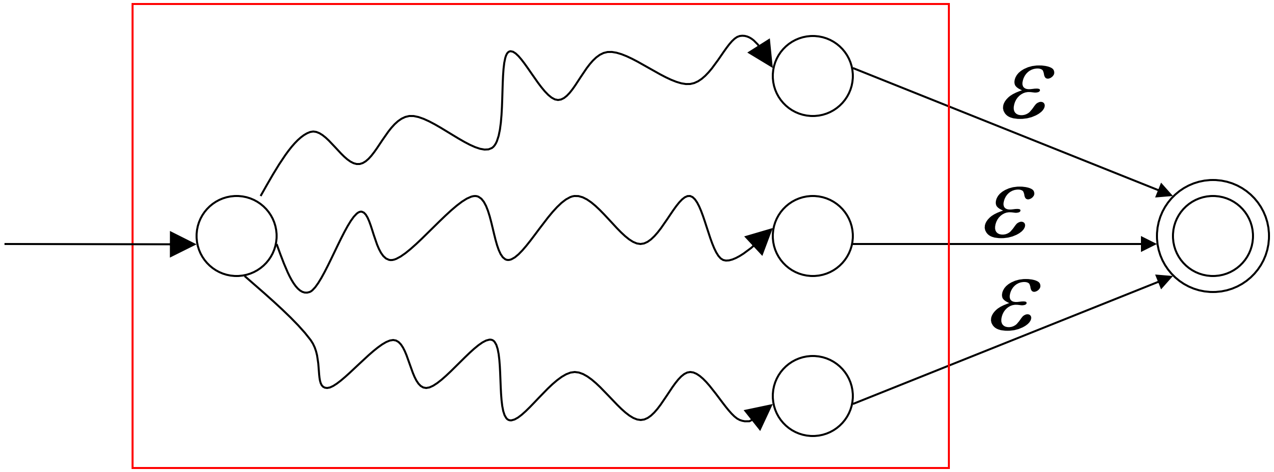
Intersection: $L_1 \cap L_2$

A useful transformation: Use single accept state

NFA with more than one accept state

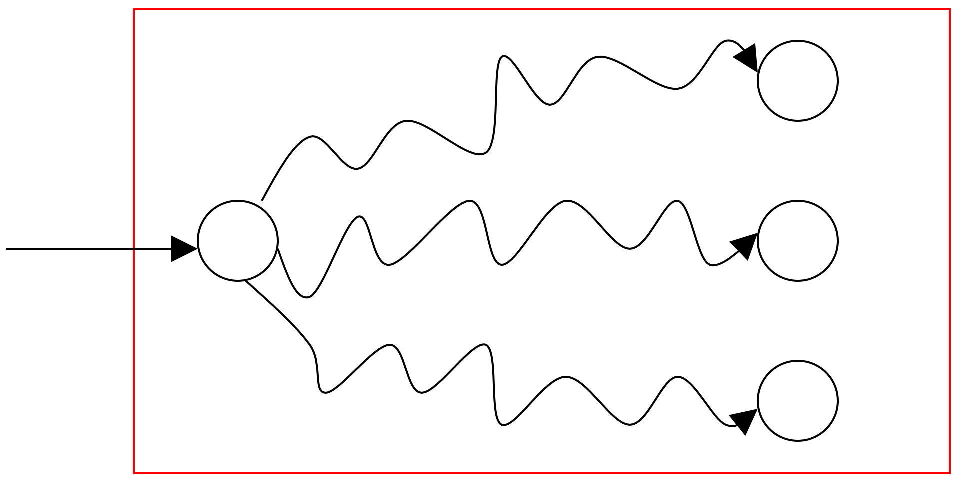


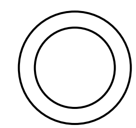
Equivalent NFA with a single accept state



Extreme case

NFA without an accept state





Add an accept state
without transitions

Take Two Languages

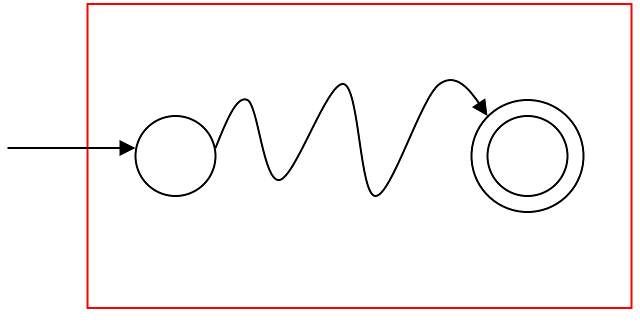
Regular language L_1

Regular language L_2

$$L(M_1) = L_1$$

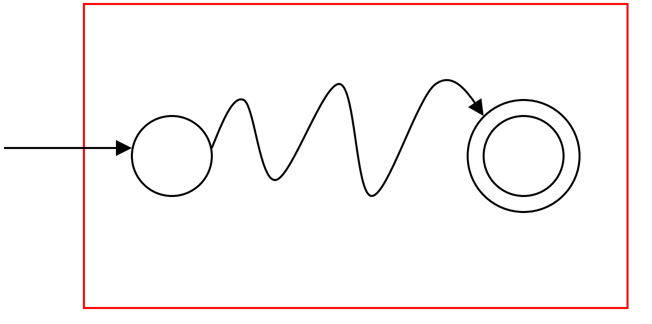
$$L(M_2) = L_2$$

NFA M_1



Single accept state

NFA M_2



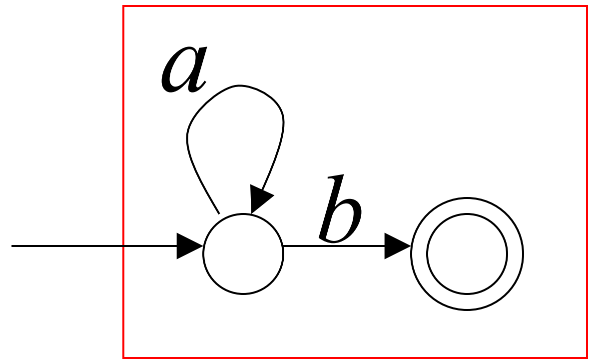
Single accept state

Example

$n \geq 0$

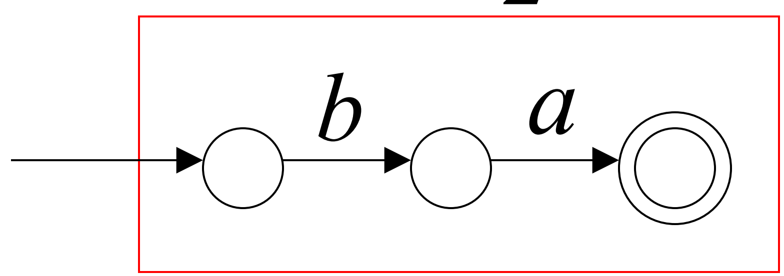
$L_1 = \{a^n b\}$

M_1



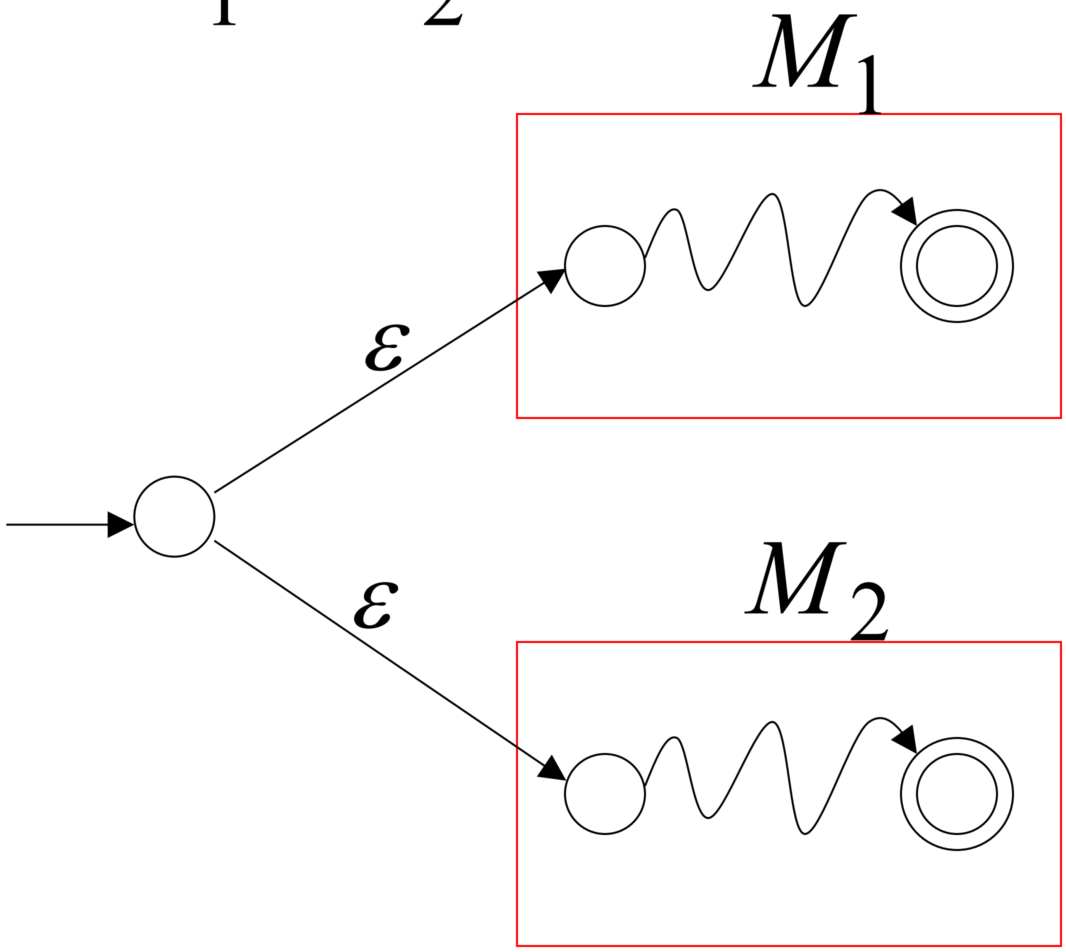
$L_2 = \{ba\}$

M_2



Union

NFA for $L_1 \cup L_2$



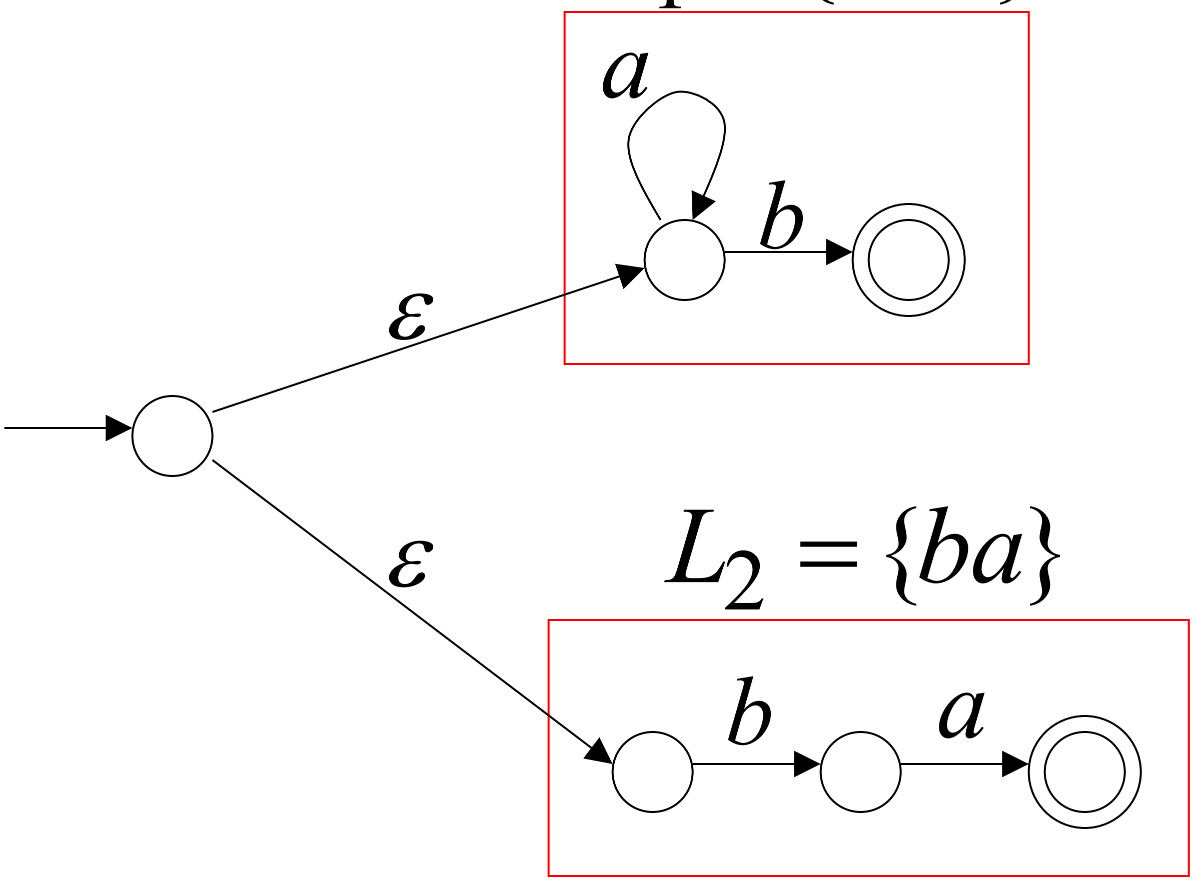
$w \in L_1 \cup L_2 \iff w \in L_1$ or $w \in L_2$

Example

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

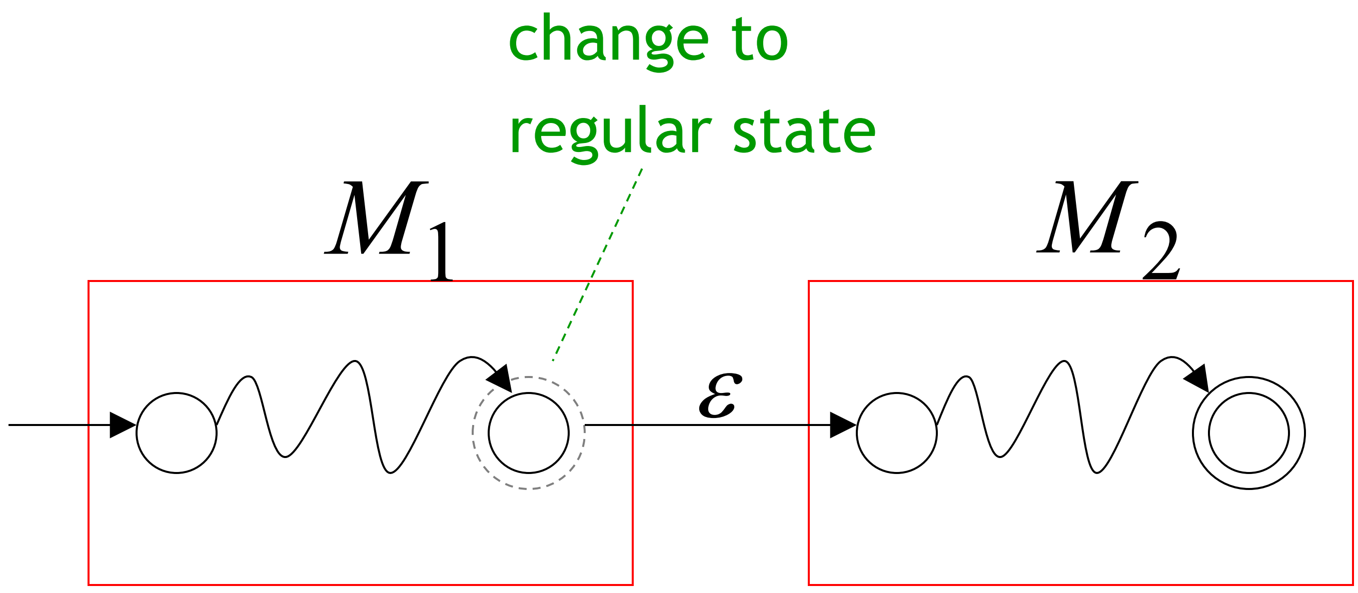
$L_1 = \{a^n b\}$

$L_2 = \{ba\}$



Concatenation

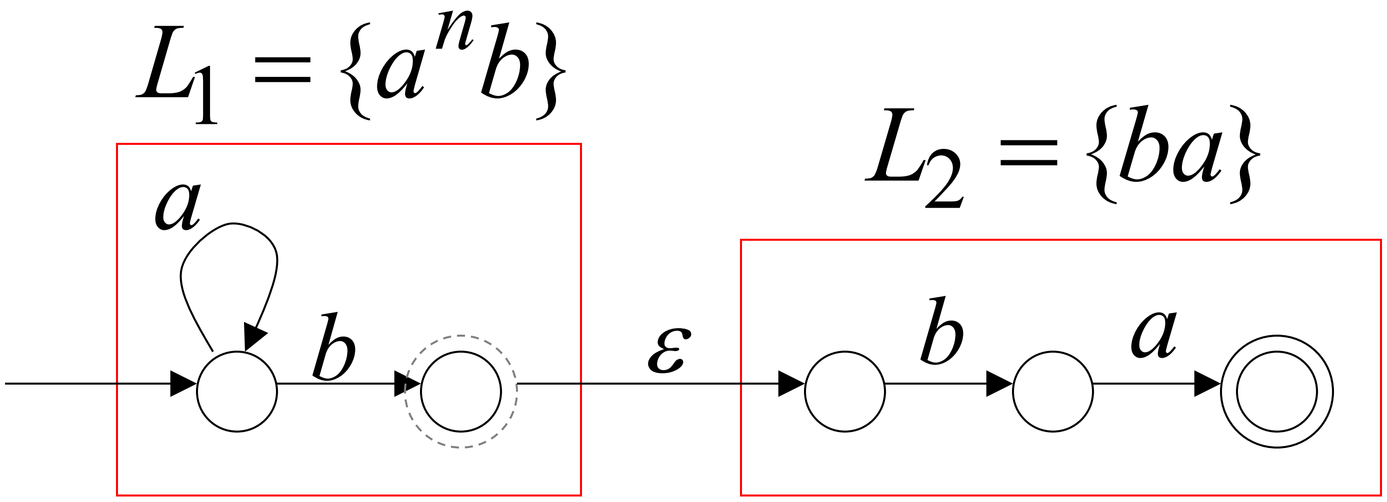
NFA for $L_1 L_2$



$w \in L_1 L_2 \Longleftrightarrow w = w_1 w_2 : \; w_1 \in L_1$ and $w_2 \in L_2$

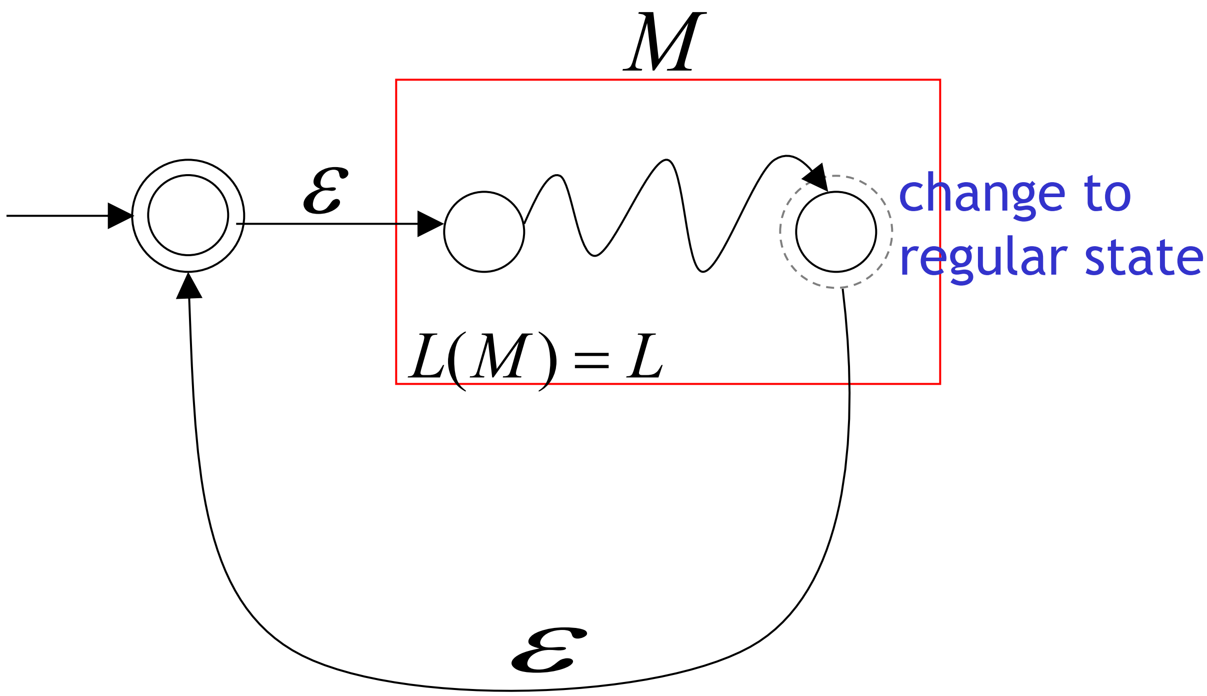
Example

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



Star Operation

NFA for L^*



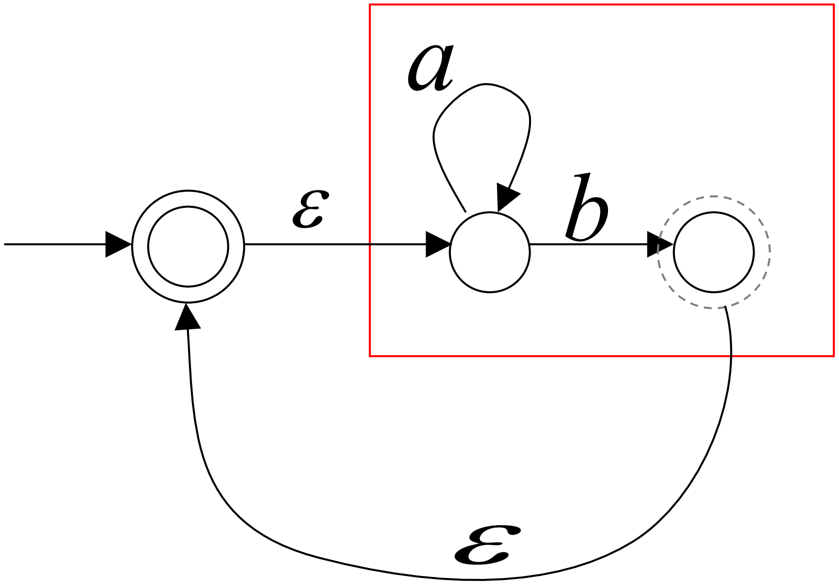
$w \in L^*$ $\Longleftrightarrow$ $w = w_1 w_2 \cdots w_k \; : \; w_i \in L$

or $w = \varepsilon$

Example

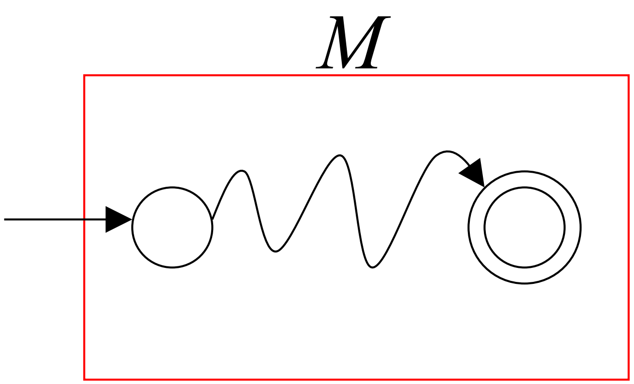
NFA for $L_1^{\,*} = \{a^n b\}^*$

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

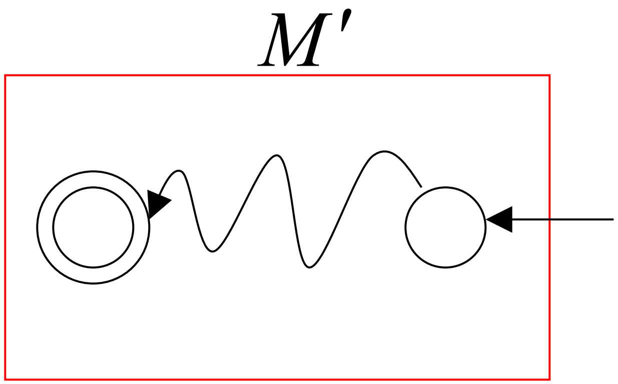


Reverse

NFA for L^R



$L(M) = L$



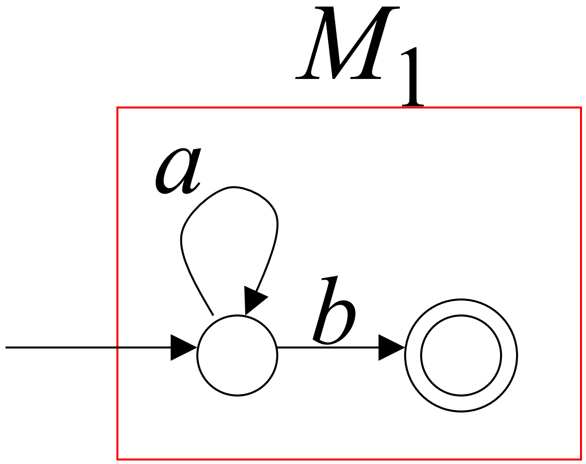
$L(M') = L^R$

1. Reverse all transitions

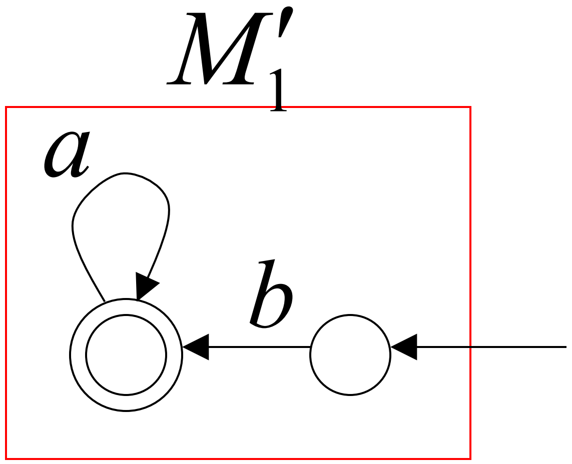
2. Make the initial state accept state
   and the accept state initial state

Example

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

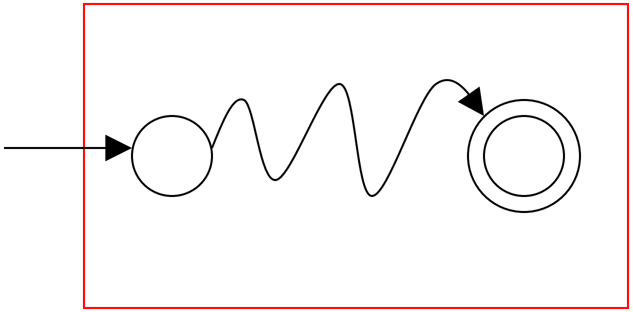


$$L_1^R = \{ba^n\}$$



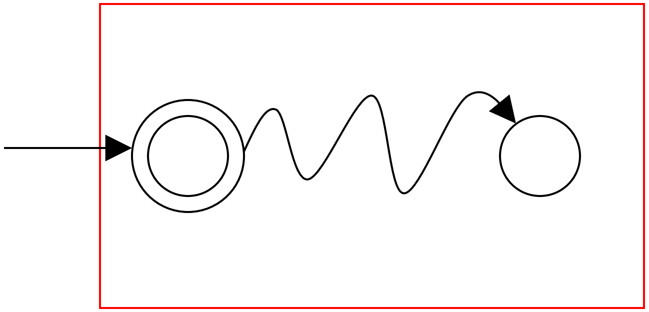
Complement

M



$L(M) = L$

M'

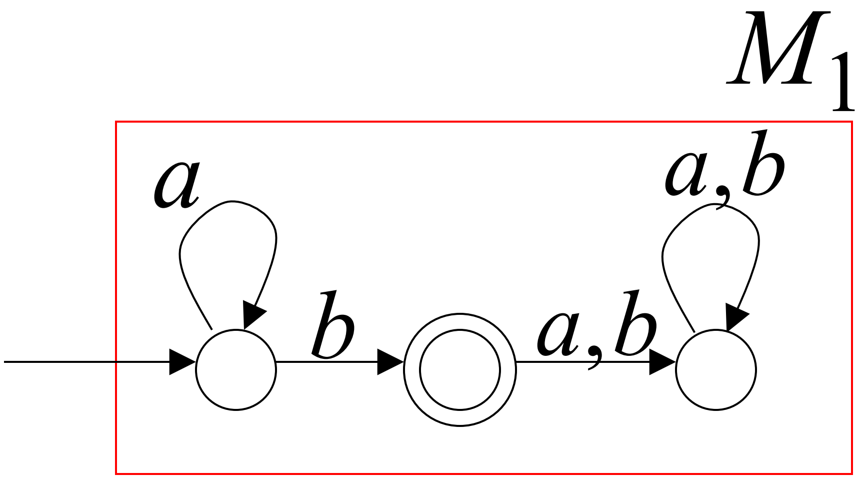


$L(M') = \overline{L}$

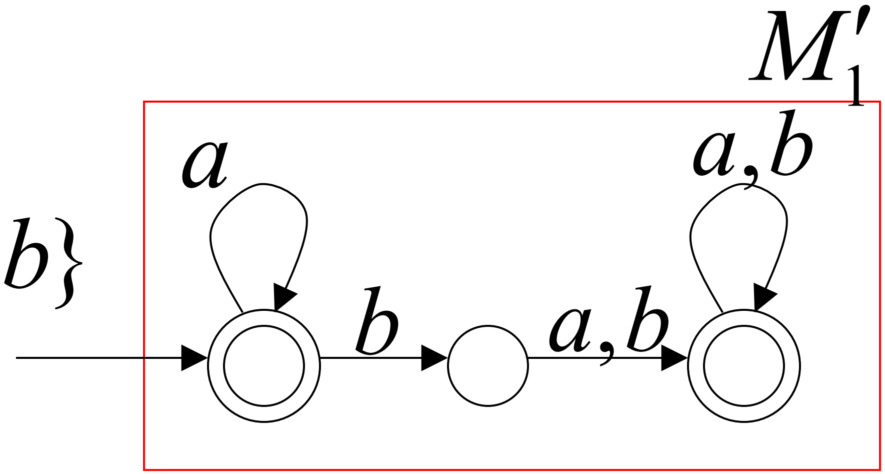
1. Take the **DFA** that accepts L
2. Make accept states regular
   and vice-versa

Example

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



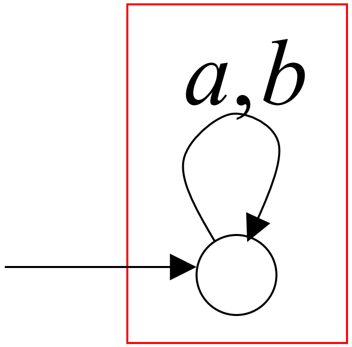
$$\overline{L_1} = \{a,b\}^* - \{a^n b\}$$



DFAs can be used for complement

Make accept states regular and vice-versa

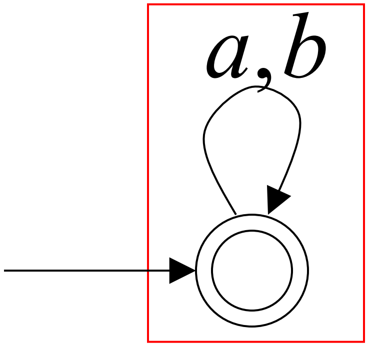
DFA M



$$L(M) = \{\}$$

$$\overline{L(M)} = \Sigma^* = \{a, b\}^*$$

DFA M'



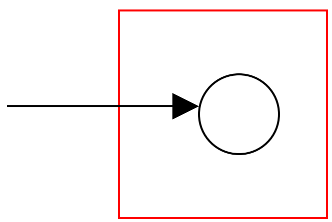
$$L(M') = \{a, b\}^* = \overline{L(M)}$$

It is the complement

NFAs cannot be used for complement

Make accept states regular
and vice-versa does not work!

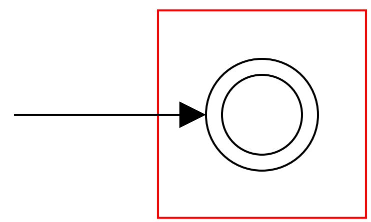
NFA M



$$L(M) = \{\}$$

$$\overline{L(M)} = \Sigma^* = \{a,b\}^*$$

NFA M'



$$L(M') = \{\varepsilon\} \neq \overline{L(M)}$$

It is not the complement

Intersection

L_1 regular

L_2 regular

we show $\Rightarrow$

$L_1 \cap L_2$

regular

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

$L_1\,,\ L_2$ regular, regular

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

DFA for L_1

Machine M_2

DFA for L_2

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

M simulates in parallel M_1 and M_2

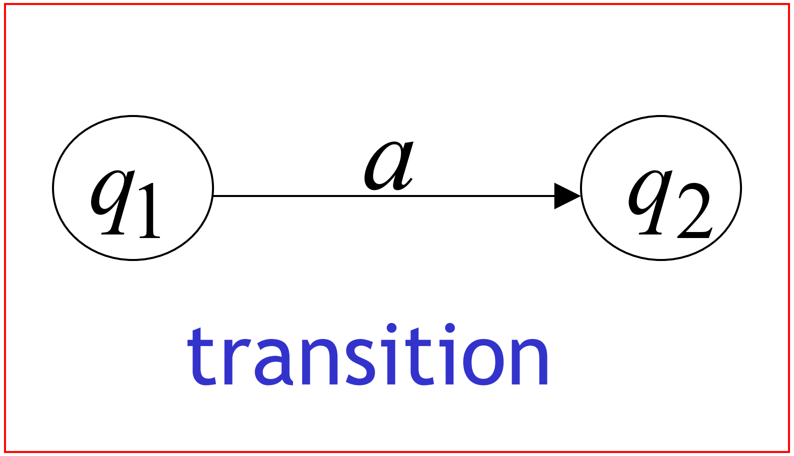
States in M

q_i, p_j

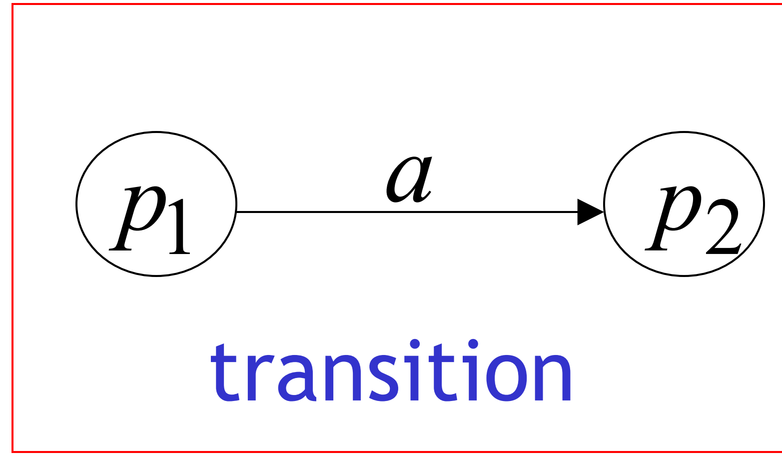
State in M_1

State in M_2

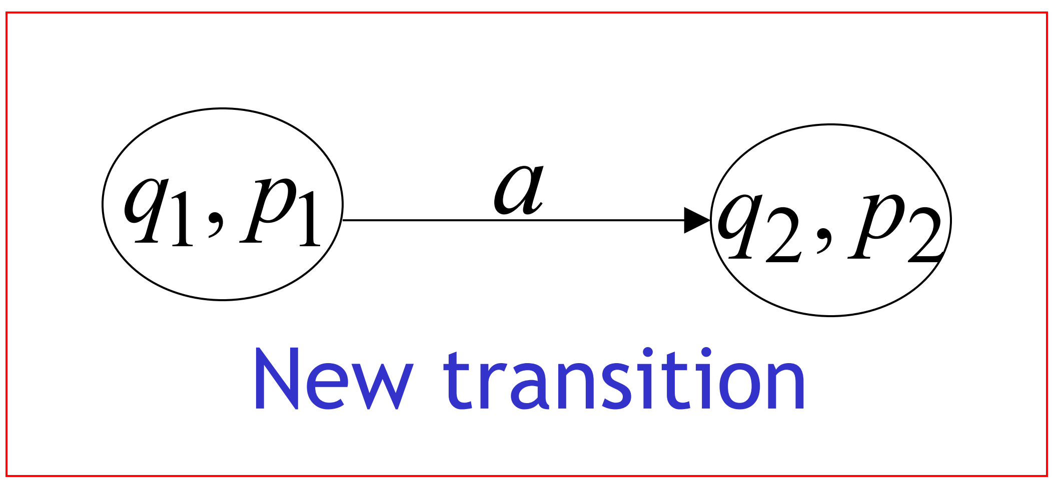
DFA M_1

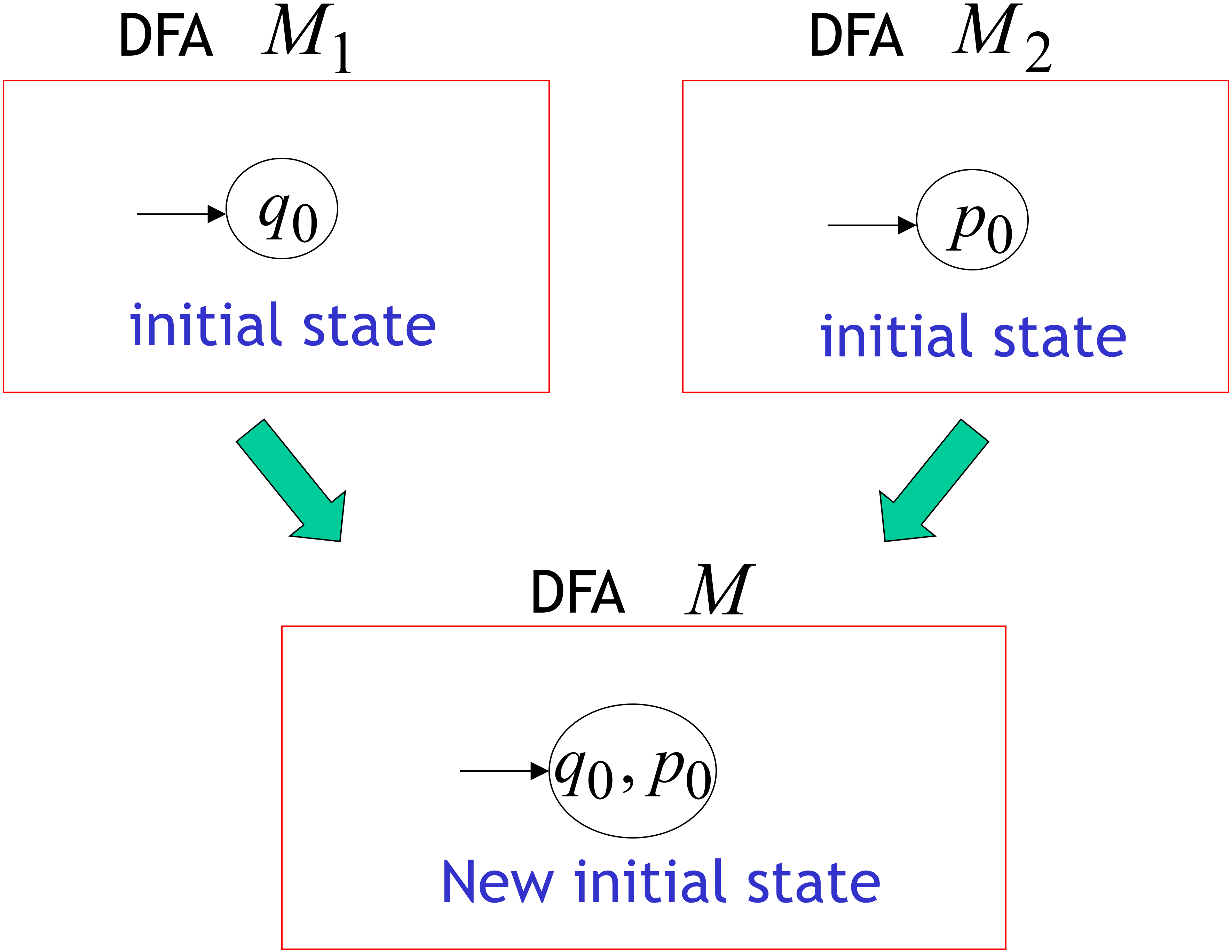


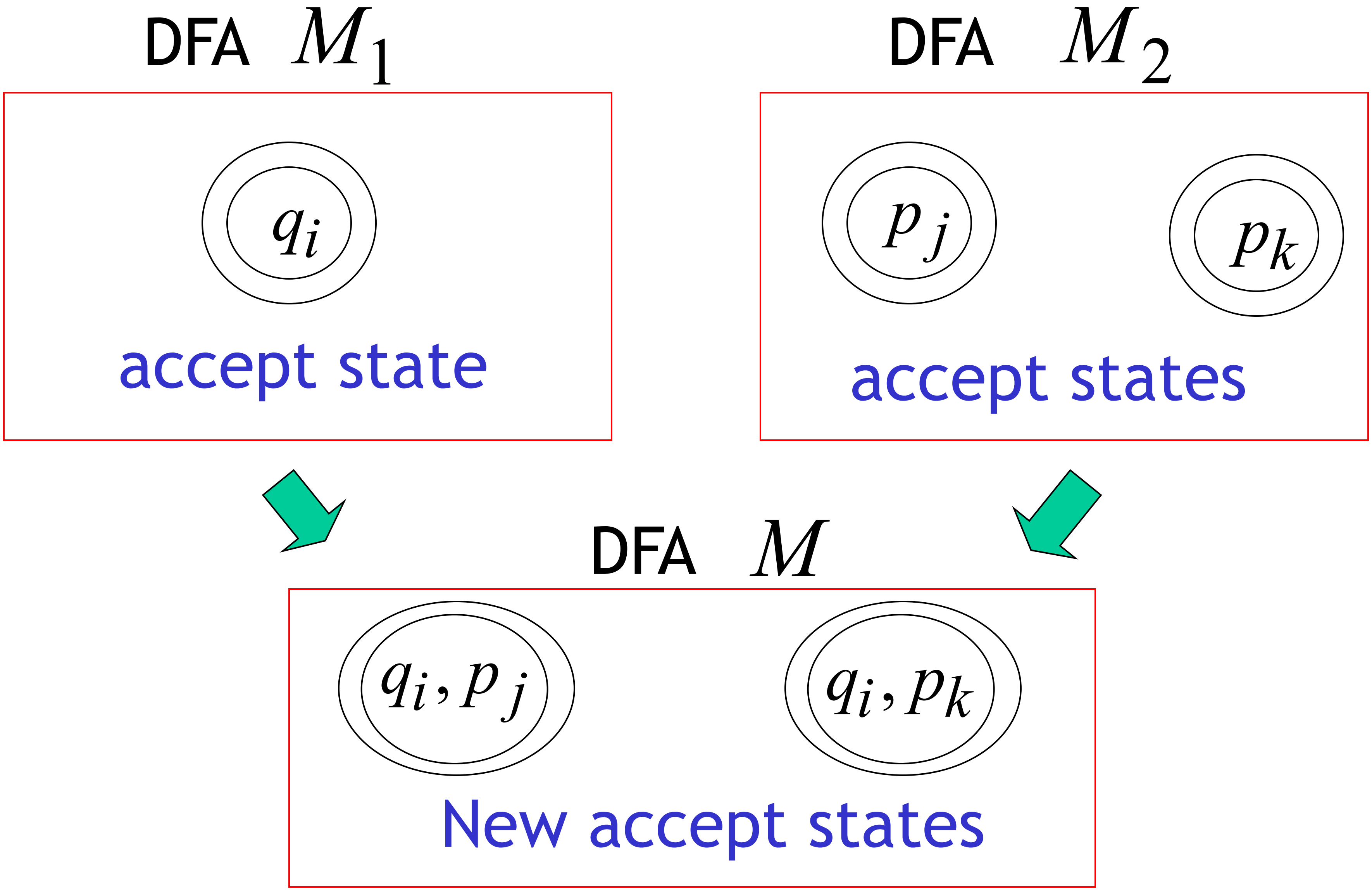
DFA M_2



DFA M



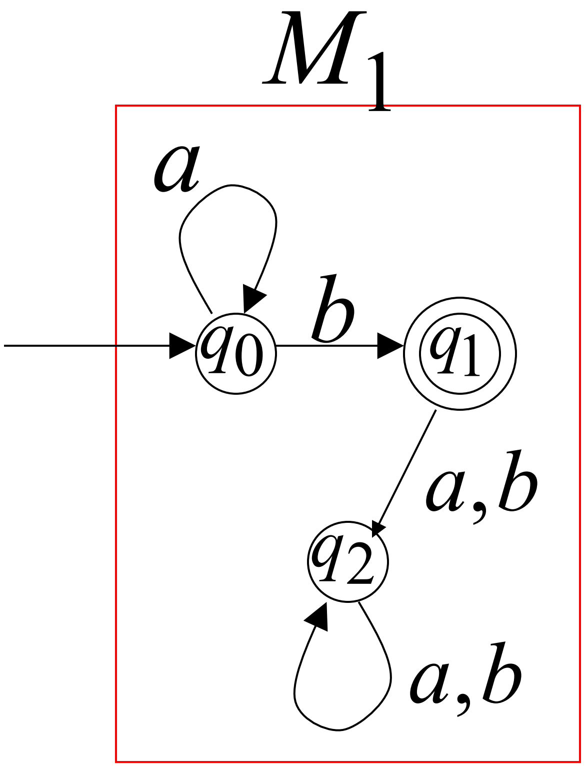

DFA M_1
q_0
initial state
DFA M_2
p_0
initial state
DFA M
q_0, p_0
New initial state

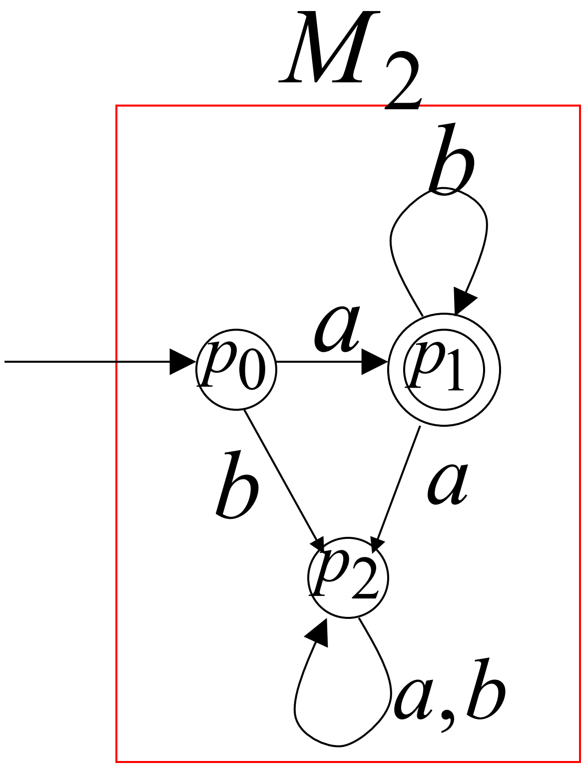

DFA M_1
q_i
accept state
DFA M_2
p_j
p_k
accept states
DFA M
q_i, p_j
q_i, p_k
New accept states

Example:

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

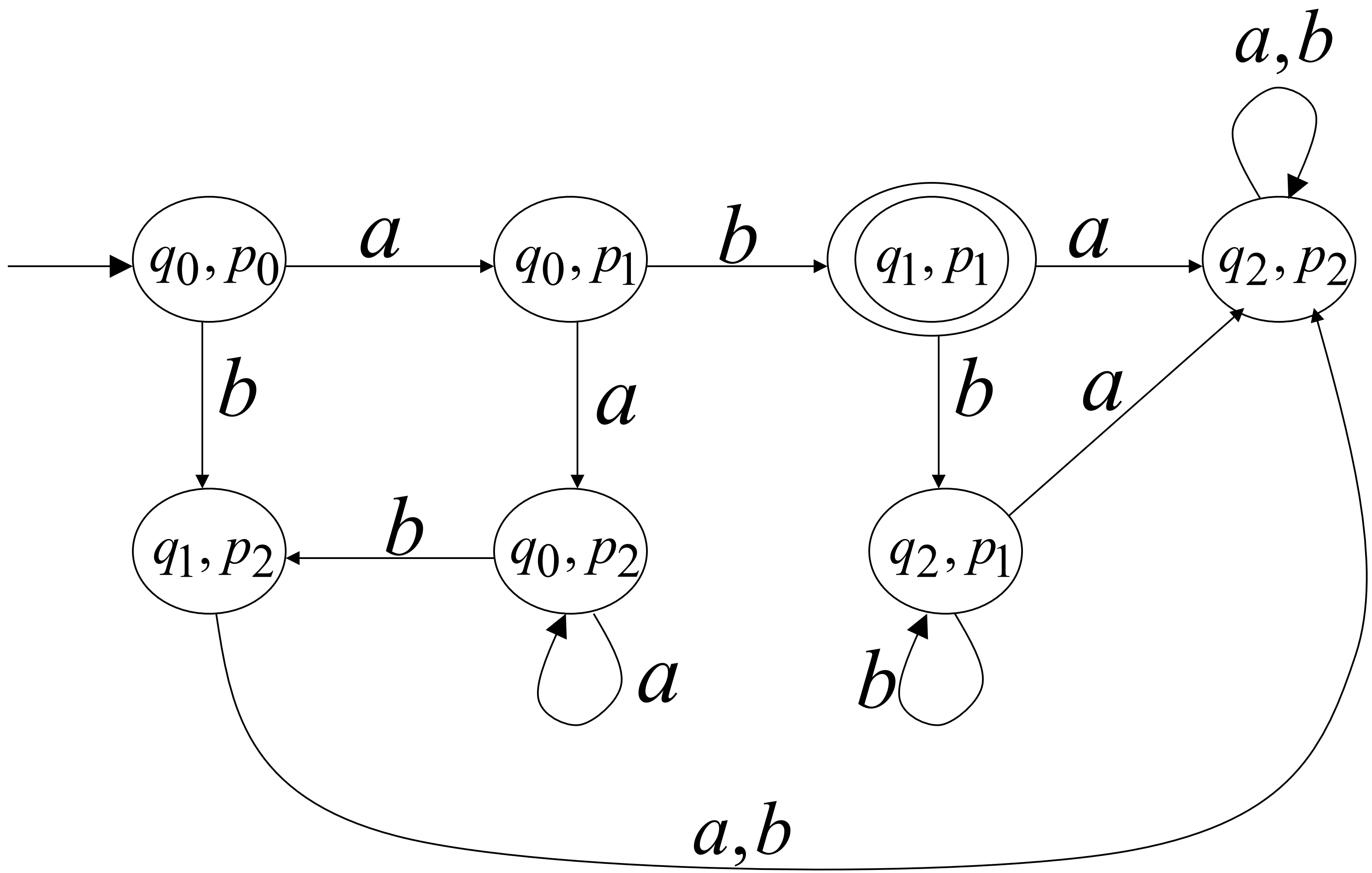
$$L_2 = \{ab^m\} \quad m \geq 0$$





DFA M for intersection

$$L(M) = \{a^n b\} \cap \{ab^m\} = \{ab\}$$

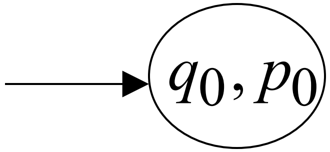


Construction Procedure for Intersection

1. Build initial state
2. For each new state and for each symbol add transition to either an existing state or create a new state and point to it.
3. Repeat step 2 until no new states are added.
4. Designate accept states.

Automaton for intersection

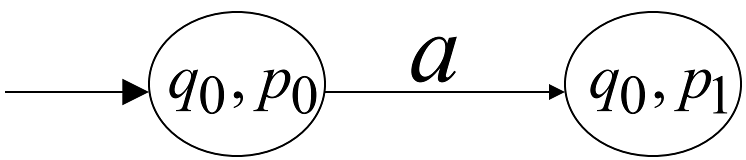
$$L = \{a^n b\} \cap \{ab^m\} = \{ab\}$$



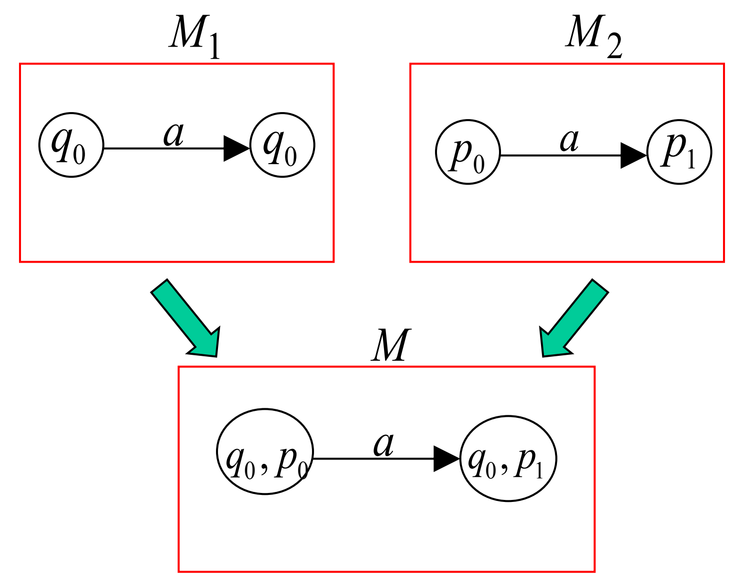
initial state

Automaton for intersection

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

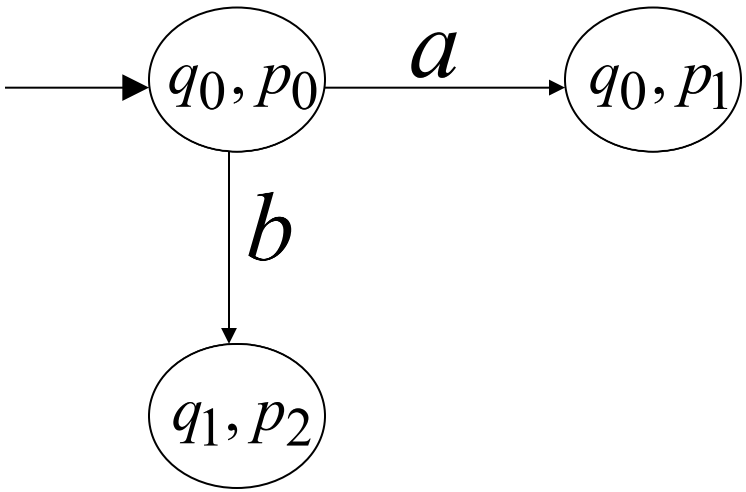


add transition and new state
for symbol a



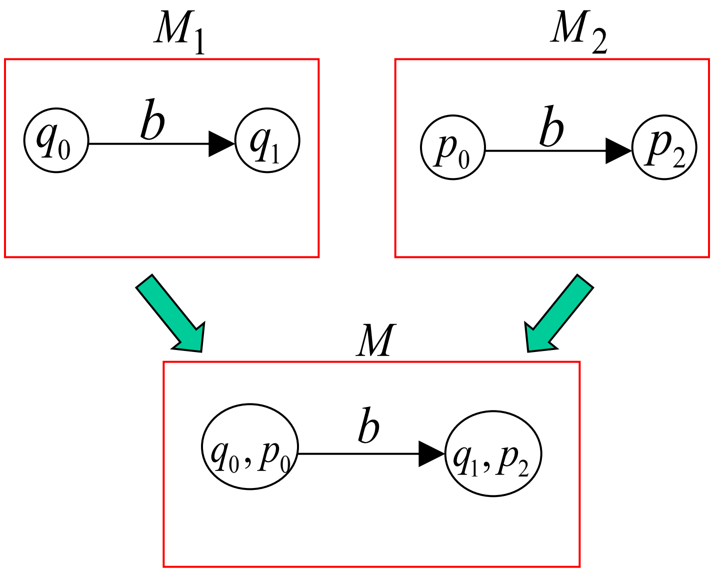
Automaton for intersection

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



add transition and new state

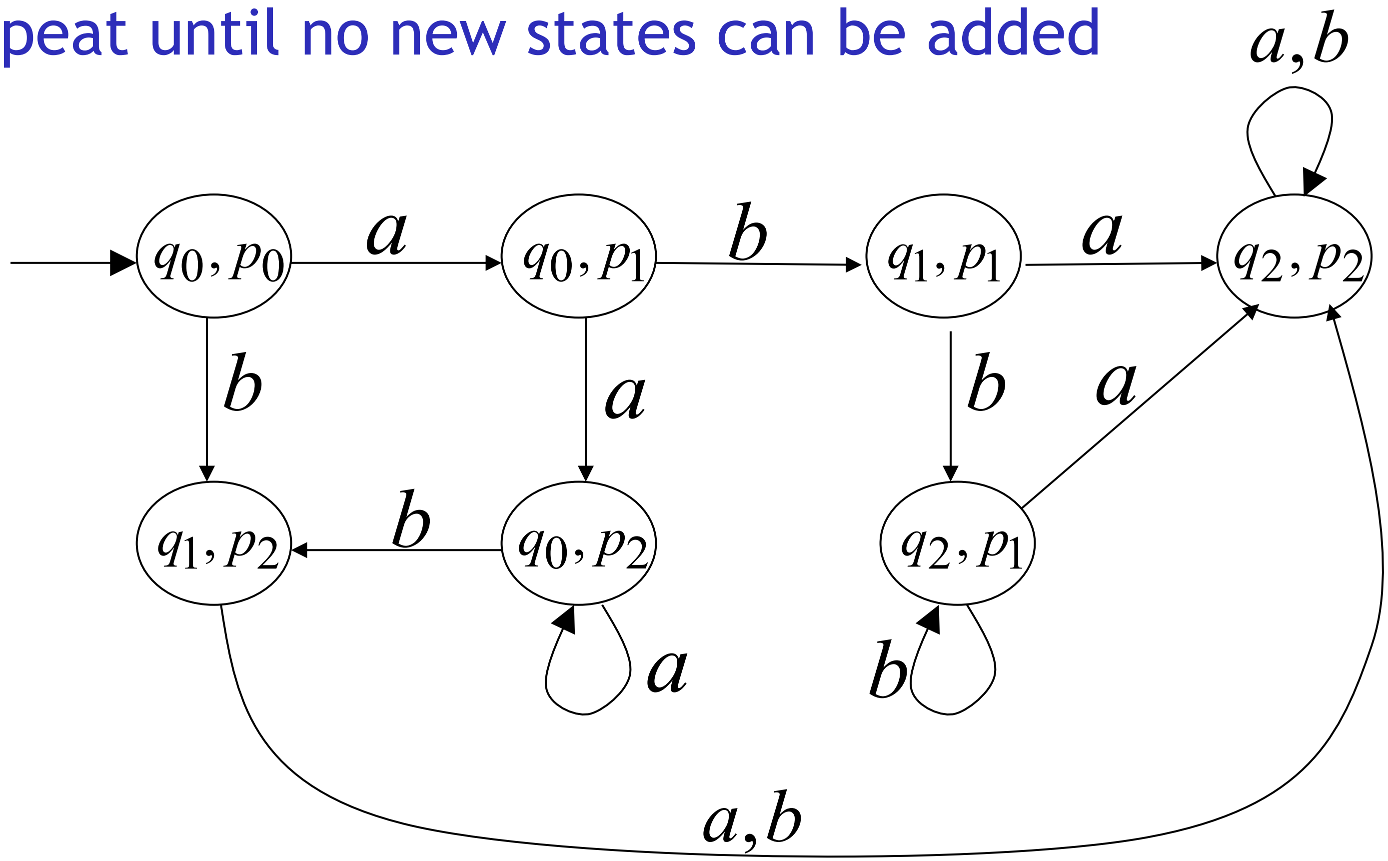
for symbol b



Automaton for intersection

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

Repeat until no new states can be added

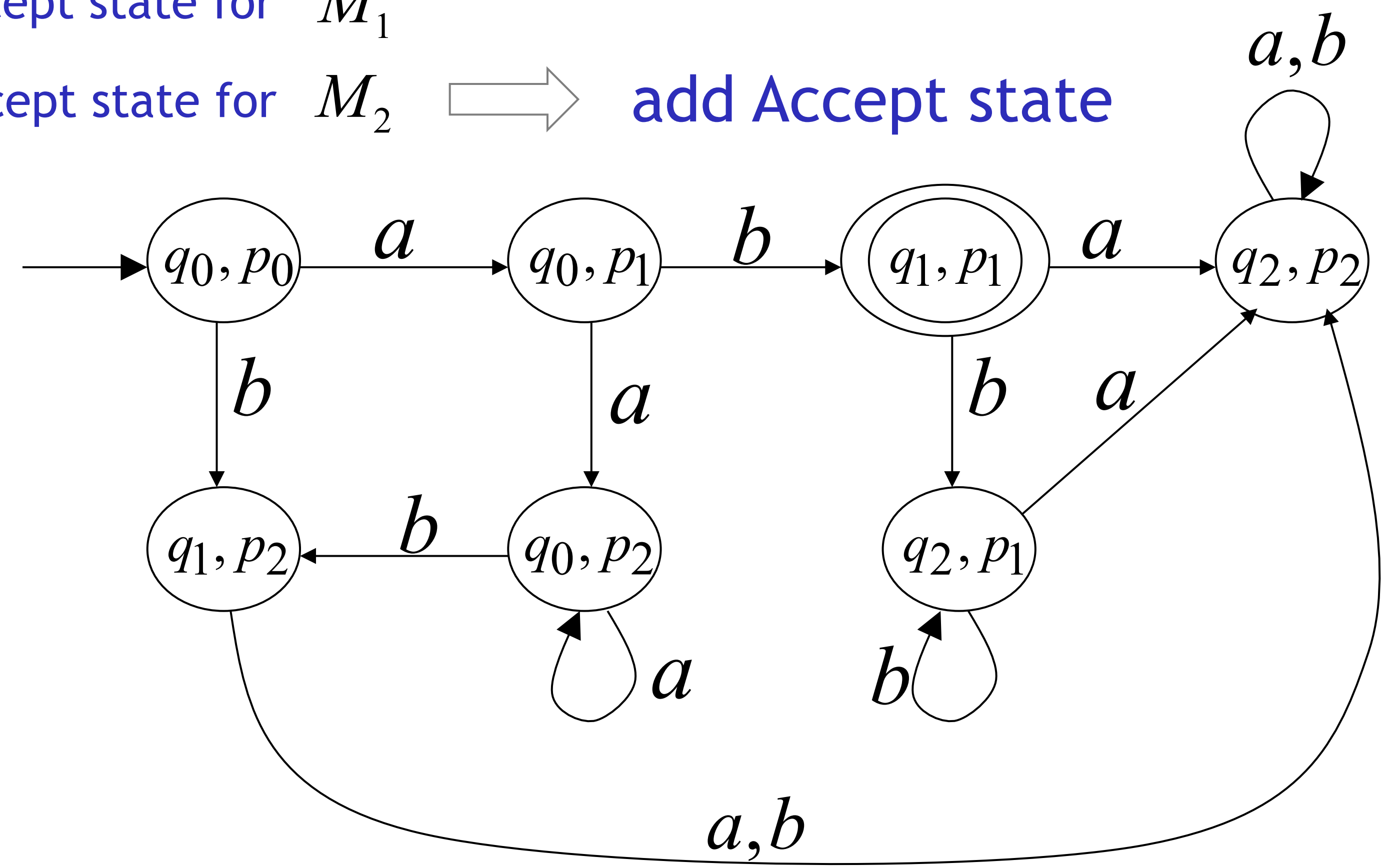


Automaton for intersection

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

q_1 accept state for M_1

p_1 accept state for M_2 ⇨ add Accept state



Intersection DFA M :

Simulates in parallel M_1 and M_2

Accepts string w if and only if:

M_1 accepts string w

and M_2 accepts string w

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