Properties of Regular Languages and Regular Expressions

For regular languages L_1 and L_2 :

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 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: $L_1 \cap L_2$

Take two languages

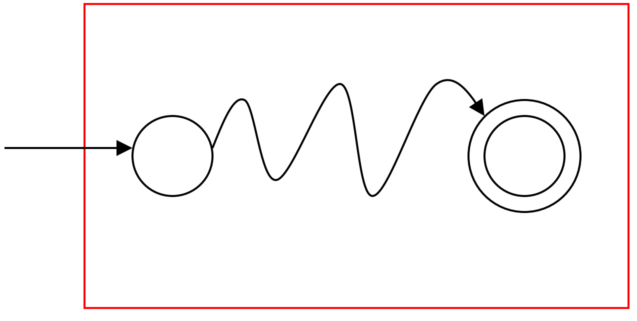
Regular language L_1

Regular language L_2

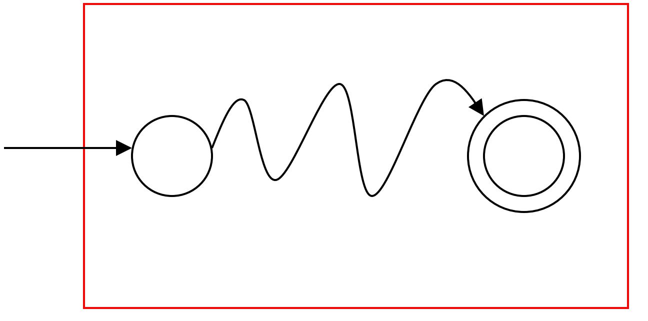
$$L(M_1) = L_1$$

$$L(M_2) = L_2$$

NFA M_1



NFA M_2



Single accepting state

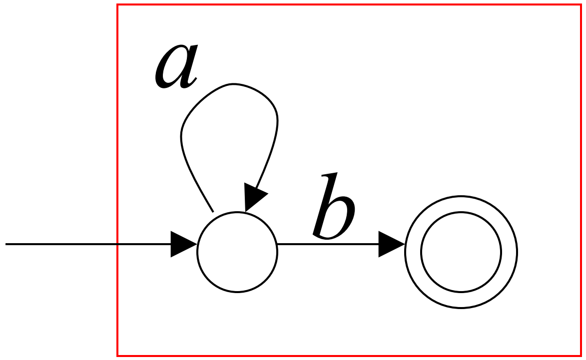
Single accepting state

Example

$n \geq 0$

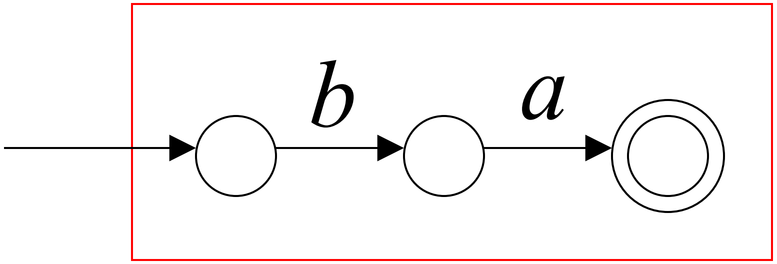
$L_1 = \{a^n b\}$

M_1



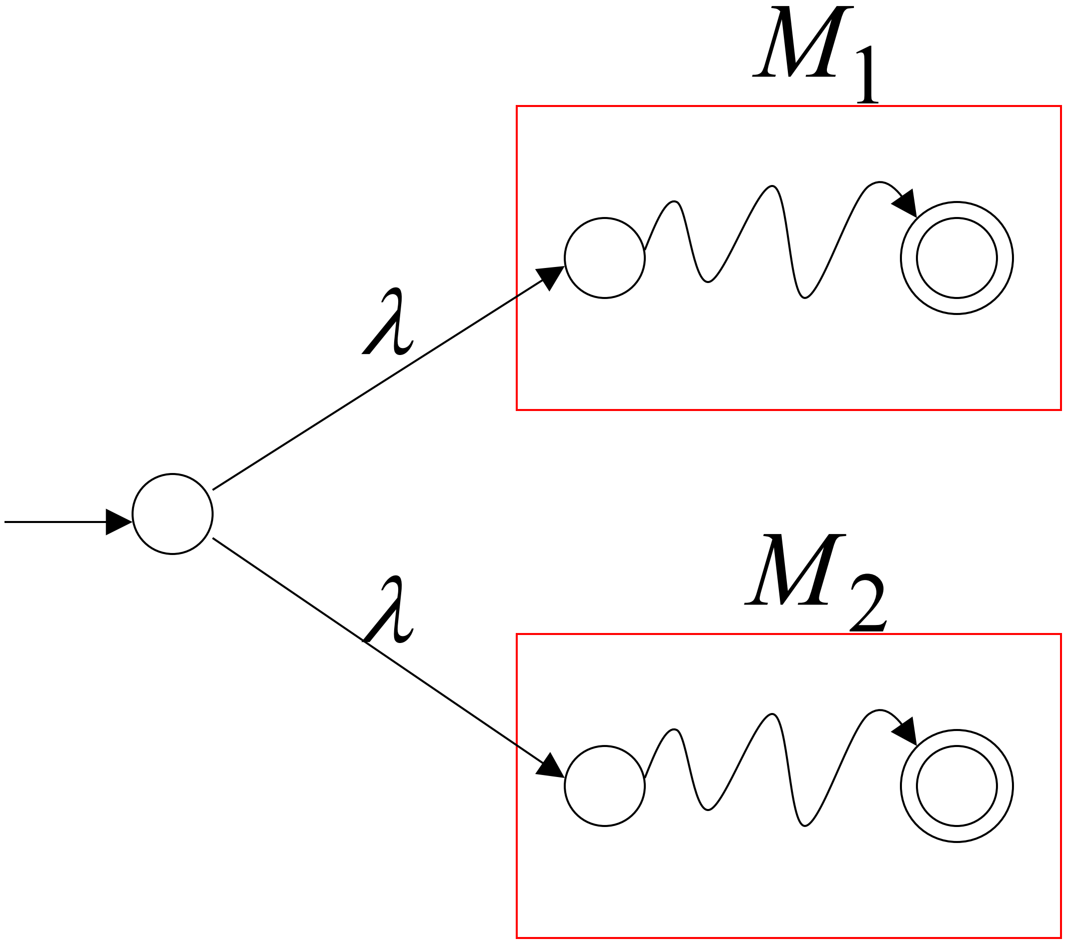
$L_2 = \{ba\}$

M_2



Union

NFA for $L_1 \cup L_2$

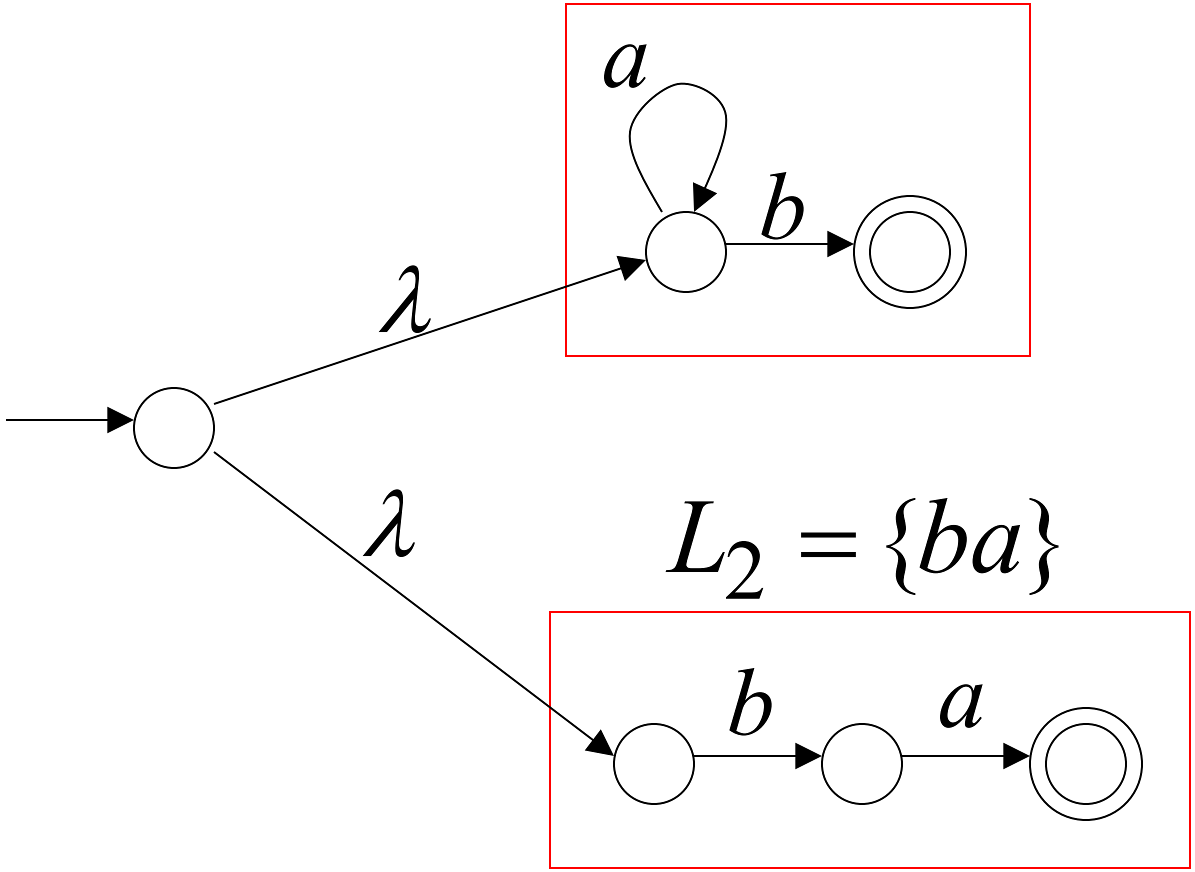


Example

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

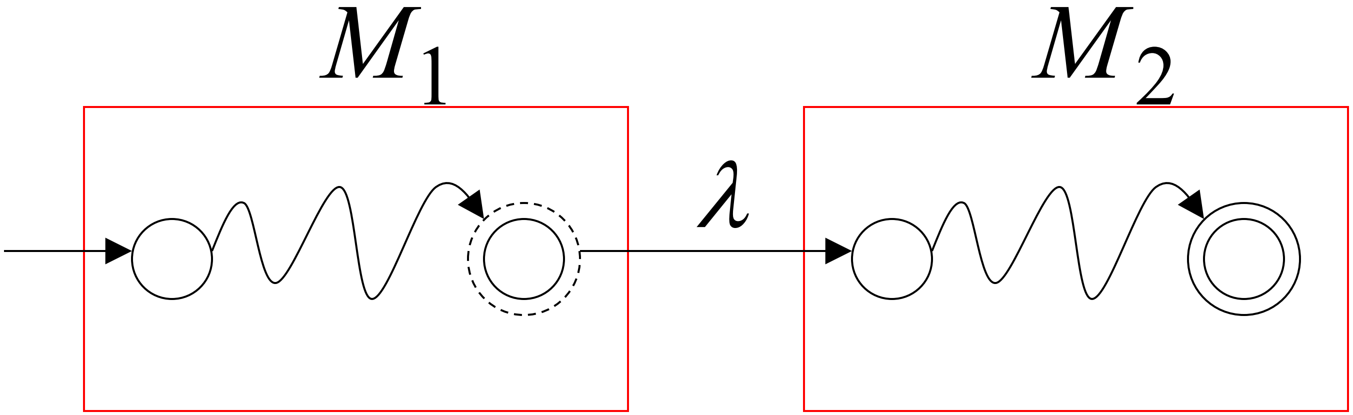
$L_1 = \{a^n b\}$

$L_2 = \{ba\}$



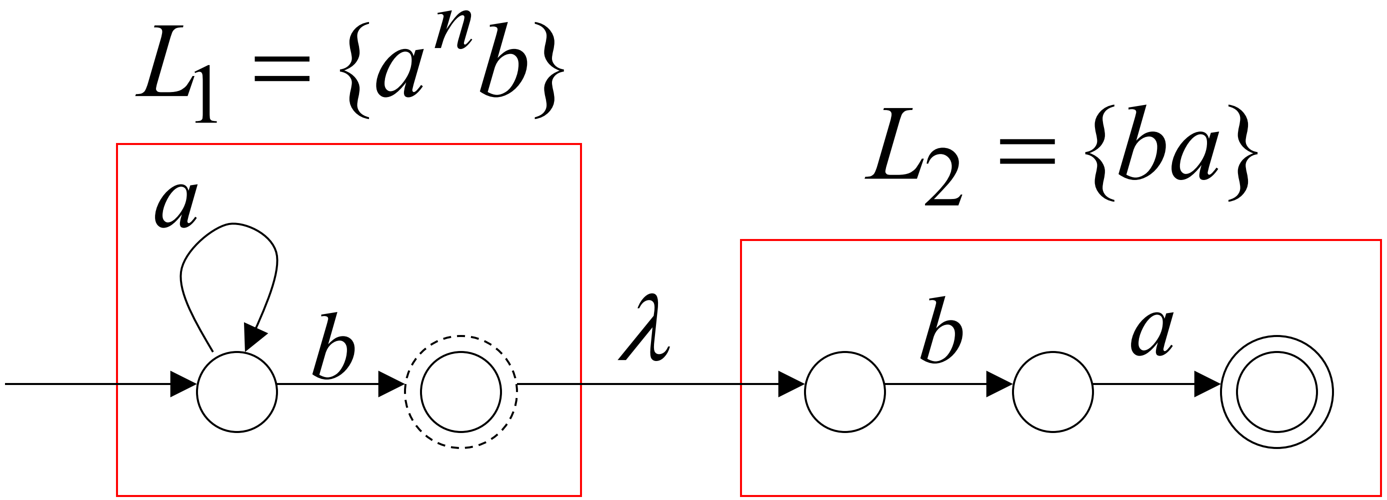
Concatenation

NFA for $L_1 L_2$



Example

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



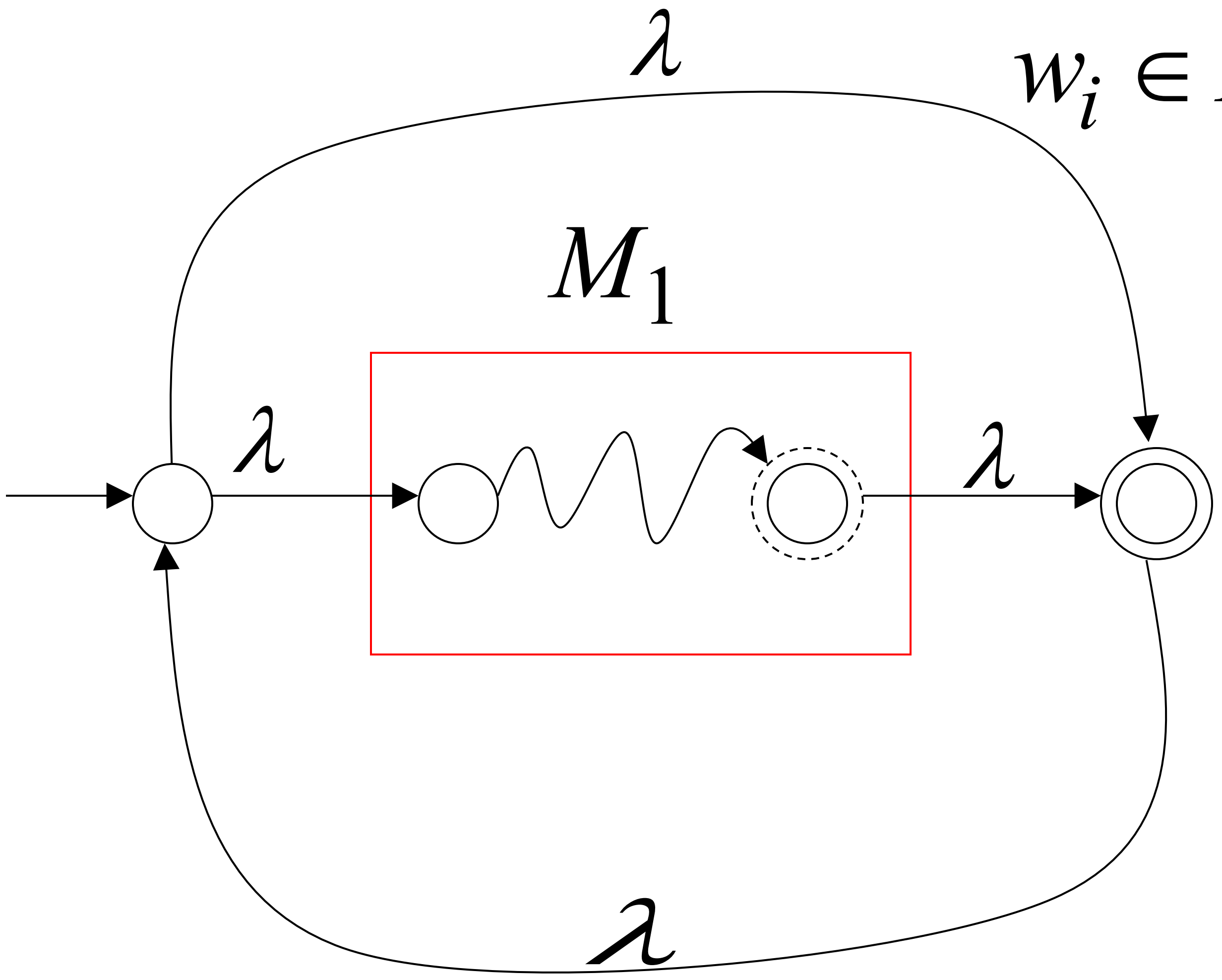
Star Operation

NFA for $L_1{}^*$

$w = w_1 w_2 \cdots w_k$

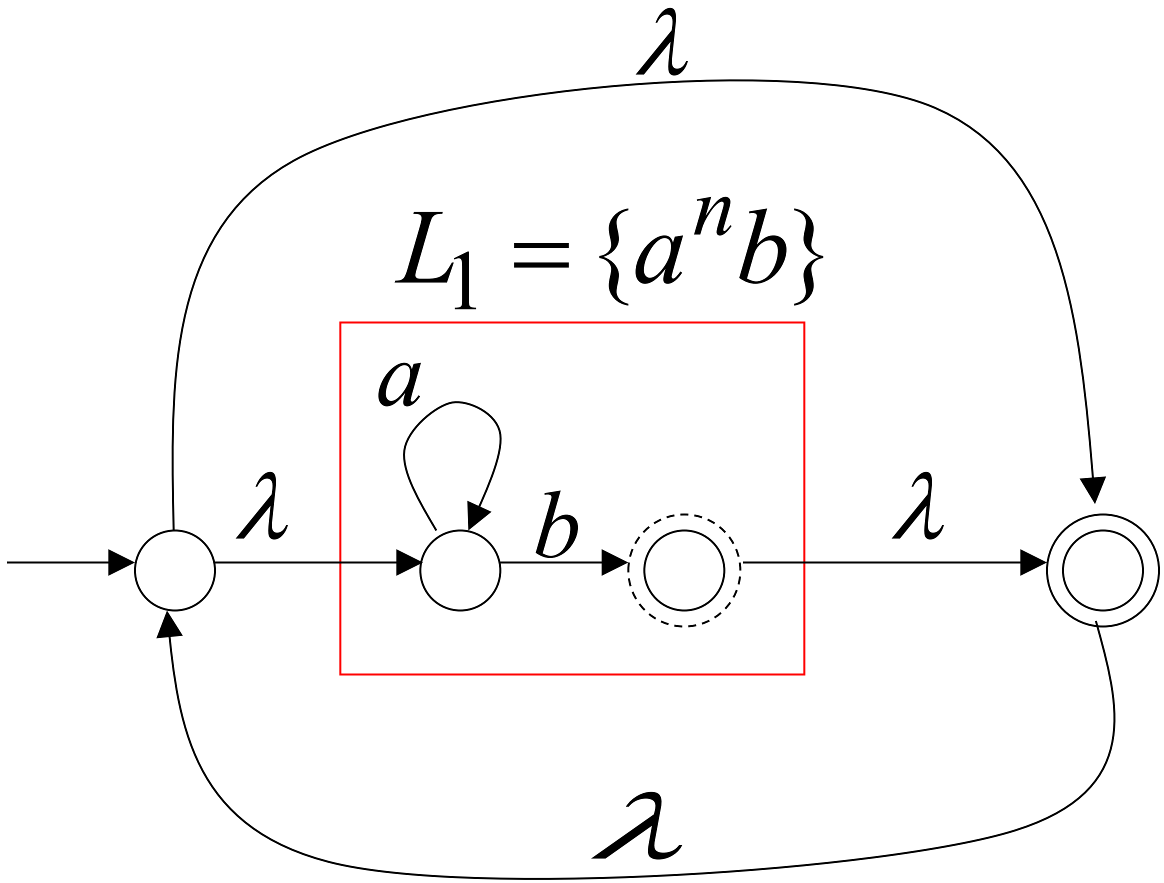
$w_i \in L_1$

$\lambda \in L_1{}^*$



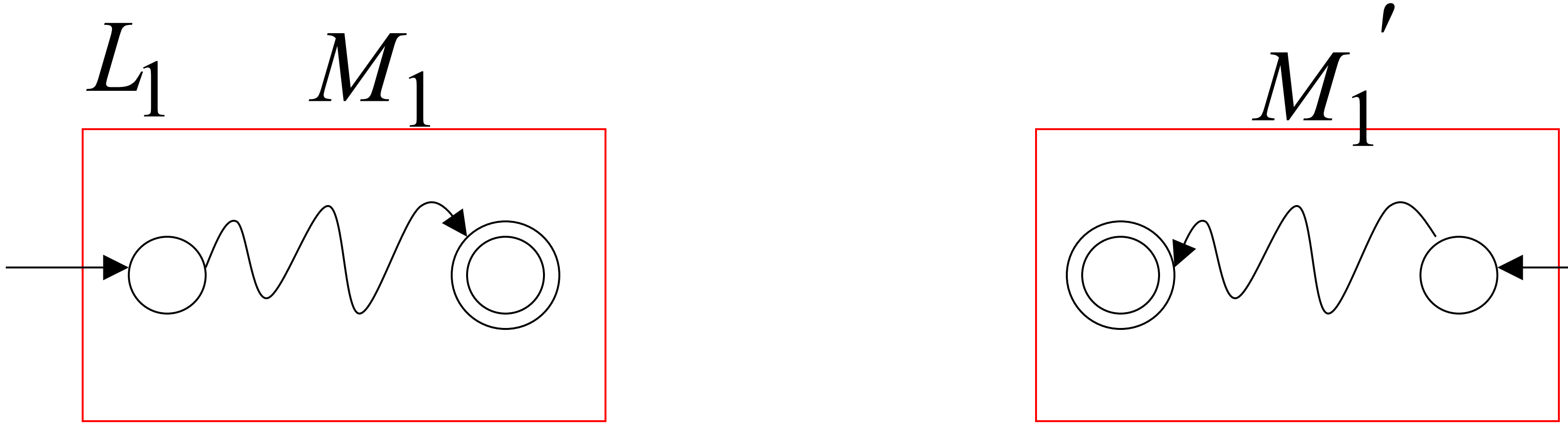
Example

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



Reverse

NFA for L_1^R



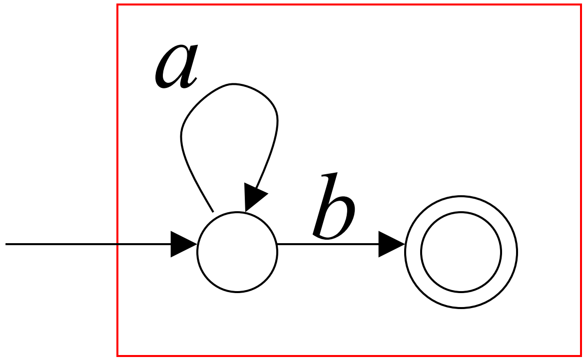
1. Reverse all transitions

2. Make initial state accepting state and vice versa

Example

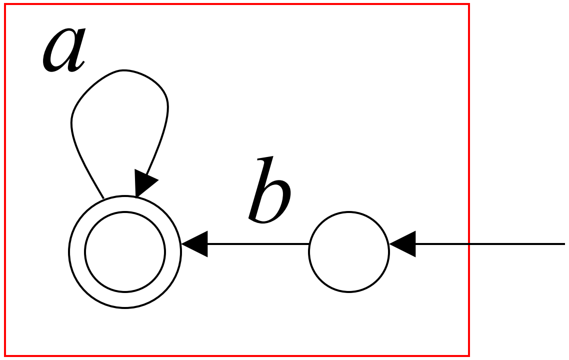
$$L_1 = \{a^n b\}$$

M_1

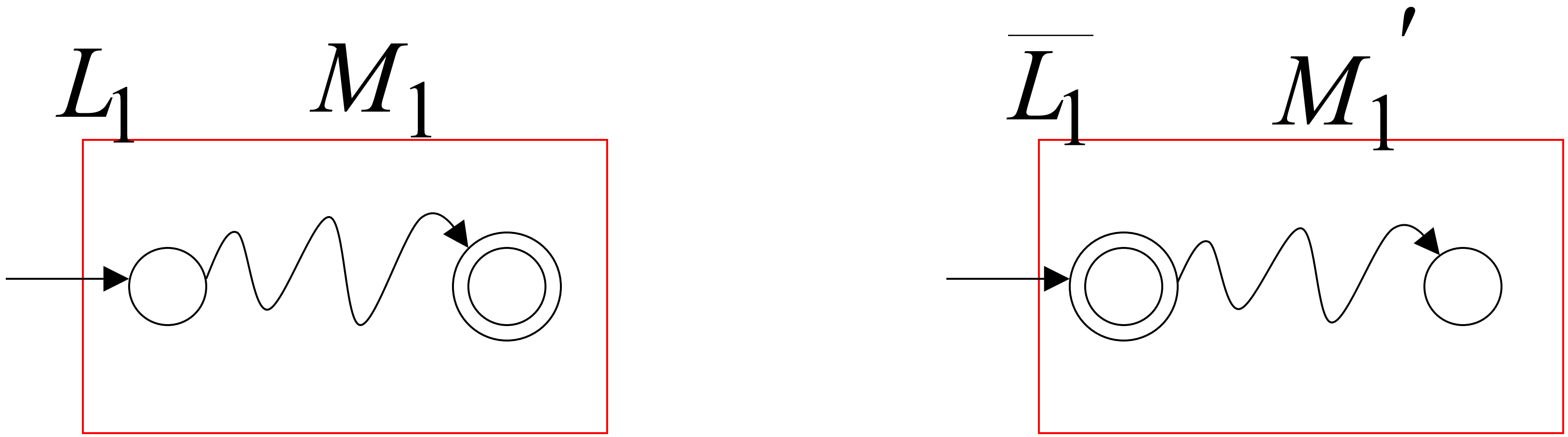


$$L_1^R = \{ba^n\}$$

M_1'



Complement



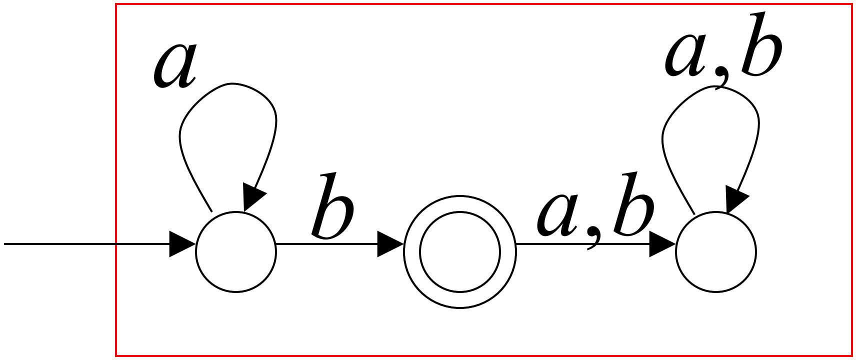
1. Take the DFA that accepts L_1

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

Example

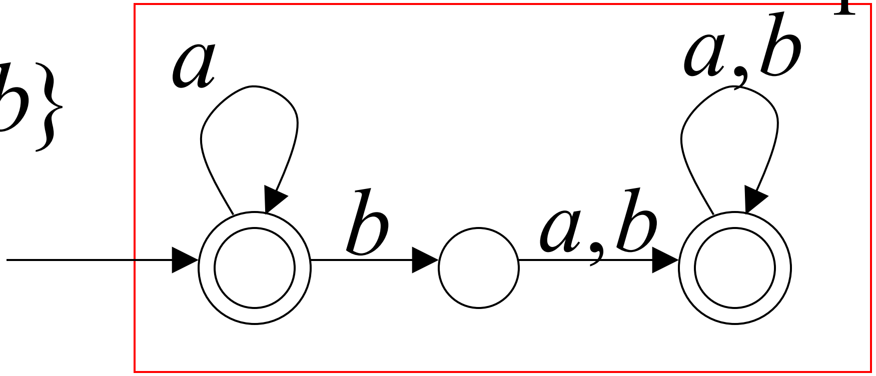
$$L_1 = \{a^n b\}$$

M_1



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

M_1'



Intersection Closure

Machine M_1

DFA for L_1

Machine M_2

DFA for L_2

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

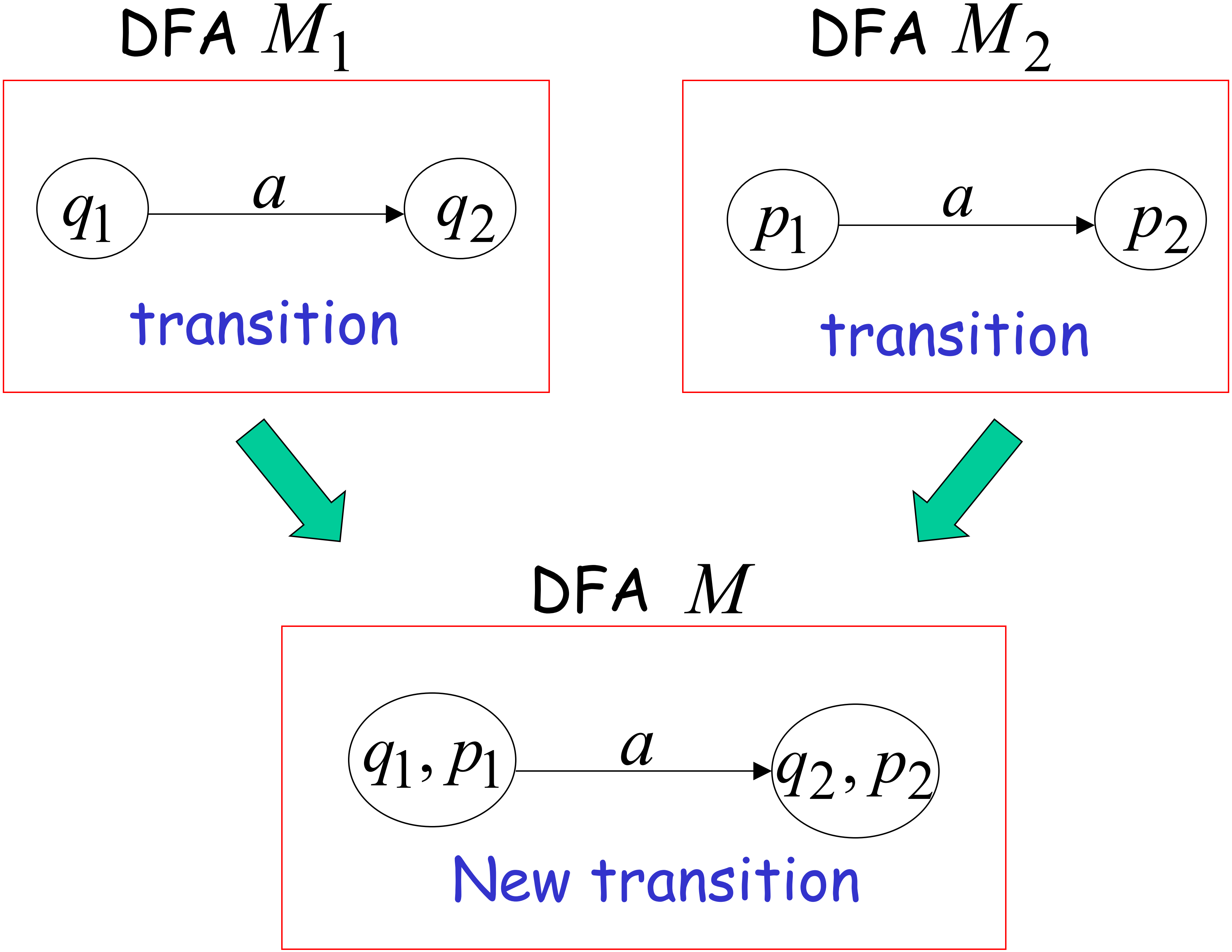
M simulates in parallel M_1 and M_2

States in M

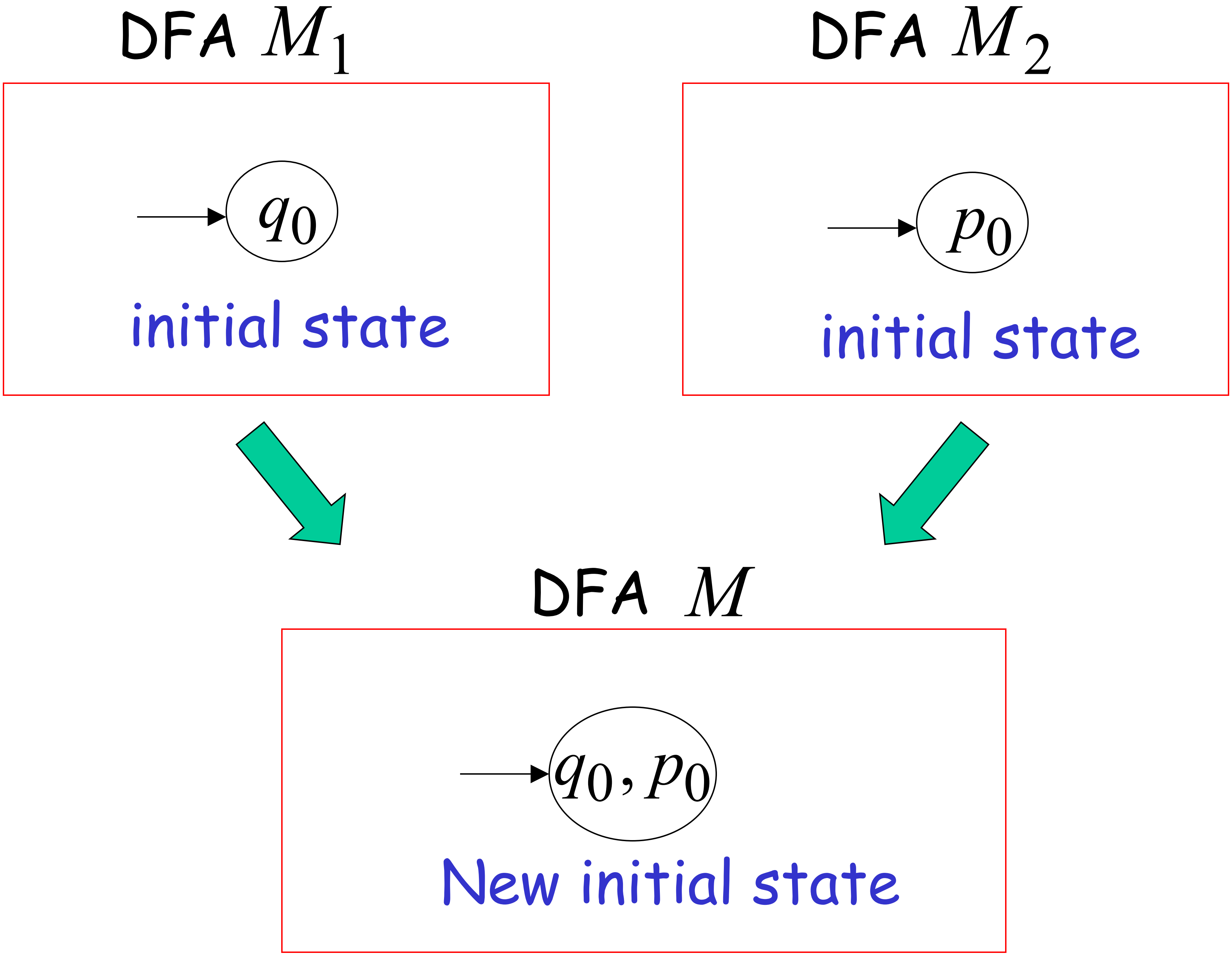
q_i, p_j

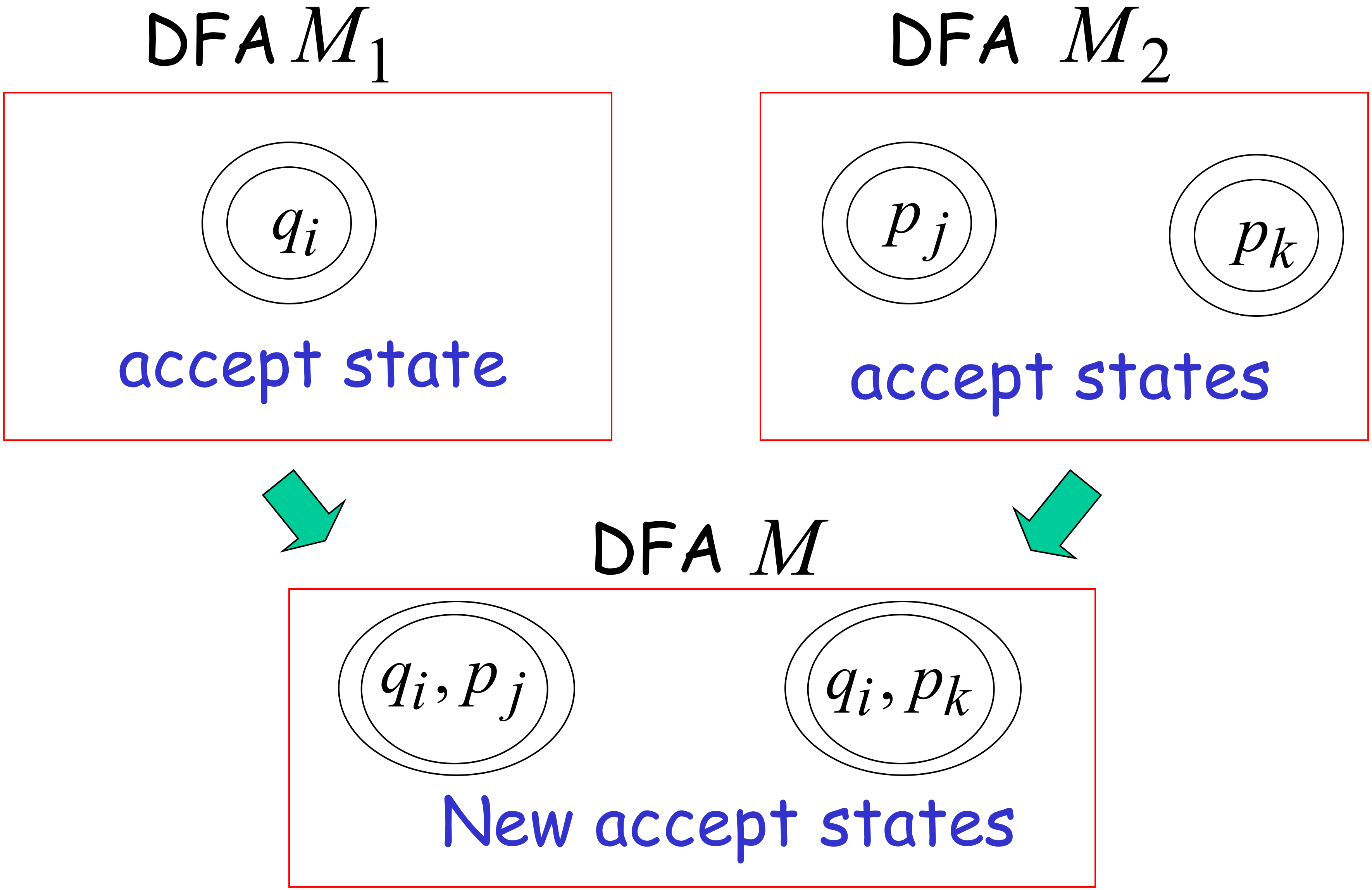
State in M_1

State in M_2



DFA M_1
q_1
a
q_2
transition
DFA M_2
p_1
a
p_2
transition
DFA M
q_1, p_1
a
q_2, p_2
New transition


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



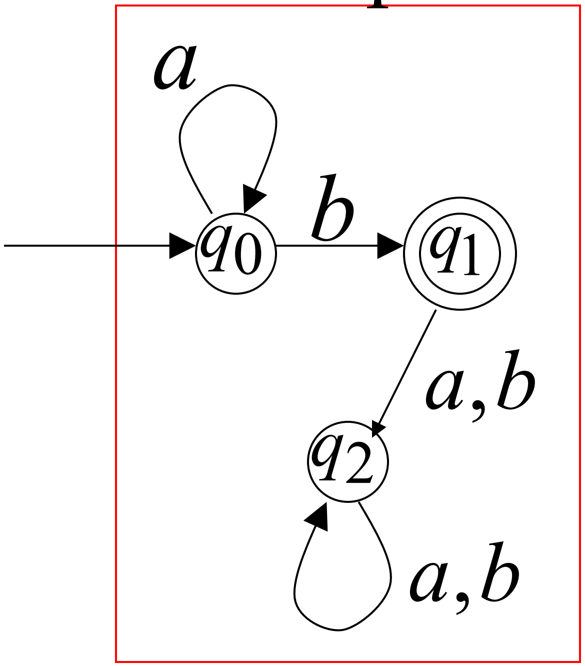
Both constituents must be accepting states

Example:

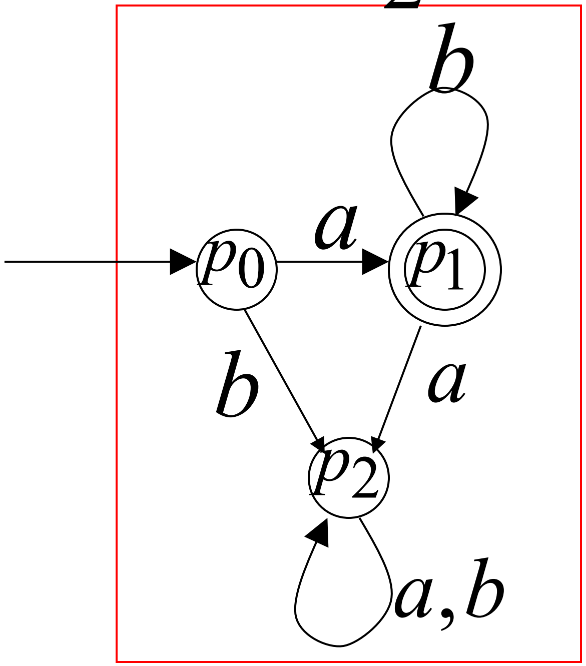
$$L_1 = \{a^n b\} \quad n \geq 0$$

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

M_1

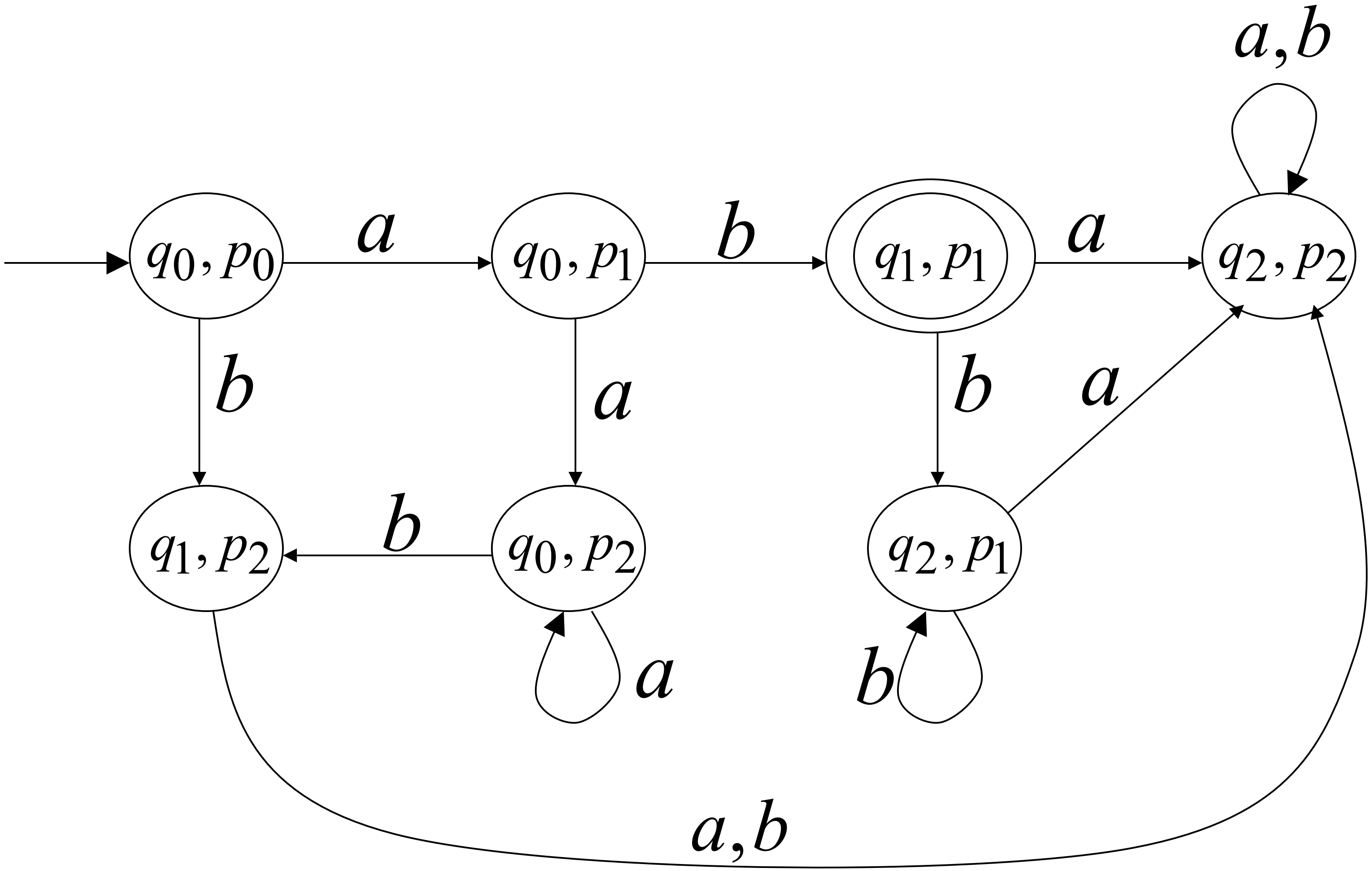


M_2



Automaton for intersection

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



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

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

Regular Expressions

Regular Expressions

Regular expressions
describe regular languages

Example: $(a+b\cdot c)^*$

describes the language

$$\{a, bc\}^* = \{\lambda, a, bc, aa, abc, bca, ...\}$$

Recursive Definition

Primitive regular expressions: $\emptyset, \quad \lambda, \quad \alpha$

Given regular expressions r_1 and r_2

$$\left.\begin{array}{l} r_1 + r_2 \\ r_1 \cdot r_2 \\ r_1{}^* \\ (r_1) \end{array}\right\} \text{Are regular expressions}$$

Examples

A regular expression: $(a+b\cdot c)^{*}\cdot(c+\emptyset)$

Not a regular expression: $(a+b+)$

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L((a+b\cdot c)^*)=\{\lambda,a,bc,aa,abc,bca,...\}$$

Definition

For primitive regular expressions:

$$L(\varnothing) = \varnothing$$

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

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

Definition (continued)

For regular expressions r_1 and r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Example

Regular expression: $(a+b)\cdot a^*$

$$
\begin{aligned}
L((a+b)\cdot a^*) &= L((a+b))\,L(a^*) \\
&= L(a+b)\,L(a^*) \\
&= (L(a)\cup L(b))\,(L(a))^* \\
&= (\{a\}\cup\{b\})\,(\{a\})^* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Example

Regular expression $r = (a+b)^*(a+bb)$

$$L(r) = \{a, bb, aa, abb, ba, bbb, ...\}$$

Example

Regular expression $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m}b: \quad n,m \geq 0\}$$

Example

Regular expression $r = (0+1)^*\,00\,(0+1)^*$

$L(r)$ = { all strings containing substring 00 }

Example

Regular expression $r = (1 + 01)^*(0 + \lambda)$

$L(r)$ = { all strings without substring 00 }

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2

are **equivalent** if $L(r_1) = L(r_2)$

Example

L = { all strings without substring 00 }

$$r_1 = (1 + 01)^*(0 + \lambda)$$

$$r_2 = (1^* 011^*)^*(0 + \lambda) + 1^*(0 + \lambda)$$

$$L(r_1) = L(r_2) = L \Longrightarrow$$

r_1 and r_2 are equivalent regular expressions

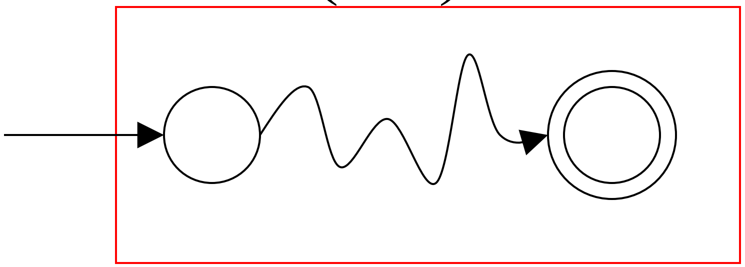
Regular Expressions and Regular Languages

Theorem

$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} = \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

Since L is regular take the NFA M that accepts it

$$L(M) = L$$



Single final state

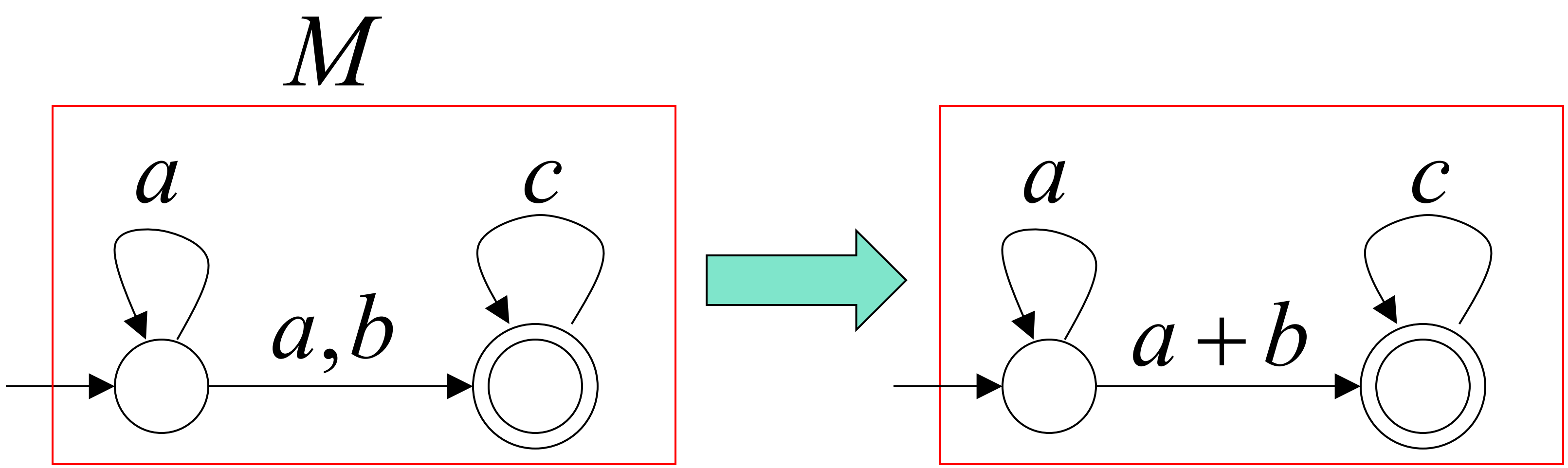
From M construct the equivalent

Generalized Transition Graph

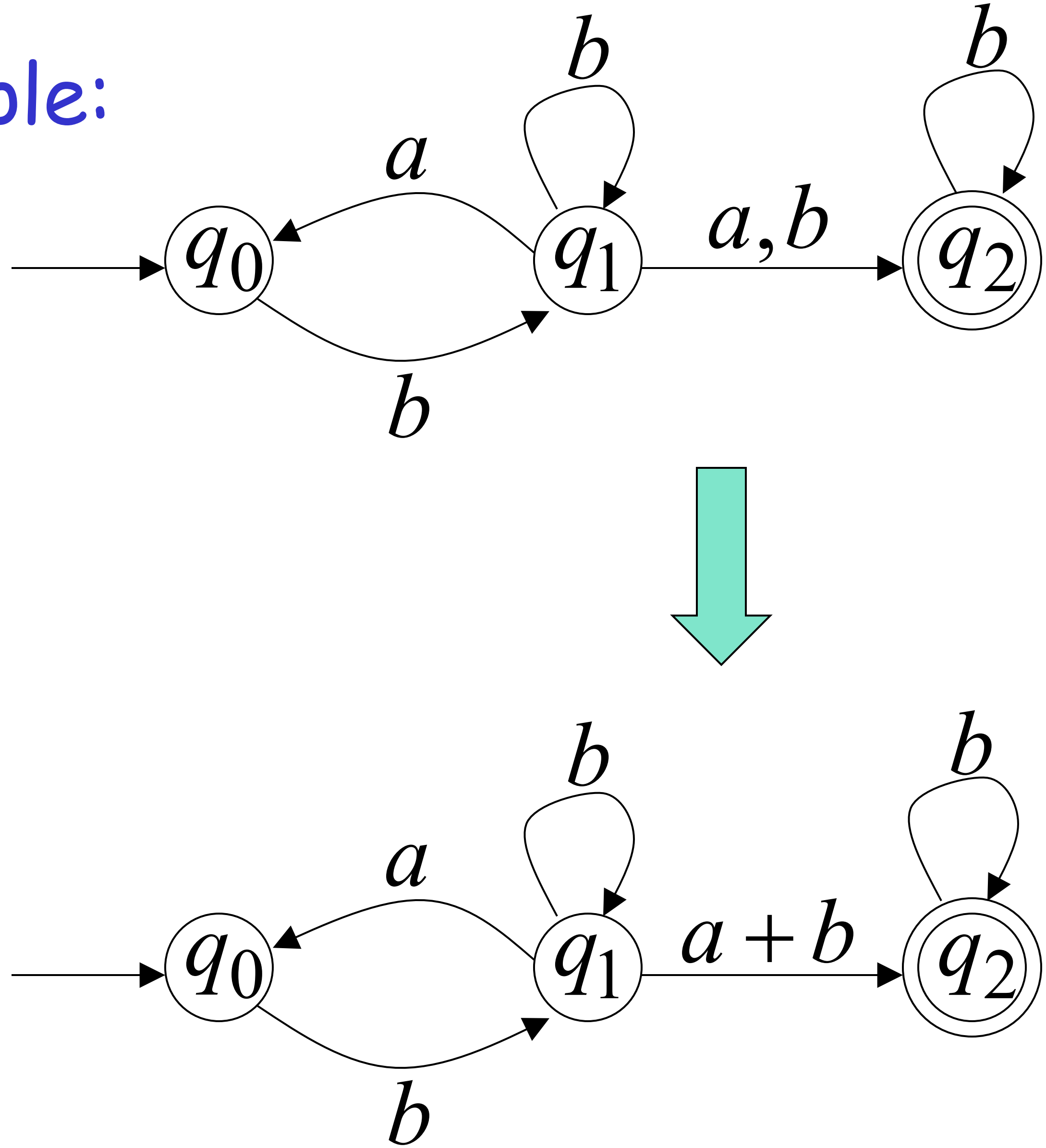
transition labels

are regular expressions

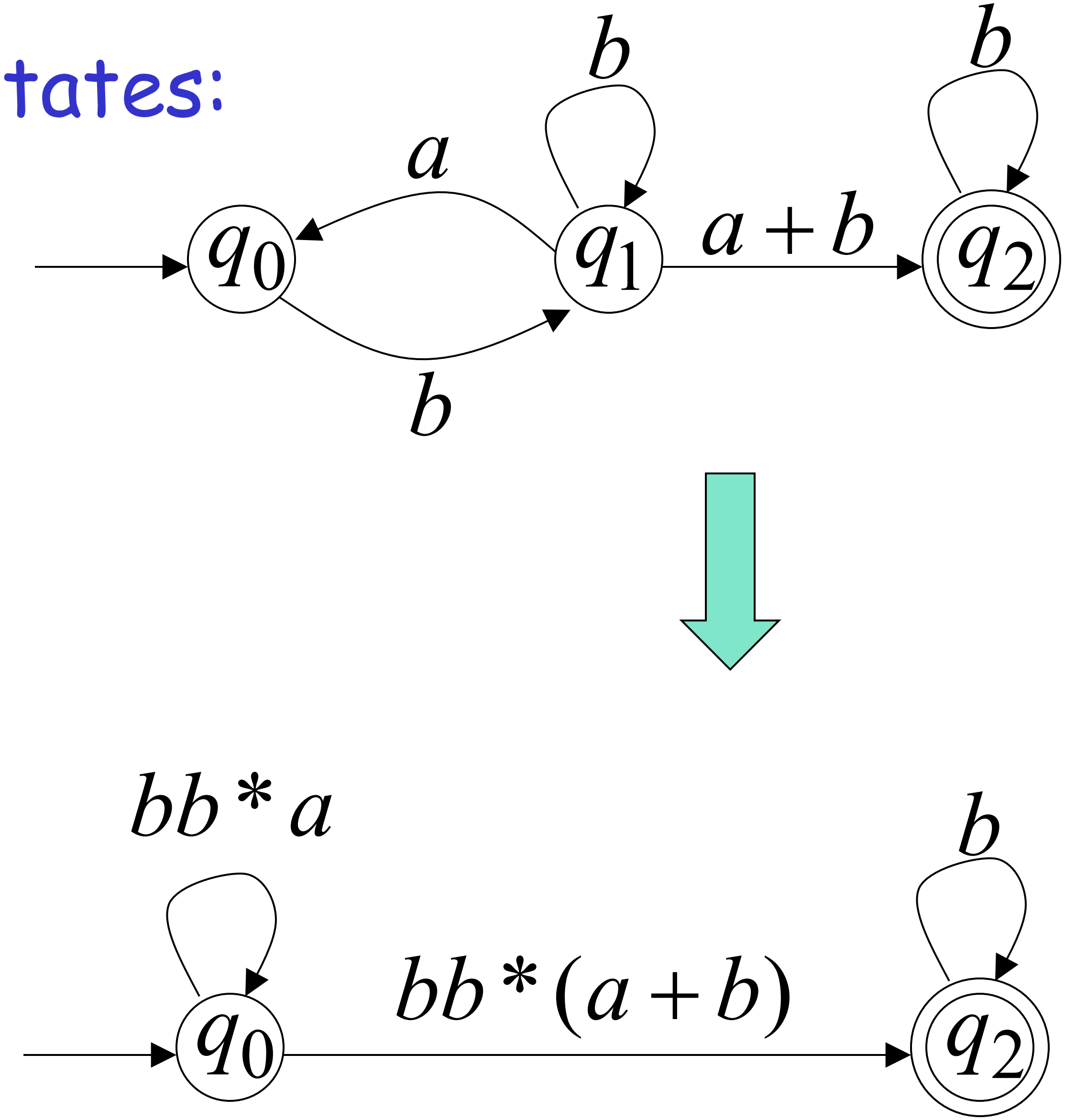
Example:



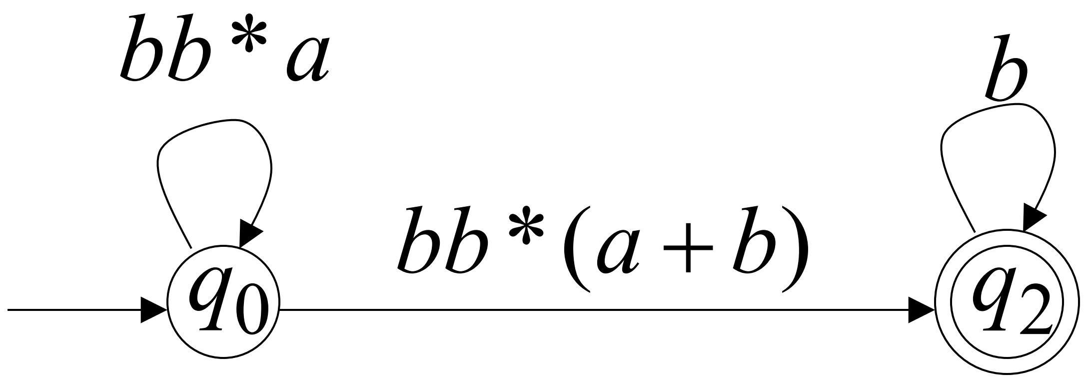
Another Example:



Reducing the states:



Resulting Regular Expression:

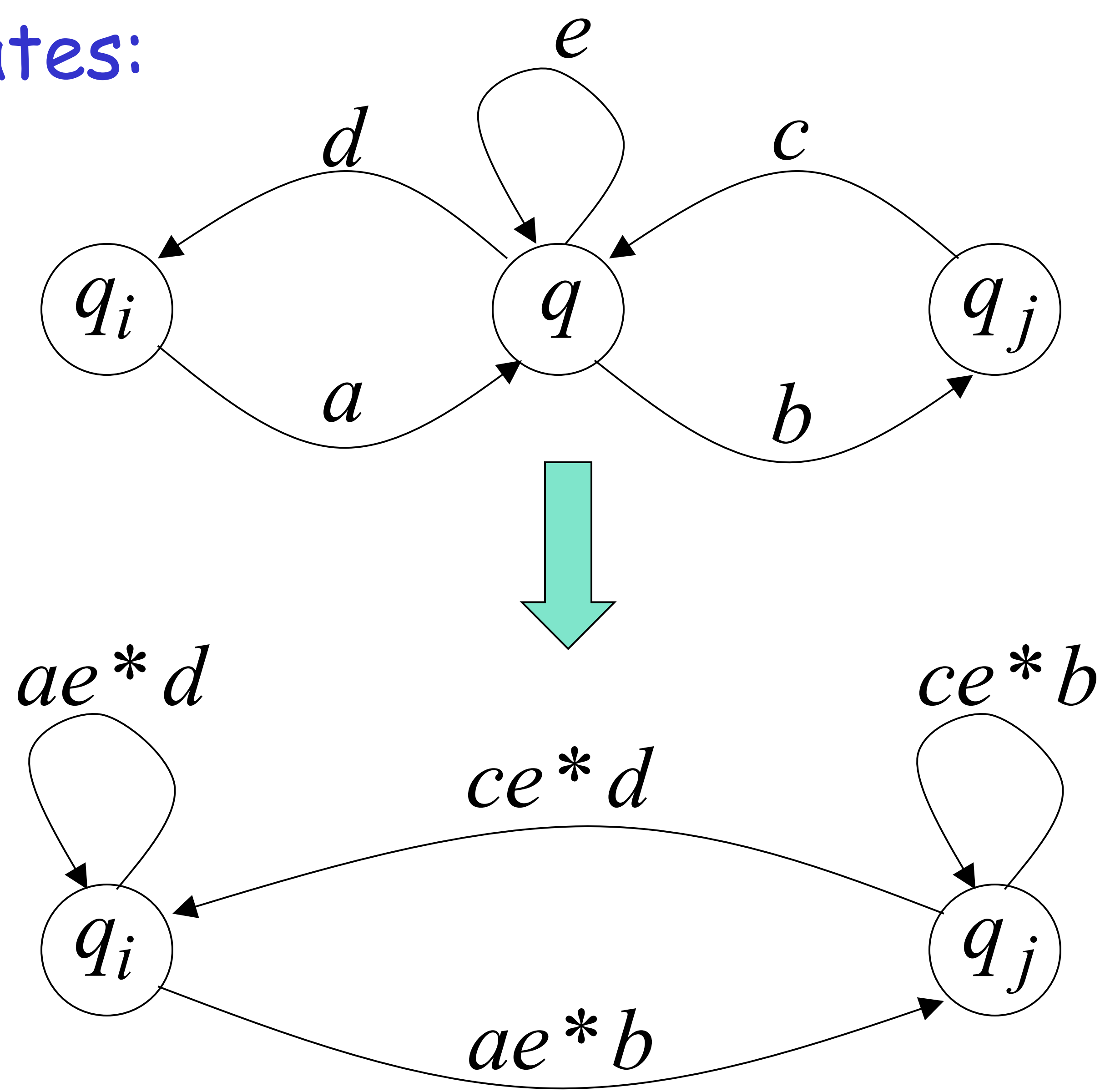


$$r = (bb^*a)^*bb^*(a+b)b^*$$

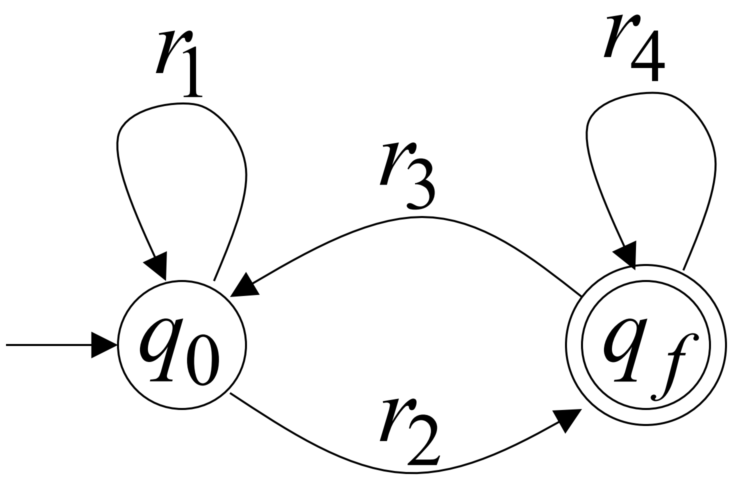
$$L(r) = L(M) = L$$

In General

Removing states:



Obtaining the final regular expression:



$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

$$L(r) = L(M) = L$$