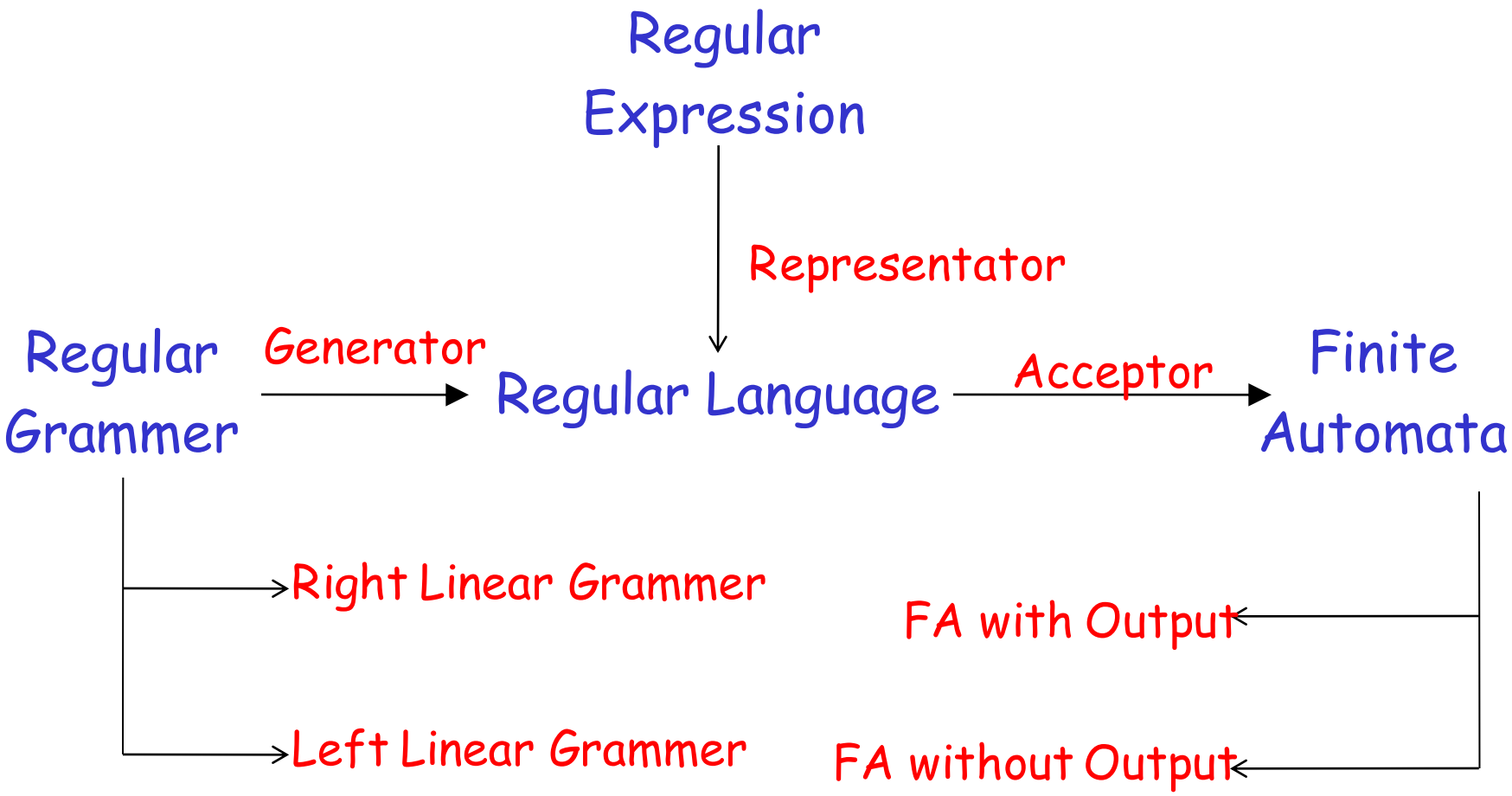
Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.


Regular Expression
Representator
Regular Grammer
Generator
Regular Language
Acceptor
Finite Automata
Right Linear Grammer
Left Linear Grammer
FA with Output
FA without Output

Regular Expressions

Regular expressions

describe regular languages

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

describes the language

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

Recursive Definition

1) + : Union 2) . : Concatenation 3) * : Kleene Closure

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

$\emptyset \longrightarrow \{\ \}$

$\lambda \longrightarrow \{\lambda\ \}$

$\alpha \longrightarrow \{\ a\ \}$

Given regular expressions r_1 and r_2

$r_1 + r_2$

$r_1 \cdot r_2$

r_1^*

(r_1)

Are regular expressions

Examples

A regular expression: $(a+b\cdot c)^*\cdot(c+\varnothing)$

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

Languages of Regular Expressions

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

Example

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

Definition

For primitive regular expressions:

$$L(\emptyset) = \emptyset$$

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

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

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

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

Example

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

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

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2

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

Regular Expressions and Regular Languages

- ***Regular Language***
- ∅
- {ε}
- {a,b}*
- {aab}*{a,ab}
- {aa,bb}U {ab,ba}{aa,bb}*

- ***Regular Expression***
- ∅
- ε
- (a+b)*
- (aab)*(a+ab)
- (aa+bb+(ab+ba)(aa+bb)*)

Theorem

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

Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.

Recall

- Regular Expression

Find RE for the Language of strings of Length 2 over the alphabet {a,b}

$=(aa+ab+ba+bb)$ or $(a+b)(a+b)$

Find RE for the Language of strings of Length at least 2 over the alphabet {a,b}

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

Find RE for the Language of strings of Length at most 2 over the alphabet {a,b}

$=(a+b+\varepsilon)(a+b+\varepsilon)$

Find RE for the Language of strings of even Length over the alphabet {a,b}

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

Find RE for the Language of strings of odd Length over the alphabet {a,b}

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

Find RE for the Language of strings which is divisible by 3 over the alphabet {a,b}

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

Find RE for the Language of strings = 2 mod 3 over the alphabet {a,b}

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

Find RE for the Language of strings of a's exactly 2 over the alphabet {a,b}

$=b^*ab^*ab^*$

- Find RE for the Language of strings of a's atleast 2 over the alphabet {a,b}
- (a+b)*a(a+b)*a(a+b)*
- Find RE for the Language of strings of a's atmost 2 over the alphabet {a,b}
- b*(a+ ε)b*(a+ ε)b*
- Find RE for the Language of strings of even length a's over the alphabet {a,b}
- (b*ab*ab*)*
- Find RE for the Language of strings which starts with a over the alphabet {a,b}
- a(a+b)*
- Find RE for the Language of strings which ends with a over the alphabet {a,b}
- (a+b)*a

- Find RE for the Language of strings which starts and ends with different symbol over the alphabet {a,b}

- (a(a+b)*b)+(b(a+b)*a)

- Find RE for the Language of strings which starts and ends with Same symbol over the alphabet {a,b}

- (a(a+b)*a)+(b(a+b)*b)

Regular Expression

Write RE for the following languages for $\Sigma = \{a,b\}$

- The language of all words
  $(a+b)^*$
- All words ending with **b**
  $(a+b)^*b$
- All words that start with **a**
  $a(a+b)^*$
- The language of all strings, not beginning with b
  $(a(a+b)^*)+ \Lambda$
- All words that start with a double letter
  $(aa+bb)(a+b)^*$
- All words that contain **at least one double letter**
  $(a+b)^*(aa+bb)(a+b)^*$

Continued..

- All words that contain **at least two a's or two b's**
  b*ab*a(a+b)* + a*ba*b(a+b)*
- All words that start and end with a double letter
  (aa+bb)(a+b)*(aa+bb)
- All words of length >=3
  (a+b)(a+b)(a+b)(a+b)* or (a+b)(a+b)(a+b)$^+$
- All words that contain exactly one **a or exactly one b**
  b*ab* + a*ba*
- All words that don't end at **ba**
  (a+b)*(aa+ab+bb)
- All strings of a's and b's in which either the strings are all b's or else there is an a followed by some b's
  b*+ab*

Continued..

- Language of all words that have at least two **a's**

  (a+b)* a (a+b)* a (a+b)*
- Language of all words that have **at least one a and at least one b**

  (a+b)* a (a+b)* b (a+b)* + (a+b)* b (a+b)* a (a+b)*
- Language of all words that have **at least one a or at least one b**

  (a+b)*a(a+b)* + (a+b)*b(a+b)*
- The languages L, of even length, defined over Σ = {a, b}

  ((a+b)(a+b))*
- The languages L, of odd length, defined over Σ = {a, b}

  ((a+b)(a+b))*(a+b)
- The strings of length 2, starting with **a**,

  aa+ab

- Give RE for Following over the alphabet {0,1} (Aug-2015 6 Marks)
  - All binary strings with at least one 0
  - All binary strings with at Most one 0
- (0+1)*0(0+1)*
- 1*(0+ε) 1*
- Language of all strings containing substring 00
- (0+1)*00(0+1)*
- The Language of Strings in {a, b}* Ending with b and Not Containing aa
- (ab+b)+(b+ ε)
- Find RE for the Language of strings which does not contains two a's together over the alphabet {a,b}
- (a+ε)(ba+b)*

Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.

Recall

- Examples on Regular Expression

- Construct regular expressions defined over the alphabet {a, b}, which denote the following languages:
- i) All strings without a double a.

- ii) All strings in which any occurrence of the symbol b, is in groups of odd numbers.

- iii) All strings in which the total number of a's is divisible by 2.

- Construct regular expressions defined over the alphabet {a, b}, which denote the following languages:
- i) All strings without a double a.
- (a+ε)(ba+b)*
- ii) All strings in which any occurrence of the symbol b, is in groups of odd numbers.
- a*b(bb)*a*
- iii) All strings in which the total number of a's is divisible by 2.
- (b* a b* a b*)* + b*

- Check the following regular expressions for equivalence and justify:
- (i) R1 = (a + bb)* (b + aa)* R2 = (a + b)*
- (ii) R1 = (a + b)* abab* R2 = b* a (a + b)* ab*

i) Let us write languages denoted by R1 and R2 as below.

- L(R1) = {ε, a, b, aa, ab, bb, abb, baa, bba, ... }
- L(R2) = {ε, a, b, aa, ab, ba, bb, ... }
- Given regular expressions R1 and R2 are not equal as the strings produced by them are not same.
- For example, string 'ba' cannot be generated using regular expression R1 which can be produced by R2 .

ii) Let us write languages denoted by R1 and R2 as below.

- L(R1) = {aba, aaba, baba, abab, ababb, ababa, ... }
- L(R2) = { aa, baa, baaa, baba, baab, ... }
- Given regular expressions R1 and R2 are not equal as the strings produced by them are not same.
- For example, string 'aa' cannot be produced by Regular Expression R1 which can be produced by R2 .

Identities Related to Regular Expressions

- Given R, P, L, Q as regular expressions, the following identities hold −
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $RR^* = R^*R$
- $R^*R^* = R^*$
- $(R^*)^* = R^*$
- $RR^* = R^*R$
- $(PQ)^*P = P(QP)^*$
- $(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$

- $R + \emptyset = \emptyset + R = R$ (The identity for union)
- $R\,\varepsilon = \varepsilon\,R = R$ (The identity for concatenation)
- $R + R = R$ (Idempotent law)
- $L\,(M + N) = LM + LN$ (Left distributive law)
- $(M + N)\,L = ML + NL$ (Right distributive law)
- $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

- Let $\Sigma^* = \{0, 1\}$. Construct regular expressions for each of the following:
- (a) L1 = $\{W \in \Sigma^* \mid W$ has at least one pair of consecutive zeros$\}$
- (b) L2 = $\{W \in \Sigma^* \mid W$ has no pair of consecutive zeros$\}$
- (c) L3 = $\{W \in \Sigma^* \mid W$ starts with either '01' or '10'$\}$
- (d) L4 = $\{W \in \Sigma^* \mid W$ consists of even number of 0's followed by odd number of 1's$\}$

- Solution
- (a) $r = [\ (1 + 0\ )^*\ (00)\ (1 + 0\ )^*\ ]^+$
- (b) $r = (0 + \varepsilon)\ (1+10)^*$
- (c) $r = (01 + 10)\ (1+0)^*$
- (d) $r = (00)^*\ 1\ (11)^*)$

Gate Question

- Which one of the following languages over the alphabet {0,1} is described by the regular expression:

  (0+1)*0(0+1)*0(0+1)*?

1. The set of all strings containing the substring 00.
2. The set of all strings containing at most two 0's.
3. The set of all strings containing at least two 0's.
4. The set of all strings that begin and end with either 0 or 1.

- GATE-CS-2009

Theorem

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

Proof:

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

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

Proof - Part 1

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

For any regular expression r

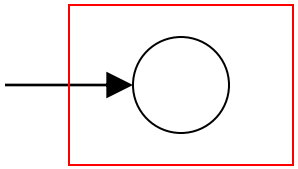
the language $L(r)$ is regular

Proof by induction on the size of r

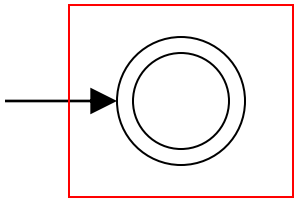
Induction Basis

Primitive Regular Expressions: $\emptyset$, λ, α

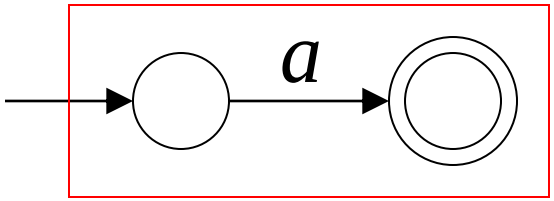
Corresponding NFAs



$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{\lambda\} = L(\lambda)$$



$$L(M_3) = \{a\} = L(a)$$

regular languages

Inductive Hypothesis

Suppose

that for regular expressions r_1 and r_2,

$L(r_1)$ and $L(r_2)$ are regular languages

Inductive Step

We will prove:

$L(r_1 + r_2)$

$L(r_1 \cdot r_2)$

$L(r_1 *)$

$L((r_1))$

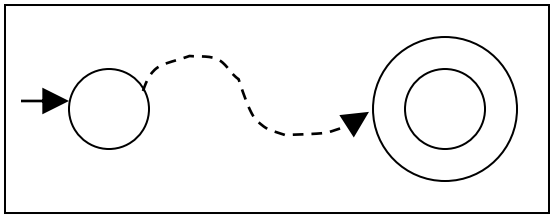
Are regular Languages

Using the regular closure of these operations, we can construct recursively the NFA M that accepts $L(M) = L(r)$

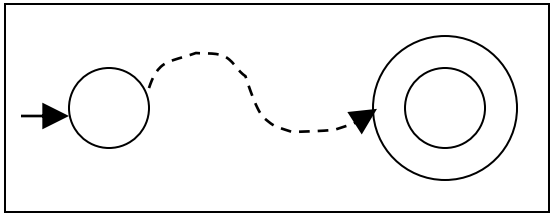
Example: $r = r_1 + r_2$

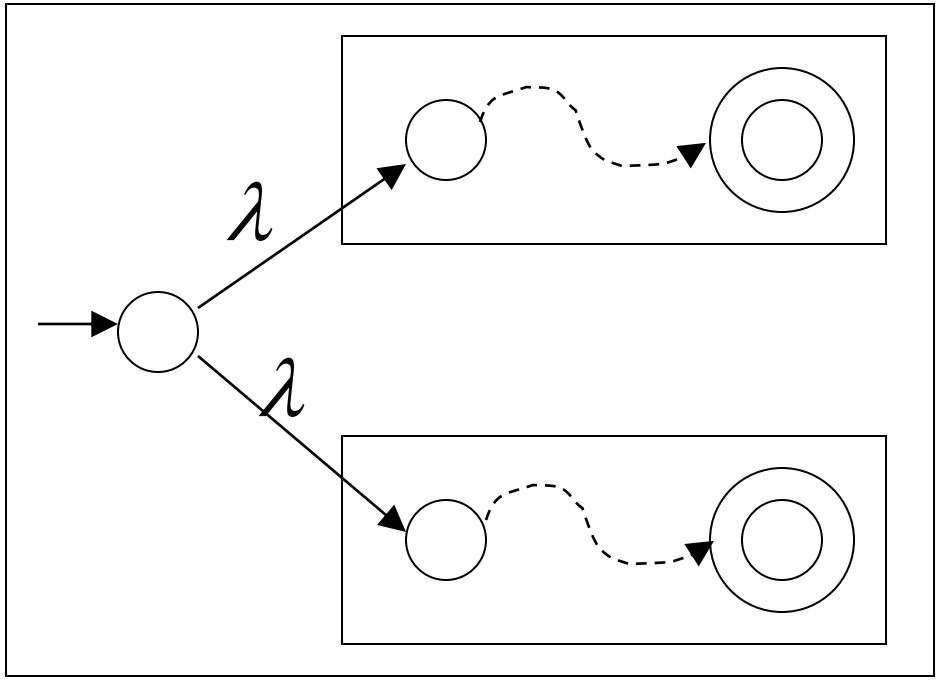
$L(M) = L(r)$

$L(M_1) = L(r_1)$



$L(M_2) = L(r_2)$

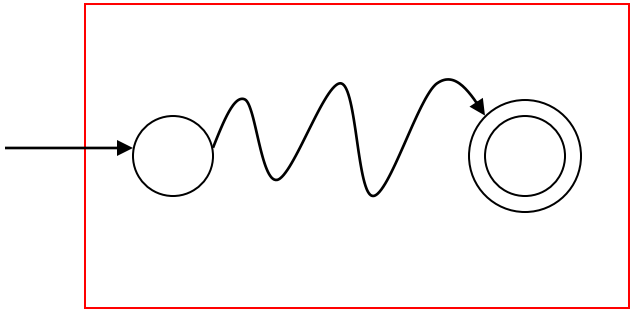




Regular Expression r1

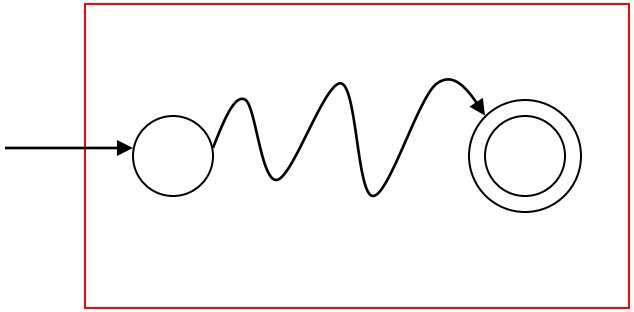
Regular Expression r2

NFA M_1



Single accepting state

NFA M_2

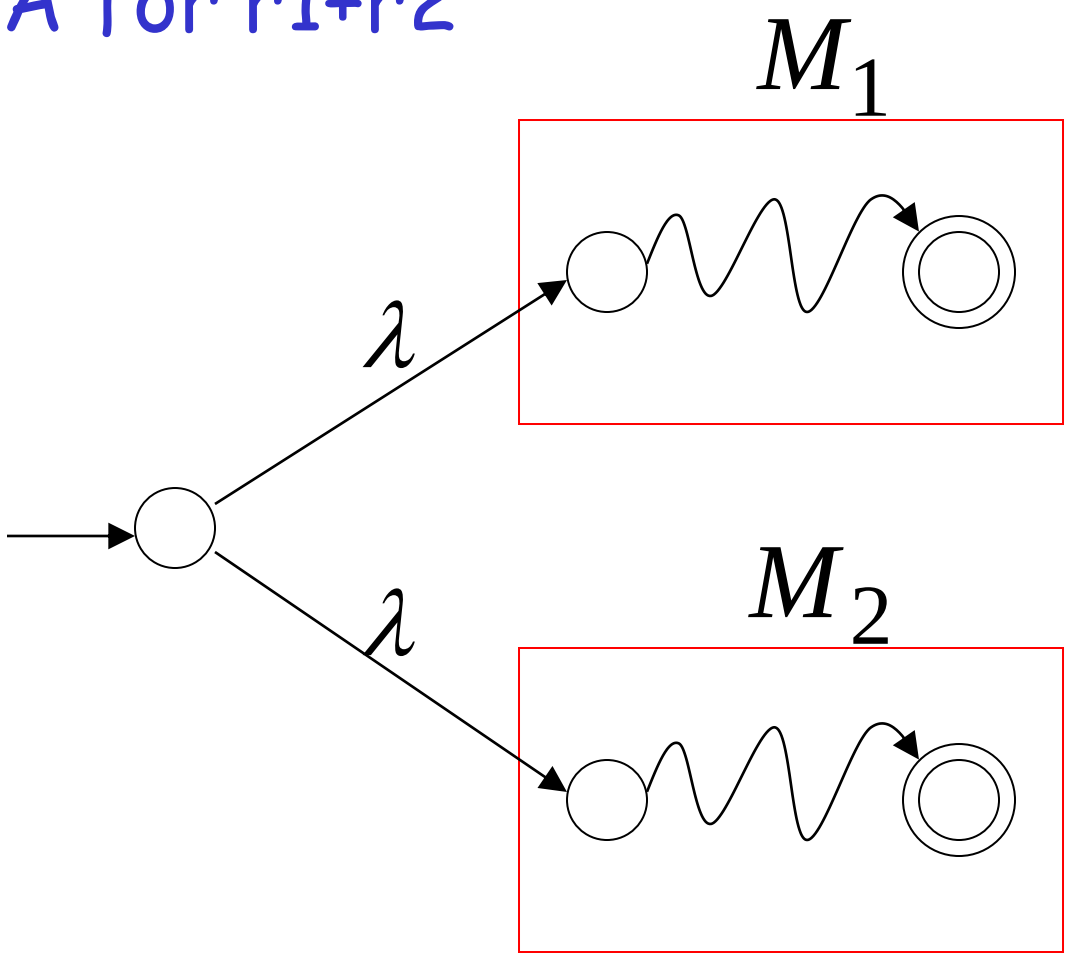


Single accepting state

Union

r1 and r2

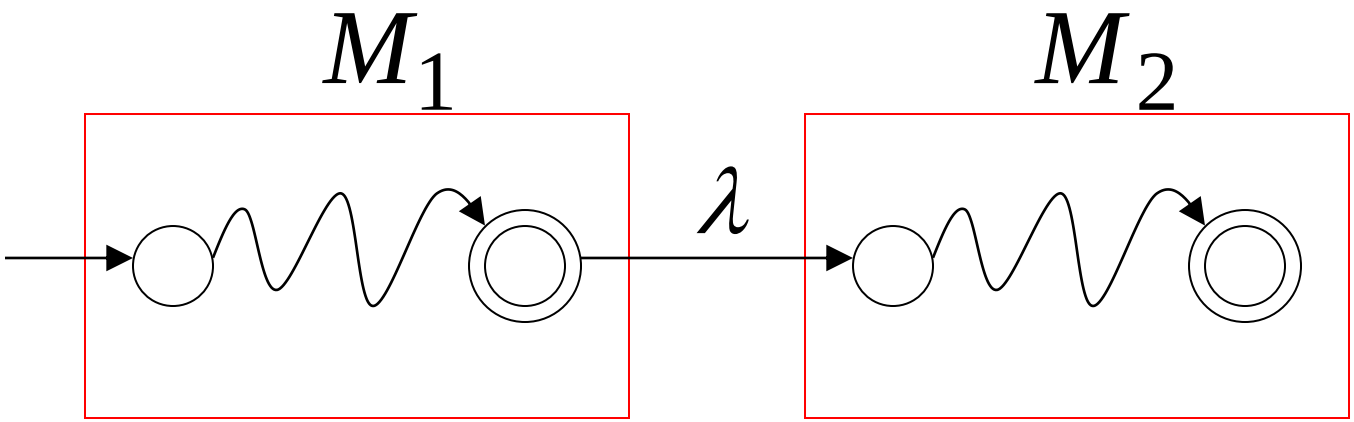
NFA for r1+r2



Concatenation

r1 and r2

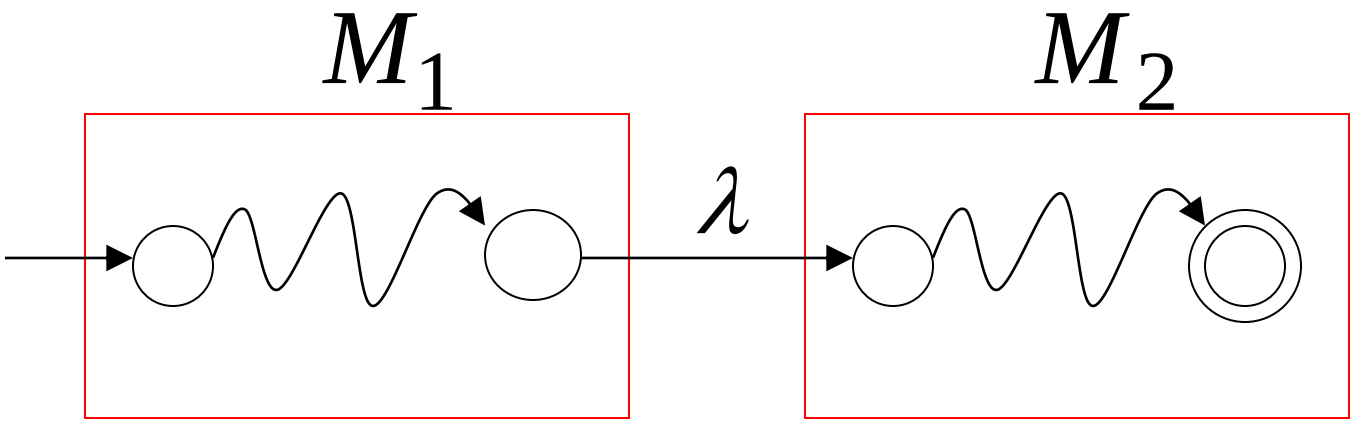
NFA for r1tr2



Concatenation

r1 and r2

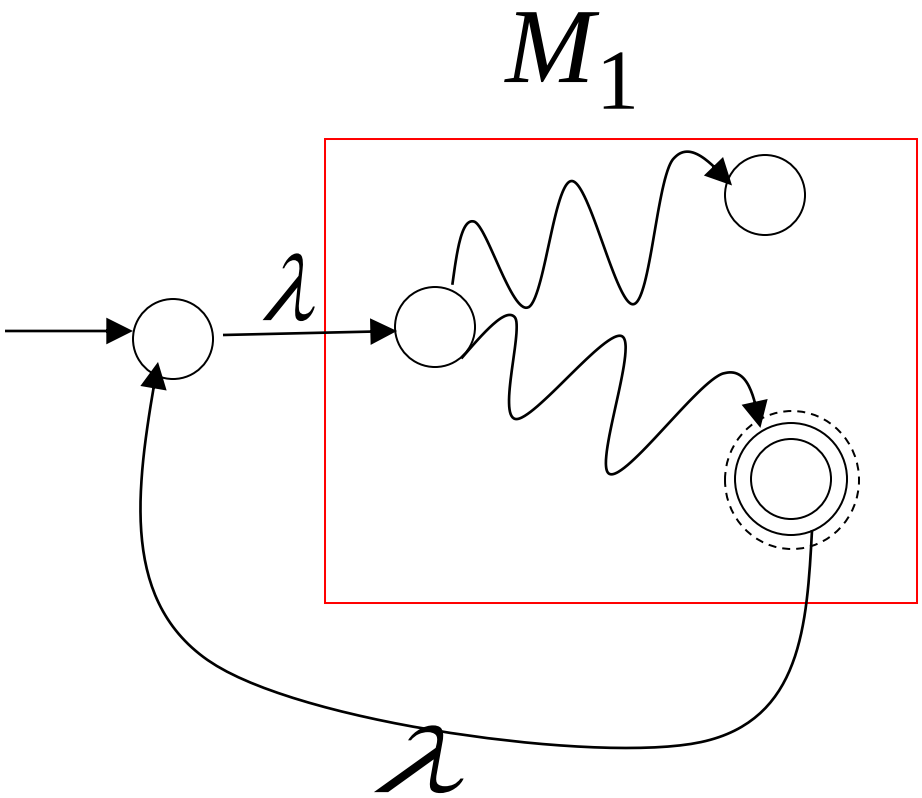
NFA for r1tr2



Star Operation

r1

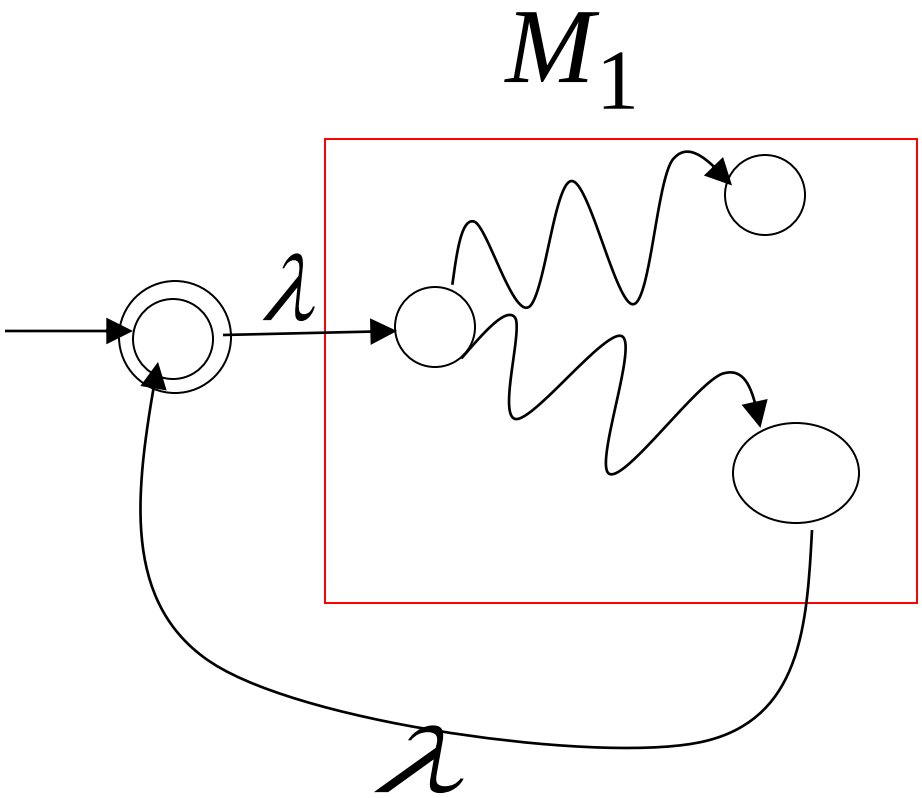
NFA for r1*



Star Operation

r1

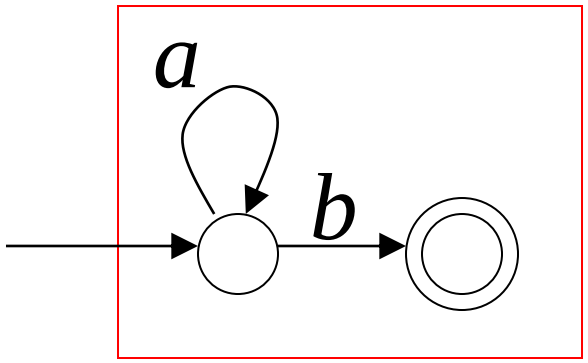
NFA for r1*



Example

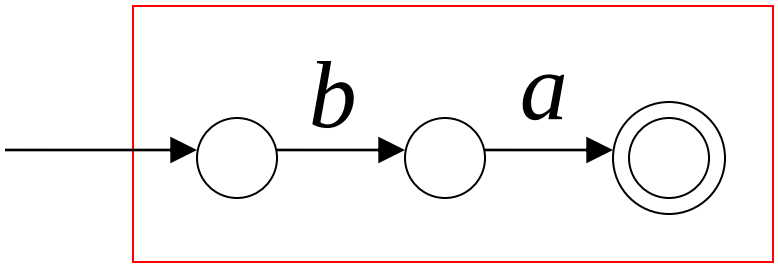
$r_1 = (a^* b)$

M_1



$r_2 = (ba)$

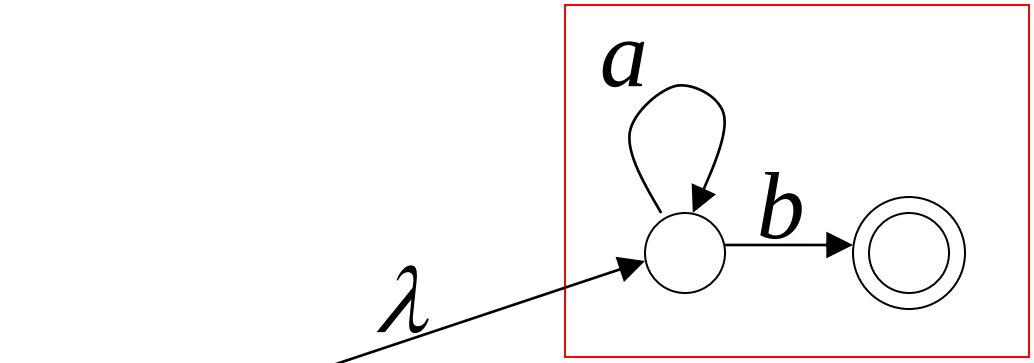
M_2



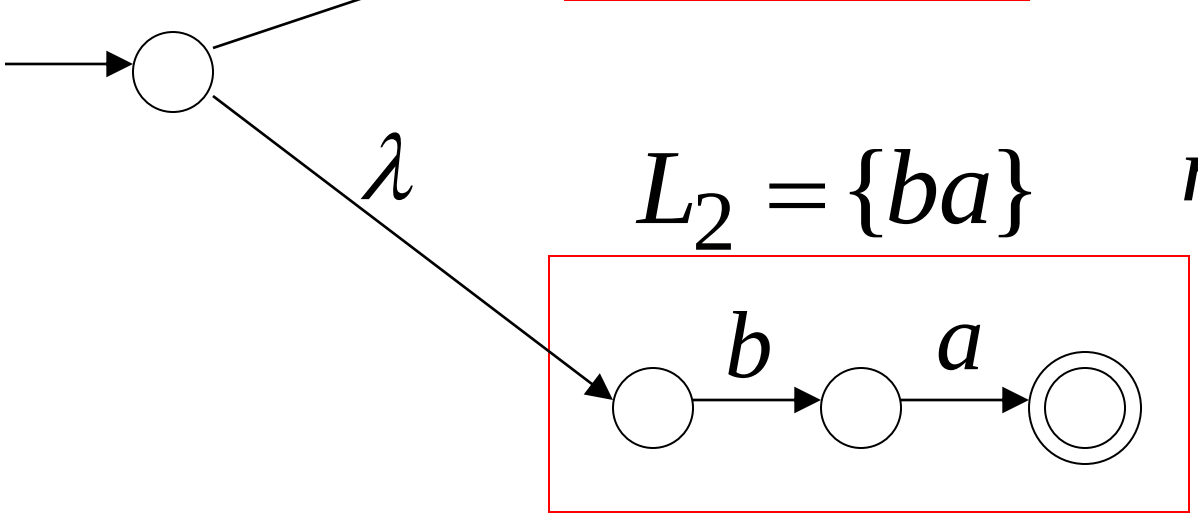
Example

NFA for $r_1 + r_2 = (a^*b) + (ba)$

$L_1 = \{a^n b\}$ $r_1 = (a^*b)$



$L_2 = \{ba\}$ $r_2 = (ba)$



Example

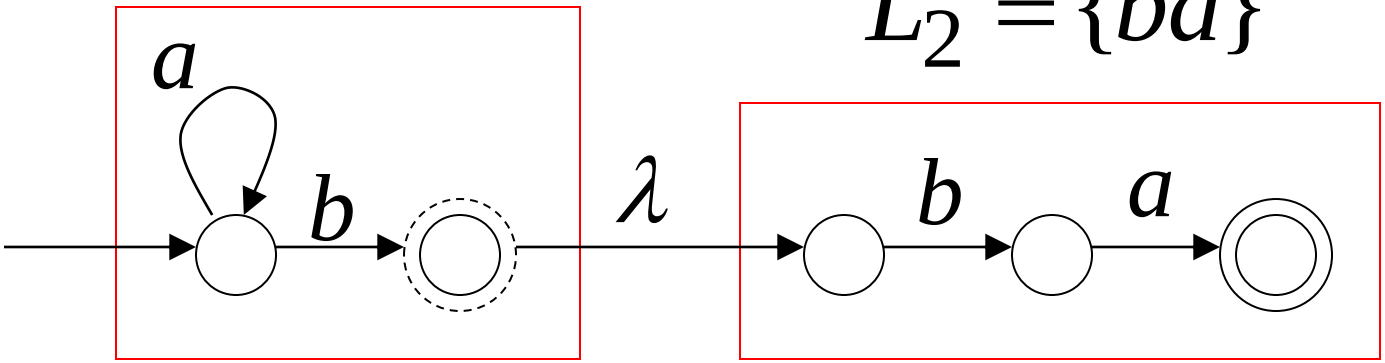
NFA for $r_1 r_2 = (a^* b)(ba) = (a^* bba)$

$r_1 = (a^* b)$

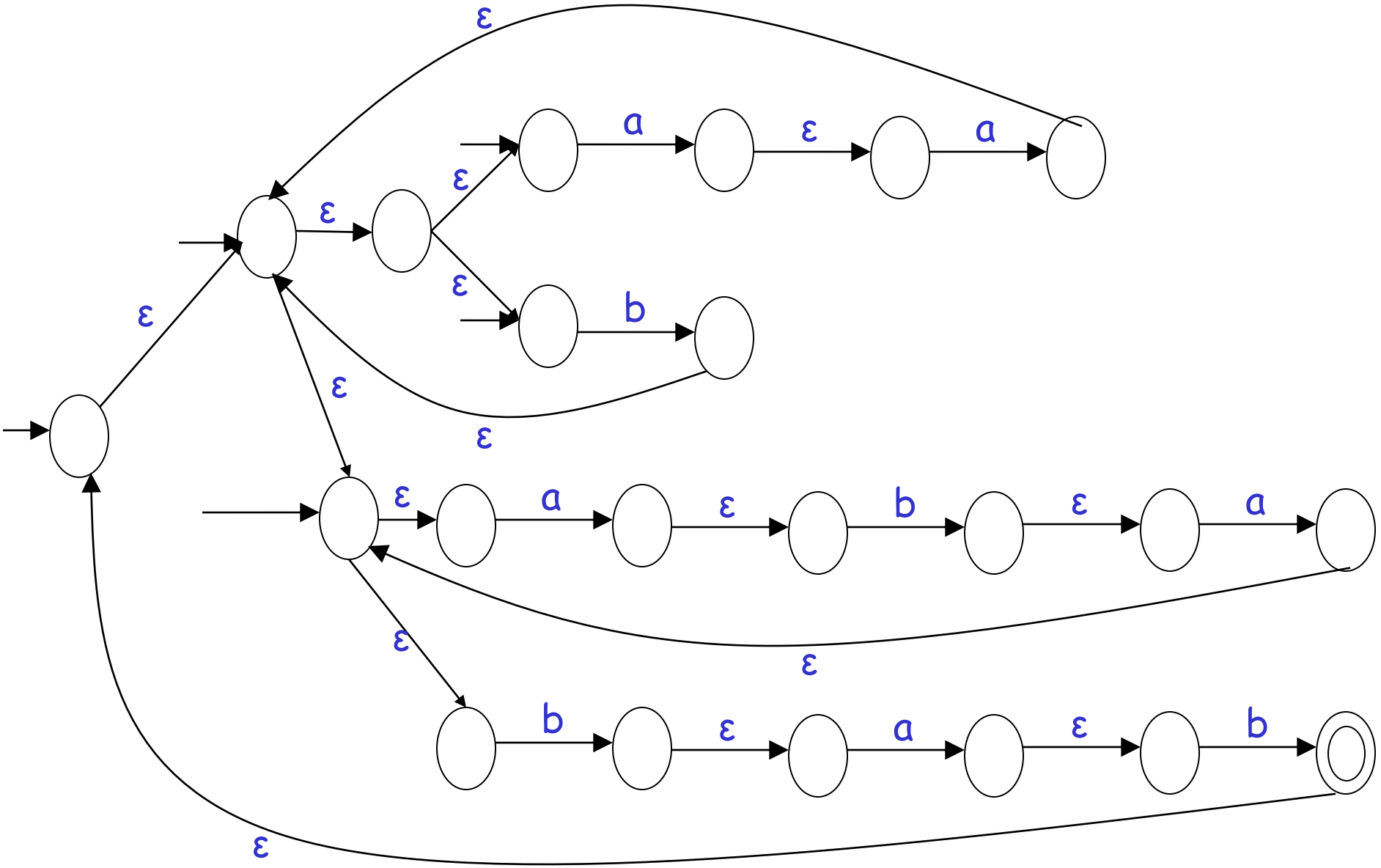
$L_1 = \{a^n b\}$

$r_2 = (ba)$

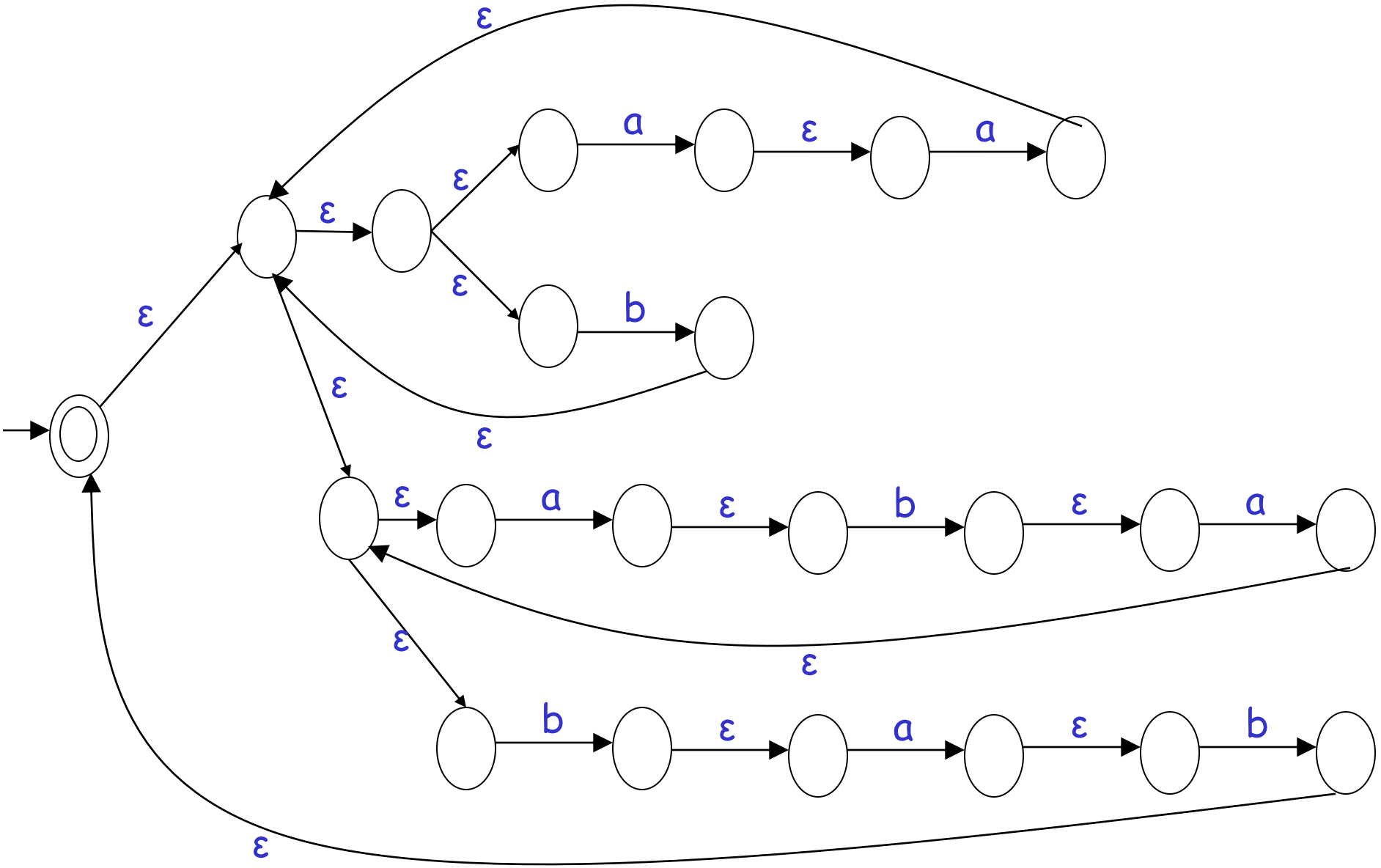
$L_2 = \{ba\}$



- An NFA Corresponding to ((aa +b)*(aba)*bab)*



- An NFA Corresponding to ((aa +b)*(aba)*bab)*



Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.

Review

- Regular Expression
- Construction of RE from Language
- Construction of Language from RE
- Equivalence in RE and RL
- Conversion of RE to RL (FSM)

Problems

- Construct an NFA with null-moves, which accepts the language defined by:
- ((0 + 1)* 10 + (00)* (11)*)*
- 01[((10)+ + 111)* + 0]* 1
- (*a* / *b*)* *ab*.

Problems

- Construct DFA for the R.E l0 + (0 + ll)
- Construct DFA for regular expression (0 + l)* (00 + ll)

Proof - Part 2

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

For any regular language L there is a regular expression r with $L(r) = L$

We will convert an NFA that accepts L to a regular expression

Arden's Theorem

In order to find out a regular expression of a Finite Automaton, we use Arden's Theorem along with the properties of regular expressions.

Statement –

Let **P**, **Q** and R be the regular expressions.

If **P** does not contain null string, then

R = Q + RP has a unique solution that is

R = QP*

Proof –

$R = Q + (Q + RP)P$ [After putting the value $R = Q + RP$]

$= Q + QP + RPP$

When we put the value of **R** recursively again and again, we get the following equation –

$R = Q + QP + QP^2 + QP^3 .....$

$R = Q (\varepsilon + P + P^2 + P^3 + ...)$

$R = QP^*$ [As P^* represents $(\varepsilon + P + P2 + P3 + ...)$]

Hence, proved.

Assumptions for Applying Arden's Theorem

- The transition diagram must not have NULL transitions

Method

Step 1 – Create equations as the following form for all the states of the DFA having n states with initial state q1 (on incoming edges only and add ε to the initial state).

$$q_1 = q_1R_{11} + q_2R_{21} + \ldots + q_nR_{n1} + \varepsilon$$

$$q_2 = q_1R_{12} + q_2R_{22} + \ldots + q_nR_{n2}$$

.....

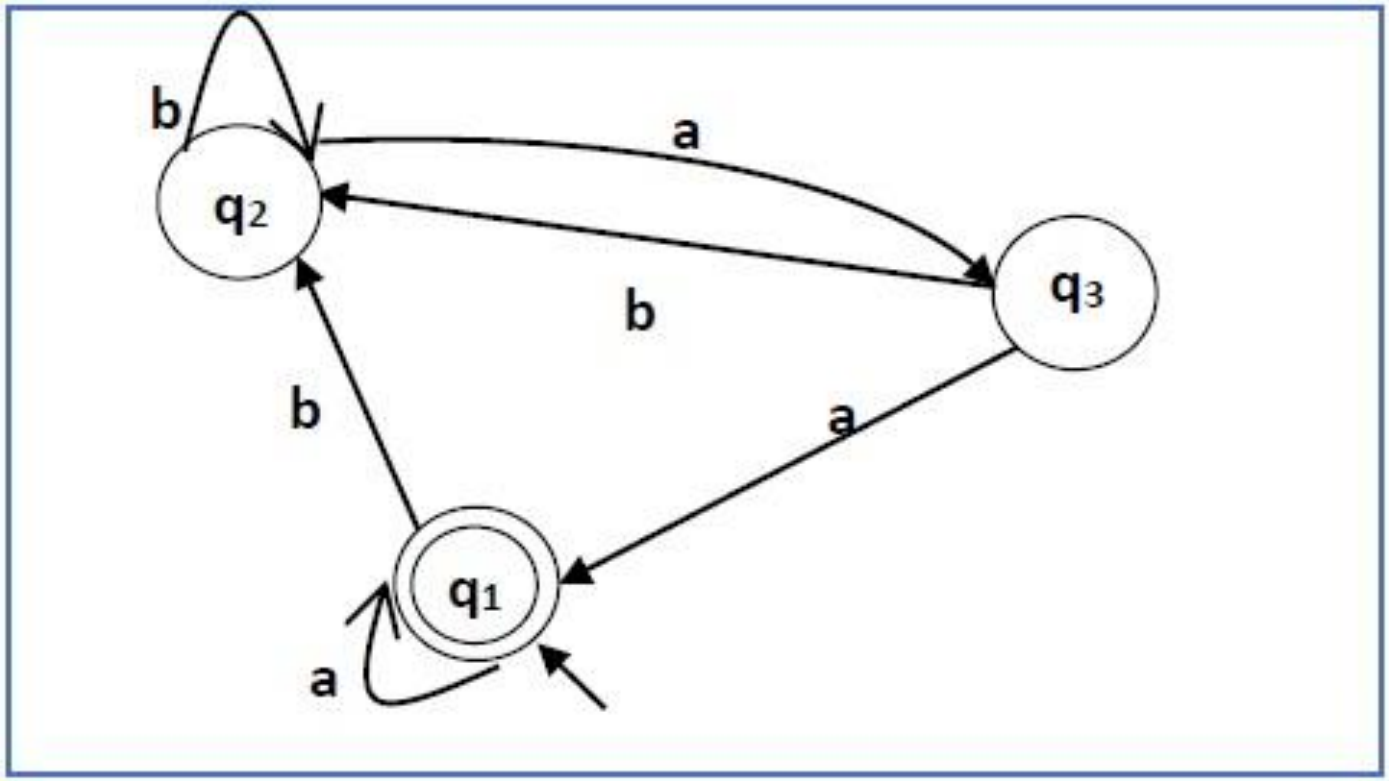
$$q_n = q_1R_{1n} + q_2R_{2n} + \ldots + q_nR_{nn}$$

R_{ij} represents the set of labels of edges from q_i to q_j, if no such edge exists, then $R_{ij} = \emptyset$

Step 2 – Solve these equations to get the equation for the final state in terms of R_{ij}

Problem

Construct a regular expression corresponding to the automata given below:



Solution –

Here the initial state is q_2 and the final state is q_1.

The equations for the three states q1, q2, and q3 are as follows –

$$q_1 = q_1a + q_3a + \varepsilon$$ (ε move is because q1 is the initial state0

$$q_2 = q_1b + q_2b + q_3b$$

$$q_3 = q_2a$$

Now, we will solve these three equations –

$$q_2 = q_1b + q_2b + q_3b$$

$$= q_1b + q_2b + (q_2a)b$$ (Substituting value of q_3)

$$= q_1b + q_2(b + ab)$$

$$= q_1b\ (b + ab)^*$$ (Applying Arden's Theorem)

$q_1 = q_1a + q_3a + \varepsilon$

$= q_1a + q_2aa + \varepsilon$ (Substituting value of q_3)

$= q_1a + q_1b(b + ab^*)aa + \varepsilon$ (Substituting value of q_2)

$= q_1(a + b(b + ab)^*aa) + \varepsilon$

$= \varepsilon\ (a + b(b + ab)^*aa)^*$

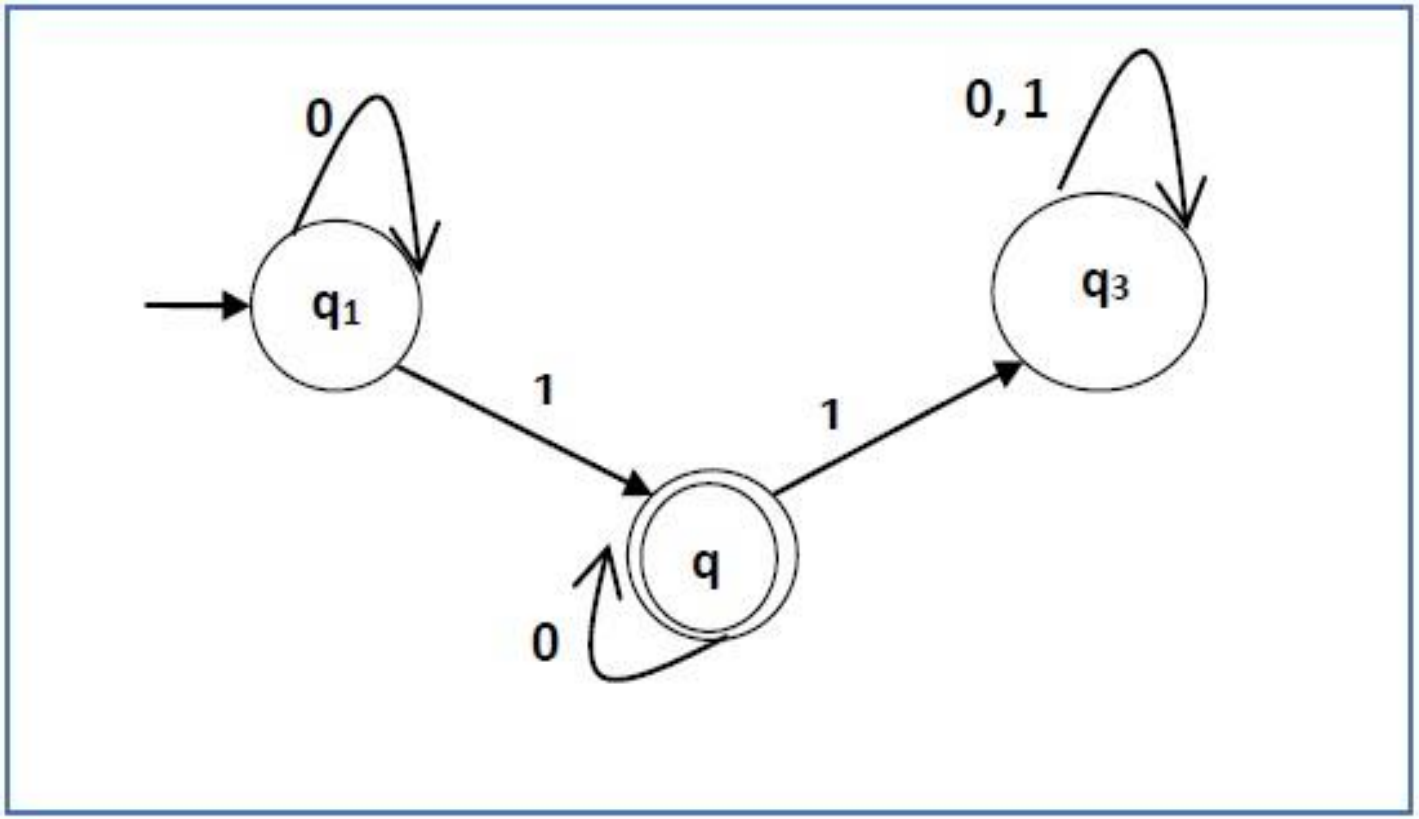
$= (a + b(b + ab)^*aa)^*$

Hence, the regular expression is

$(a + b(b + ab)^*aa)^*$.

Problem

Construct a regular expression corresponding to the automata given below:



Solution –

Here the initial state is q_1 and the final state is q_2

Now we write down the equations –

$$q_1 = q_1 0 + \varepsilon$$

$$q_2 = q_1 1 + q_2 0$$

$$q_3 = q_2 1 + q_3 0 + q_3 1$$

Now, we will solve these three equations –

$$q_1 = \varepsilon 0^* \qquad [As, \varepsilon R = R]$$

So, $q_1 = 0^*$

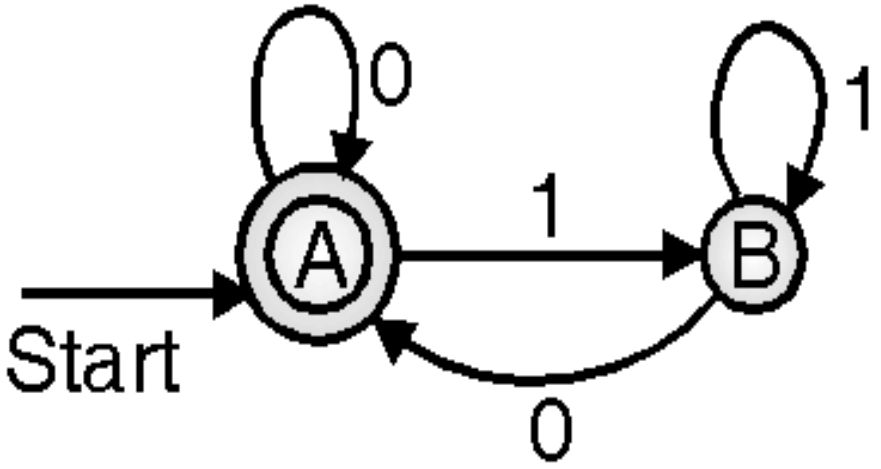
$$q_2 = 0^* 1 + q_2 0$$

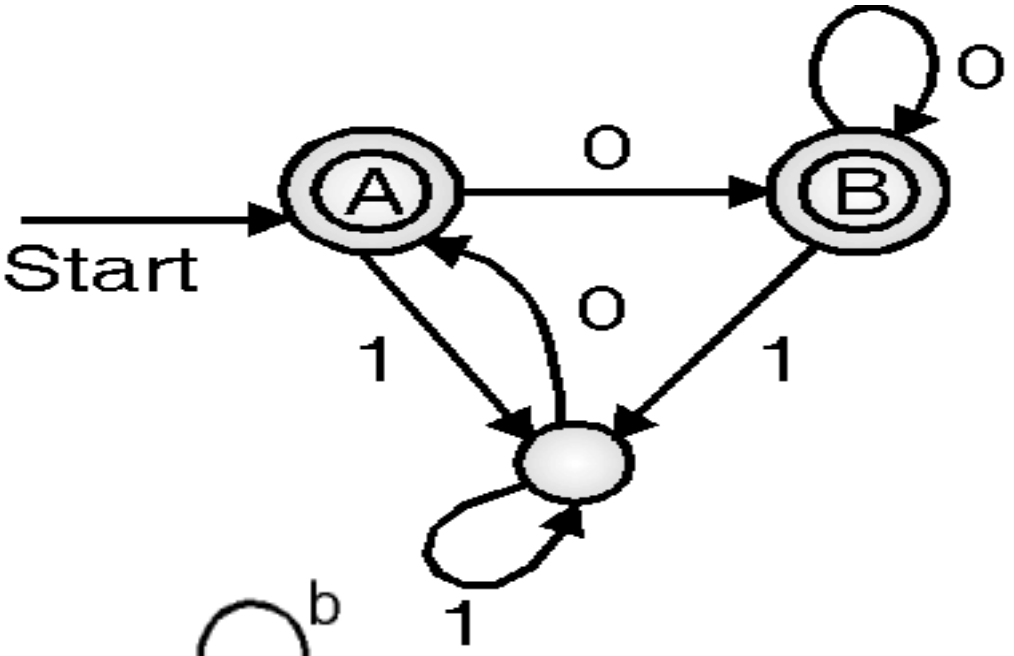
So, $q_2 = 0^* 1 (0)^*$ [By Arden's theorem]

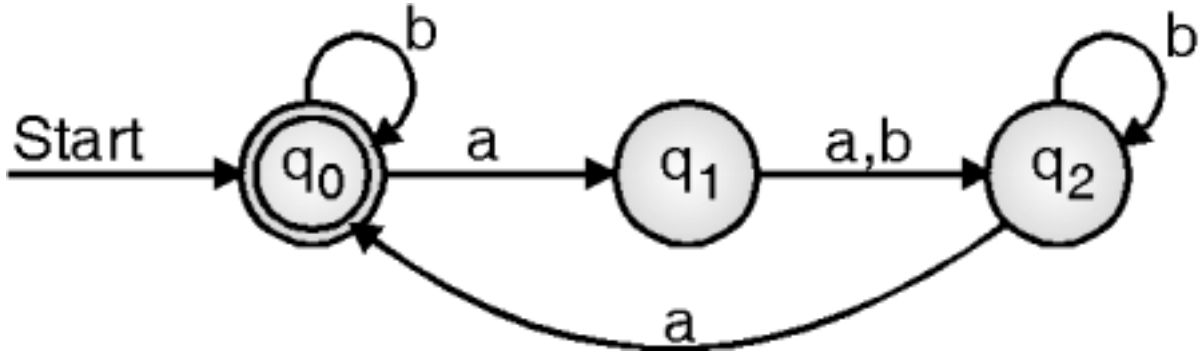
Hence, the regular expression is

$$0^* 1 0^*.$$

- Construct the regular expressions for the following DFAs:

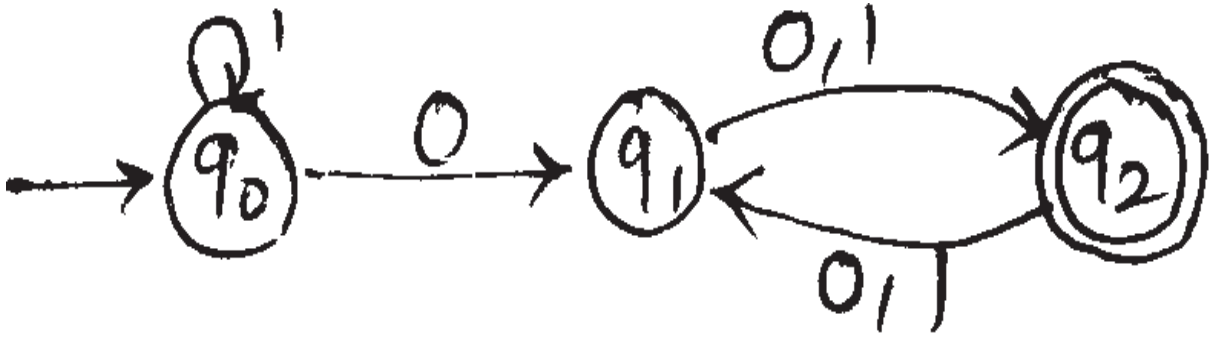


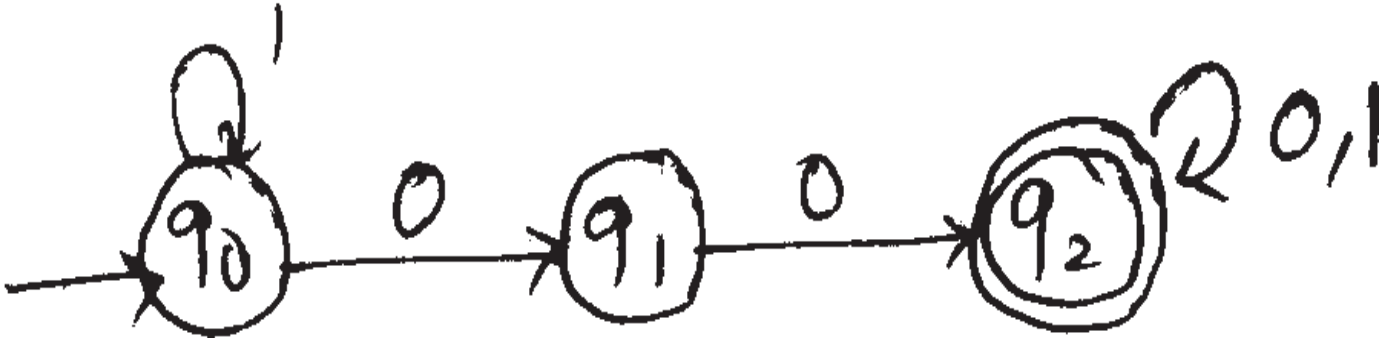




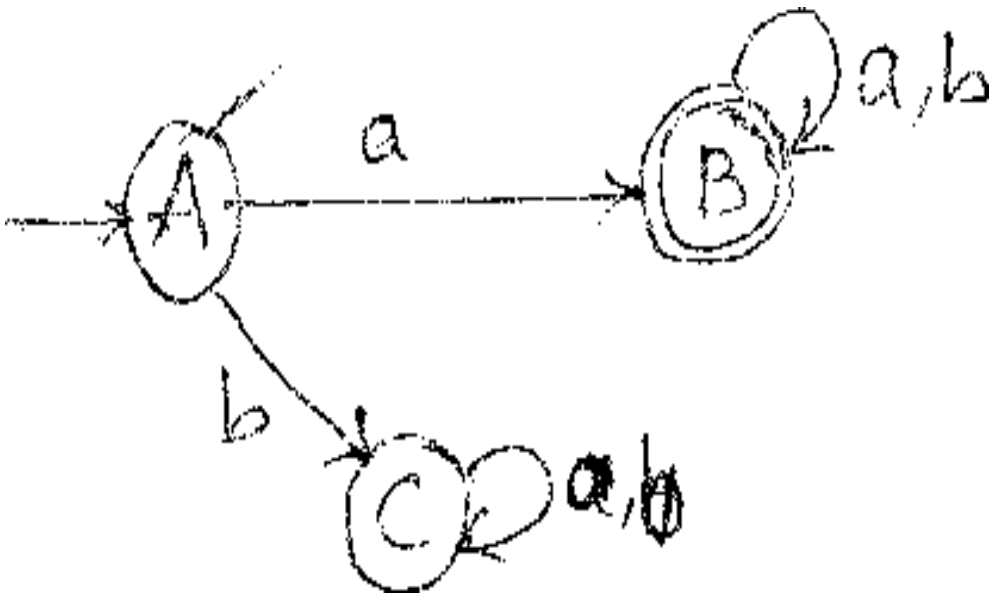
Find the regular expression for the following .

(Nov-2017 4 Marks)



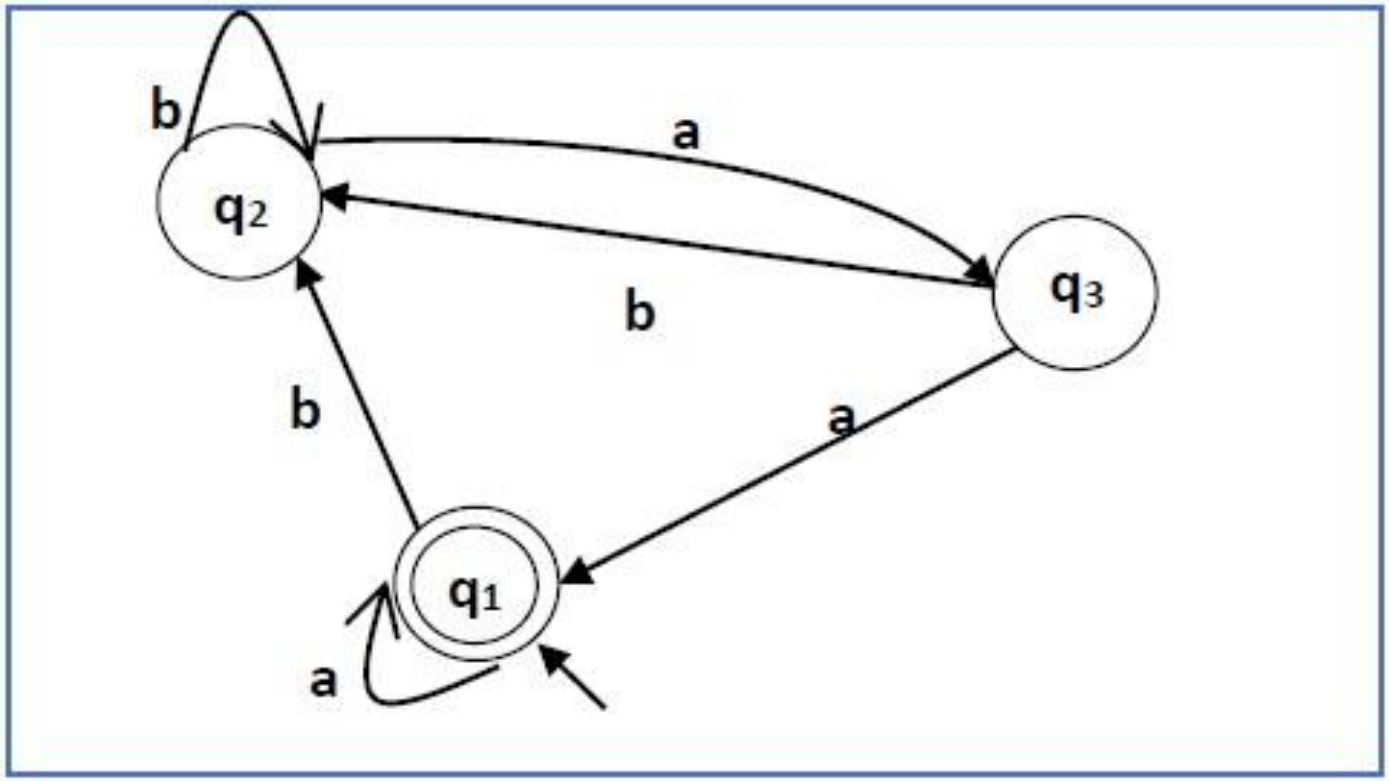


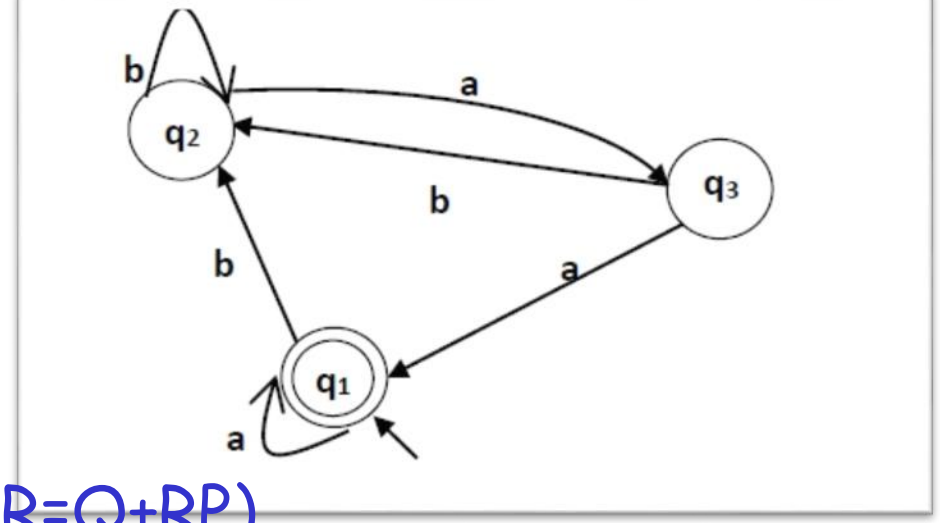
Construct Regular Expression for the following transition diagram using Arden's theorem.(Nov 2014 4 Marks)



Problem

Construct a regular expression corresponding to the automata given below:





$q_1 = q_1a + q_3a + \varepsilon$

$q_2 = q_1b + q_2b + q_3b$

$q_3 = q_2a$

$q_2 = q_1b + q_2b + q_3b$

$q_2 = q_1b + q_2b + q_2ab$

$q_2 = q_1b + q_2(b + ab)$ .....(R=Q+RP)

$q_2 = q_1b(b + ab)^*$ ..... (R=QP*)

$q_1 = q_1a + q_3a + \varepsilon$

$= q_1a + q_2aa + \varepsilon$ (Substituting value of q_3)

$= q_1a + q_1b(b + ab^*)aa + \varepsilon$ (Substituting value of q_2)

$= q_1(a + b(b + ab)^*aa) + \varepsilon$

$= \varepsilon\ (a + b(b + ab)^*aa)^*$

$= (a + b(b + ab)^*aa)^*$

Hence, the regular expression is

$(a + b(b + ab)^*aa)^*$.

Regular Languages: *Grand Unification*

$$L(NFA-\lambda s) = L(NFAs)$$
$$= L(DFAs)$$

(Parallel Simulation)
(Rabin and Scott's work)

$$L(FA) = L(RE)$$

(Collapsing graphs; Structural Induction)
(S. Kleene's work)

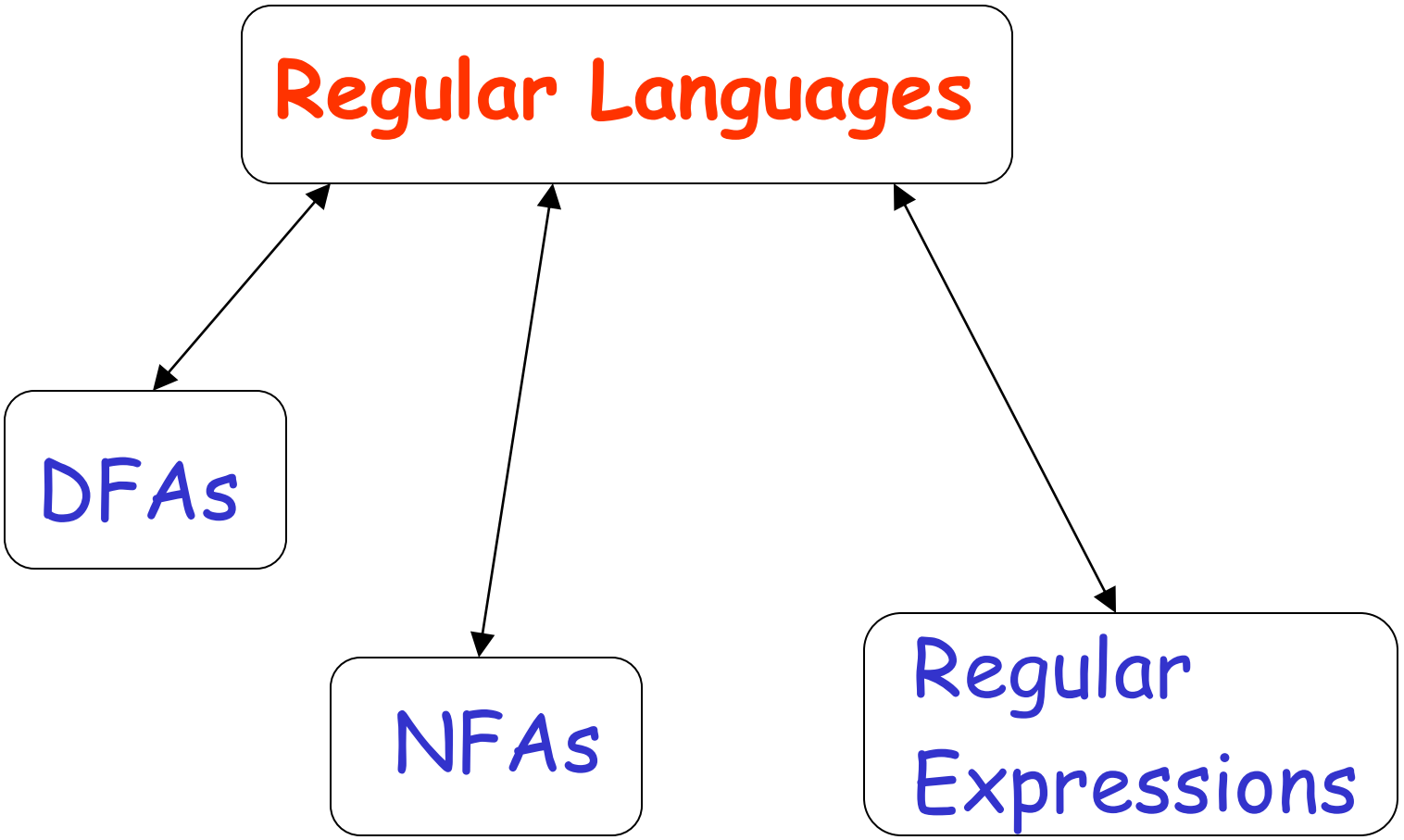
$$L(FA) = L(RG)$$

(Construction)

$$L(RG) = L(RE)$$

(Solving linear equations) study later

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

(DFA, NFA, or Regular Expression)

Non-regular languages

$$\{a^n b^n : \ n \geq 0\}$$

$$\{vv^R : \ v \in \{a,b\}^*\}$$

Regular languages

a^*b

$b^*c + a$

$b + c(a+b)^*$

$etc...$

Non-regular languages

(Pumping Lemma)

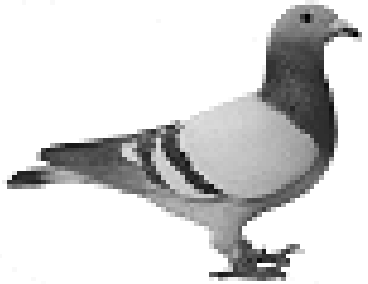
How can we prove that a language L is not regular?

Prove that there is no DFA or NFA or RE that accepts L

Difficulty: this is not easy to prove

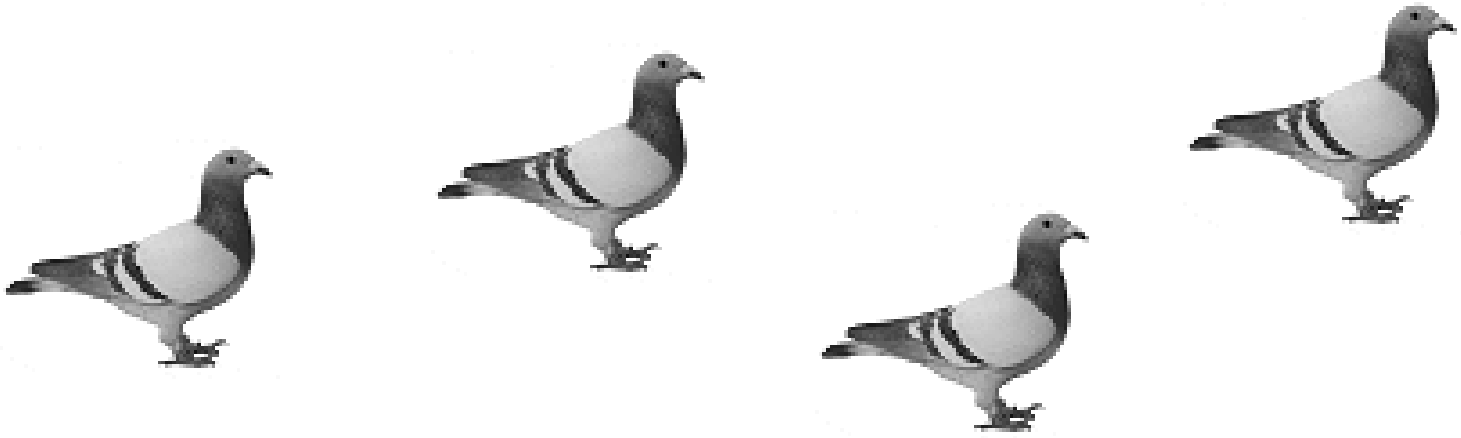
(since there is an infinite number of them)

Solution: use the Pumping Lemma !!!

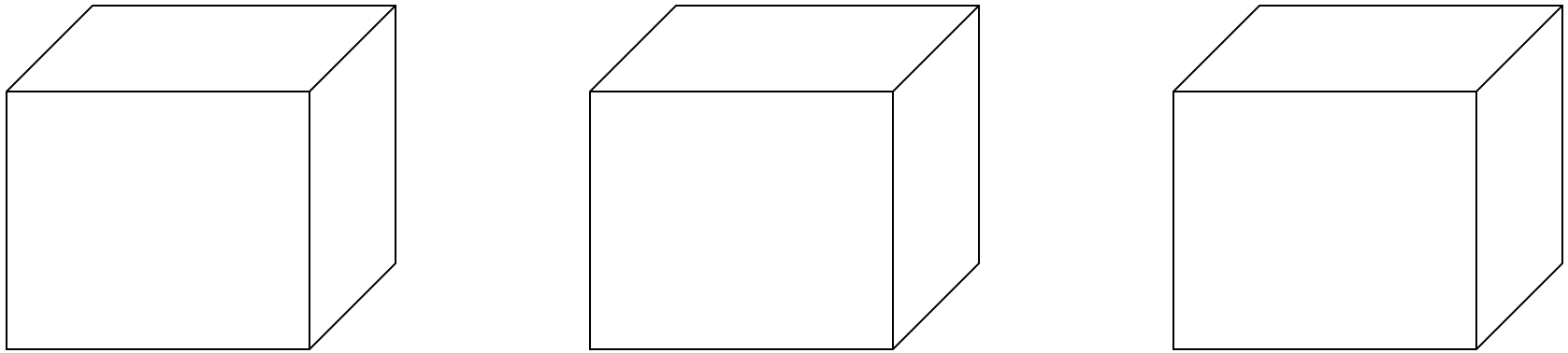


The Pigeonhole Principle

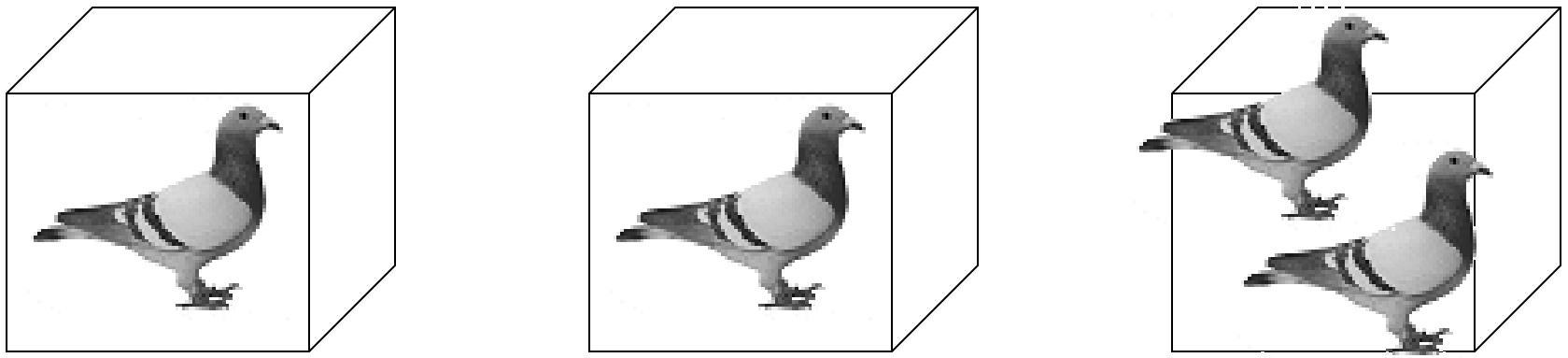
4 pigeons



3 pigeonholes

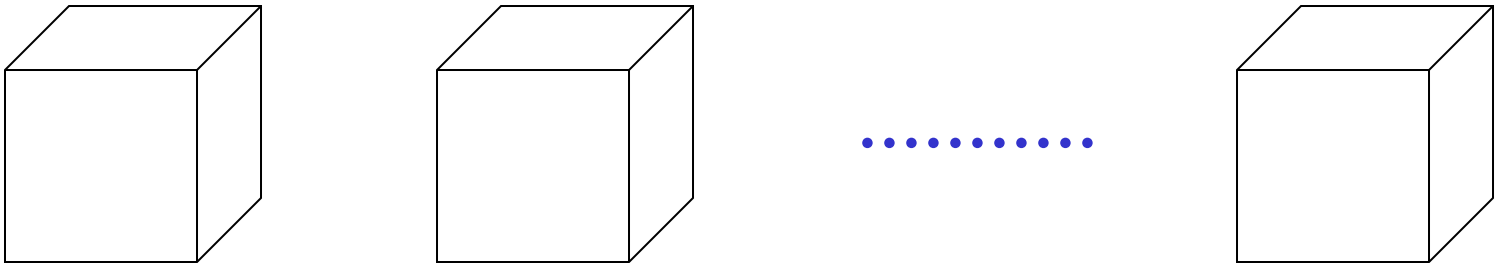


A pigeonhole must
contain at least two pigeons



n pigeons

m pigeonholes $n > m$



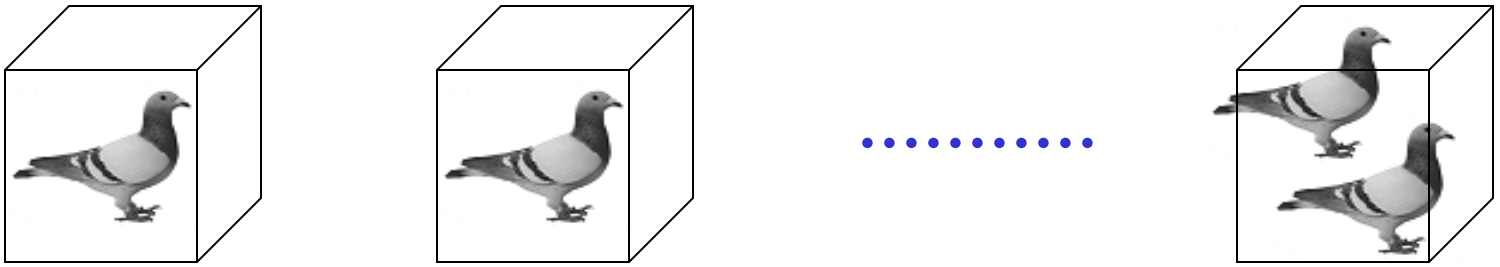
The Pigeonhole Principle

n pigeons

m pigeonholes

$n > m$

There is a pigeonhole with at least 2 pigeons

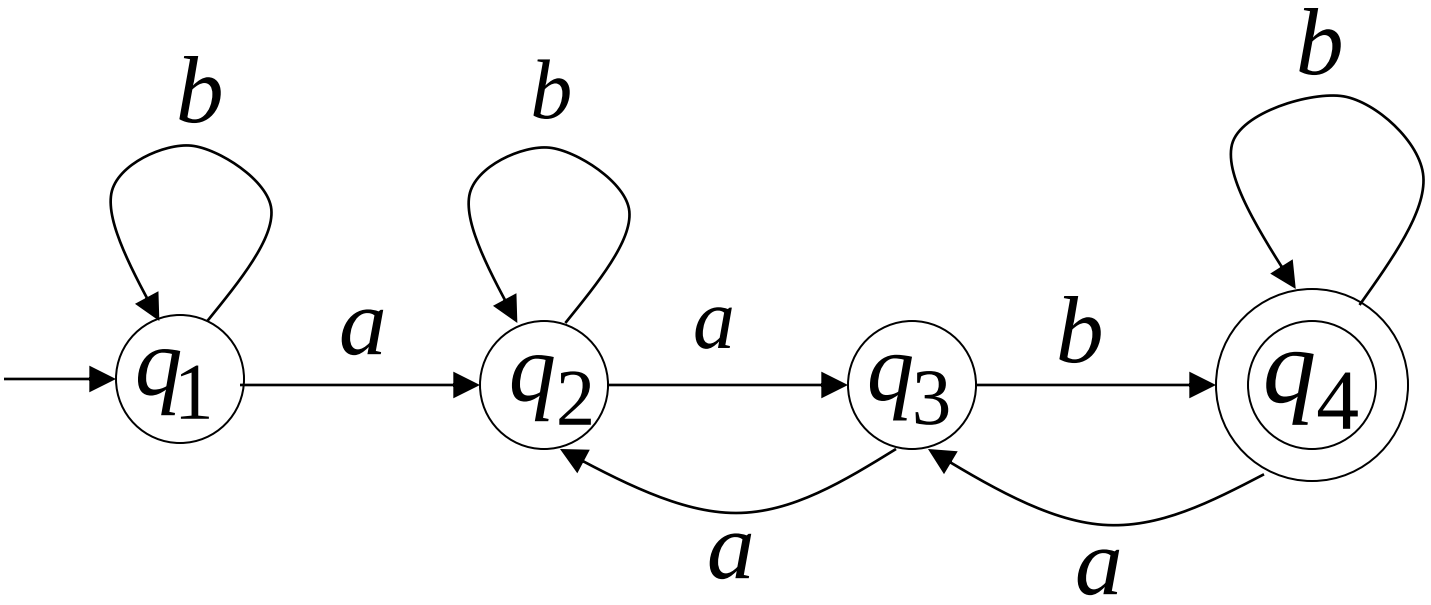


The Pigeonhole Principle

and

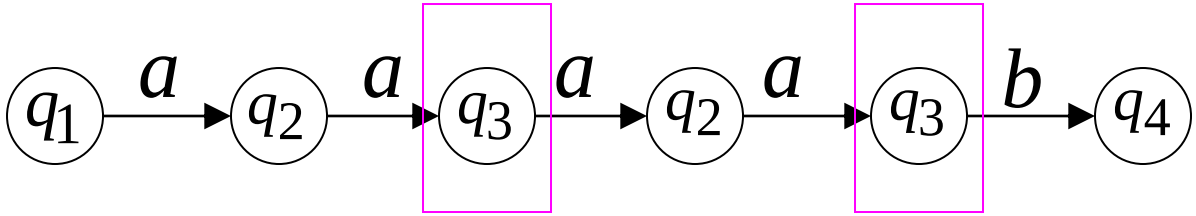
DFAs

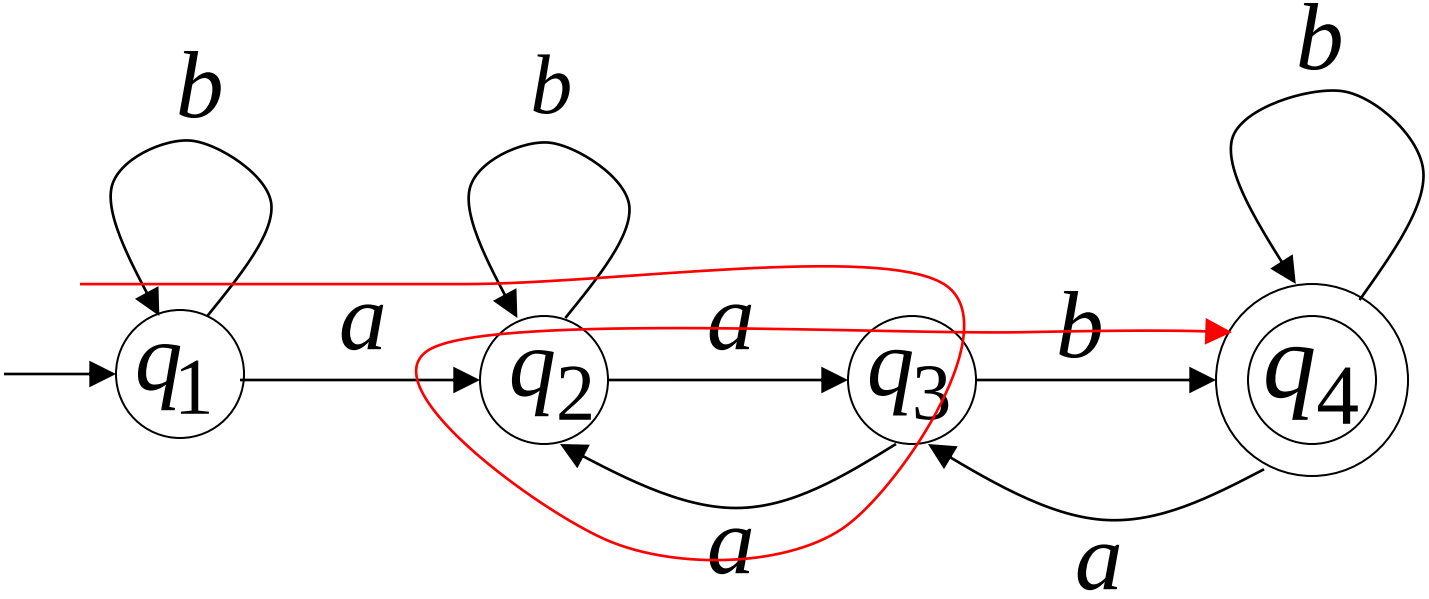
Consider a DFA with 4 states



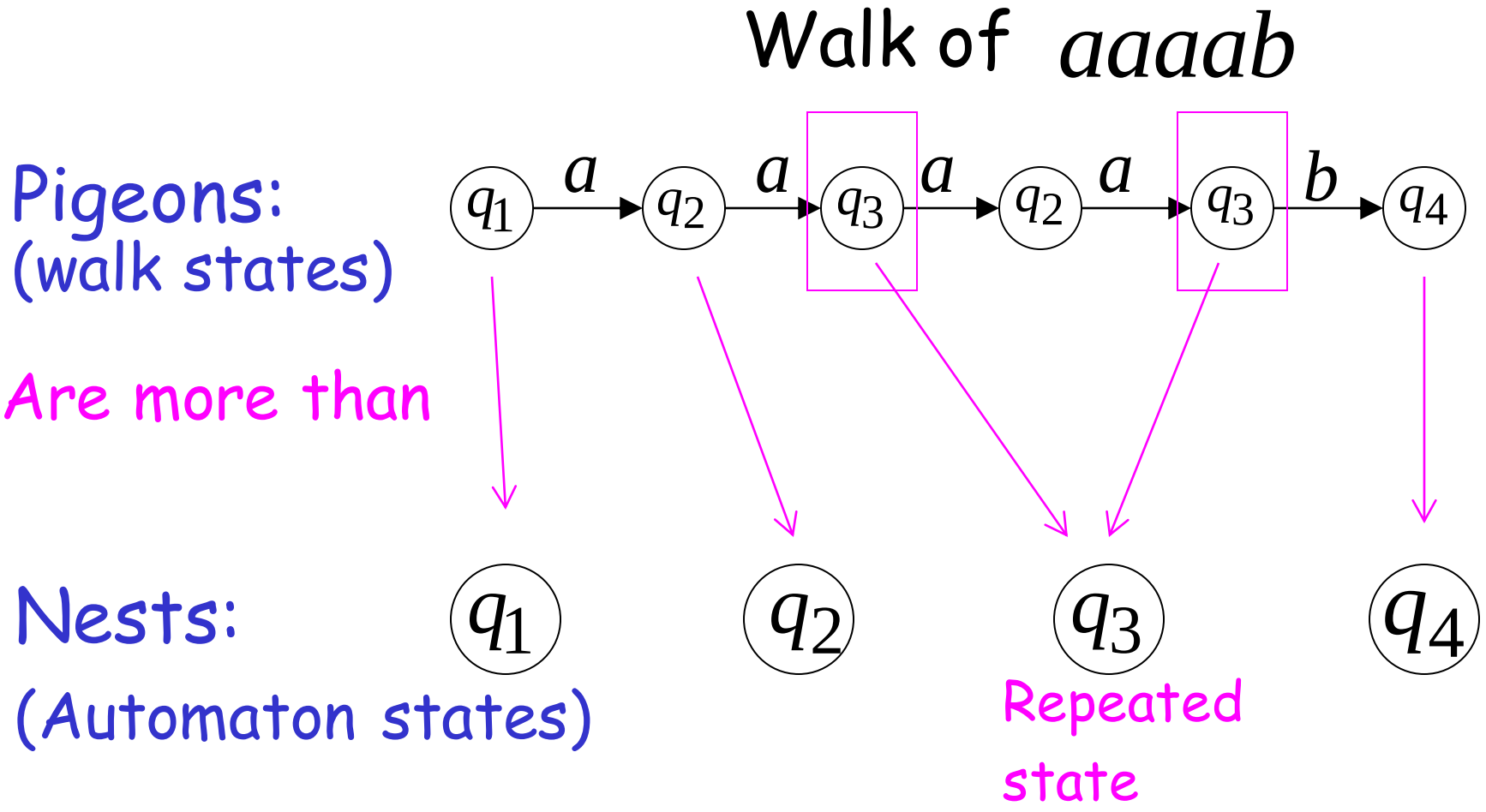
Consider the walk of a "long" string: $aaaab$
(length at least 4)

A state is repeated in the walk of $aaaab$





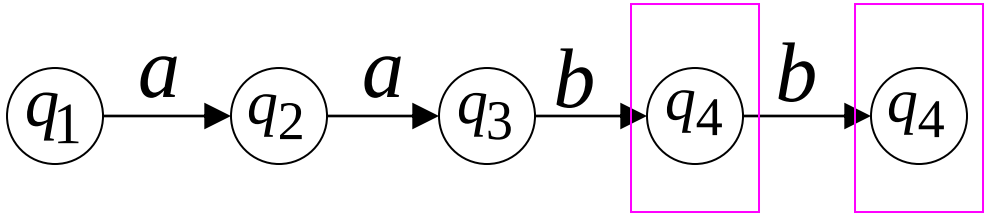
The state is repeated as a result of the pigeonhole principle

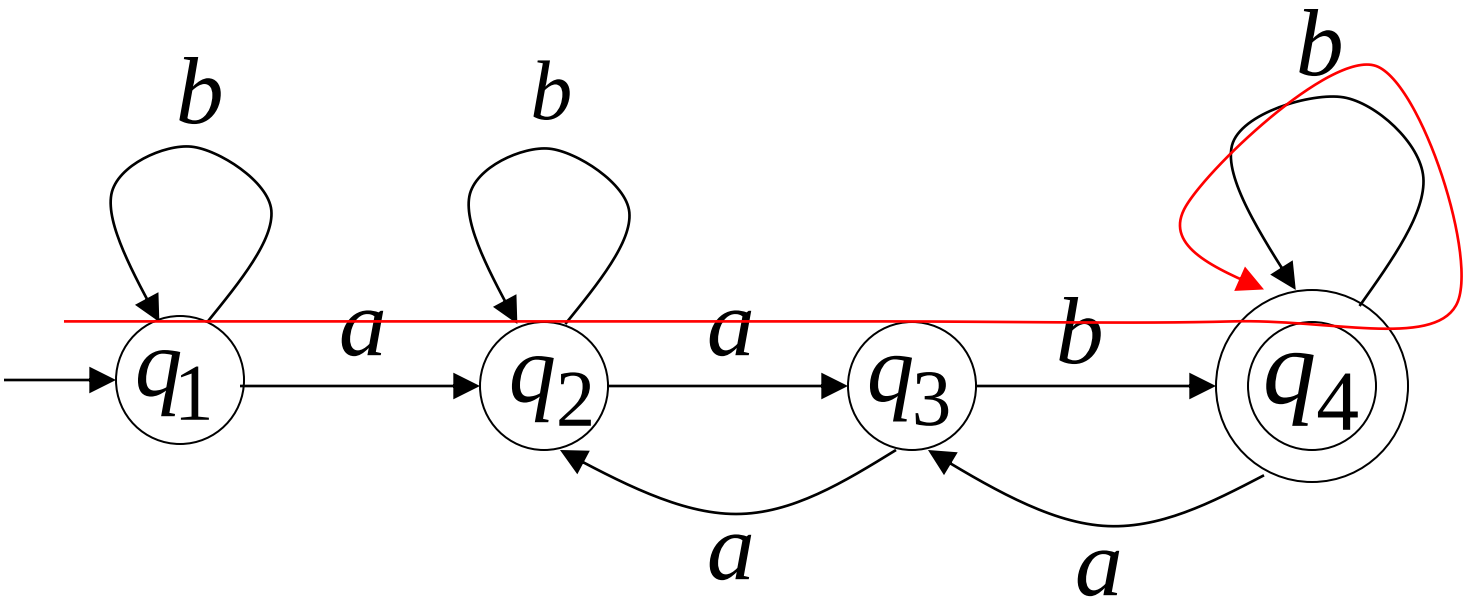


Consider the walk of a "long" string: $aabb$
(length at least 4)

Due to the pigeonhole principle:

A state is repeated in the walk of $aabb$



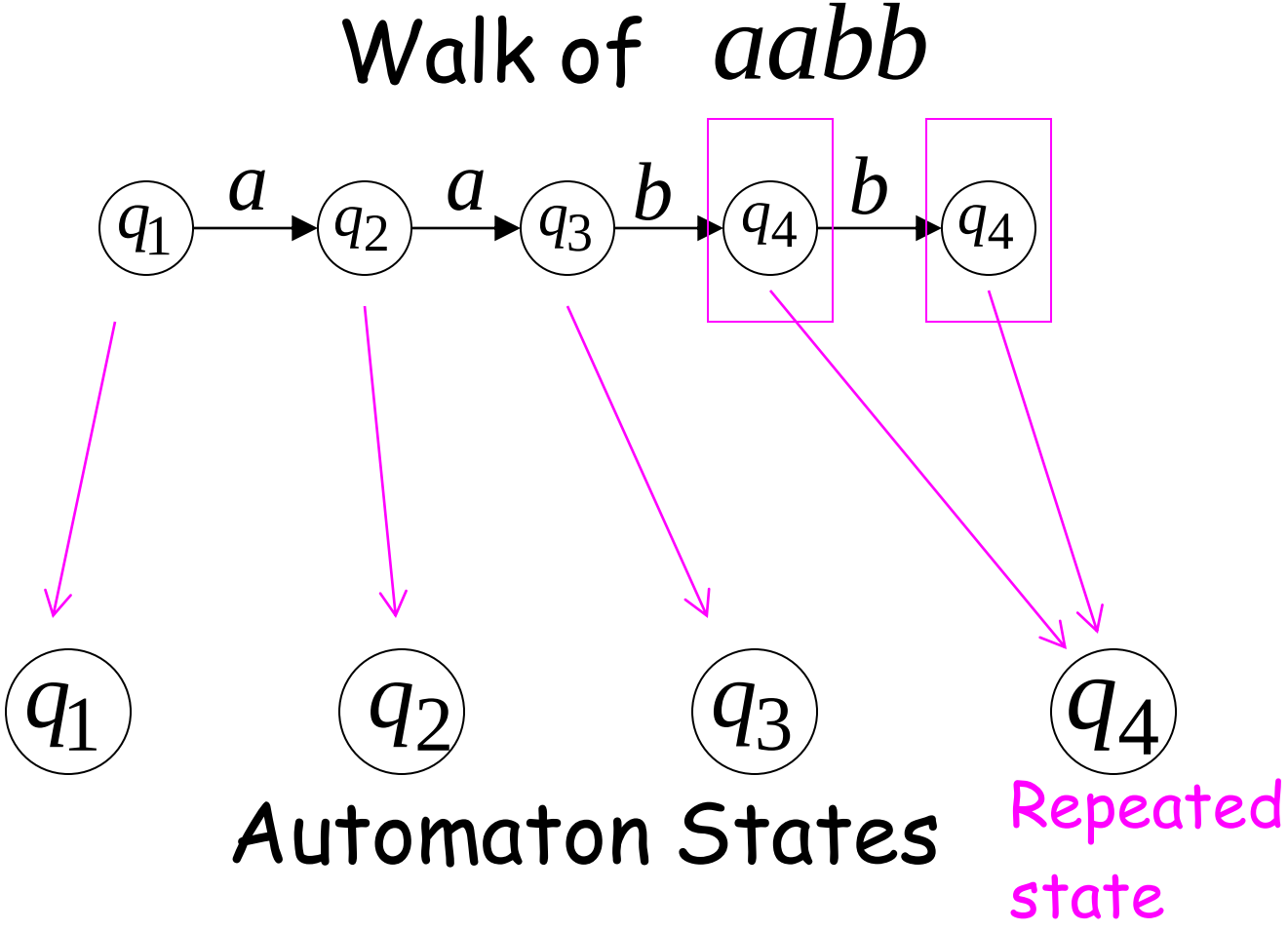


The state is repeated as a result of the pigeonhole principle

Pigeons:
(walk states)

Are more than

Nests:
(Automaton states)

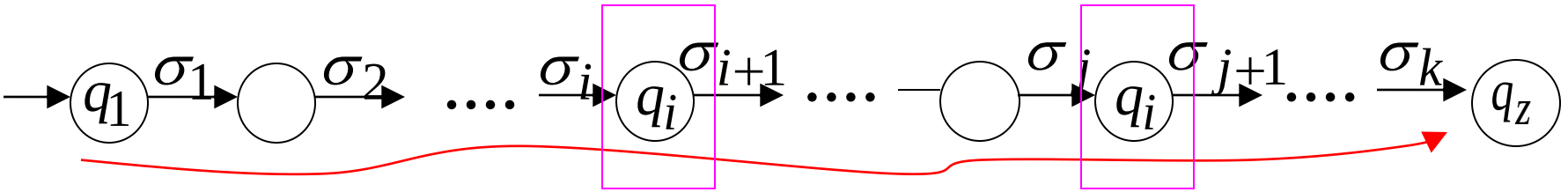


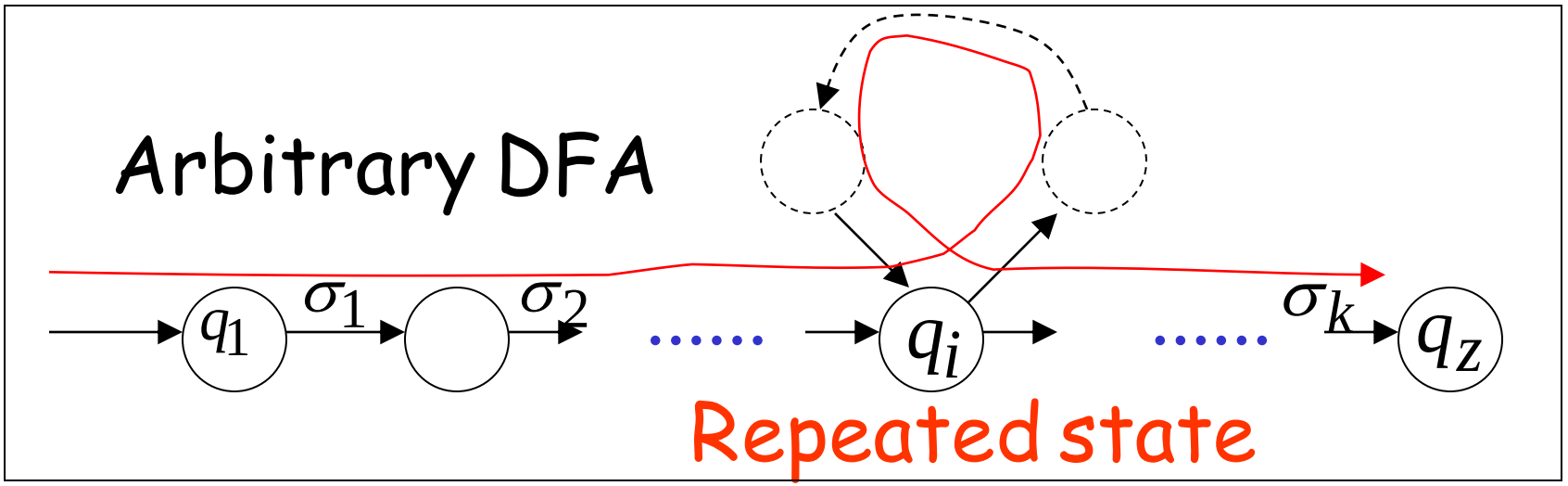
In General: If $|w| \geq$ #states of DFA,

by the pigeonhole principle,

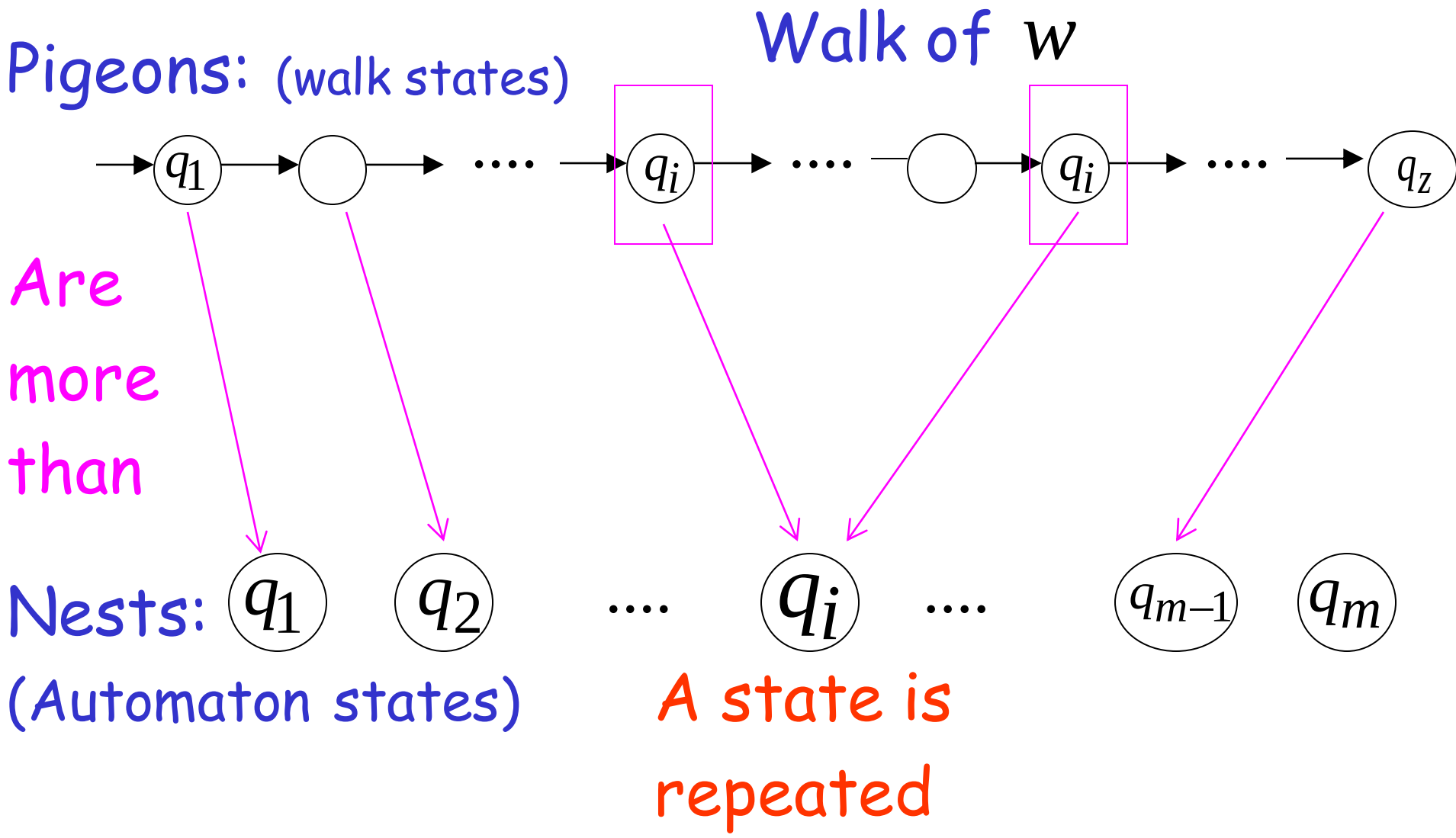
a state is repeated in the walk w

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





$$|w| \geq \#\text{states of DFA} = m$$

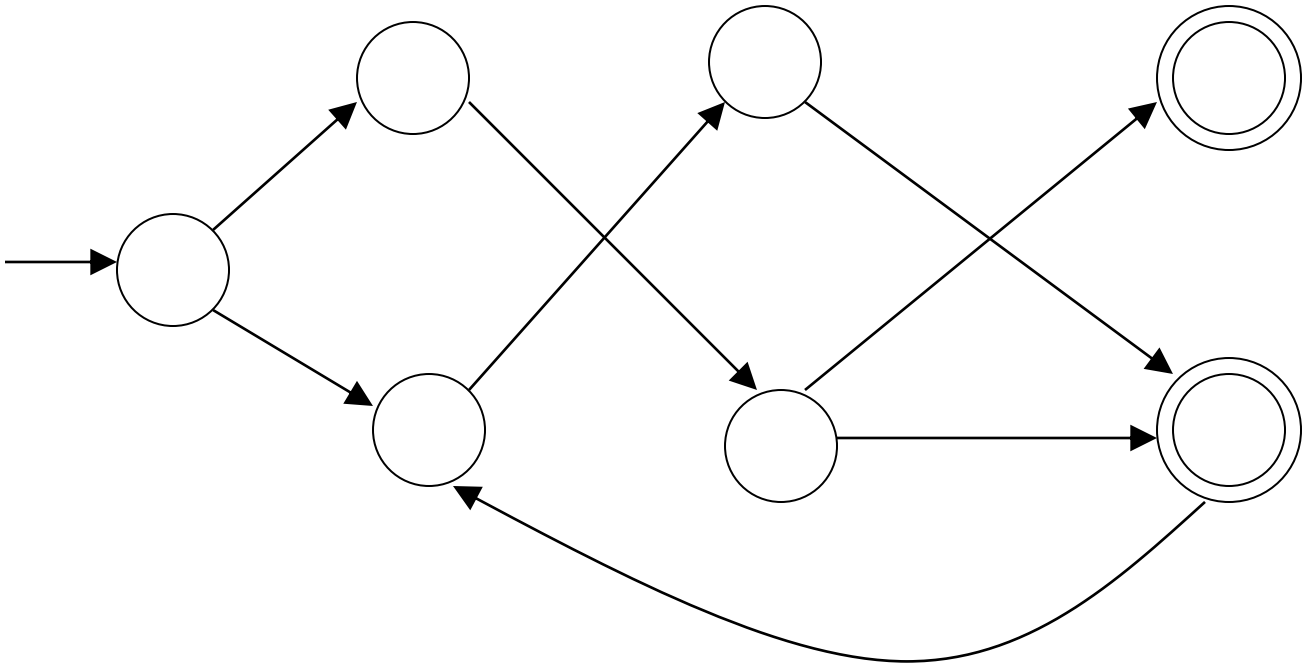


The Pumping Lemma

Take an infinite regular language L

(contains an infinite number of strings)

There exists a DFA that accepts L

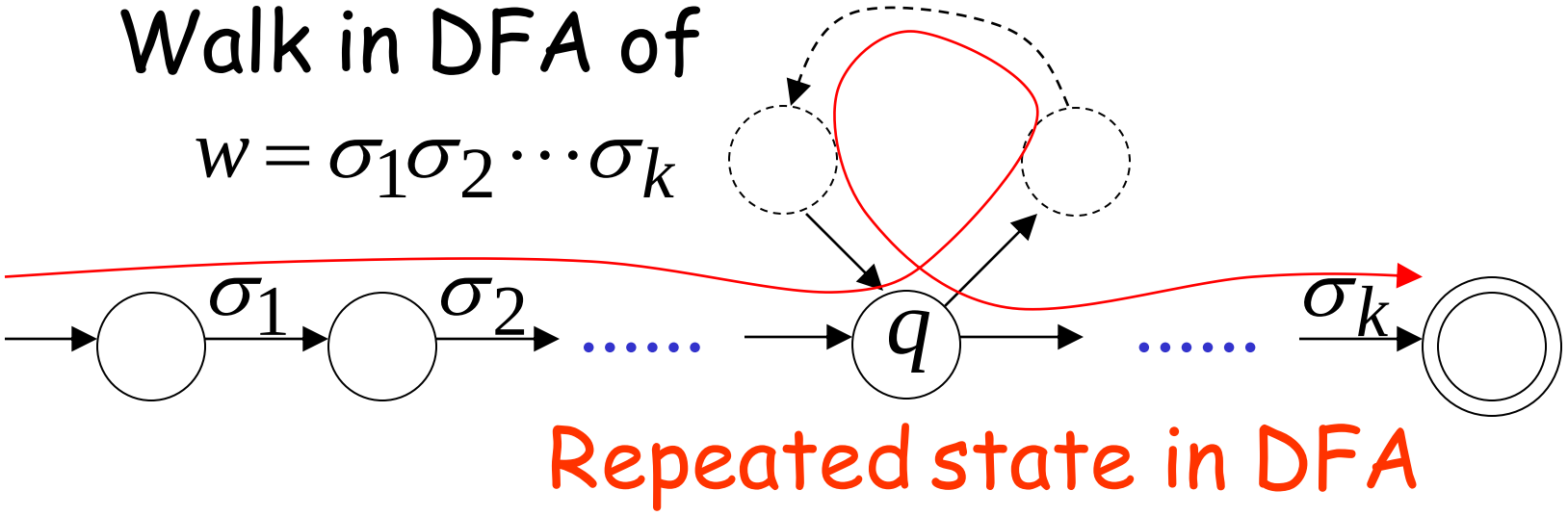


m states

Take string $w \in L$ with $|w| \geq m$

(number of states of DFA)

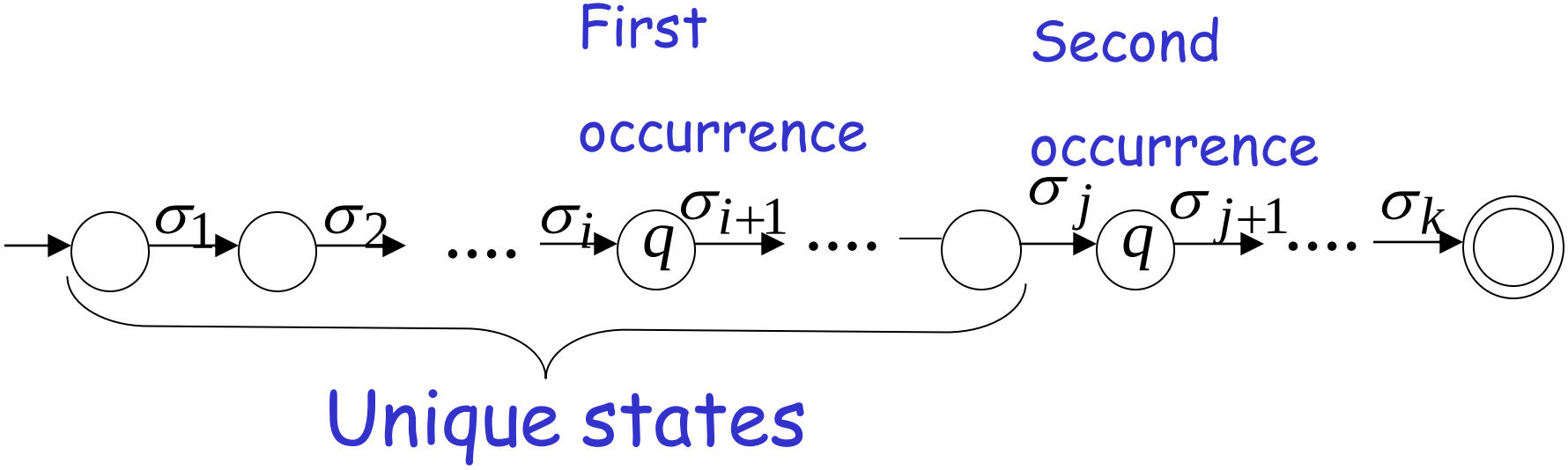
then, at least one state is repeated in the walk of w



There could be many states repeated

Take q to be the first state repeated

One dimensional projection of walk w :



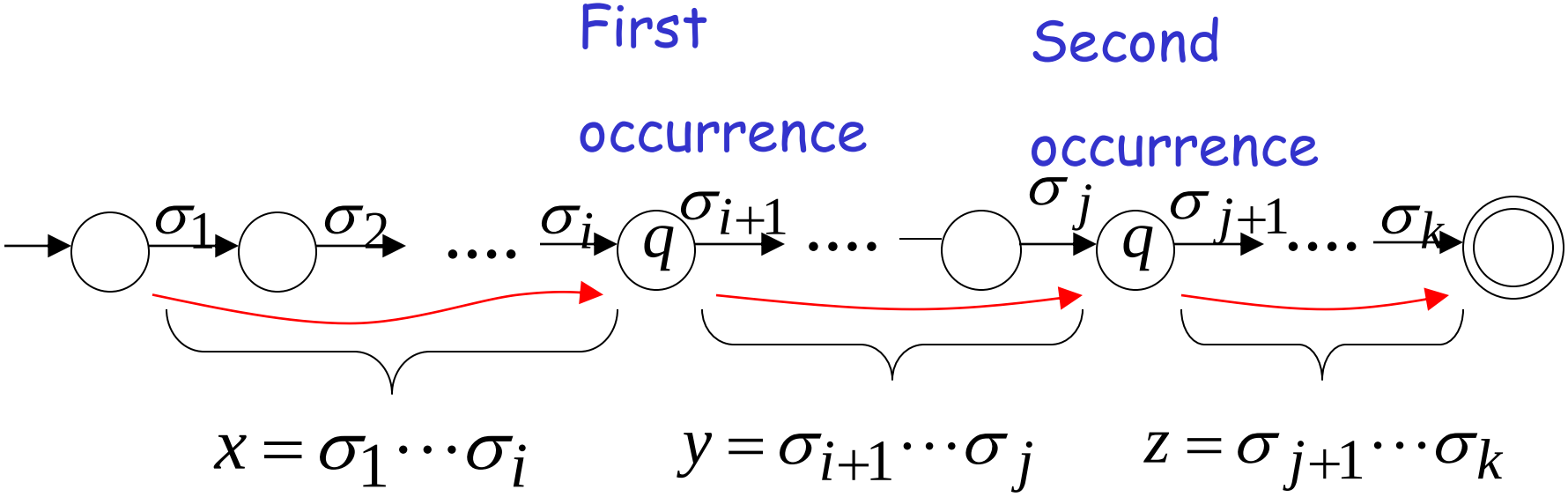
Non-regular languages

Pumping Lemma for Regular Languages

- It is a necessary condition.
  - Every regular language satisfies it.
  - If a language violates it, it is not regular.
    - RL => PL not PL => not RL
- It is *not* a sufficient condition.
  - Not every non-regular language violates it.
    - not RL =>? PL or not PL (no conclusion)

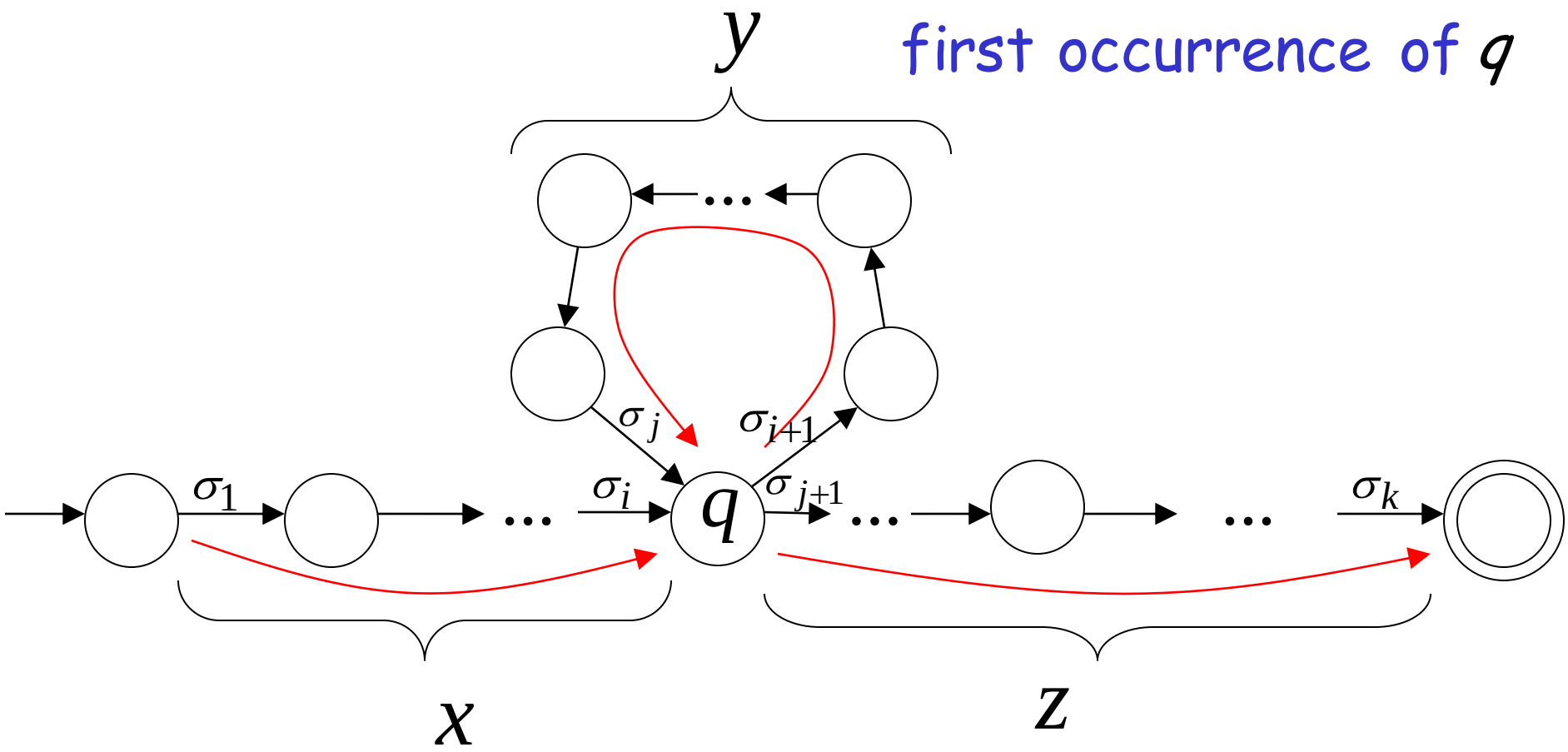
We can write $w = xyz$

One dimensional projection of walk w :

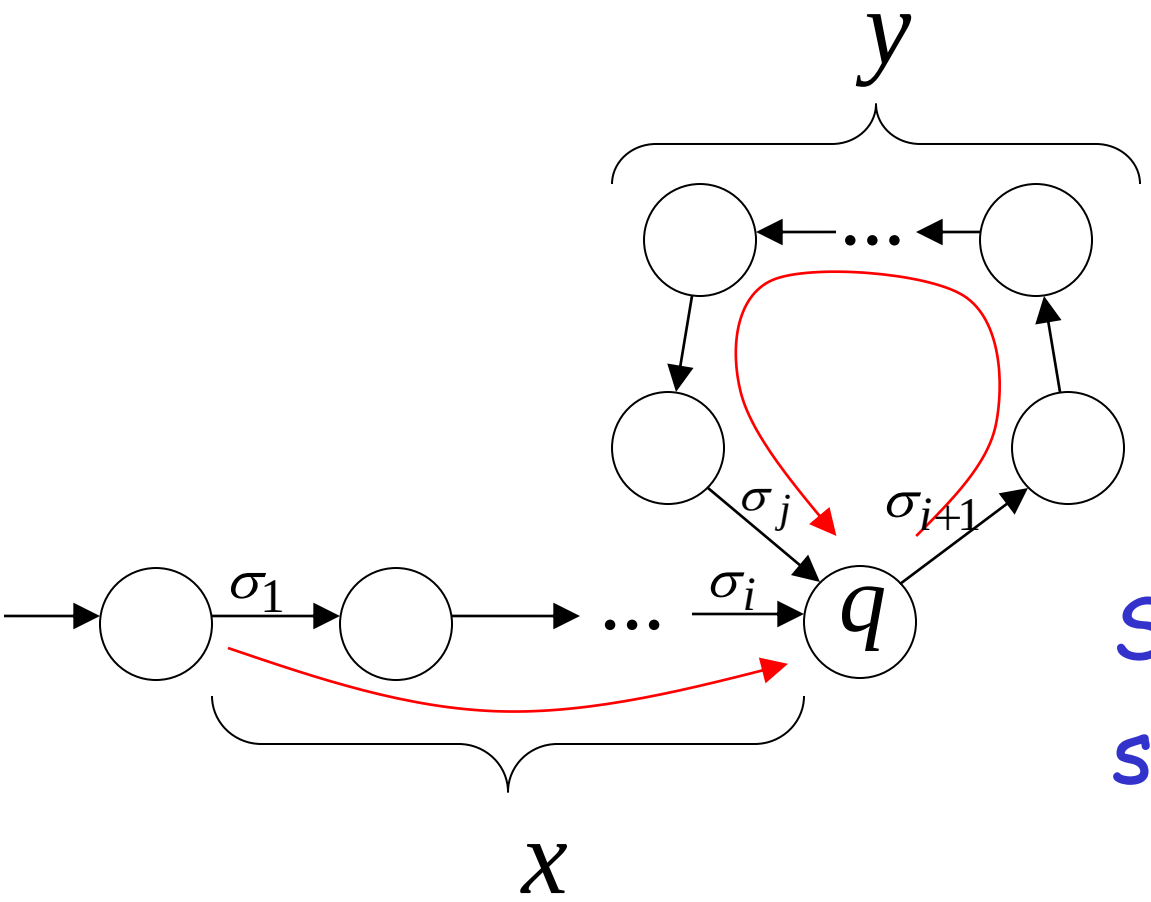


In DFA: $w = x\ y\ z$

contains only
first occurrence of q



Observation: length $|xy| \leq m$ number of states of DFA



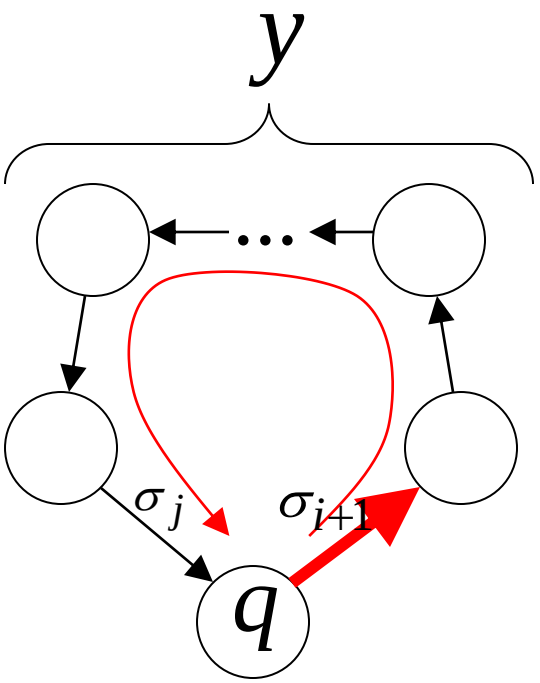
Unique States

Since, in xy no state is repeated

(except q)

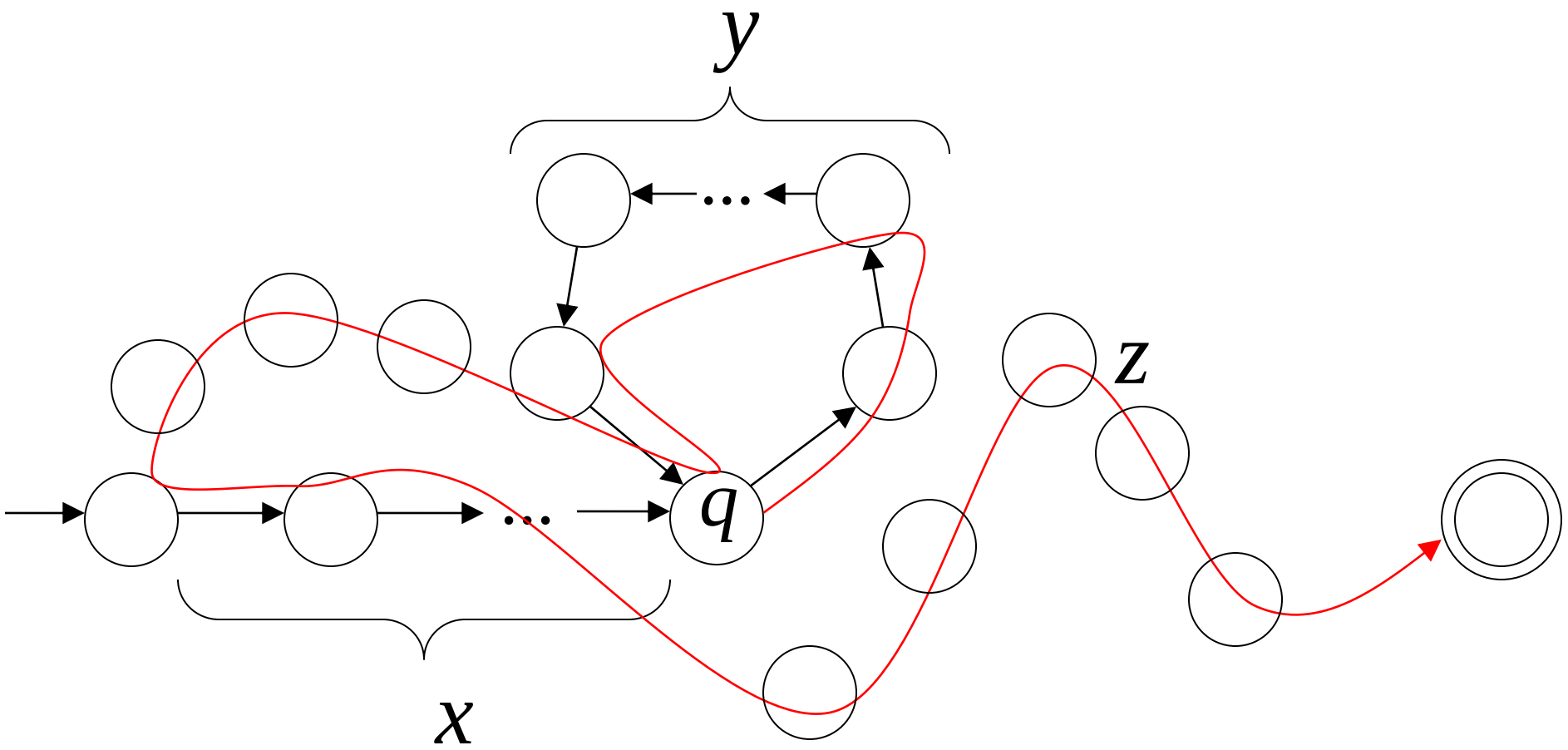
Observation: length $|y| \geq 1$

Since there is at least one transition in loop



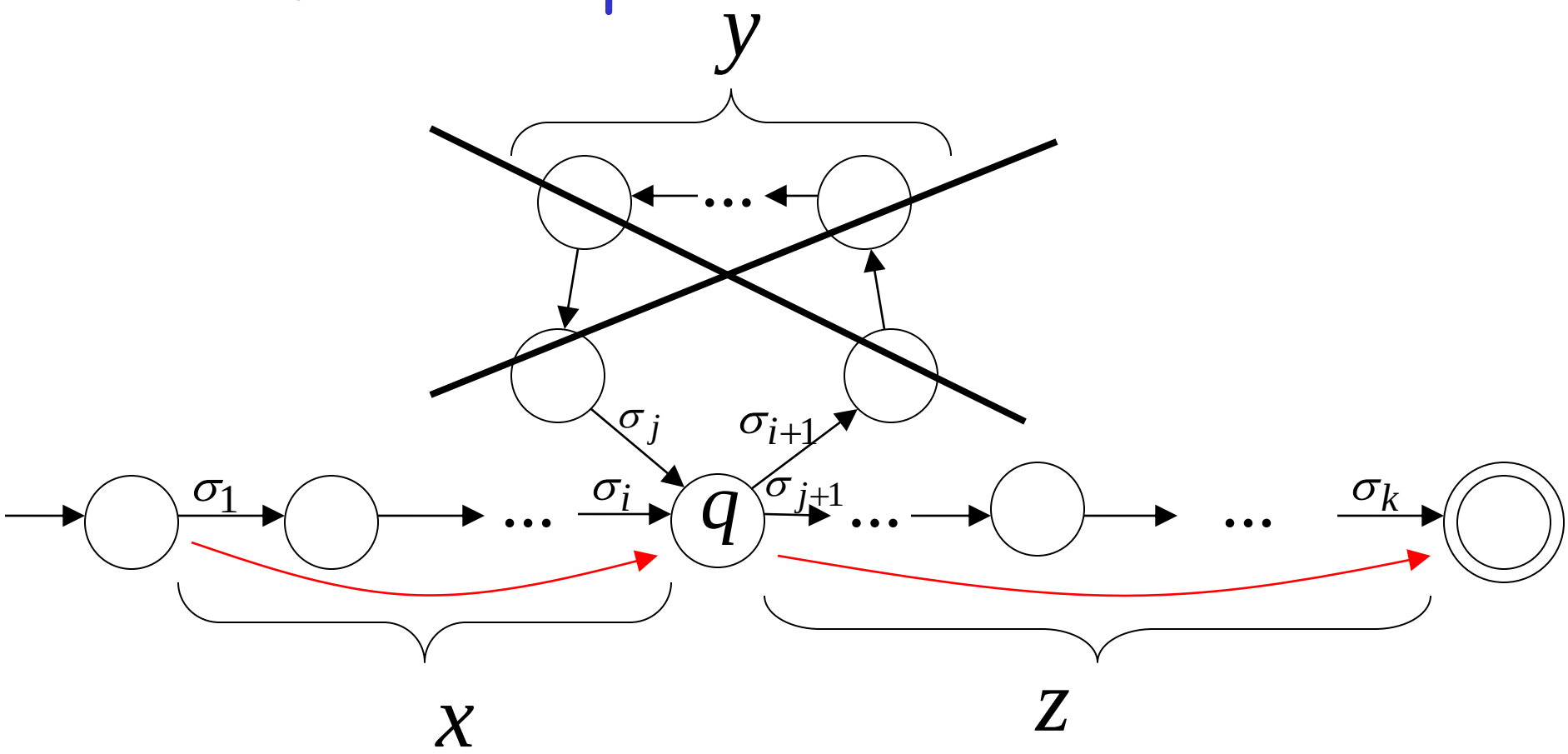
We do not care about the form of string z

z may actually overlap with the paths of x and y



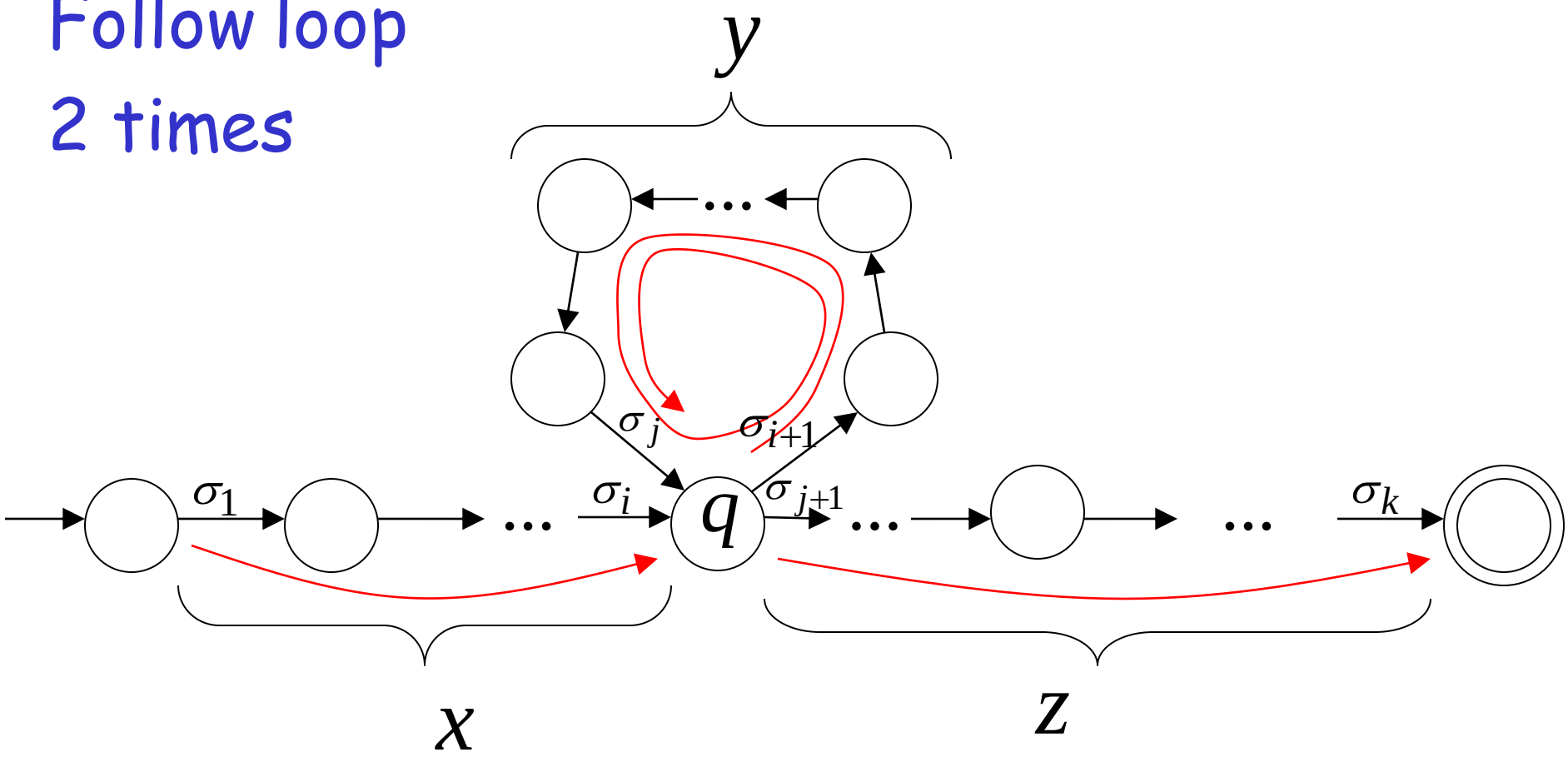
Additional string: The string $x\ z$ is accepted

Do not follow loop



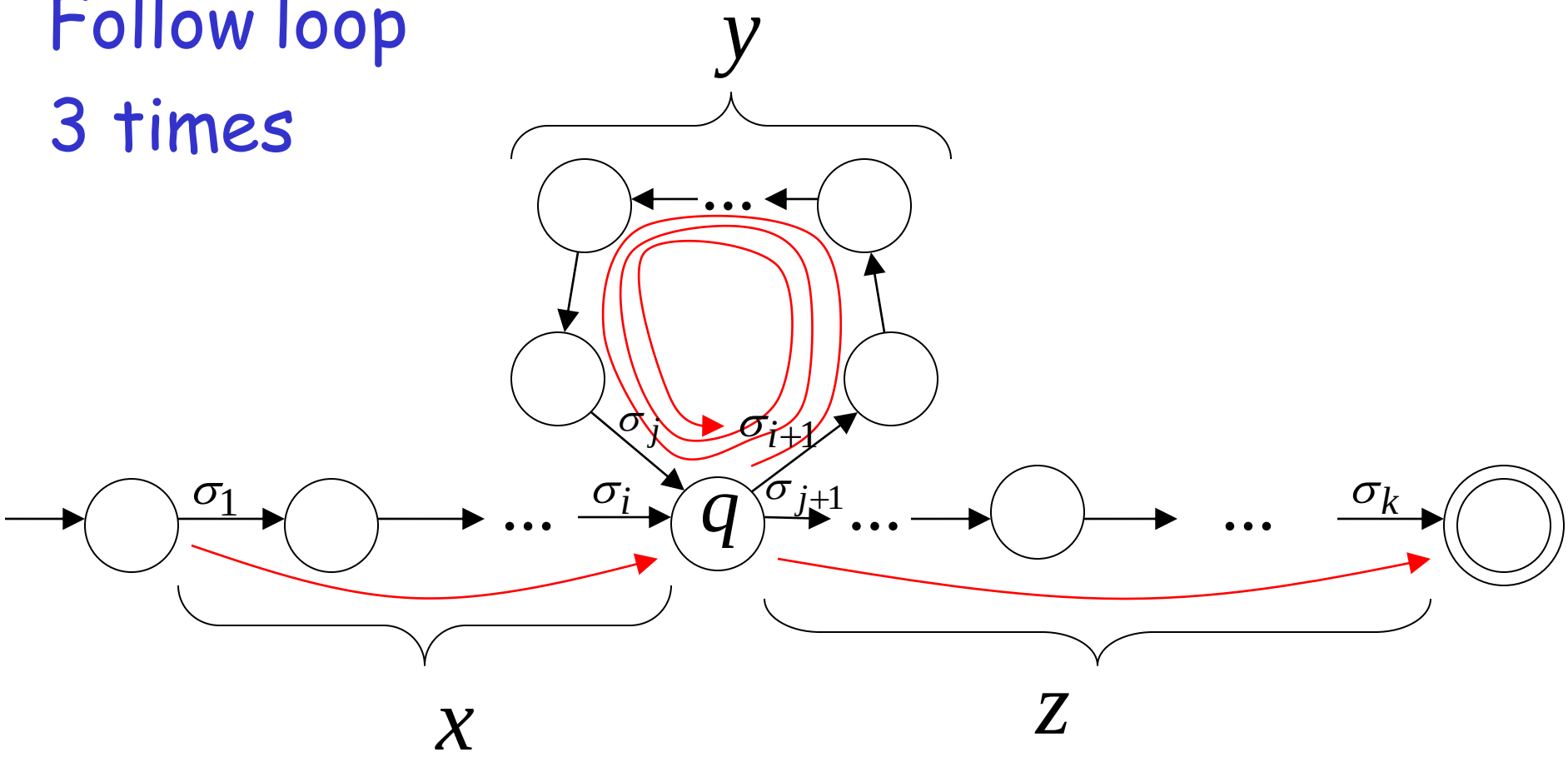
Additional string: The string $x\ y\ y\ z$ is accepted

Follow loop 2 times



Additional string: The string $x\ y\ y\ y\ z$ is accepted

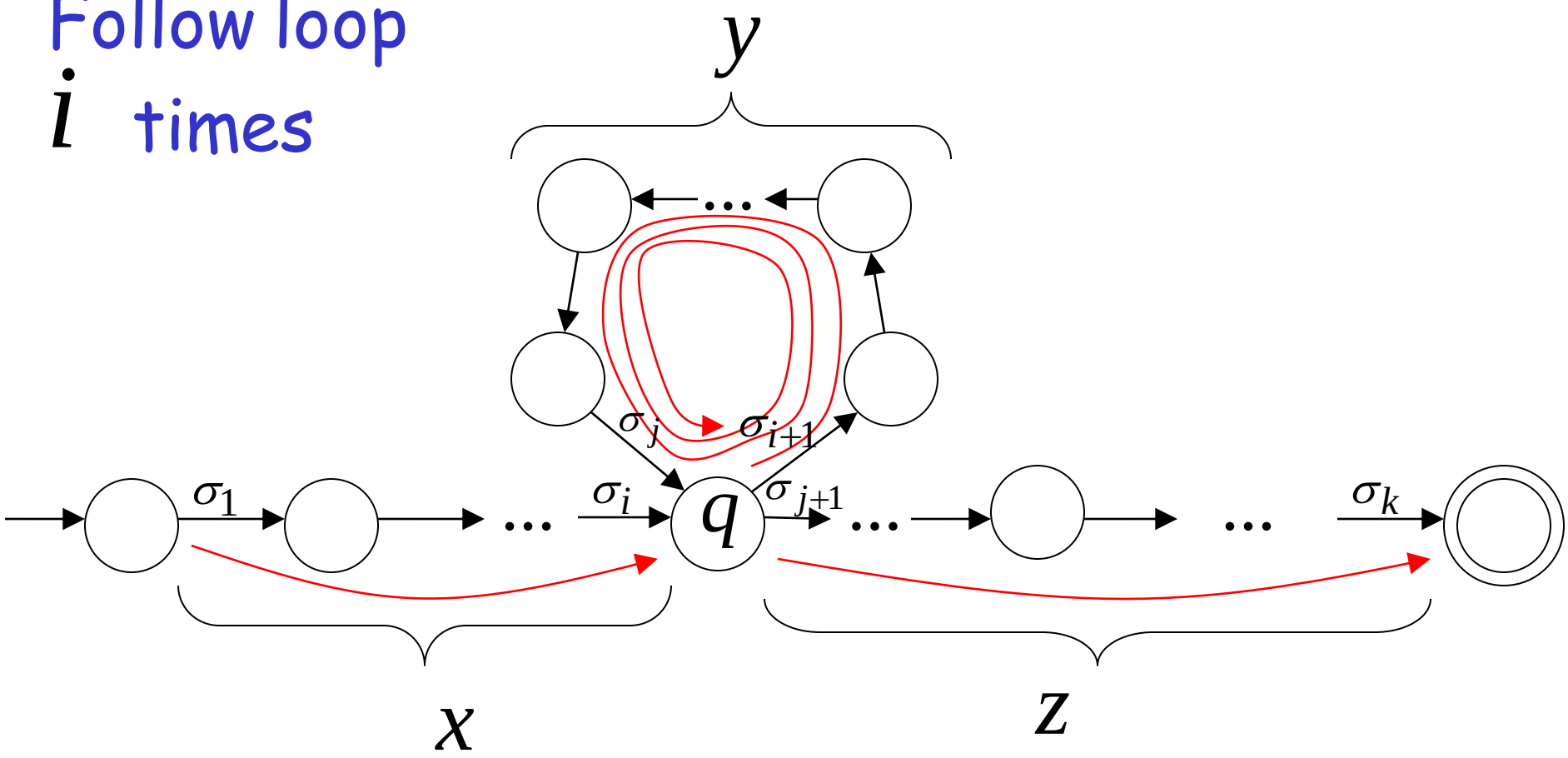
Follow loop 3 times



In General: The string $x\,y^i\,z$

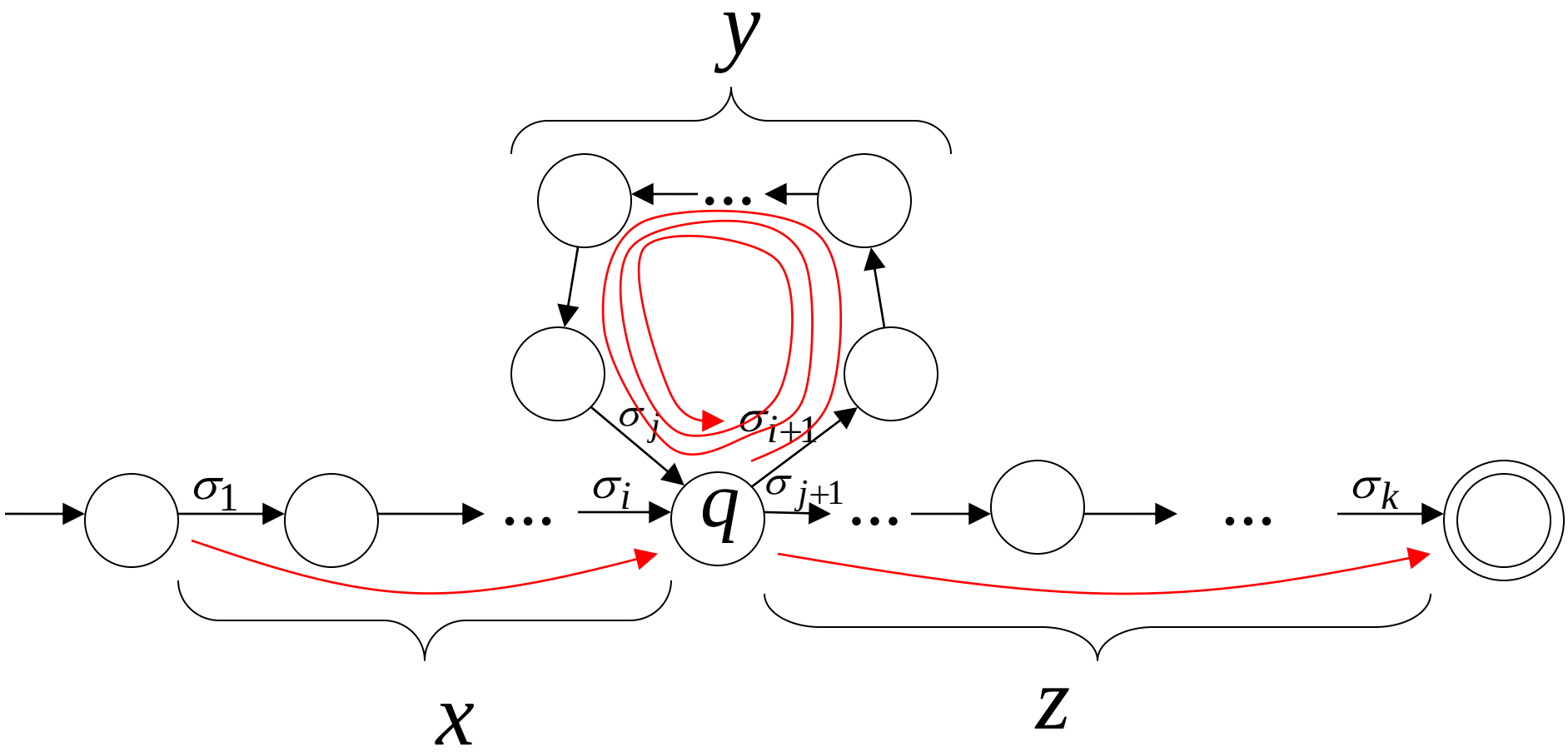
is accepted $i = 0, 1, 2, \ldots$

Follow loop i times

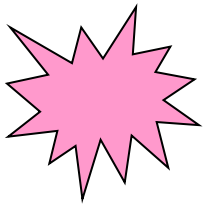


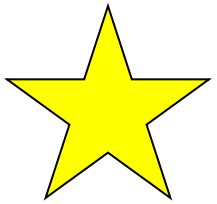
Therefore: $x\, y^i\, z \in L \qquad i = 0, 1, 2, \ldots$

Language accepted by the DFA

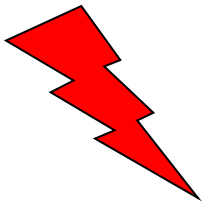


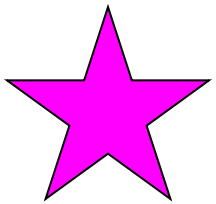
In other words, we described:





The Pumping Lemma !!!





The Pumping Lemma:

- Given a infinite regular language L
- there exists an integer m (critical length)
- for any string $w \in L$ with length $|w| \geq m$
- we can write $w = x\ y\ z$
- with $|x\ y| \leq m$ and $|y| \geq 1$
- such that: $x\ y^i\ z \in L \qquad i = 0, 1, 2, \ldots$

In the book:

Critical length m = Pumping length p

Applications

of

the Pumping Lemma

Observation:

Every language of finite size has to be regular

(we can easily construct an NFA
that accepts every string in the language)

Therefore, every non-regular language
has to be of infinite size

(contains an infinite number of strings)

Suppose you want to prove that
An infinite language L is not regular

1. Assume the opposite: L is regular

2. The pumping lemma should hold for L

3. Use the pumping lemma to obtain a contradiction

4. Therefore, L is not regular

Explanation of Step 3: How to get a contradiction

1. Let m be the critical length for L

2. Choose a particular string $w \in L$ which satisfies the length condition $|w| \geq m$

3. Write $w = xyz$

4. Show that $w' = xy^i z \notin L$ for some $i \neq 1$

5. This gives a contradiction, since from pumping lemma $w' = xy^i z \in L$

Note: It suffices to show that only one string $w \in L$ gives a contradiction

You don't need to obtain contradiction for every $w \in L$

Example of Pumping Lemma application

Theorem: The language $L = \{a^n b^n : n \geq 0\}$ is not regular

Proof: Use the Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

Let m be the critical length for L

Pick a string w such that: $w \in L$

and length $|w| \geq m$

We pick $w = a^m b^m$

From the Pumping Lemma:

we can write $w = a^m b^m = x\ y\ z$

with lengths $|x\ y| \leq m, \quad |y| \geq 1$

$$w = xyz = a^m b^m = \overbrace{a\ldots a}^{m}\overbrace{a\ldots a}^{m}\overbrace{a\ldots a}^{m}\overbrace{b\ldots b}^{m}$$

$$x \quad y \quad z$$

Thus: $y = a^k, \quad 1 \leq k \leq m$

$$x\ y\ z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\ y^i\ z \in L$

$$i = 0, 1, 2, \ldots$$

Thus: $x\ y^2\ z \in L$

$$x\,y\,z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^2\,z \in L$

$$xy^2z = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots a}_{y}\underbrace{a\ldots a}_{y}\underbrace{a\ldots a}_{z}}^{m+k}\overbrace{b\ldots b}^{m} \in L$$

Thus: $a^{m+k}b^m \in L$

$$a^{m+k}b^m \in L \qquad k \geq 1$$

BUT: $L = \{a^n b^n : n \geq 0\}$

$$a^{m+k}b^m \notin L$$

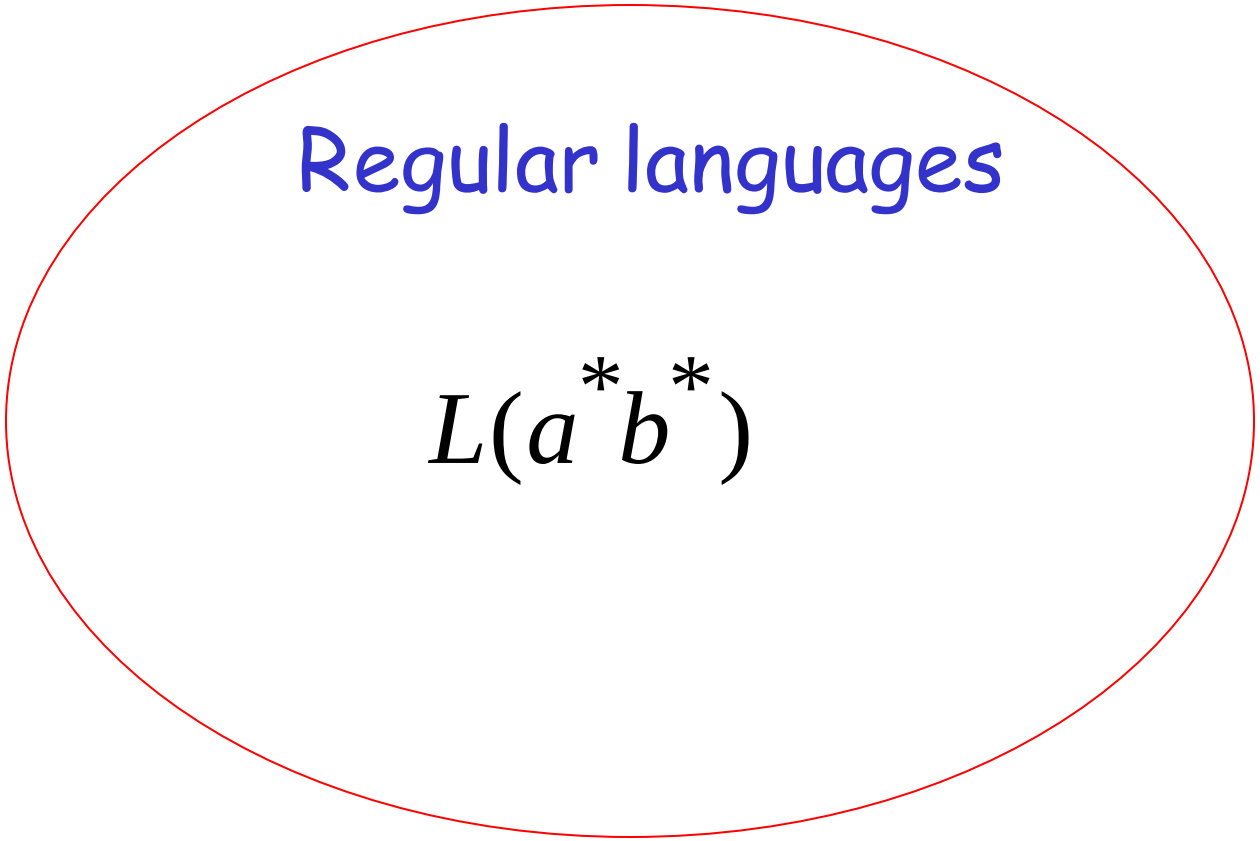
CONTRADICTION!!!

Therefore: Our assumption that L is a regular language is not true

Conclusion: L is not a regular language

END OF PROOF

Non-regular language $\{a^n b^n : \ n \geq 0\}$



More Applications

of

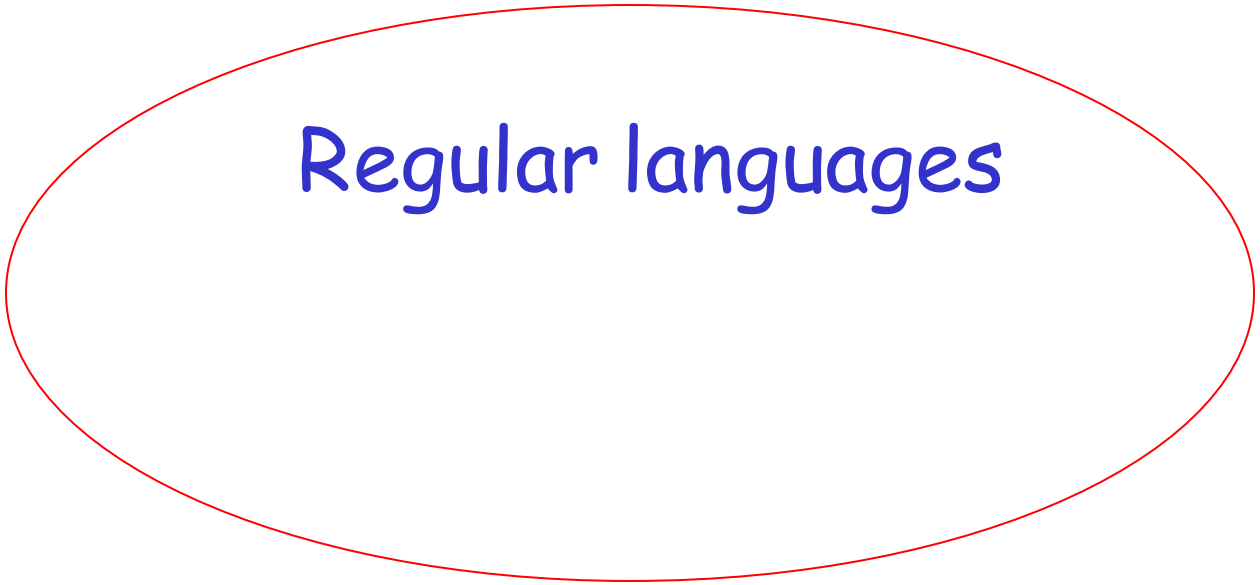
the Pumping Lemma

The Pumping Lemma:

- Given a infinite regular language L
- there exists an integer m (critical length)
- for any string $w \in L$ with length $|w| \geq m$
- we can write $w = x\ y\ z$
- with $|x\ y| \leq m$ and $|y| \geq 1$
- such that: $x\ y^i\ z \in L \qquad i = 0, 1, 2, \ldots$

Non-regular languages

$$L = \{vv^R : v \in \Sigma^*\}$$



Theorem: The language

$$L = \{vv^R : v \in \Sigma^*\} \qquad \Sigma = \{a, b\}$$

is not regular

Proof: Use the Pumping Lemma

$$L = \{vv^R : v \in \Sigma^*\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{vv^R : v \in \Sigma^*\}$$

Let m be the critical length for L

Pick a string w such that: $w \in L$

and length $|w| \geq m$

We pick $w = a^m b^m b^m a^m$

From the Pumping Lemma:

we can write: $w = a^m b^m b^m a^m = x\ y\ z$

with lengths: $|x\ y| \le m, \quad |y| \ge 1$

$$w = xyz = \overbrace{a\ldots a}^{x}\overbrace{a\ldots a}^{y}\overbrace{\ldots a}^{m}\overbrace{b\ldots b}^{m}\overbrace{b\ldots b}^{m}\overbrace{a\ldots a}^{m}$$

$$\underbrace{\qquad}_{x}\quad\underbrace{\qquad}_{y}\qquad\underbrace{\qquad\qquad\qquad\qquad}_{z}$$

Thus: $y = a^k, \quad 1 \le k \le m$

$$x\,y\,z = a^m b^m b^m a^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^i\,z \in L$

$$i = 0, 1, 2, \ldots$$

Thus: $x\,y^2\,z \in L$

$$x\,y\,z = a^m b^m b^m a^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^2\,z \in L$

$$xy^2z = \overbrace{a\ldots a}^{x}\overbrace{a\ldots a}^{y}\overbrace{a\ldots a}^{y}\overbrace{\ldots a}^{m+k-3}\overbrace{b\ldots b}^{m}\overbrace{b\ldots b}^{m}\overbrace{a\ldots a}^{m} \in L$$

Thus: $a^{m+k} b^m b^m a^m \in L$

$$a^{m+k}b^m b^m a^m \in L \qquad k \geq 1$$

BUT: $L = \{vv^R : v \in \Sigma^*\}$

$$a^{m+k}b^m b^m a^m \notin L$$

CONTRADICTION!!!

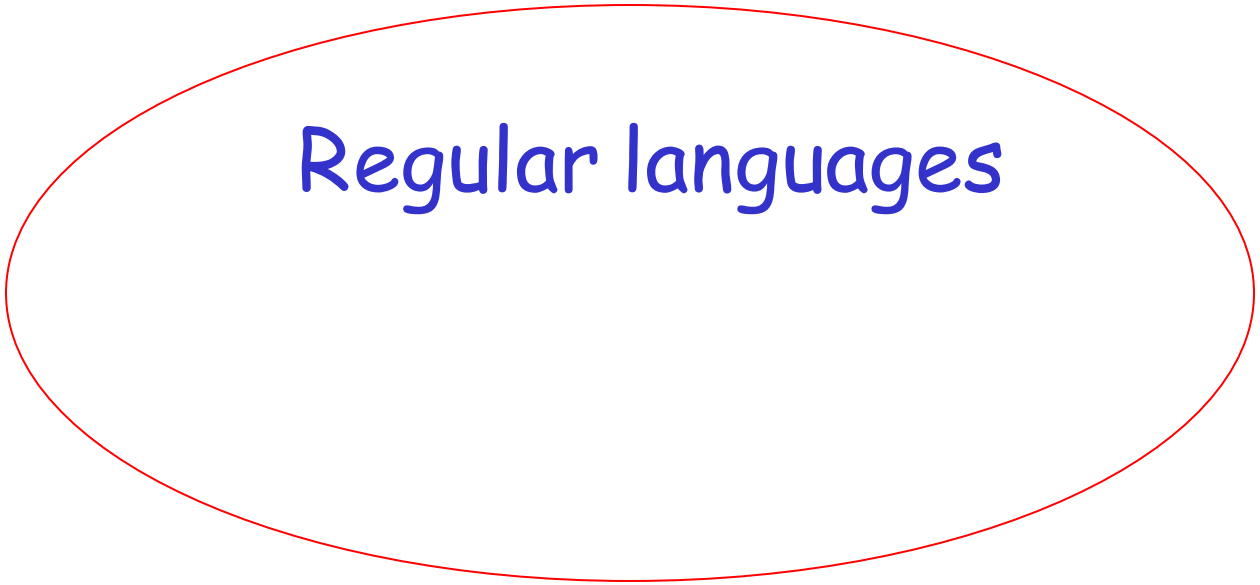
Therefore: Our assumption that L is a regular language is not true

Conclusion: L is not a regular language

END OF PROOF

Non-regular languages

$$L = \{a^n b^l c^{n+l} : n, l \geq 0\}$$



Theorem: The language

$$L = \{a^n b^l c^{n+l} : \; n, l \geq 0\}$$

is not regular

Proof: Use the Pumping Lemma

$$L = \{a^n b^l c^{n+l} : n, l \geq 0\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{a^n b^l c^{n+l} : n, l \geq 0\}$$

Let m be the critical length of L

Pick a string w such that: $w \in L$ and

length $|w| \geq m$

We pick $w = a^m b^m c^{2m}$

From the Pumping Lemma:

We can write $w = a^m b^m c^{2m} = x\ y\ z$

With lengths $|x\ y| \leq m, \quad |y| \geq 1$

$$w = xyz = \overbrace{a\ldots a}^{x}\overbrace{a\ldots a}^{y}\overbrace{a\ldots a}^{m}\overbrace{b\ldots b}^{m}\overbrace{c\ldots c}^{2m}\overbrace{c\ldots c}^{z}$$

Thus: $y = a^k, \quad 1 \leq k \leq m$

$$x\,y\,z = a^m b^m c^{2m} \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^i\,z \in L$

$$i = 0, 1, 2, \ldots$$

Thus: $x\,y^0\,z = xz \in L$

$$x\,y\,z = a^m b^m c^{2m} \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $xz \in L$

$$xz = \overbrace{a\ldots a}^{m-k}\overbrace{a\ldots a}^{m}\overbrace{b\ldots b}^{2m}c\ldots cc\ldots c \in L$$

x z

Thus: $a^{m-k}b^m c^{2m} \in L$

$$a^{m-k}b^m c^{2m} \in L \qquad k \geq 1$$

BUT: $L = \{a^n b^l c^{n+l} : n, l \geq 0\}$

$$a^{m-k}b^m c^{2m} \notin L$$

CONTRADICTION!!!

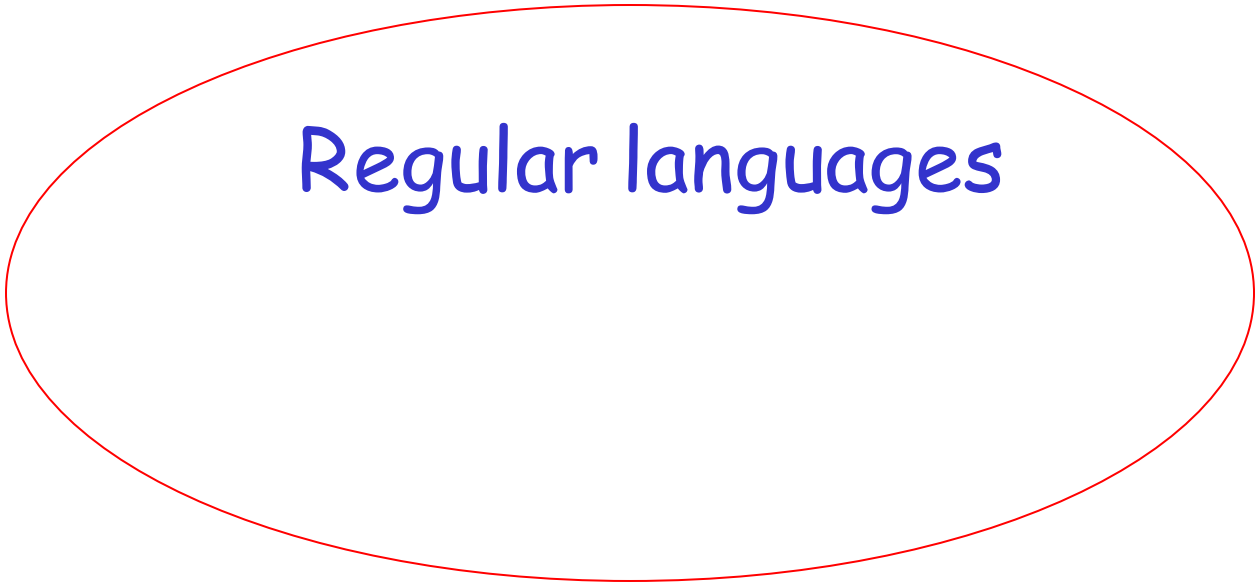
Therefore: Our assumption that L is a regular language is not true

Conclusion: L is not a regular language

END OF PROOF

Non-regular languages

$$L = \{a^{n!} : \ n \geq 0\}$$



Theorem: The language $L = \{a^{n!} : n \geq 0\}$

is not regular

$$n! = 1 \cdot 2 \cdots (n-1) \cdot n$$

Proof: Use the Pumping Lemma

$$L = \{a^{n!} : \ n \geq 0\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{a^{n!} : \ n \geq 0\}$$

Let m be the critical length of L

Pick a string w such that: $w \in L$

length $|w| \geq m$

We pick $w = a^{m!}$

From the Pumping Lemma:

We can write $w = a^{m!} = x\ y\ z$

With lengths $|x\ y| \le m, \quad |y| \ge 1$

$$w = xyz = a^{m!} = \overbrace{a\ldots a}^{x}\overbrace{a\ldots a}^{y}\overbrace{a\ldots a}^{m}\overbrace{a\ldots a\ldots a}^{m!-m}$$

Thus: $y = a^k, \quad 1 \le k \le m$

$$x\,y\,z = a^{m!} \qquad y = a^{k},\ \ 1 \le k \le m$$

From the Pumping Lemma: $x\,y^{i}\,z \in L$

$$i = 0, 1, 2, \ldots$$

Thus: $x\,y^{2}\,z \in L$

$$x\,y\,z = a^{m!} \qquad y = a^{k},\ \ 1 \le k \le m$$

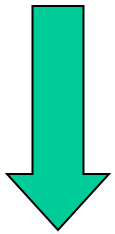
From the Pumping Lemma: $x\,y^2\,z \in L$

$$xy^2z = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots a}_{y}\underbrace{a\ldots a}_{y}\underbrace{a\ldots a}_{z}}^{m+k}\overbrace{a\ldots a}^{m!-m} \in L$$

Thus: $a^{m!+k} \in L$

$$a^{m!+k} \in L \qquad 1 \le k \le m$$

Since: $L = \{a^{n!} : \ n \ge 0\}$



There must exist p such that:

$$m!+k = p!$$

$$a^{m!+k} \in L \qquad 1 \le k \le m$$

BUT: $L = \{a^{n!} : n \ge 0\}$

$$a^{m!+k} \notin L$$

CONTRADICTION!!!

Therefore: Our assumption that L is a regular language is not true

Conclusion: L is not a regular language

END OF PROOF

Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.

Review

- Pumping Lemma

Closure Properties of Regular Languages

Regular Languages

- If $\sum$ is an alphabet, the set R of regular languages over $\sum$ is defined as follows.
  - The language $\emptyset$ is an element of R, and for every $a \in$ L , the language $\{a\}$ is in R.
  - For any two languages L_1 and L_2 in R, the three languages $L_1 \cup L_2$, L_1L_2, and L_1* are elements of R.

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 regular Languages

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

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: L_1^*

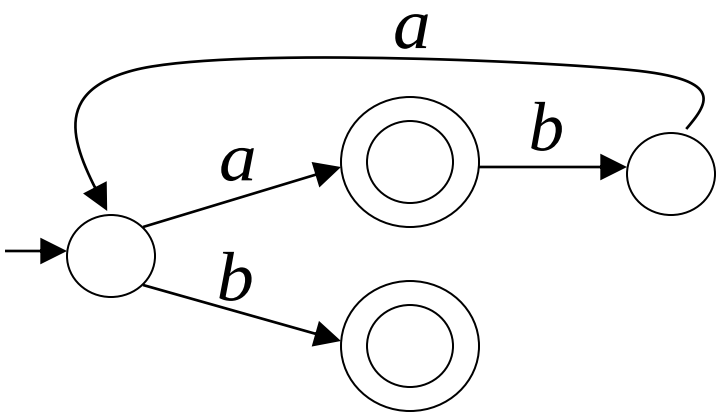
Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

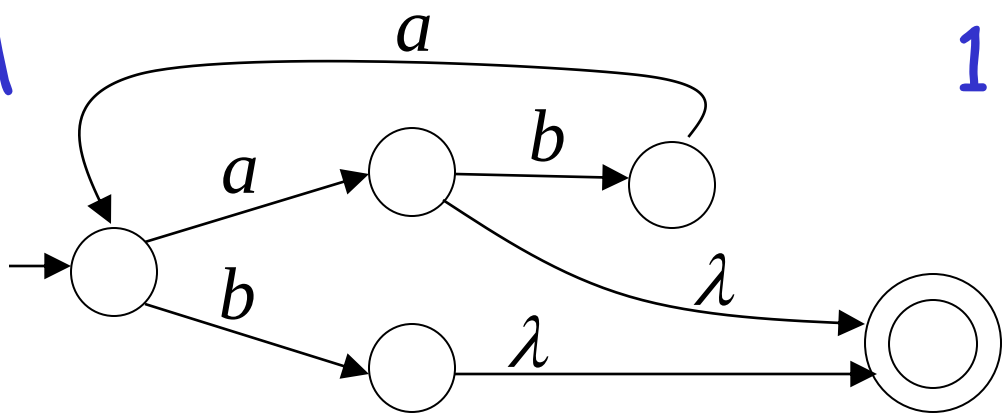
A useful transformation: use one accept state

NFA



2 accept states

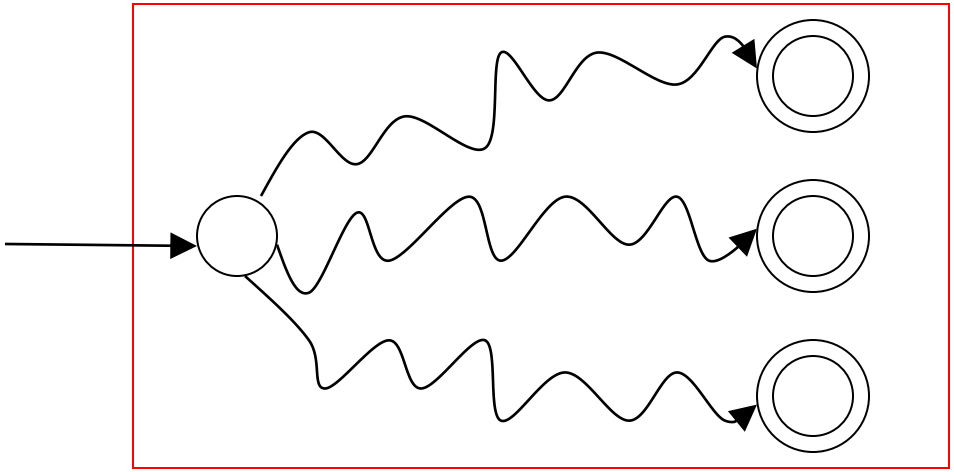
Equivalent NFA



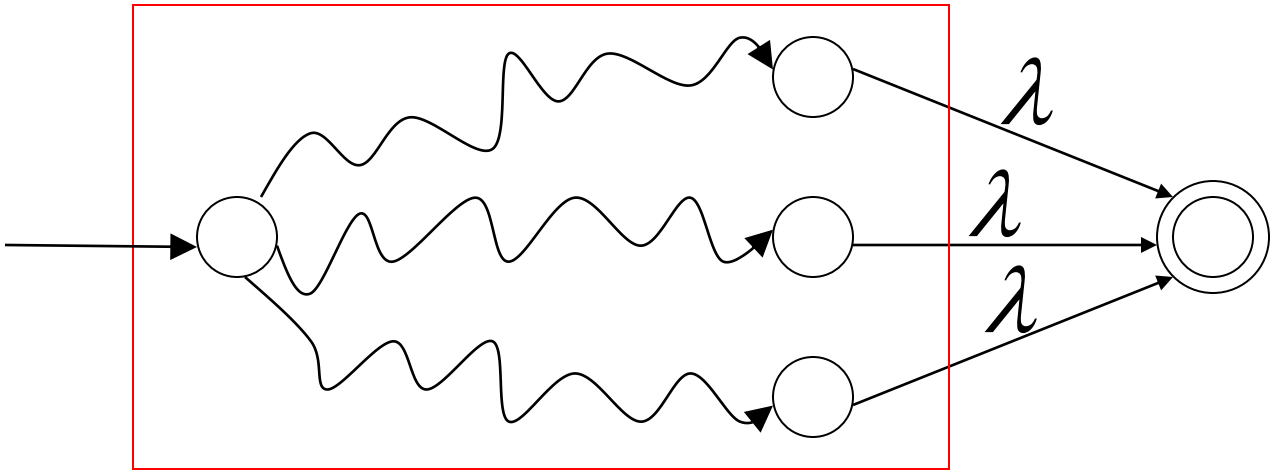
1 accept state

In General

NFA



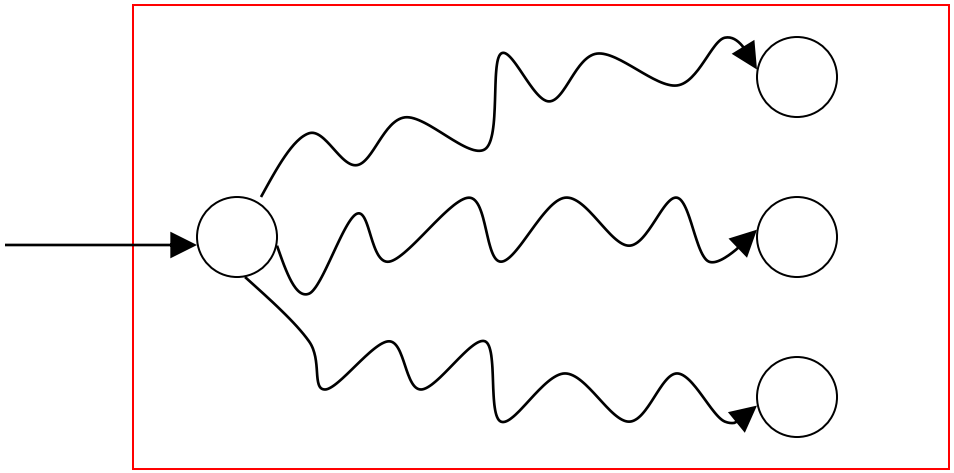
Equivalent NFA

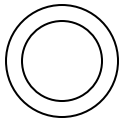


Single accepting state

Extreme case

NFA without accepting state





Add an accepting state without transitions

Take two languages

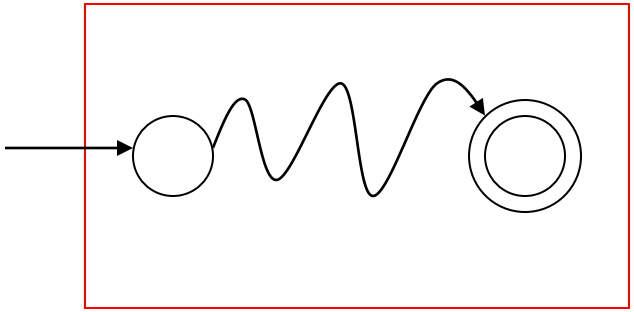
Regular language L_1

Regular language L_2

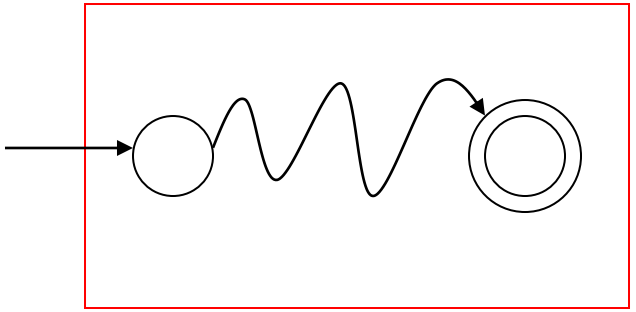
$$L(M_1) = L_1$$

$$L(M_2) = L_2$$

NFA M_1



NFA M_2



Single accepting state

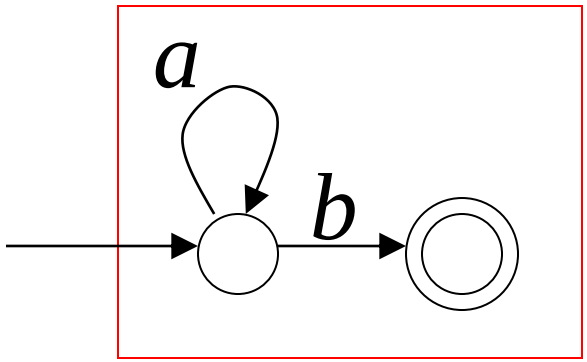
Single accepting state

Example

M_1

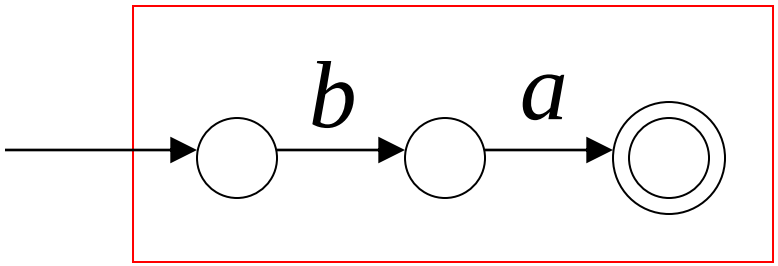
$n \geq 0$

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



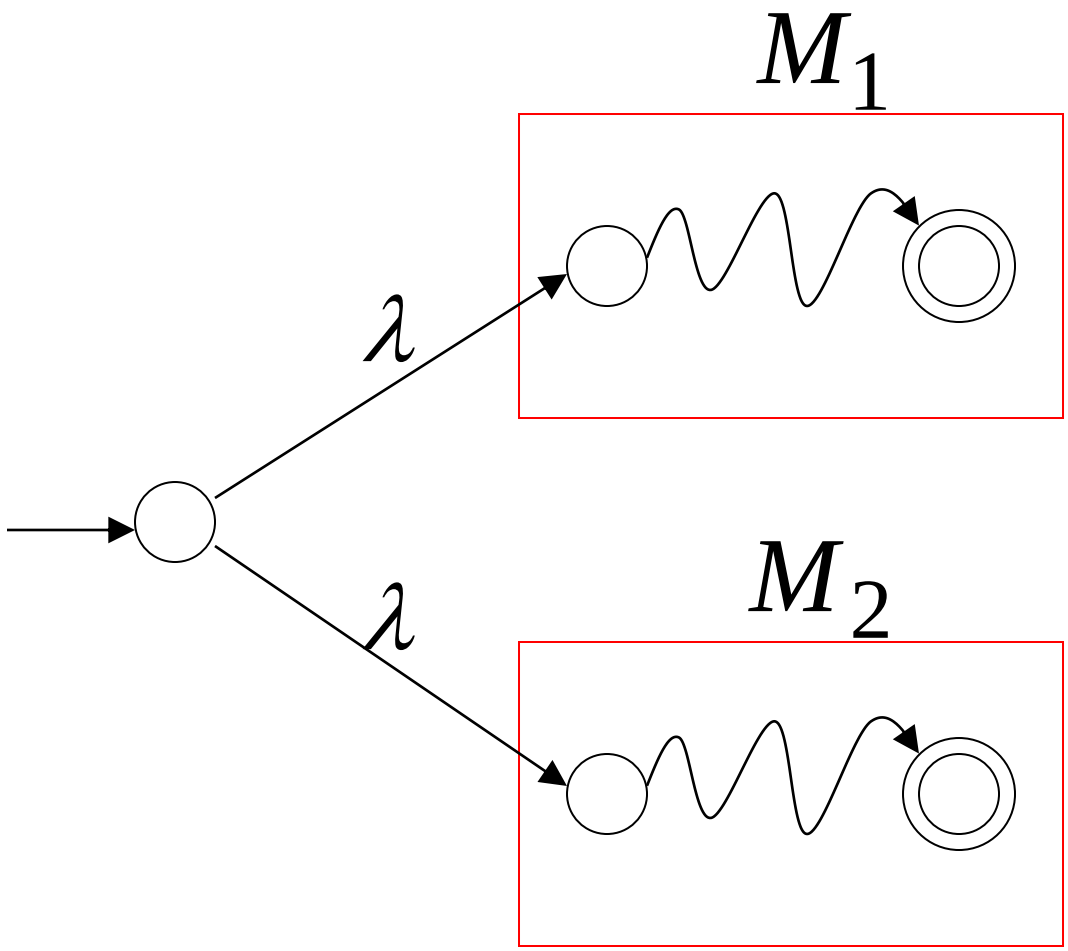
M_2

$L_2 = \{ba\}$



Union

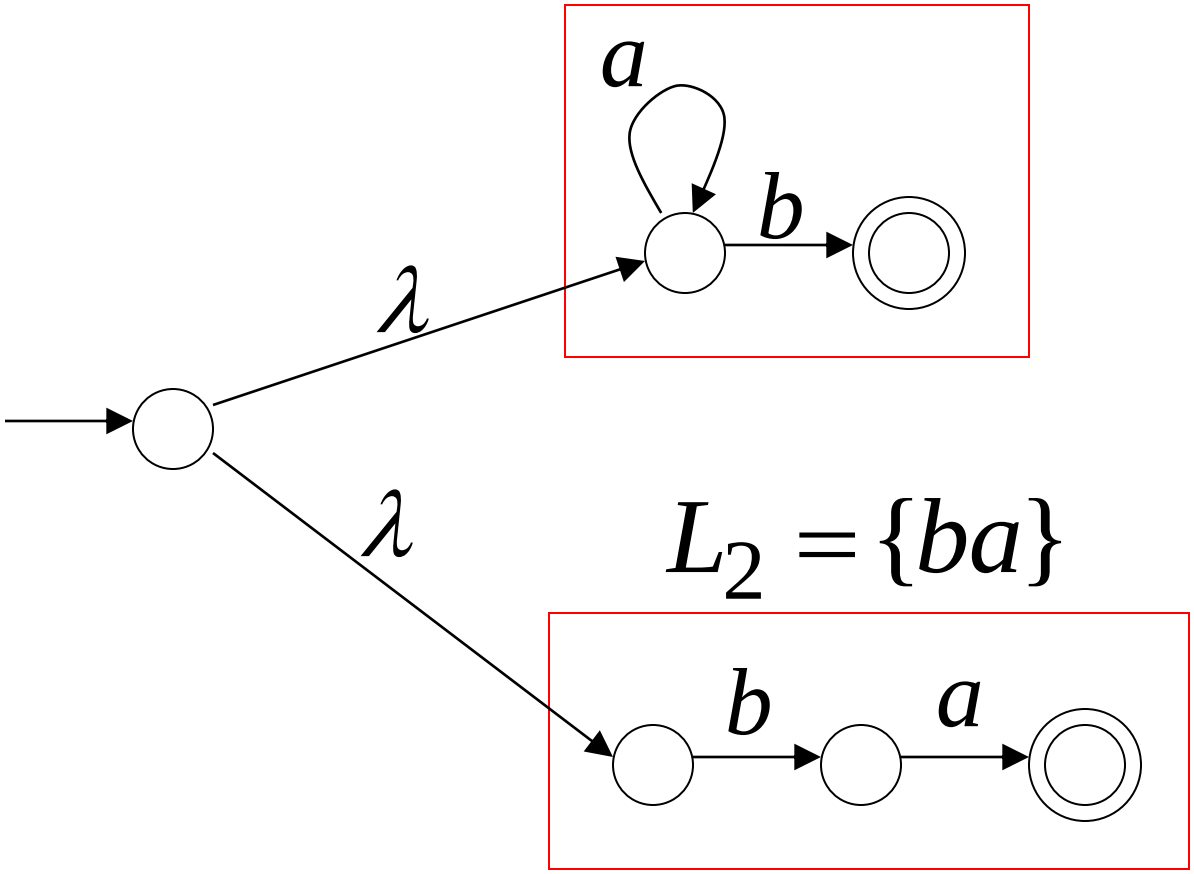
NFA for $L_1 \cup L_2$



Example

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

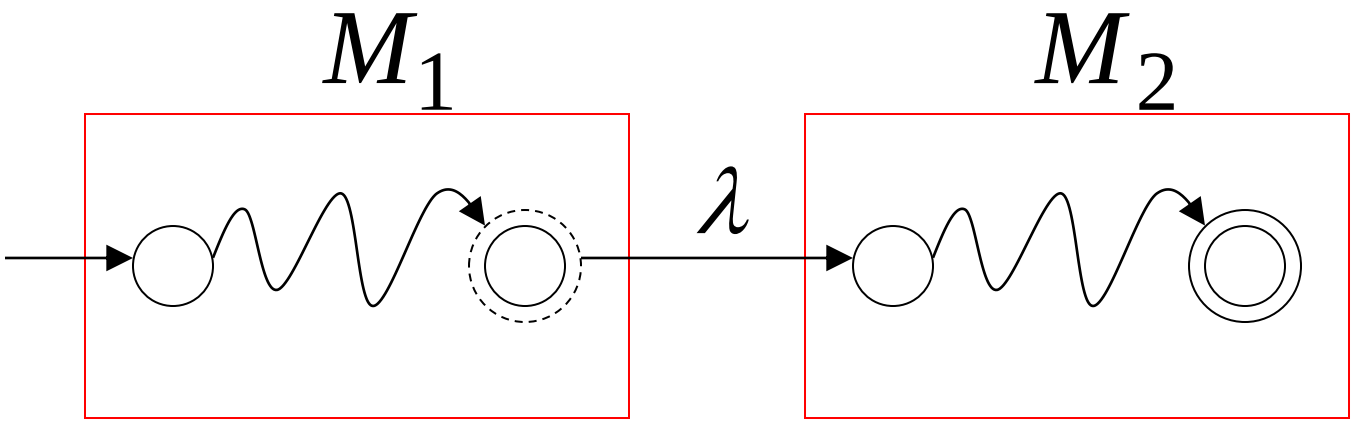
$L_1 = \{a^n b\}$



$L_2 = \{ba\}$

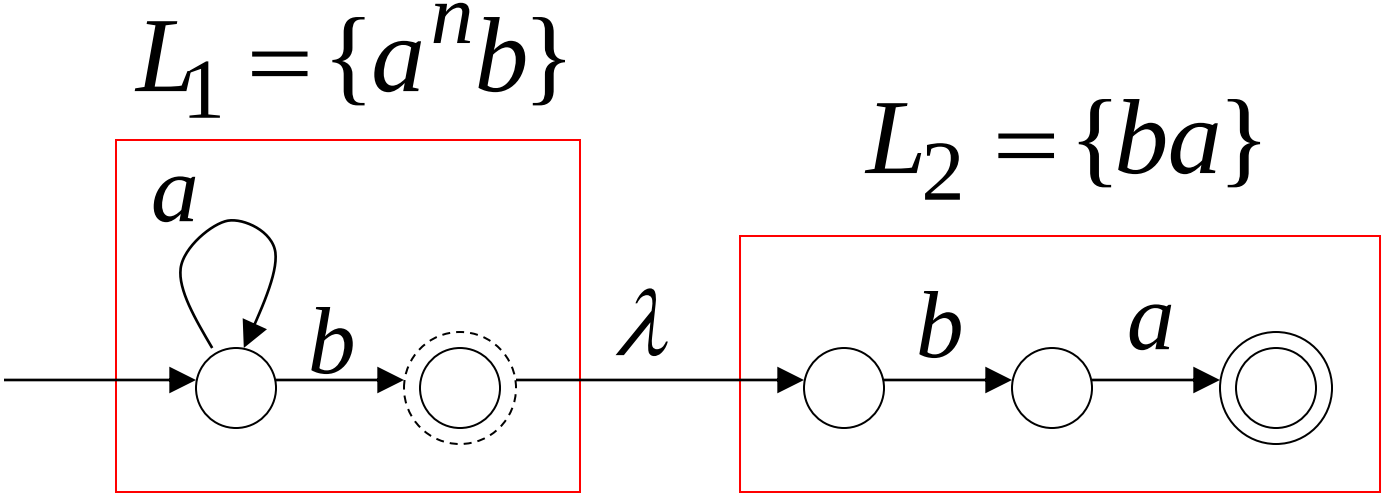
Concatenation

NFA for $L_1 L_2$



Example

NFA for $L_1L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$



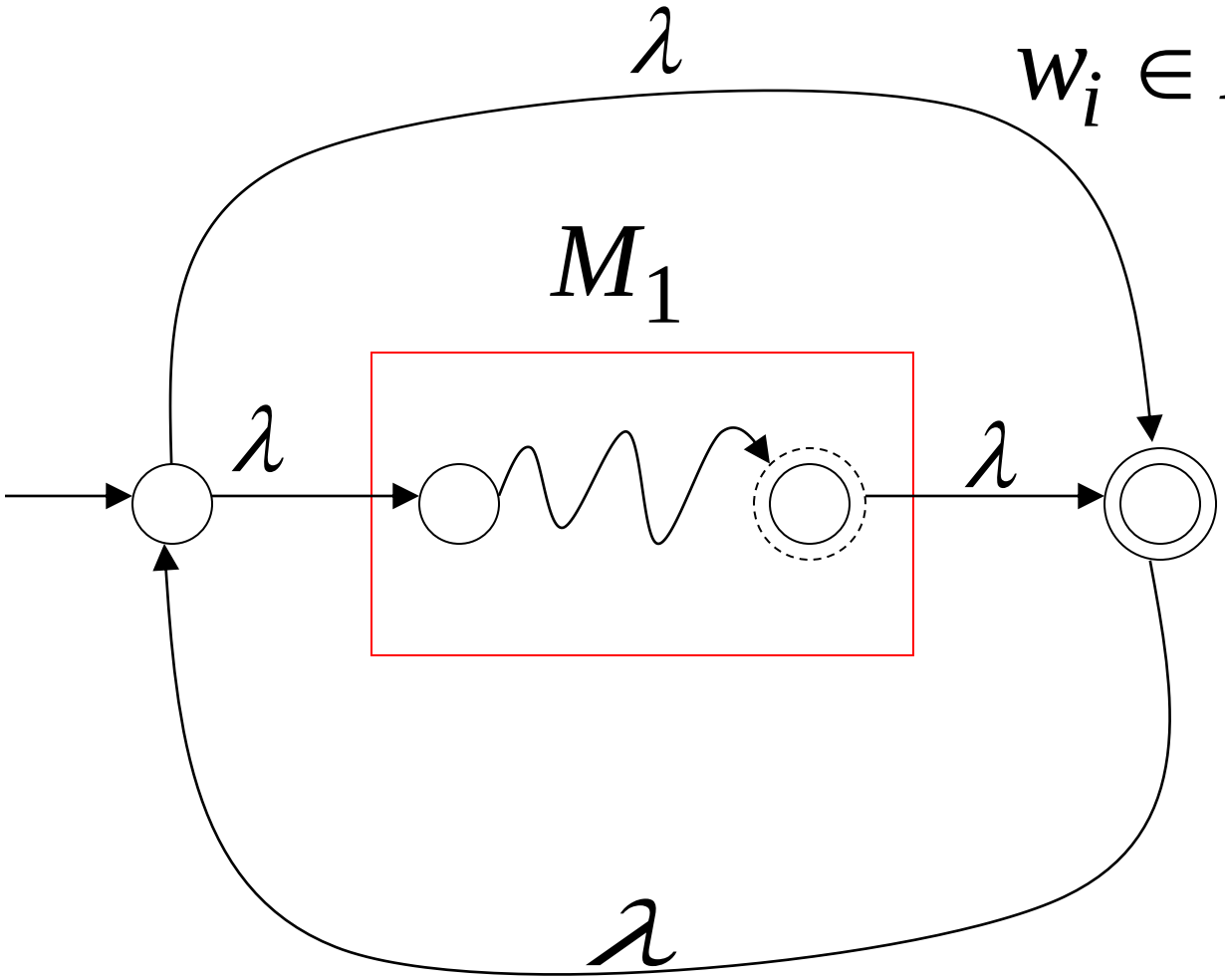
Star Operation

NFA for L_1*

$w = w_1 w_2 \cdots w_k$

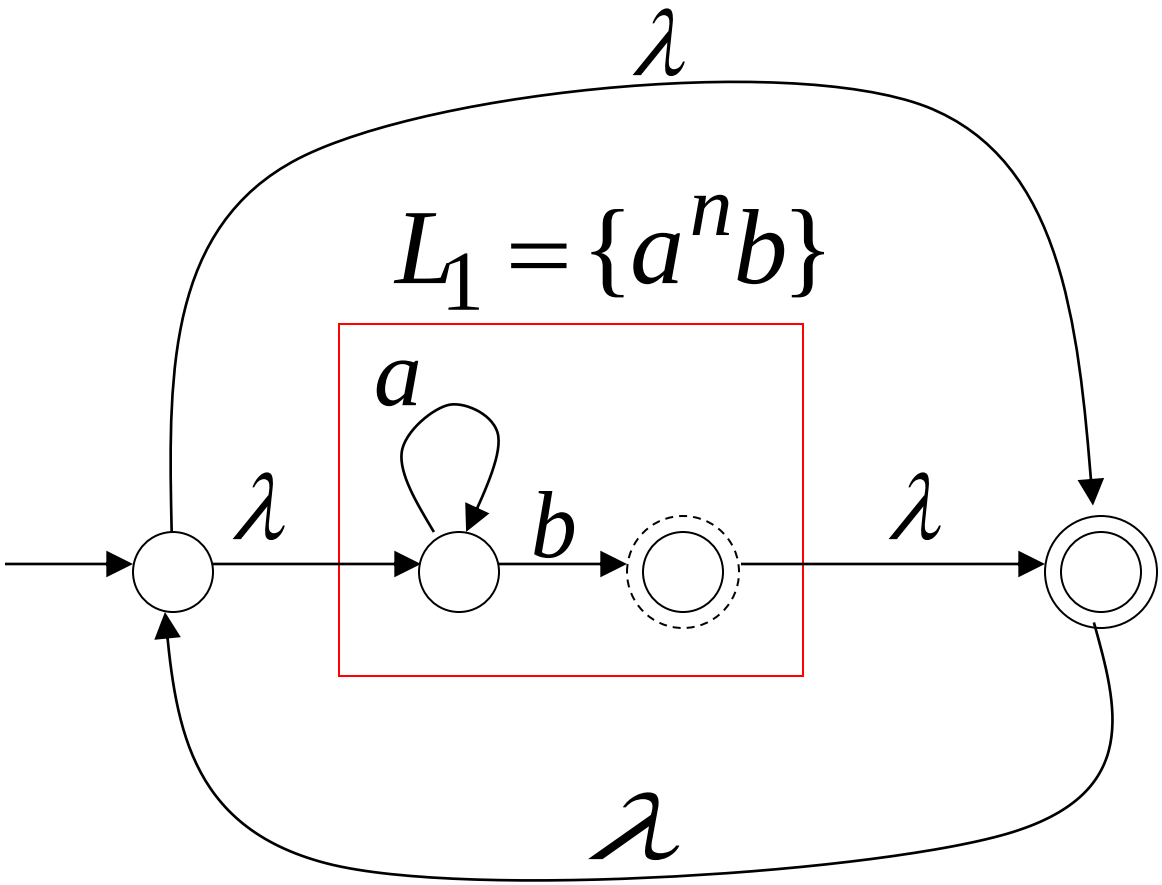
$w_i \in L_1$

$\lambda \in L_1*$



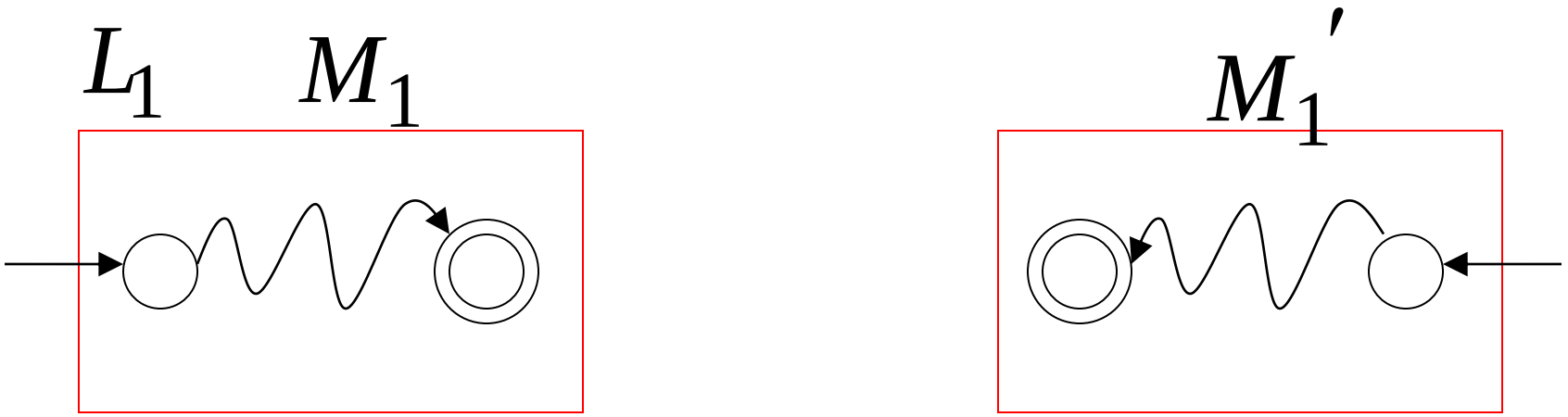
Example

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



Reverse

NFA for L_1^R



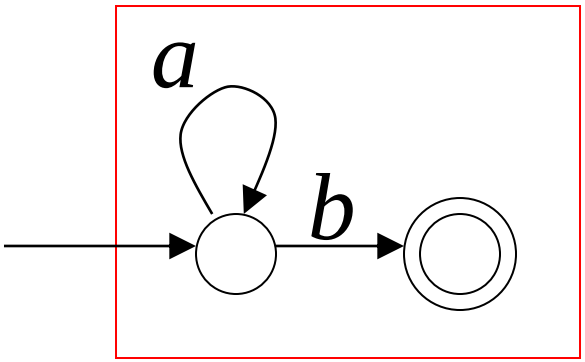
1. Reverse all transitions

2. Make initial state accepting state and vice versa

Example

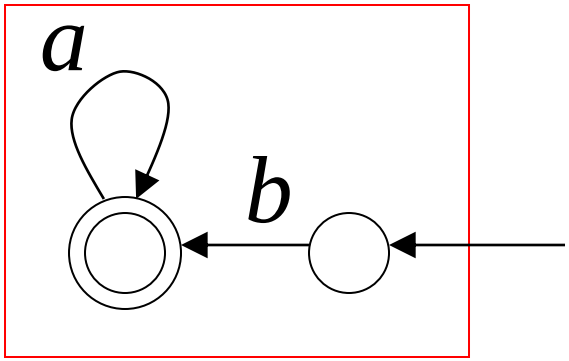
$L_1 = \{a^n b\}$

M_1

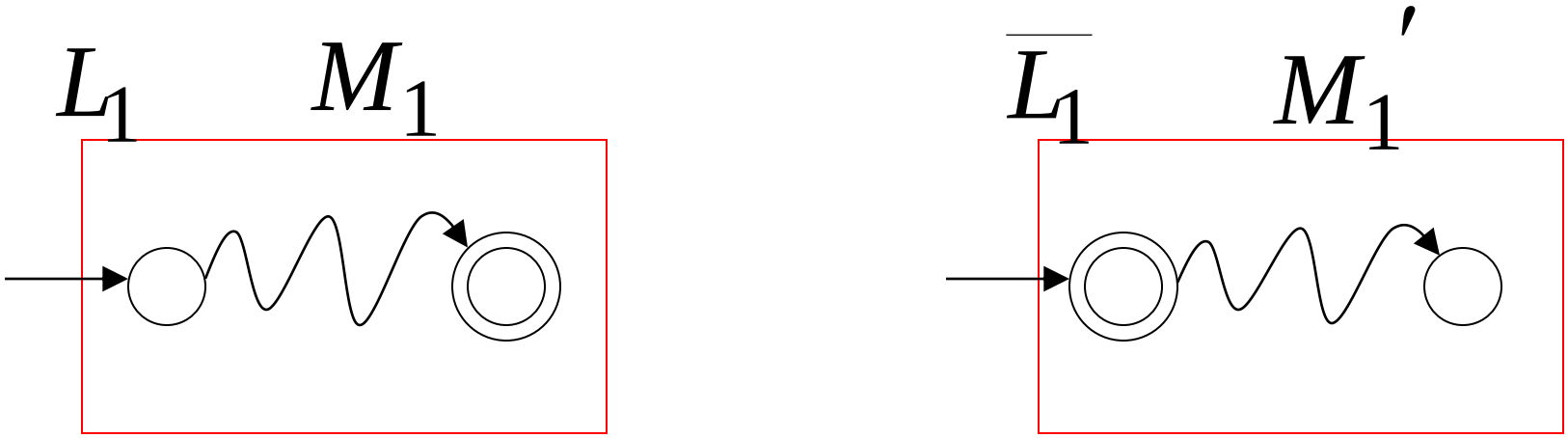


$L_1{}^R = \{b a^n\}$

$M_1{}'$



Complement



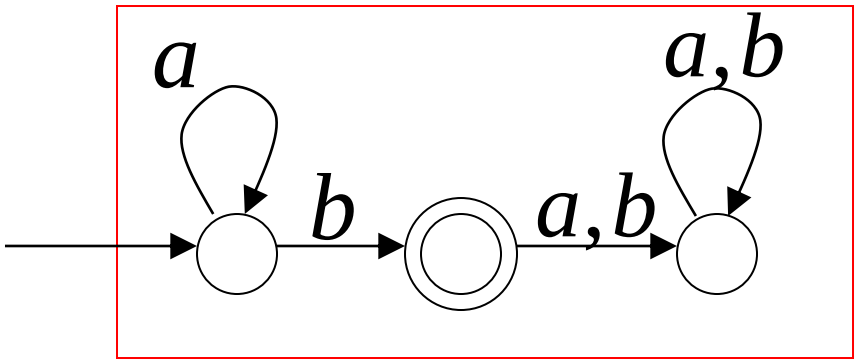
1. Take the DFA that accepts L_1

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

Example

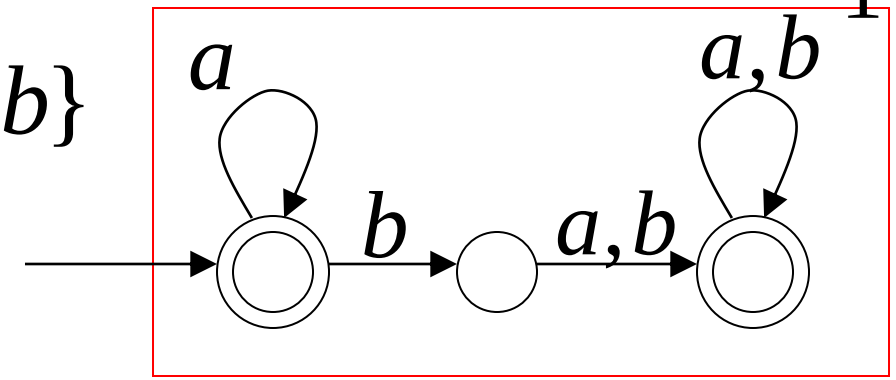
$L_1 = \{a^n b\}$

M_1

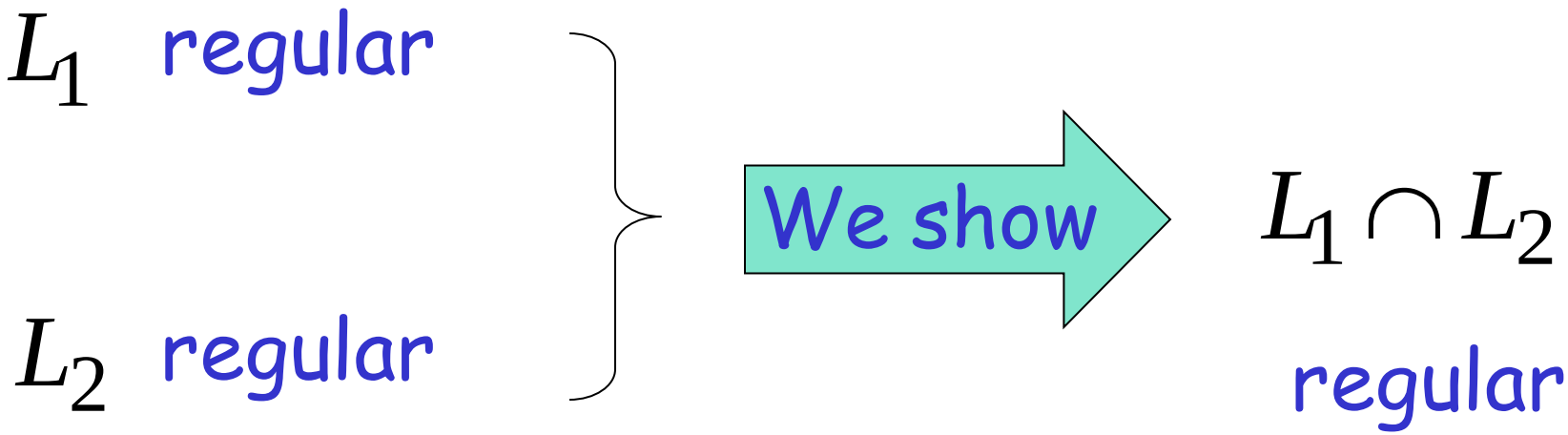


$\overline{L_1} = \{a,b\}^* - \{a^n b\}$

M_1'



Intersection



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

L_1, L_2 regular

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

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

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

$L_1 \cap L_2$ 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

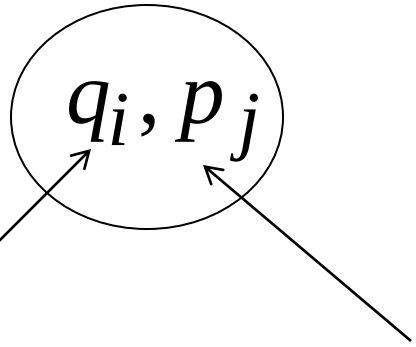
Machine M_2

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

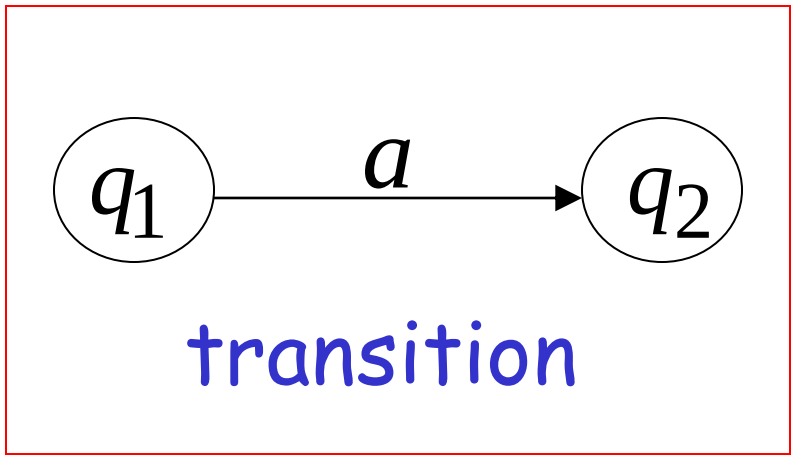
States in M



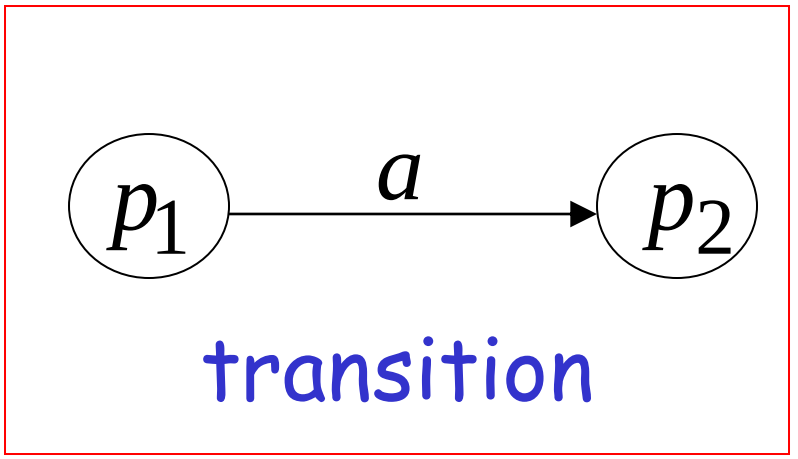
State in M_1

State in M_2

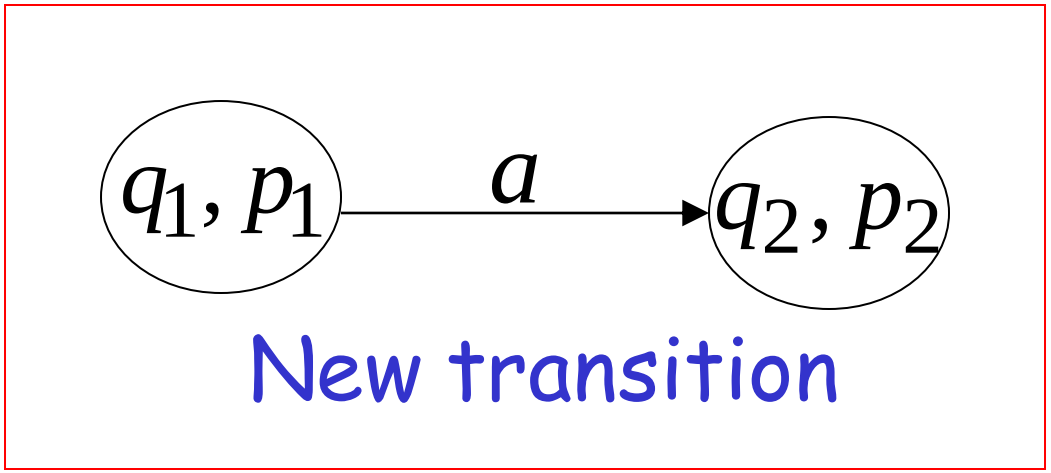
DFA M_1

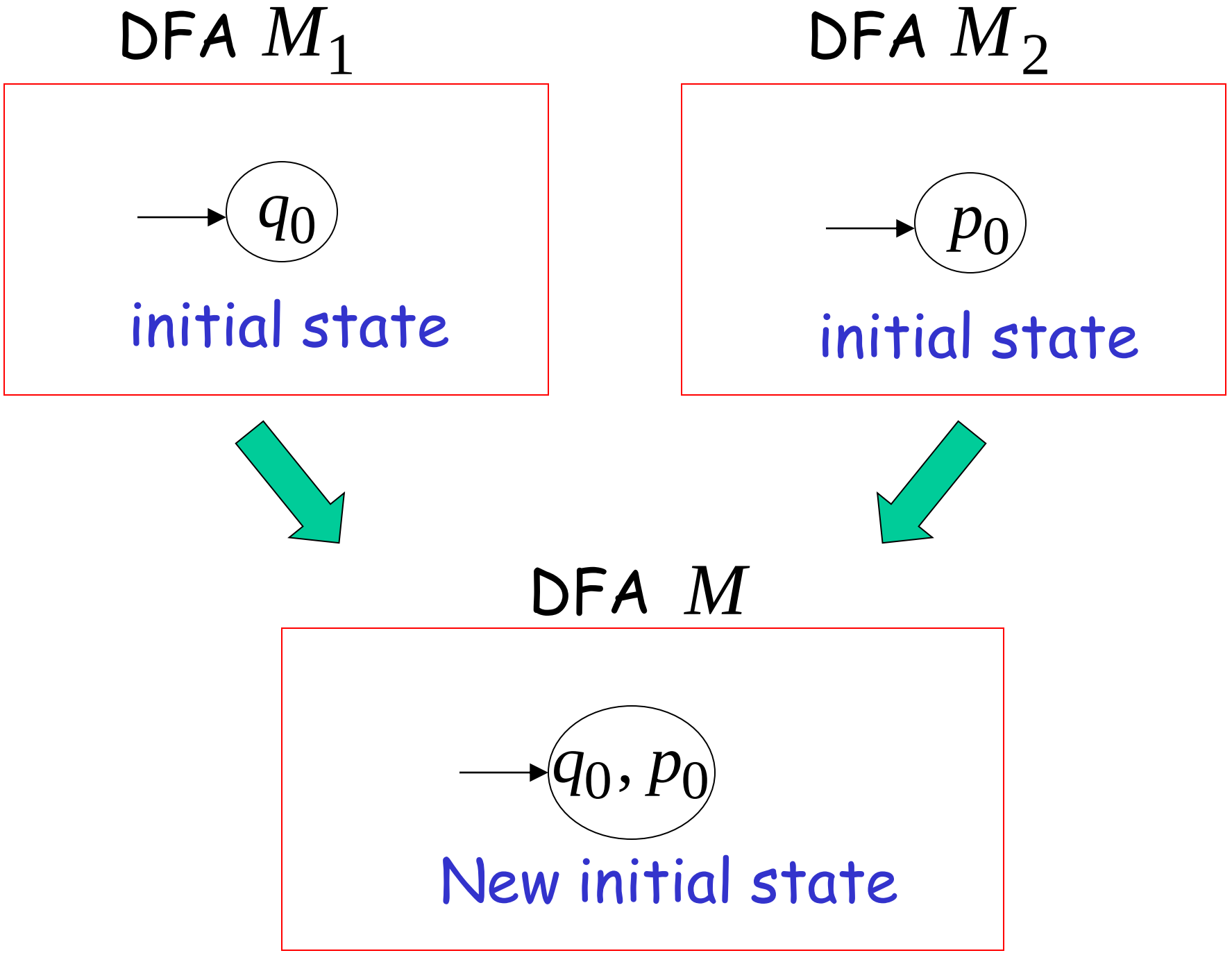


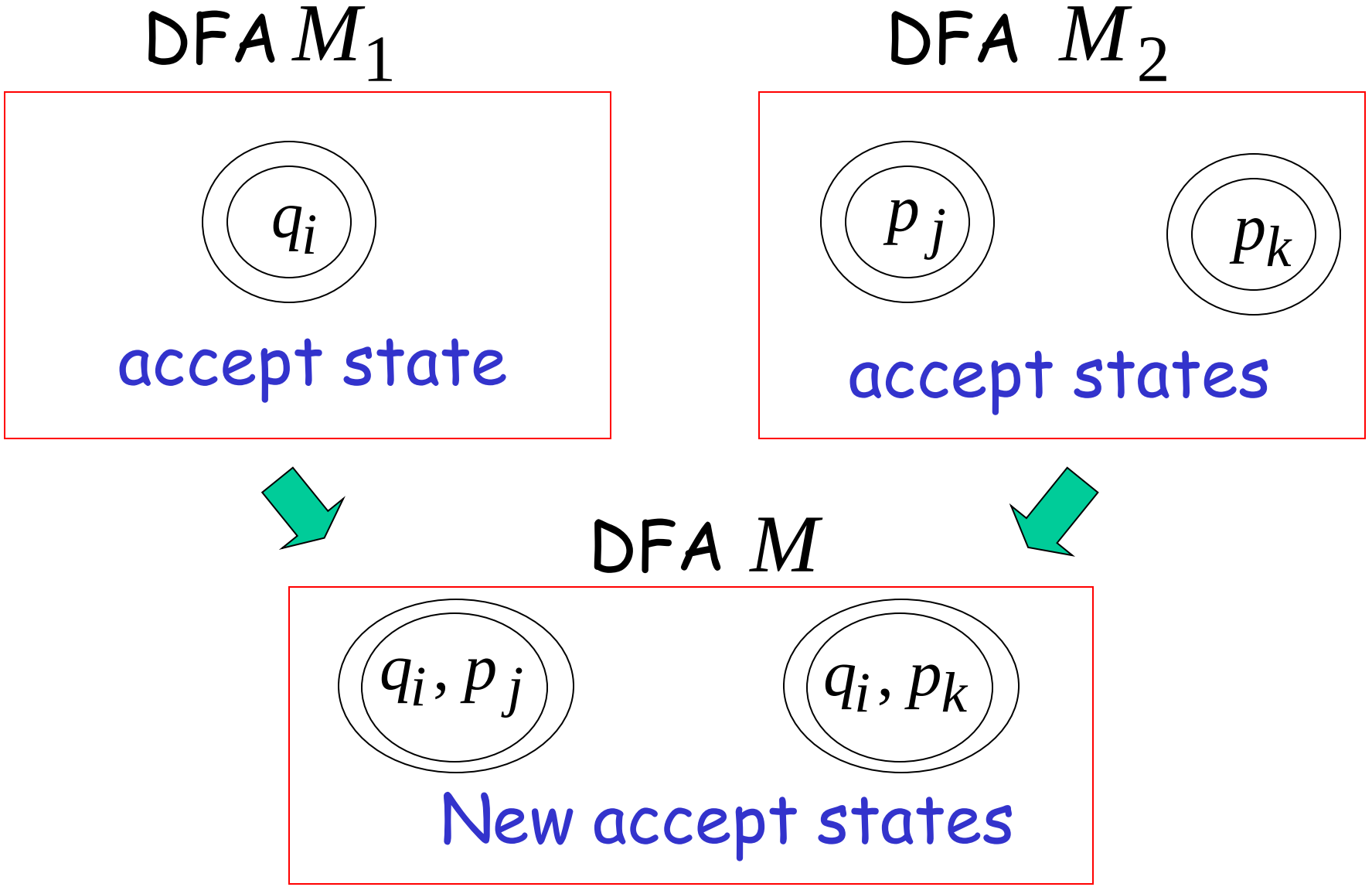
DFA M_2



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



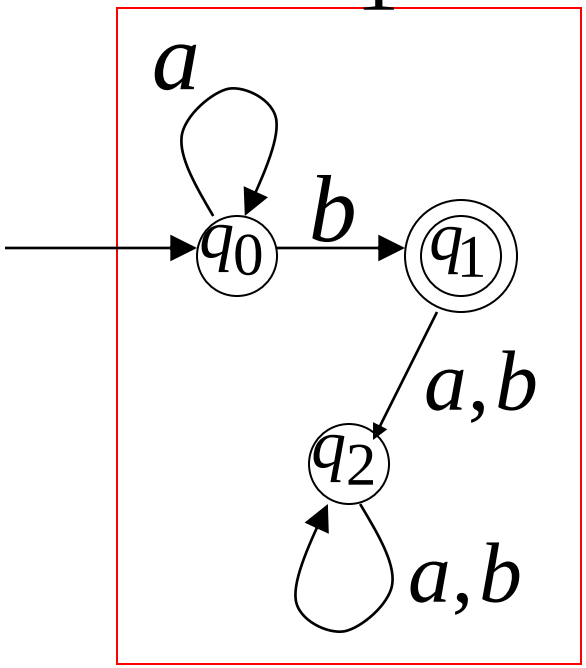
Both constituents must be accepting states

Example:

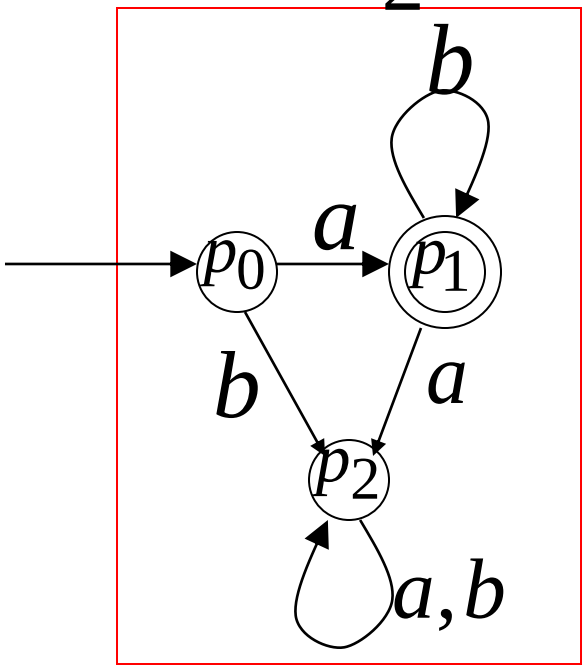
$$L_1 = \{a^n b\} \qquad n \geq 0$$

$$L_2 = \{a b^m\} \qquad m \geq 0$$

M_1

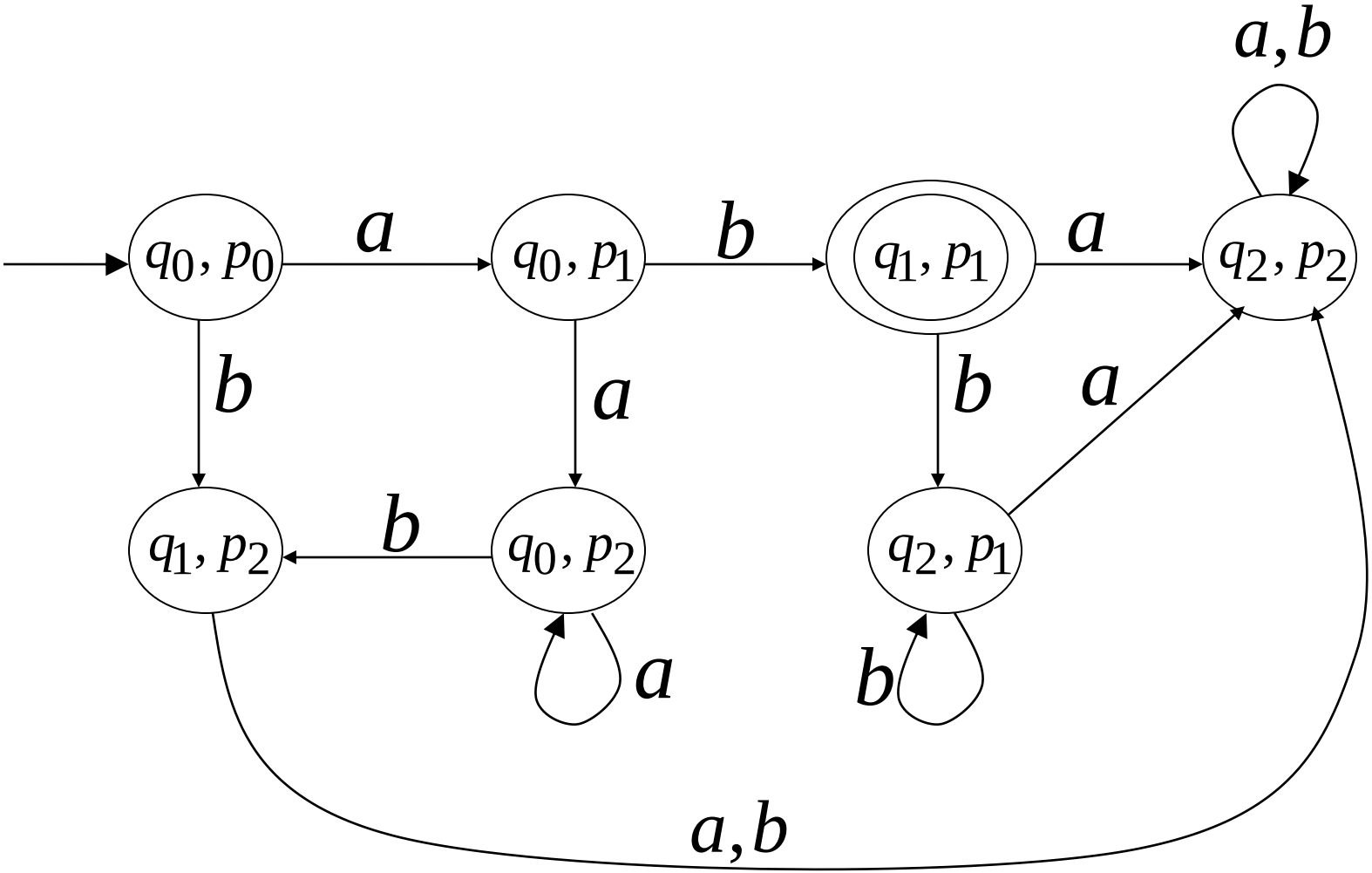


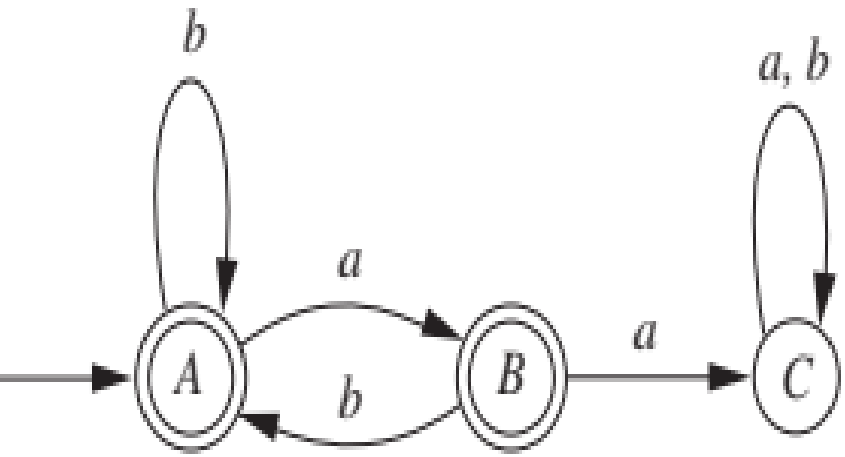
M_2

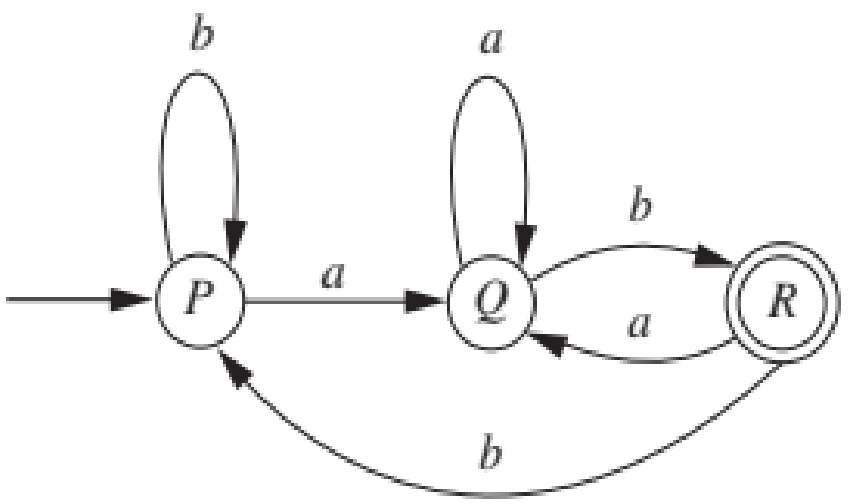


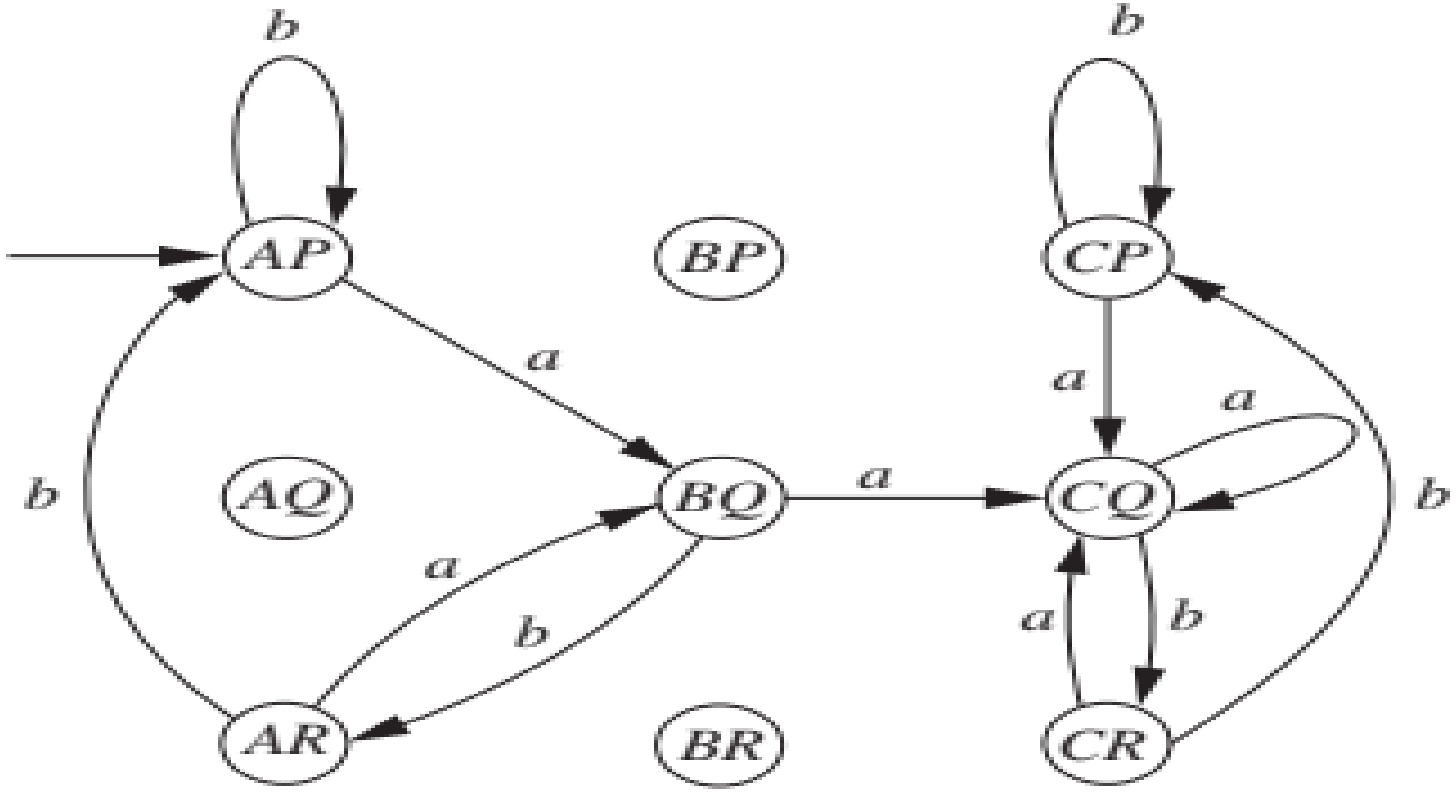
Automaton for intersection

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

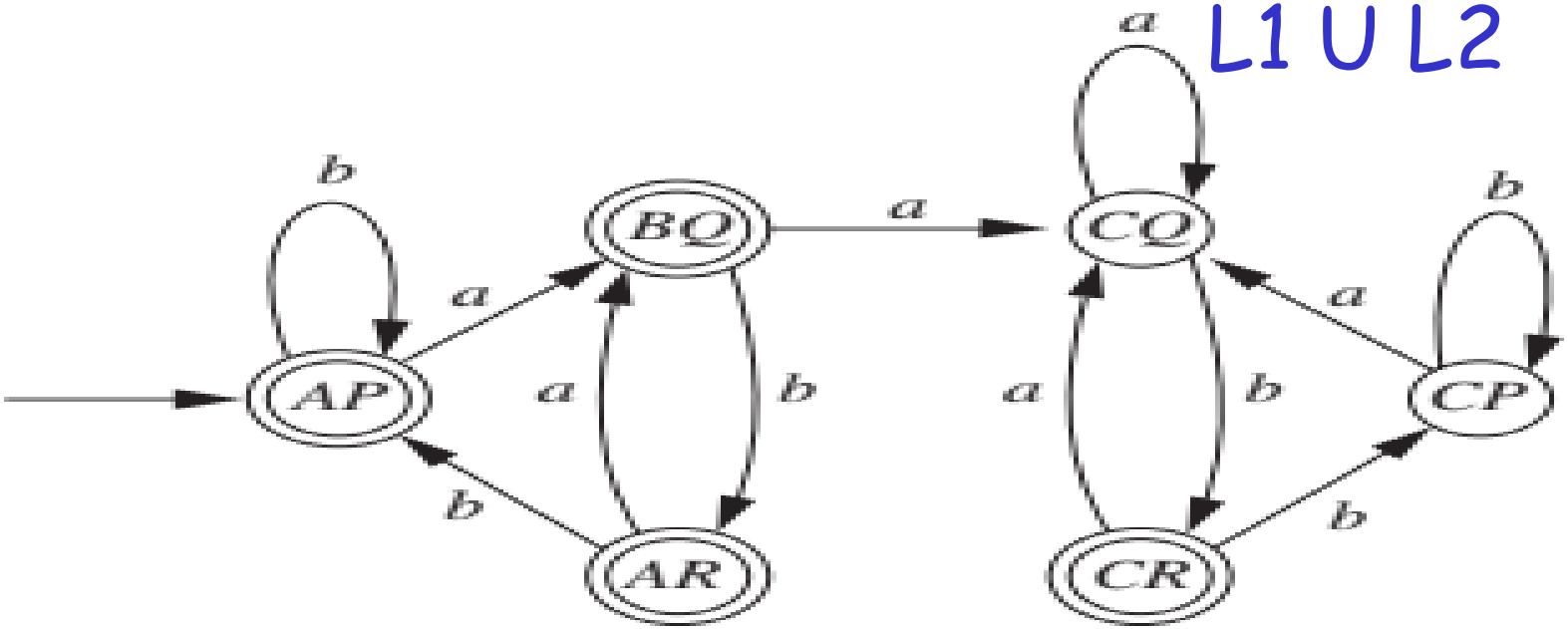



b
a, b
a
A
b
B
a
C

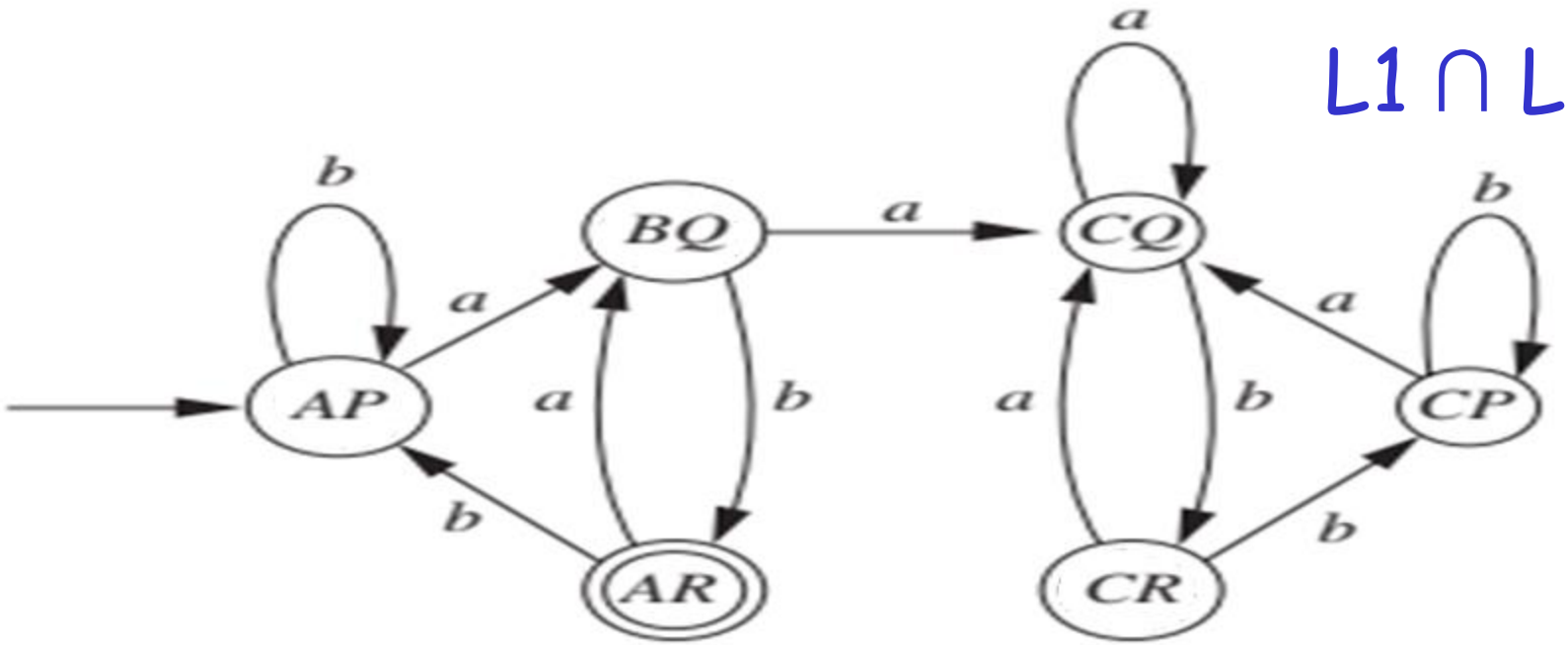

b
a
b
P
a
Q
R
a
b


b
b
AP
BP
CP
a
a
a
b
AQ
BQ
a
CQ
b
a
b
a
b
AR
BR
CR

L1 U L2



L1 ∩ L2



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

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

Pune Institute of Computer Technology, Pune

THEORY OF COMPUTATION

UNIT II:

Regular Expressions

Jagadish Kashinath Kamble **(M.Tech, IIT Kharagpur)**
Assistant Professor,
Dept. of Information Technology,
Pune **I**nstitute of **C**omputer **T**echnology, Pune.

Review

- Closure Properties

Applications of Regular Expression

- Regular Expressions in Lexical Analysis
- Regular Expressions in Web Search Engines
- Regular Expressions in Software Engineering

Summary

- Regular Expression
- Construction of RE from Language
- Construction of Language from RE
- RE and DFA
- Conversion of RE to DFA
- Conversion of DFA to RE (Arden's Theorm)
- Pumping Lemma
- Closure Properties of RL
- Applications of RE

- Thank You….