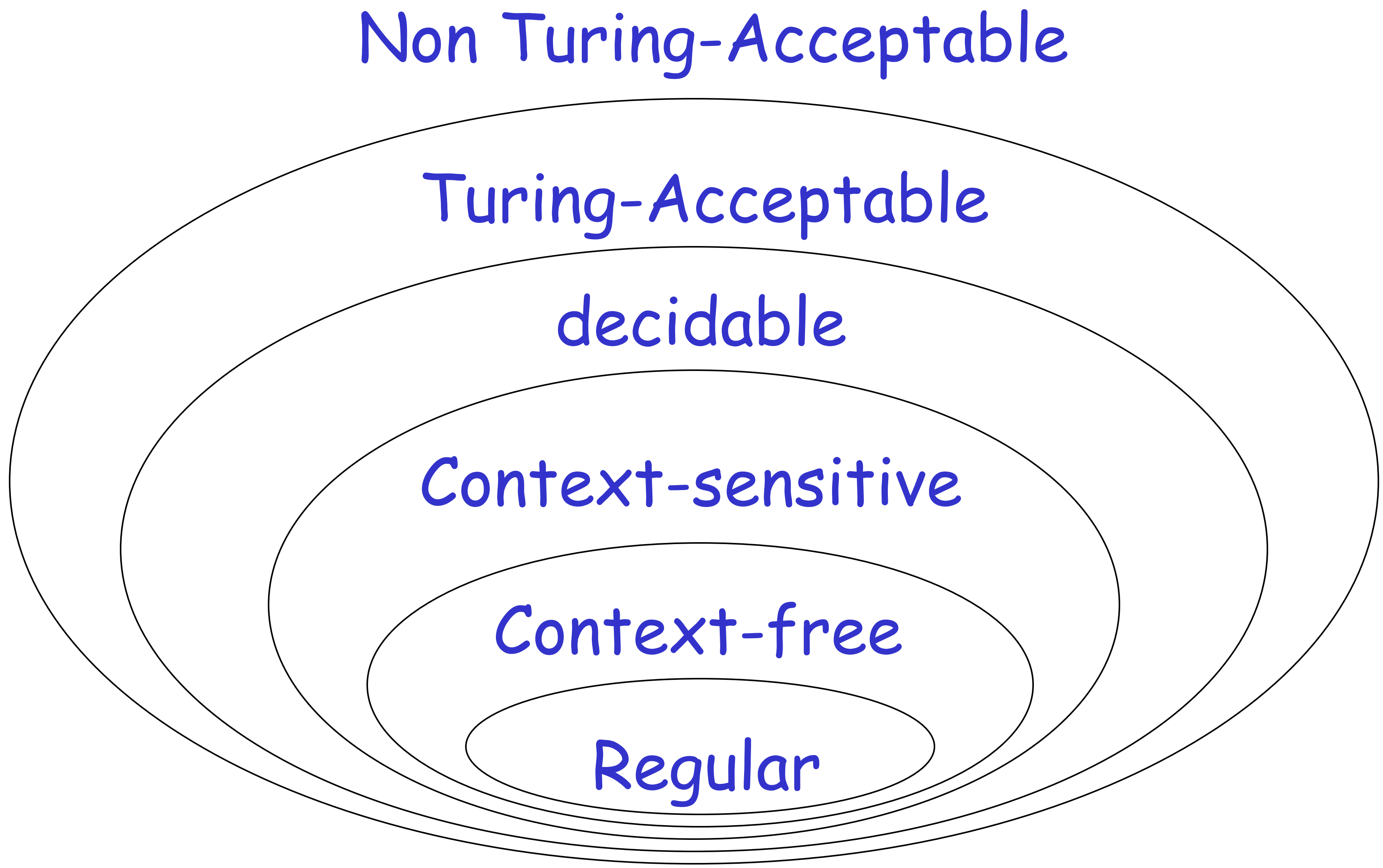
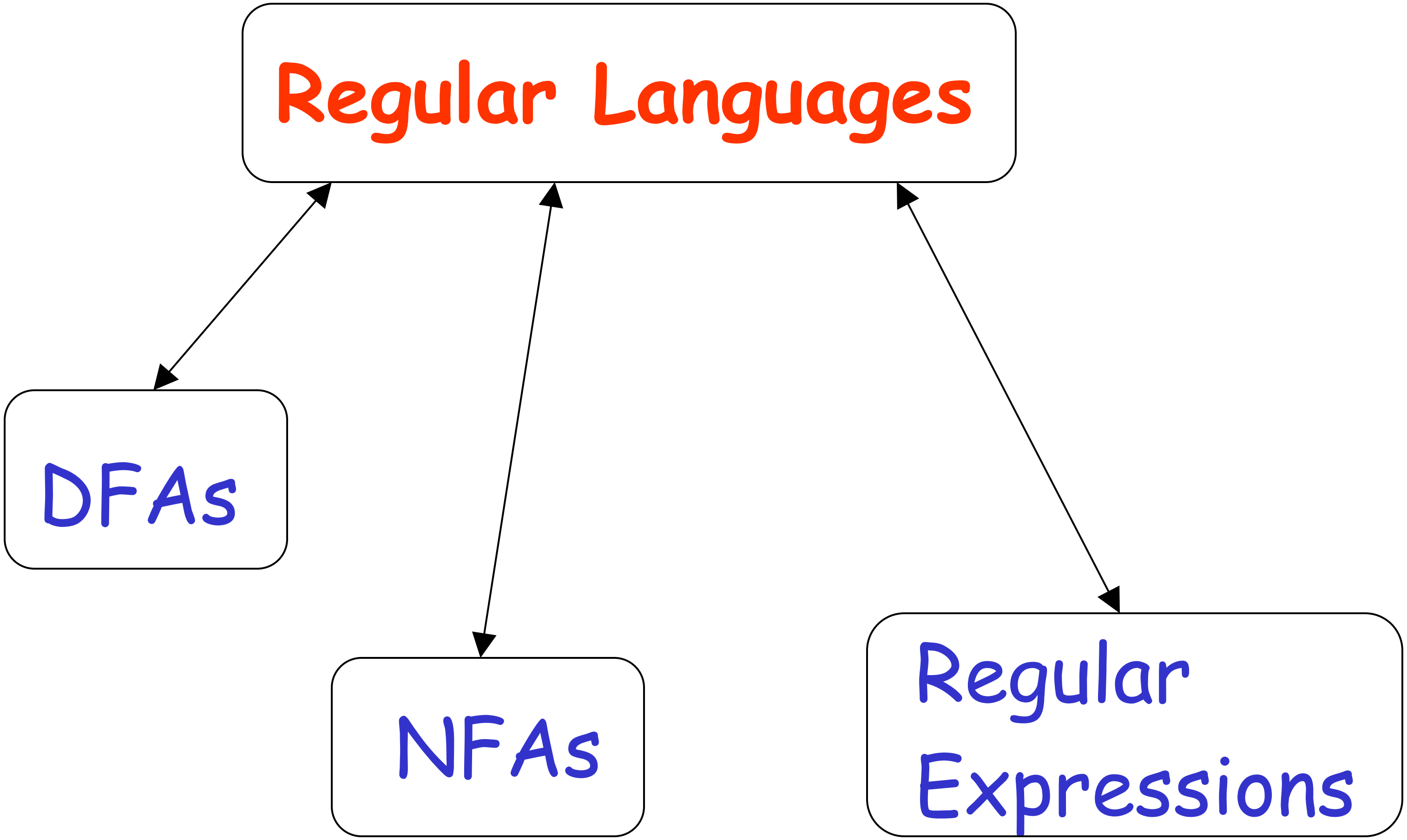
Regular Languages, Regular Expressions, and Pumping Lemma

The Chomsky Hierarchy



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)

For regular languages L_1 and L_2
we will prove that:

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

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

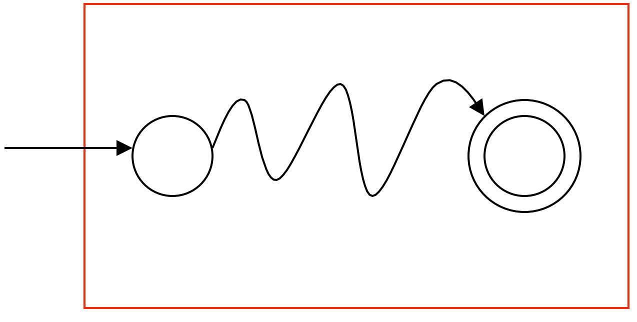
Regular language L_1

Regular language L_2

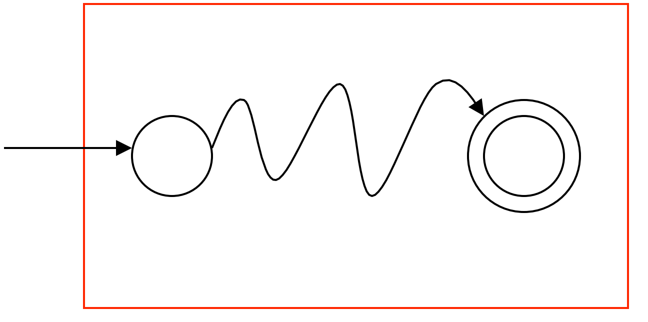
$$L(M_1) = L_1$$

$$L(M_2) = L_2$$

NFA M_1



NFA M_2



Single accepting state

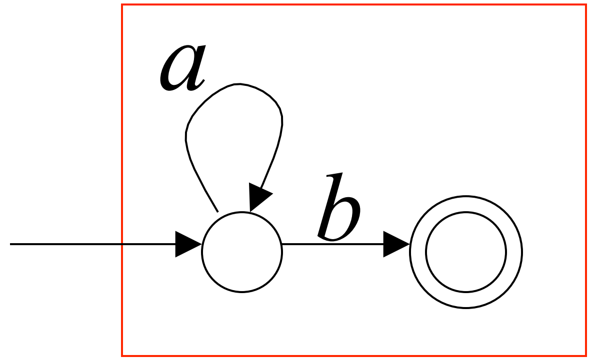
Single accepting state

Example

$n \geq 0$

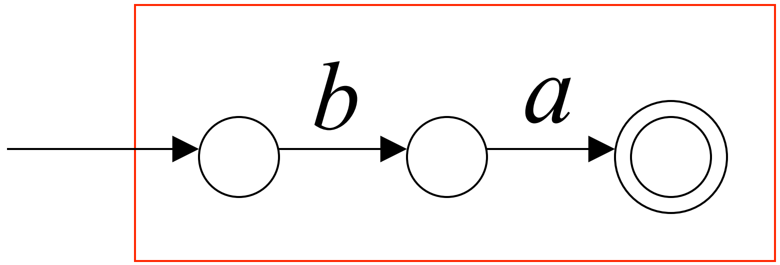
$L_1 = \{a^n b\}$

M_1



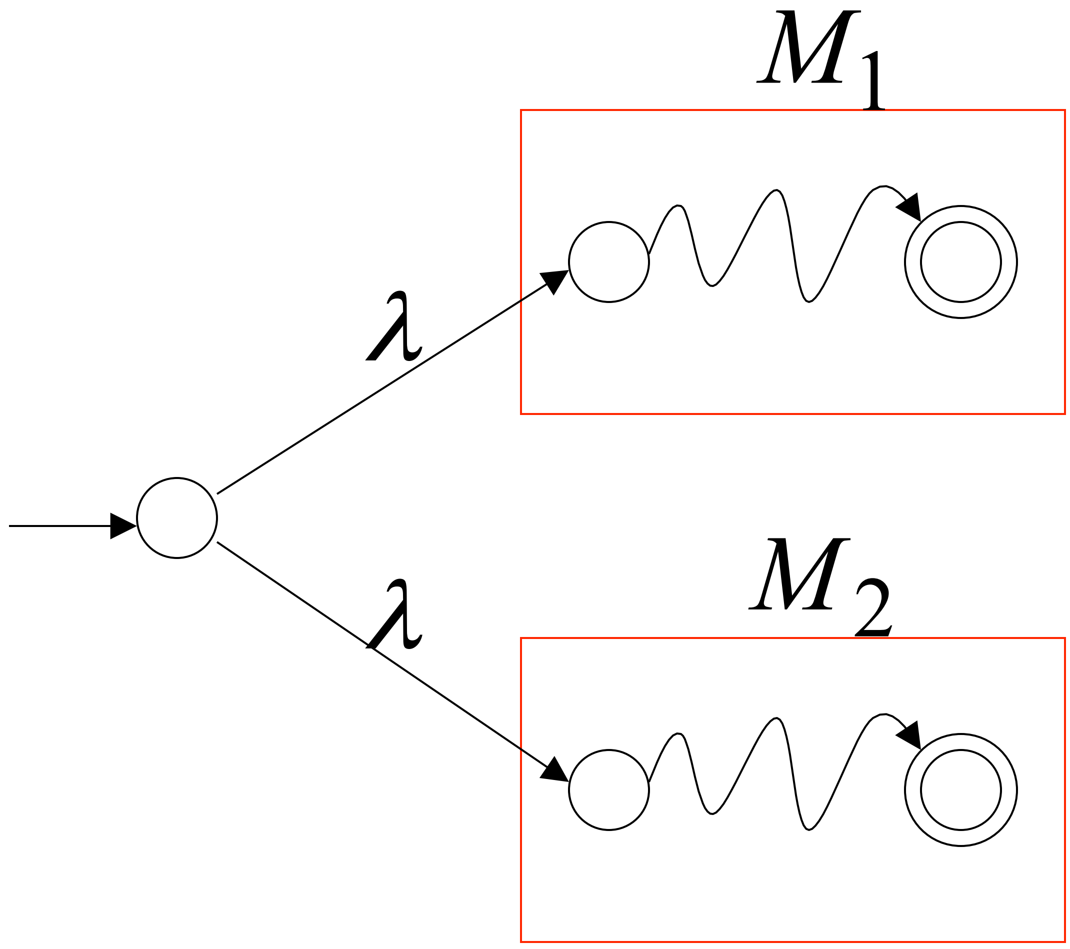
$L_2 = \{ba\}$

M_2



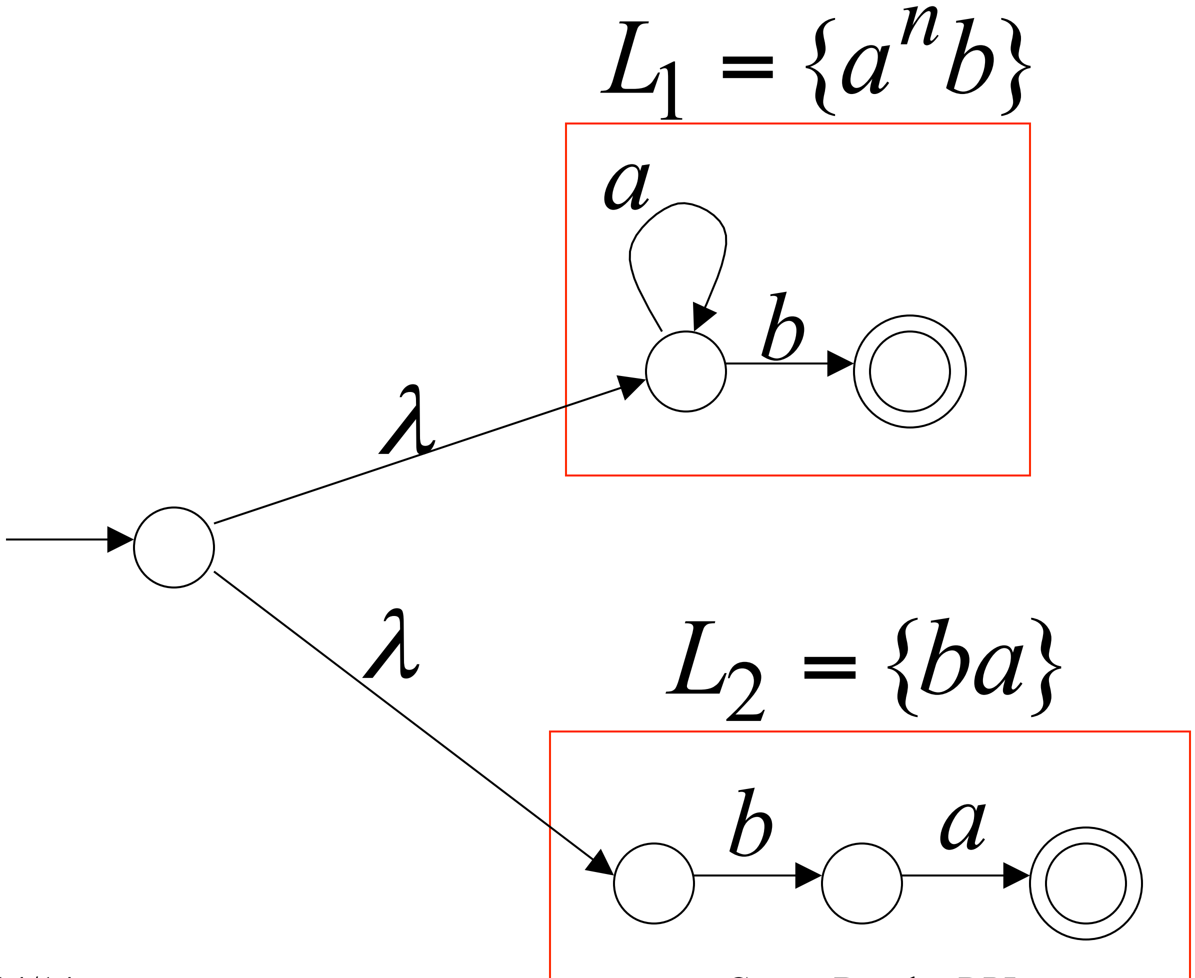
Union

NFA for $L_1 \cup L_2$



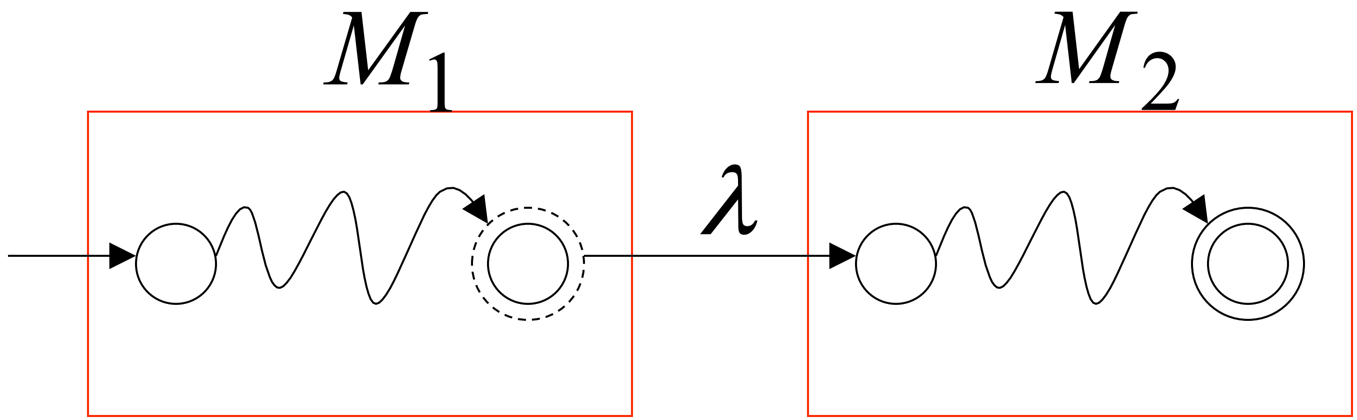
Example

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



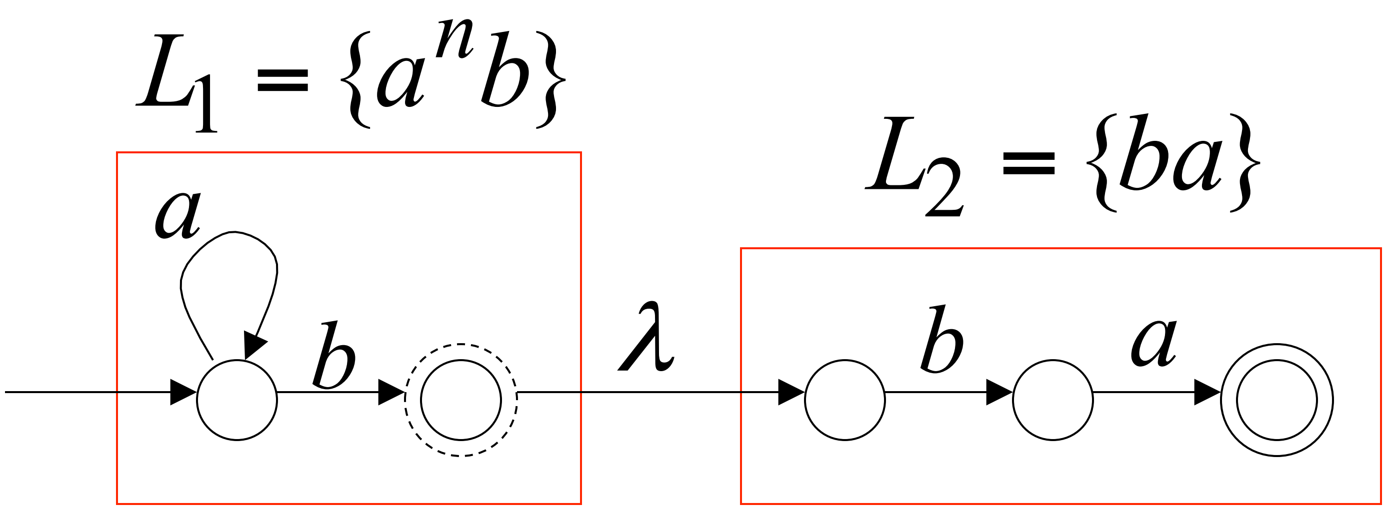
Concatenation

NFA for $L_1 L_2$



Example

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

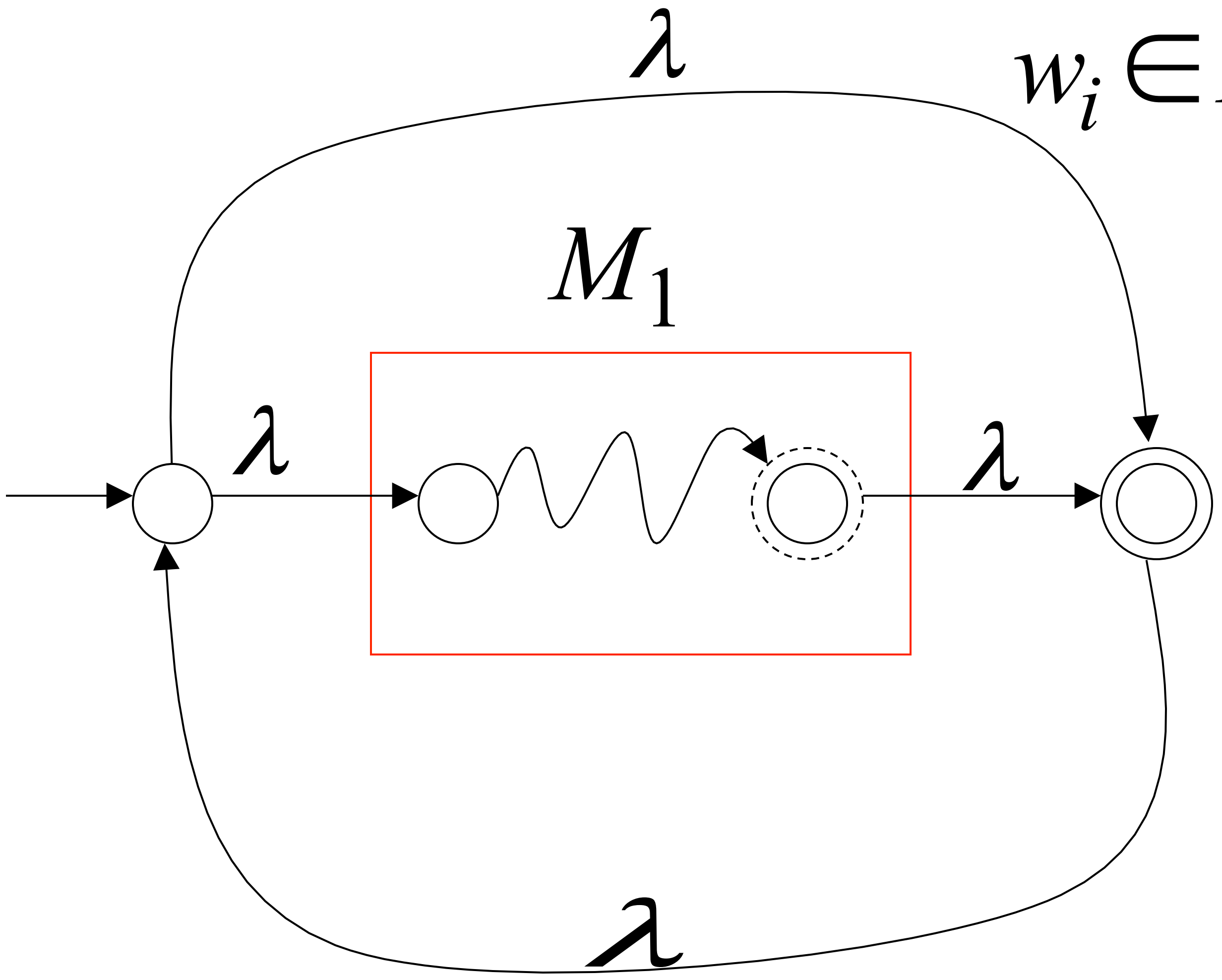


Star Operation

NFA for $L_1{}^*$

$w = w_1 w_2 \cdots w_k$

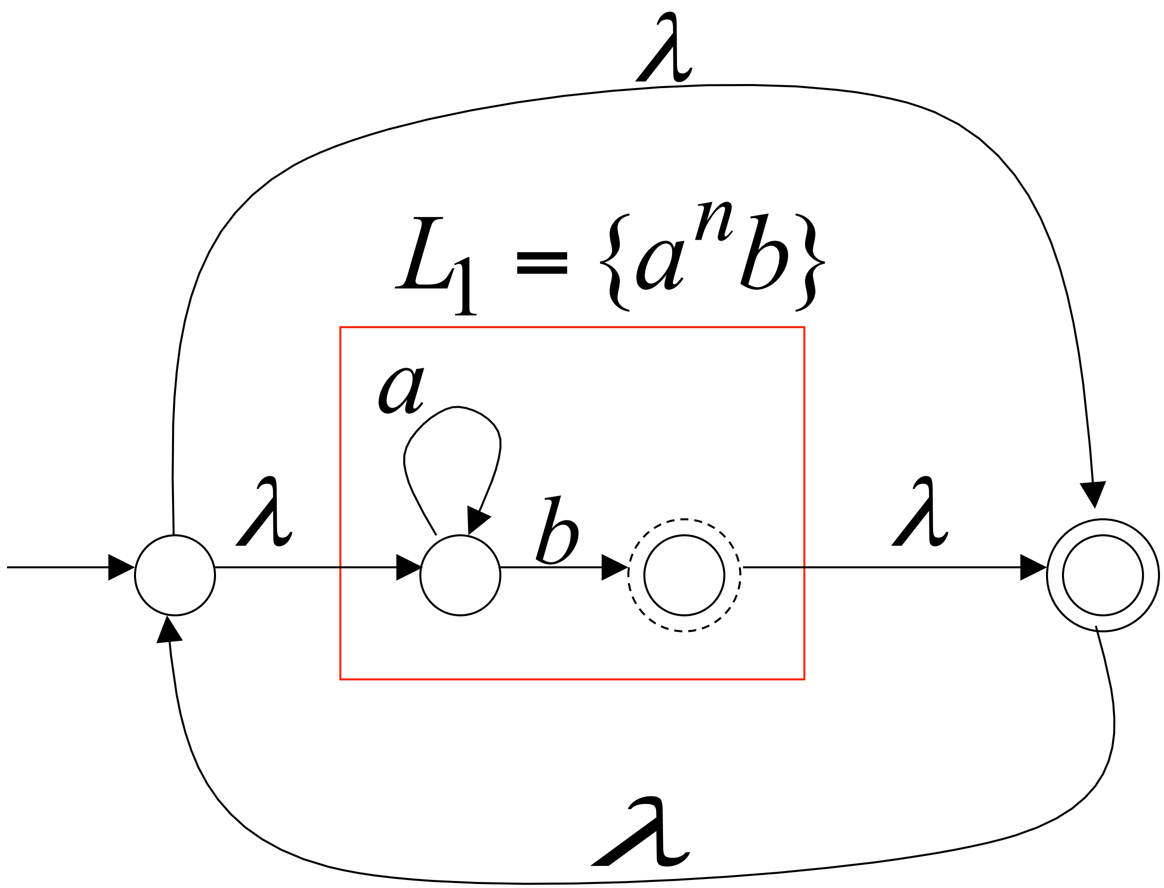
$w_i \in L_1$



$\lambda \in L_1{}^*$

Example

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



Reverse

NFA for L_1^R

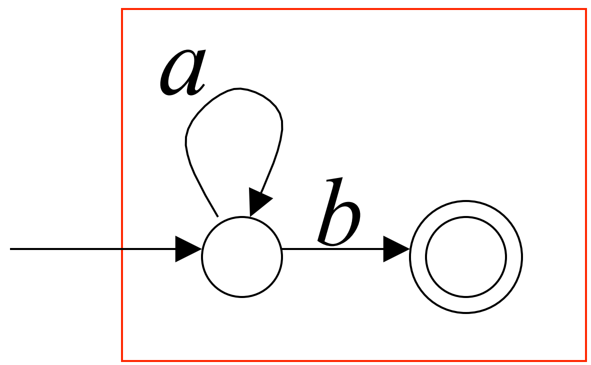
1. Reverse all transitions

2. Make initial state accepting state and vice versa

Example

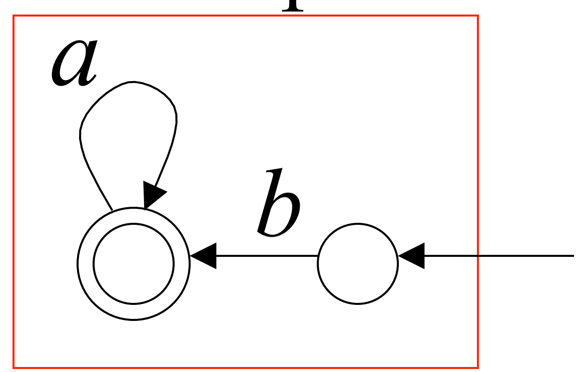
$$L_1 = \{a^n b\}$$

M_1

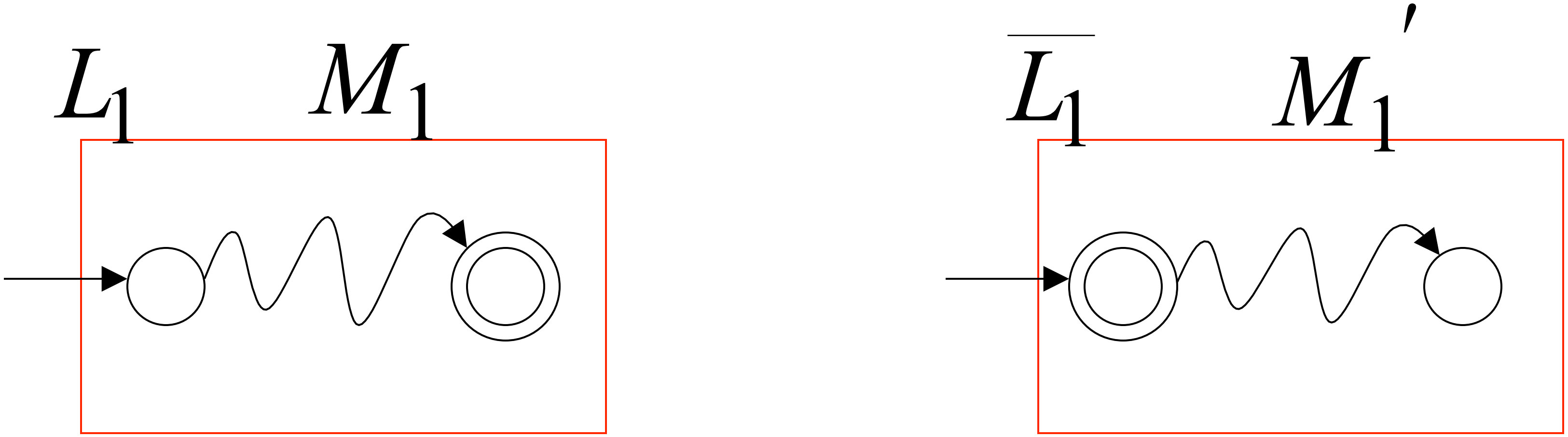


$$L_1^R = \{ba^n\}$$

M_1'



Complement

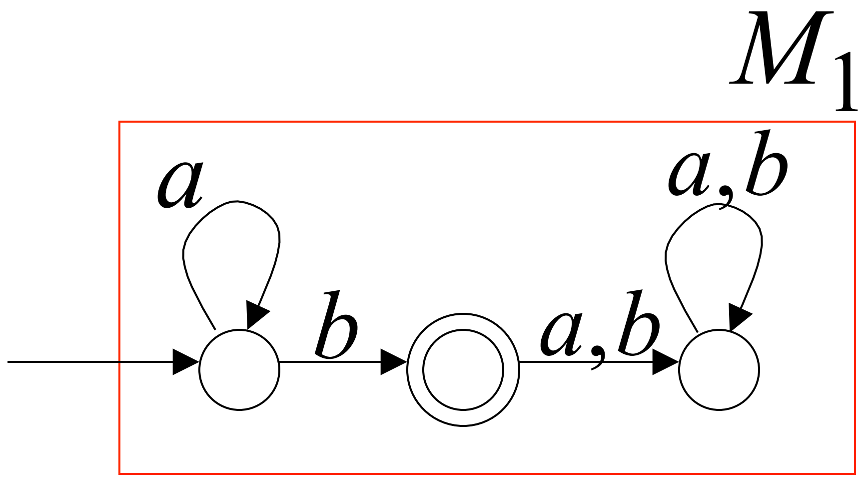


1. Take the DFA that accepts L_1

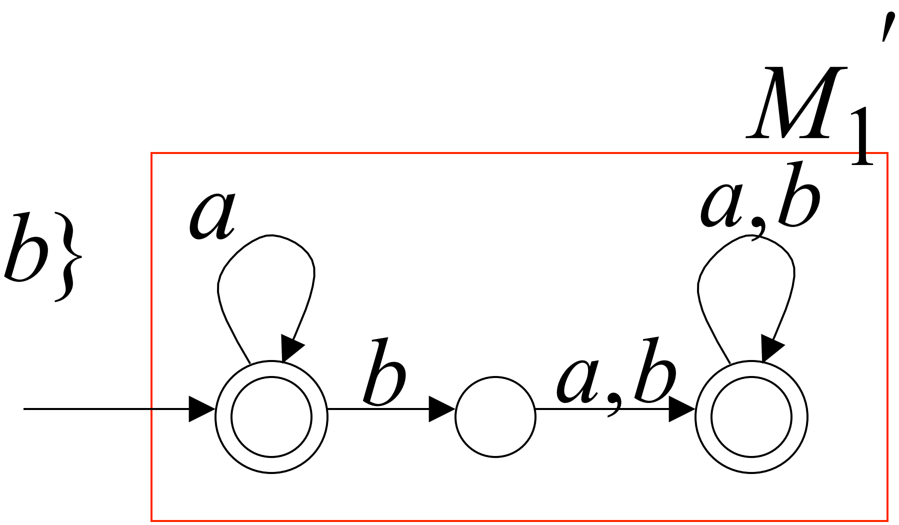
2. Make accepting states non-final, and vice-versa

Example

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



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



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

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

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 $L_1 \cap L_2$

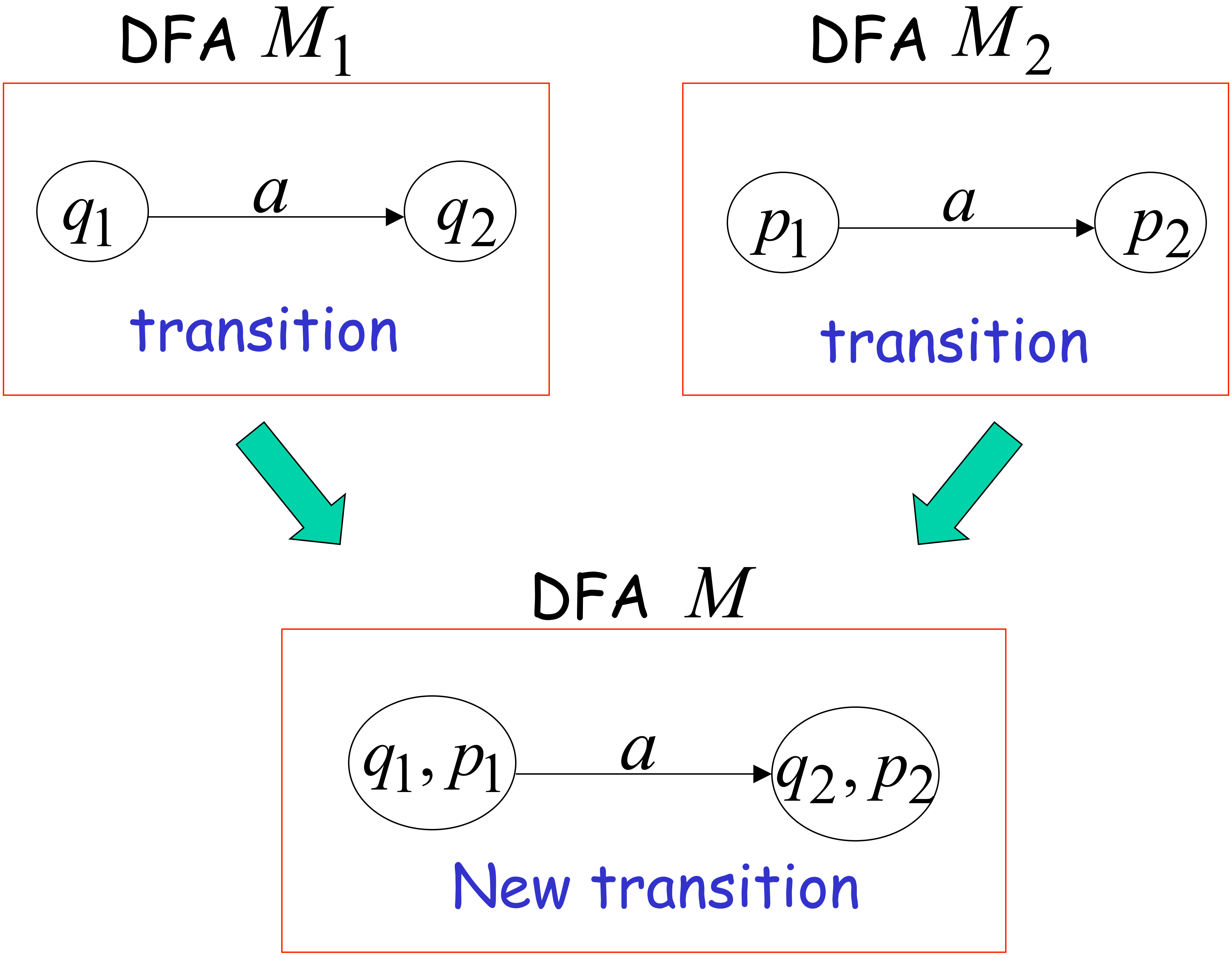
M simulates in parallel M_1 and M_2

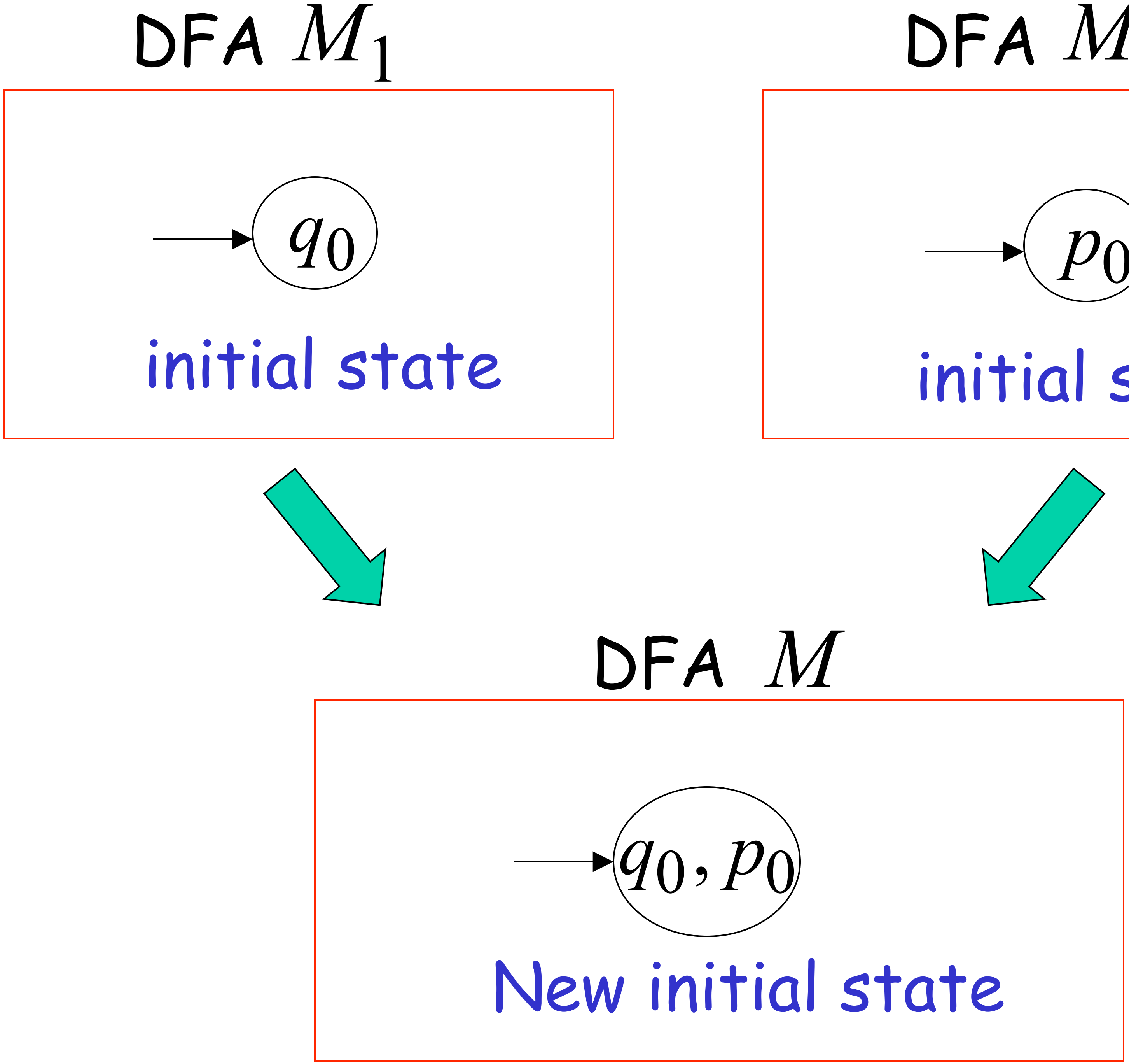
States in M

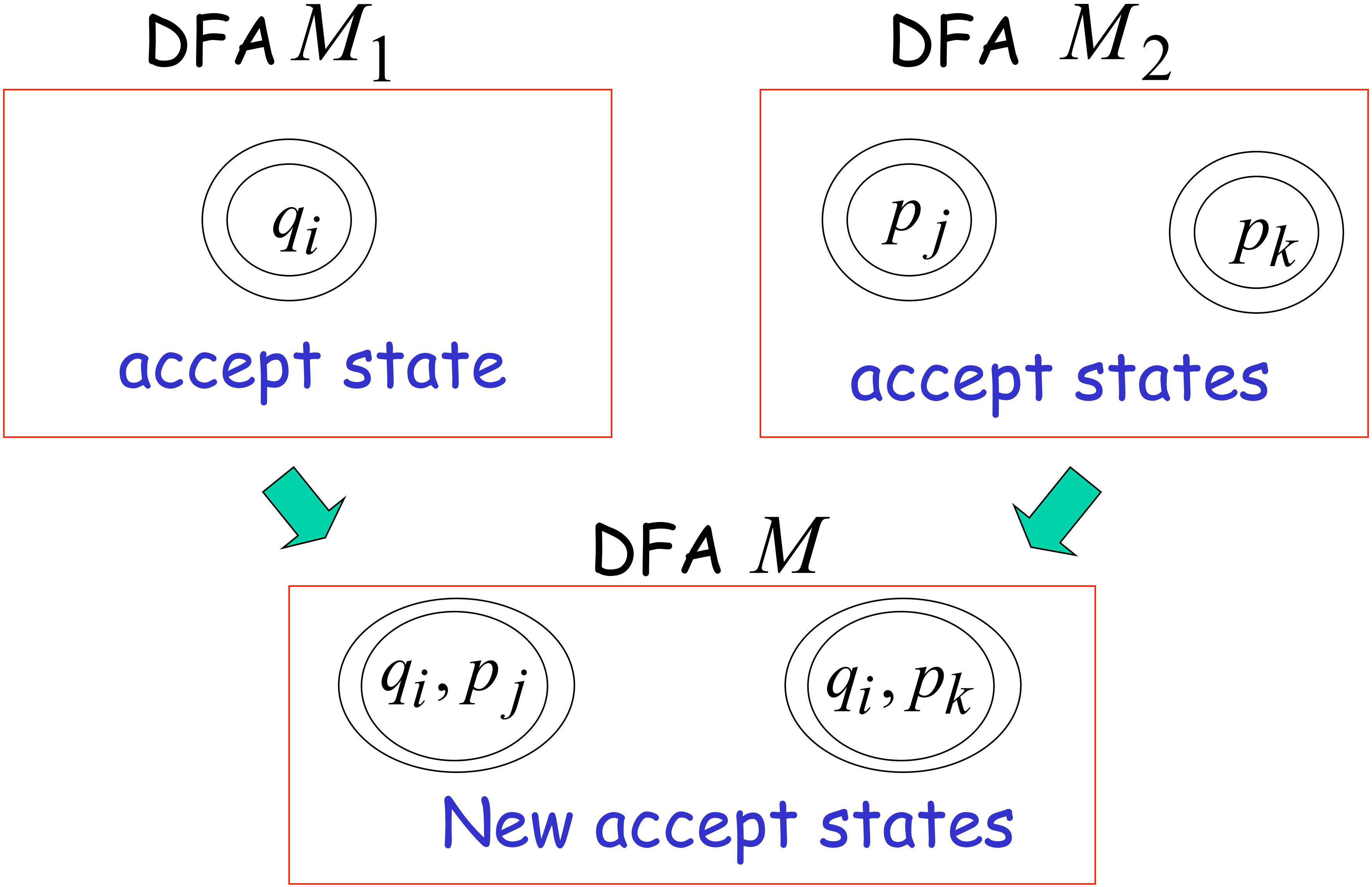
q_i, p_j

State in M_1

State in M_2


DFA M_1
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



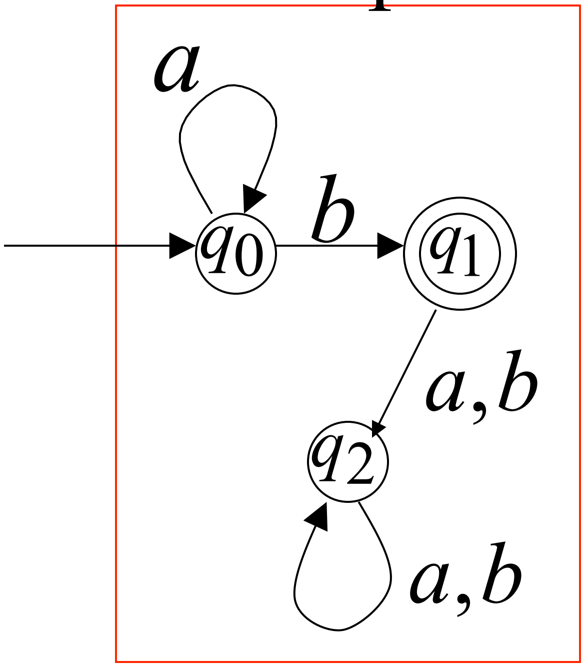
Both constituents must be accepting states

Example:

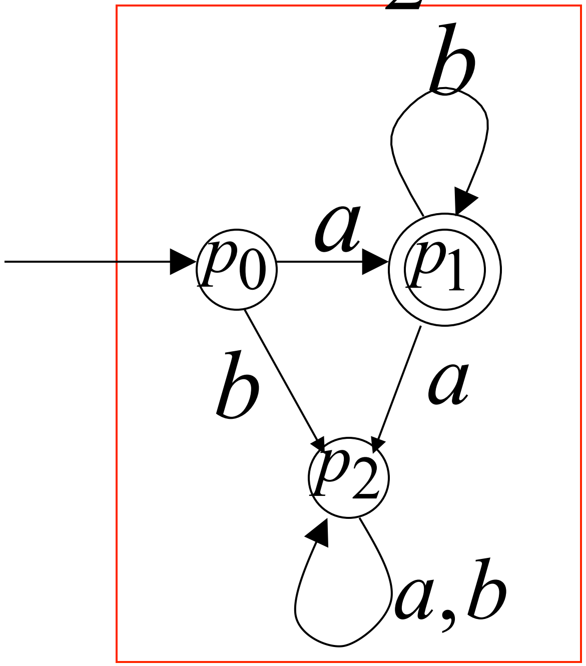
$$L_1 = \{a^n b\} \quad n \geq 0$$

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

M_1

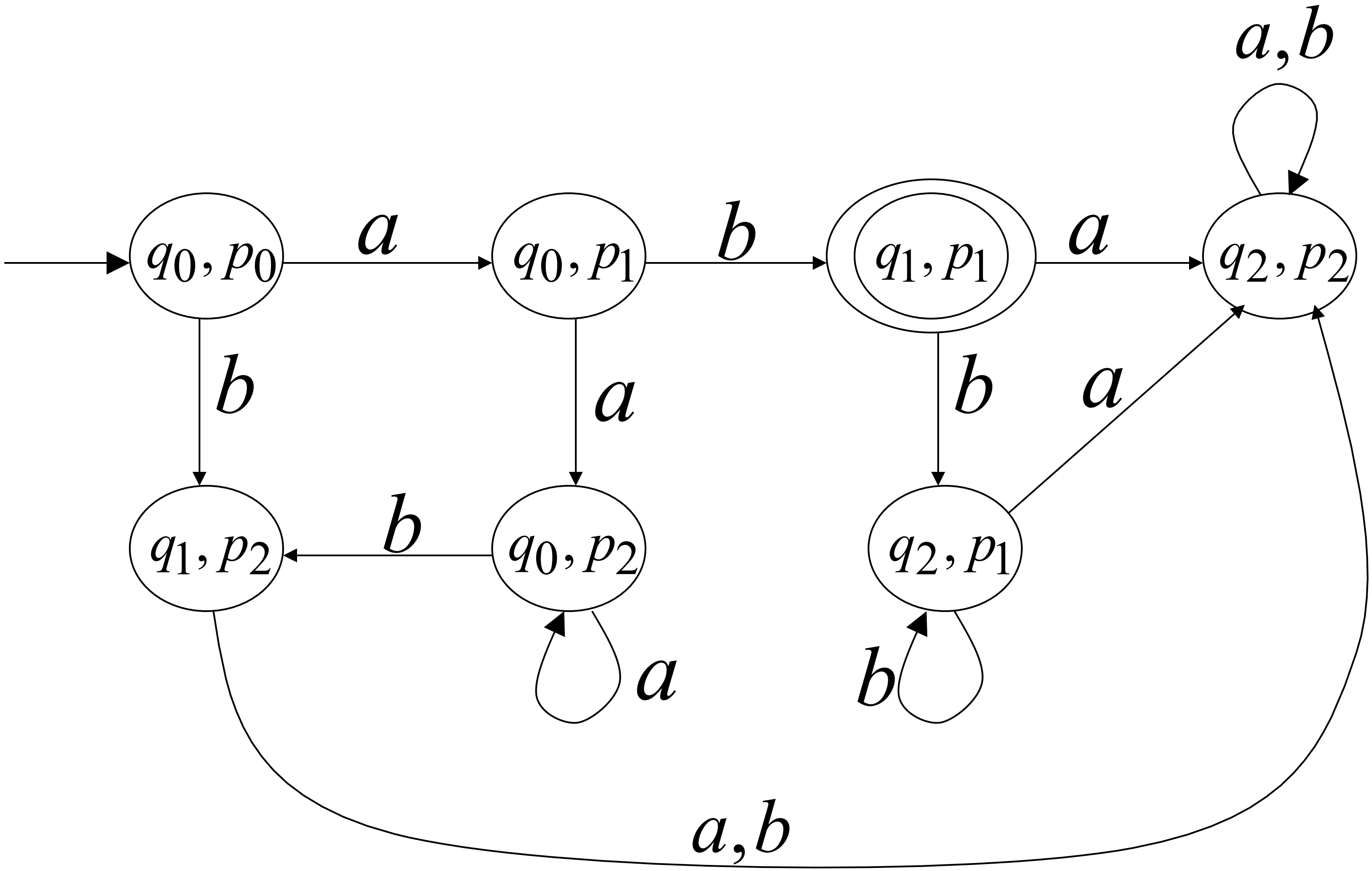


M_2



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 Languages and 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: $(a+b\cdot c)^{*}\cdot(c+\varnothing)$

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

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

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

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

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

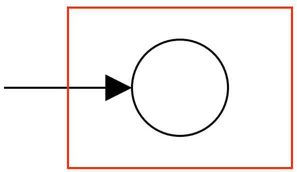
Proof by induction on the size of r

Induction Basis

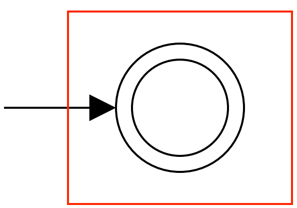
Primitive Regular Expressions: $\emptyset, \quad \lambda, \quad \alpha$

Corresponding

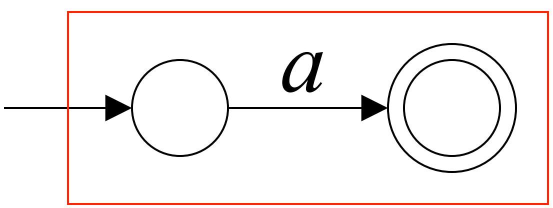
NFAs



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



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



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

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

By definition of regular expressions:

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

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

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

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

By inductive hypothesis we know:

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

We also know:

Regular languages are closed under:

Union	$L(r_1) \cup L(r_2)$
Concatenation	$L(r_1)L(r_2)$
Star	$(L(r_1))^*$

Therefore:

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

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

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

Are regular languages

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

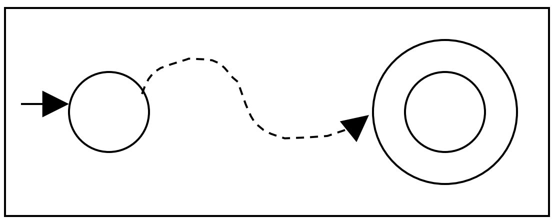
is trivially a regular language (by induction hypothesis)

End of Proof-Part 1

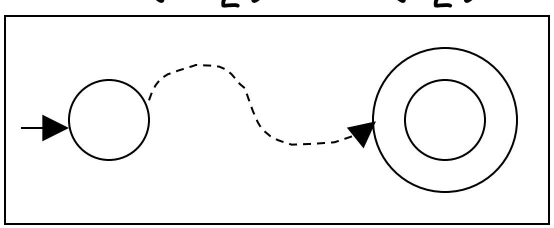
Using the regular closure of these operations, we can construct recursively the NFA M that accepts $L(M) = L(r)$

Example: $r = r_1 + r_2$

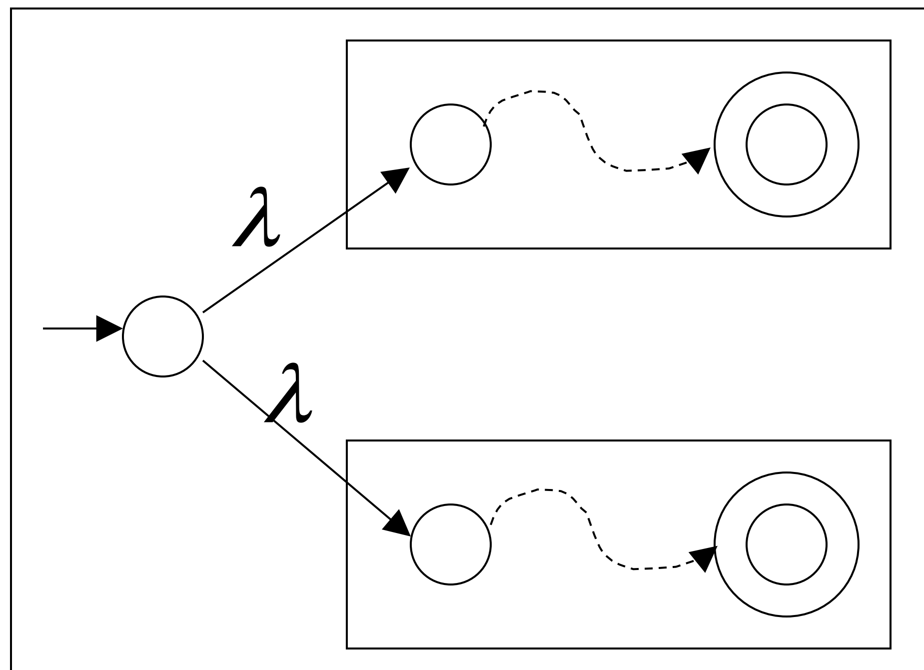
$L(M_1) = L(r_1)$



$L(M_2) = L(r_2)$



$L(M) = L(r)$



Proof - Part 2

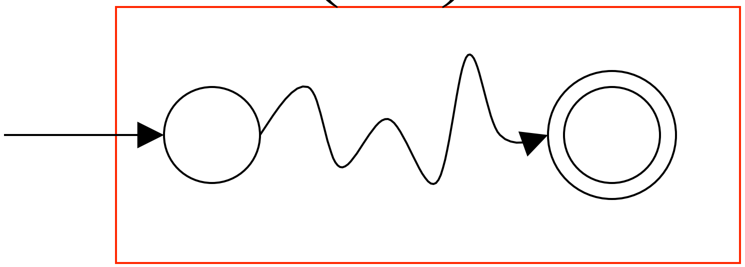
$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

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

Since L is regular, there is a NFA M that accepts it

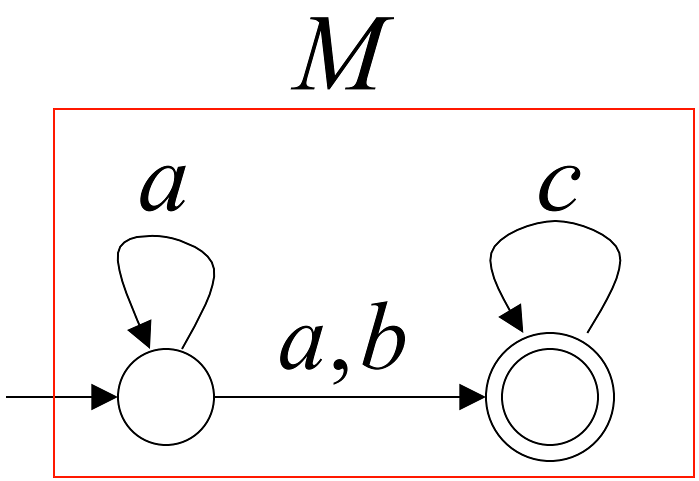
$$L(M) = L$$

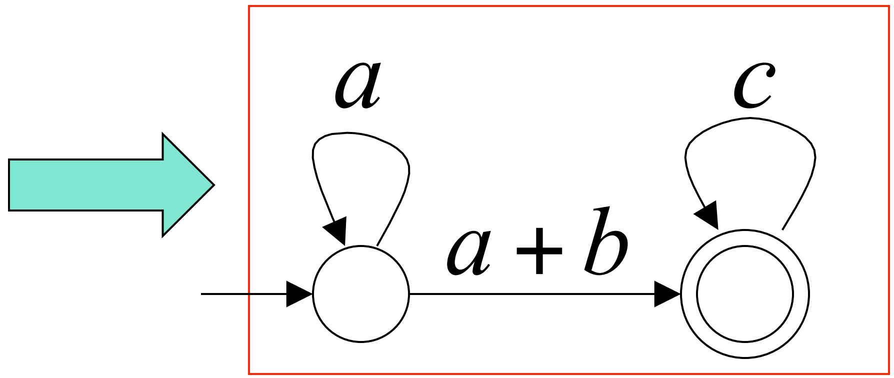


Take it with a single final state

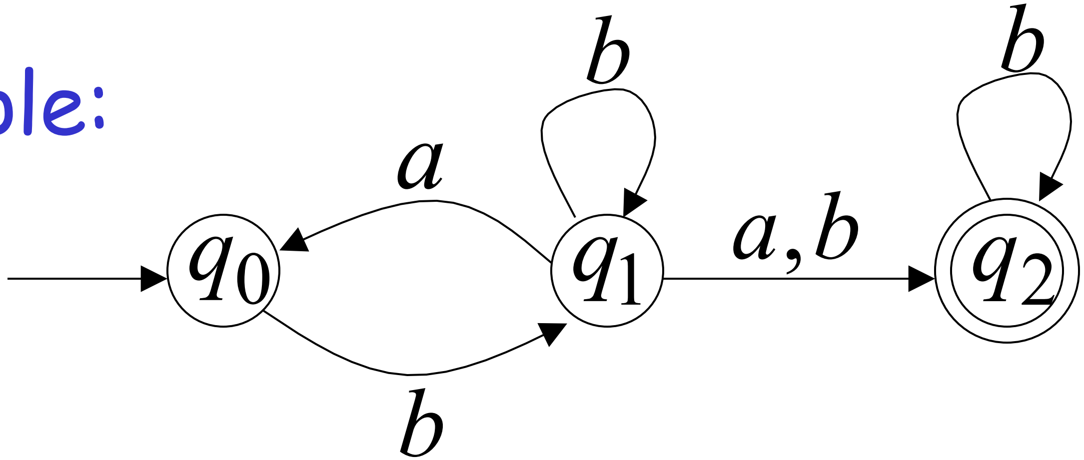
From M construct the equivalent **Generalized Transition Graph** in which transition labels are regular expressions

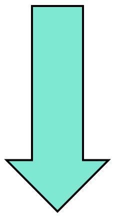
Example:



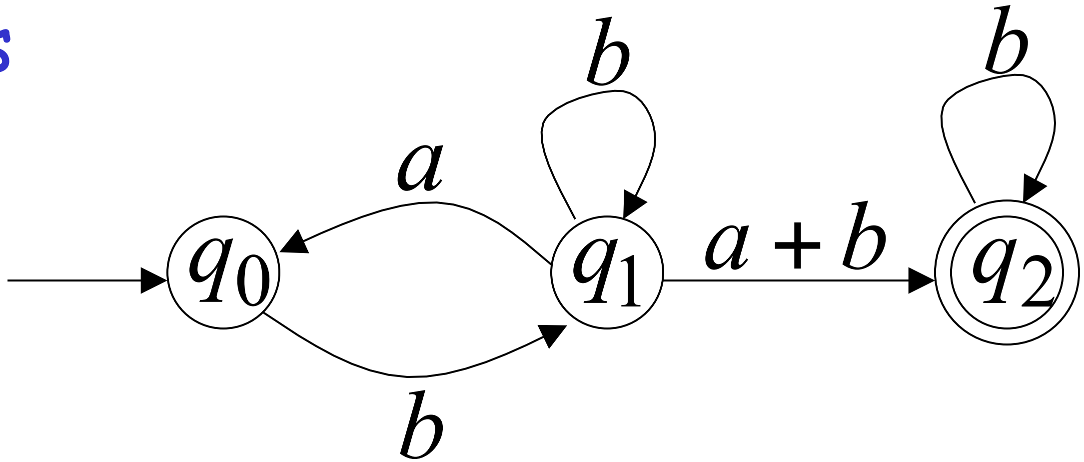


Another Example:

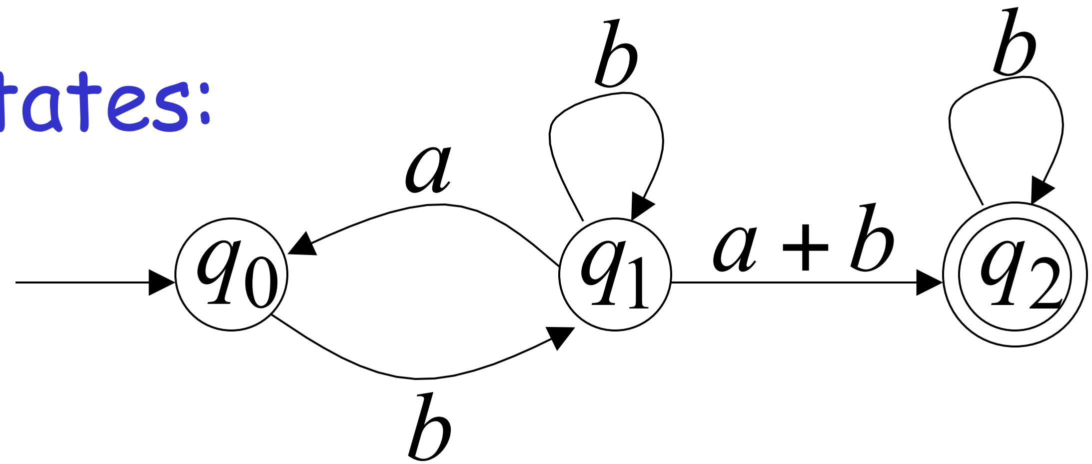




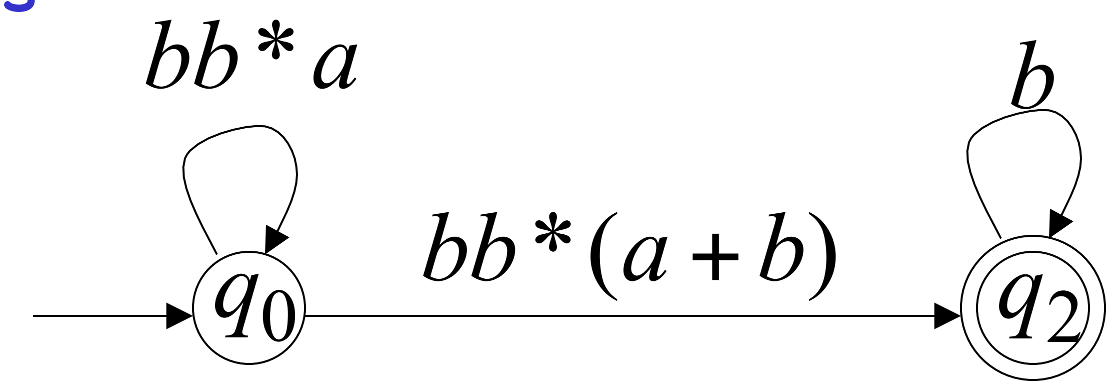
Transition labels are regular expressions



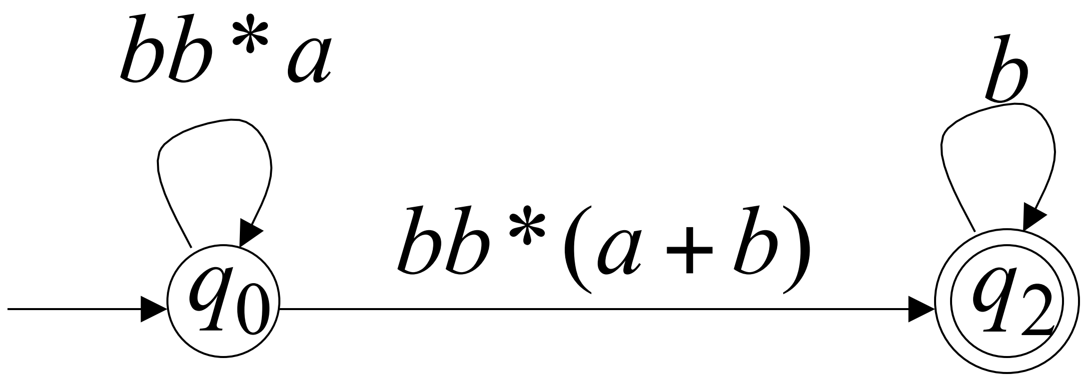
Reducing the states:



Transition labels are regular expressions



Resulting Regular Expression:

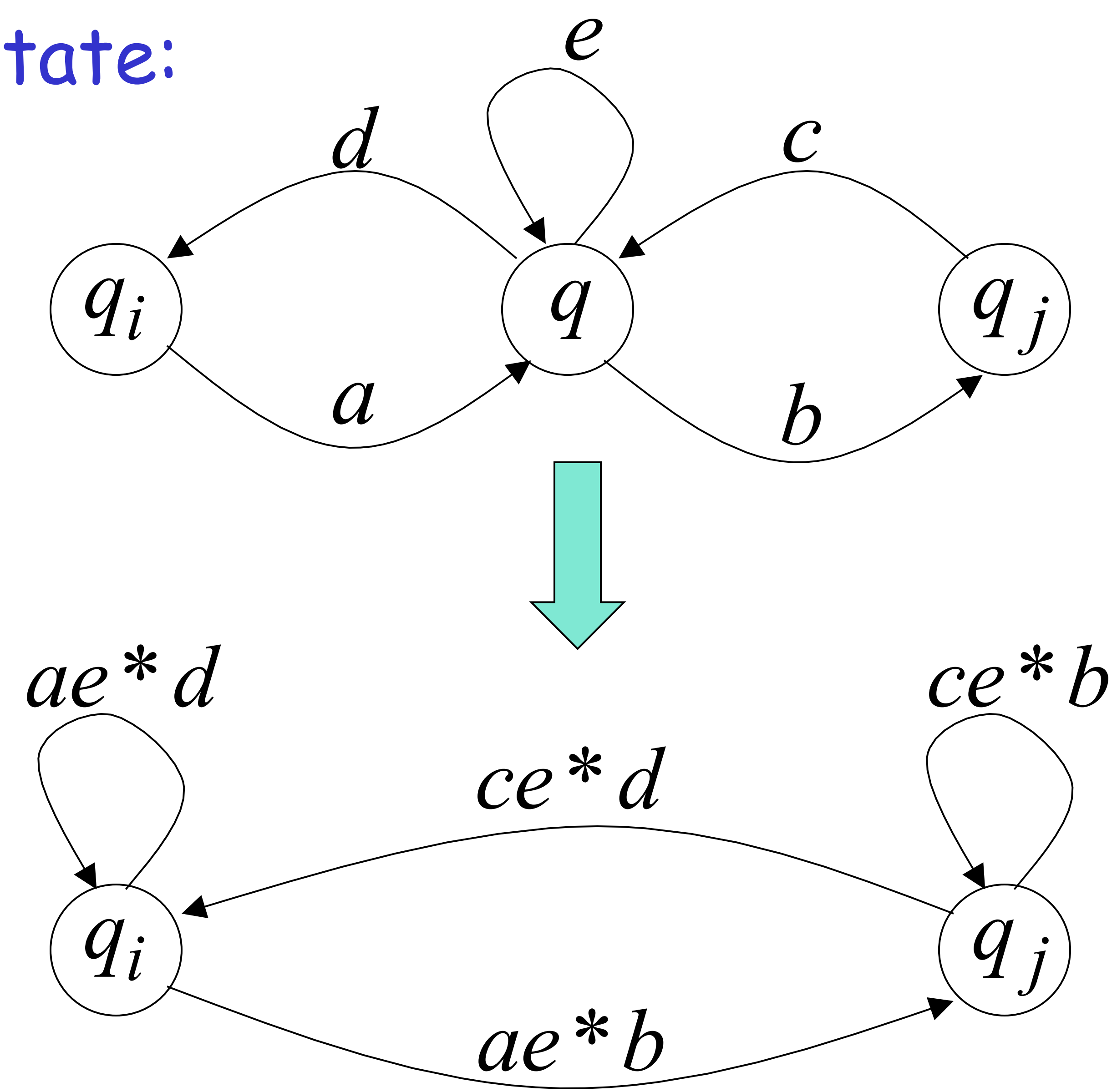


$$r = (bb^*a)^*bb^*(a+b)b^*$$

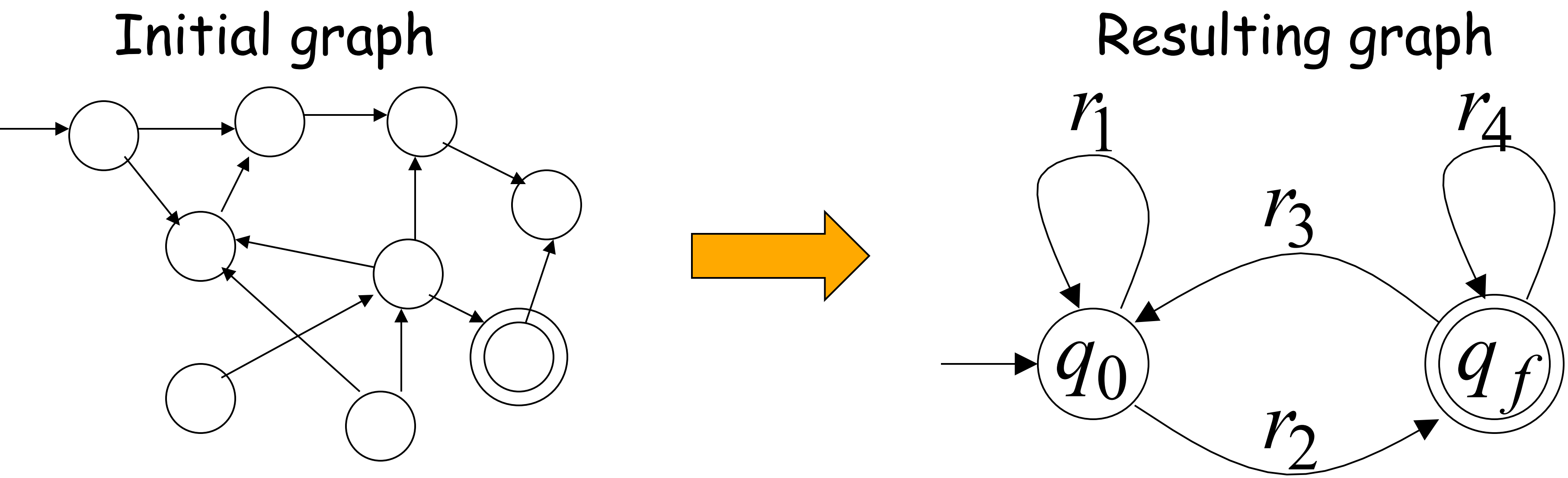
$$L(r) = L(M) = L$$

In General

Removing a state:



By repeating the process until two states are left, the resulting graph is



The resulting regular expression:

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

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

End of Proof-Part 2

Non-regular languages

(Pumping Lemma)

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

Non-regular languages

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

Regular languages

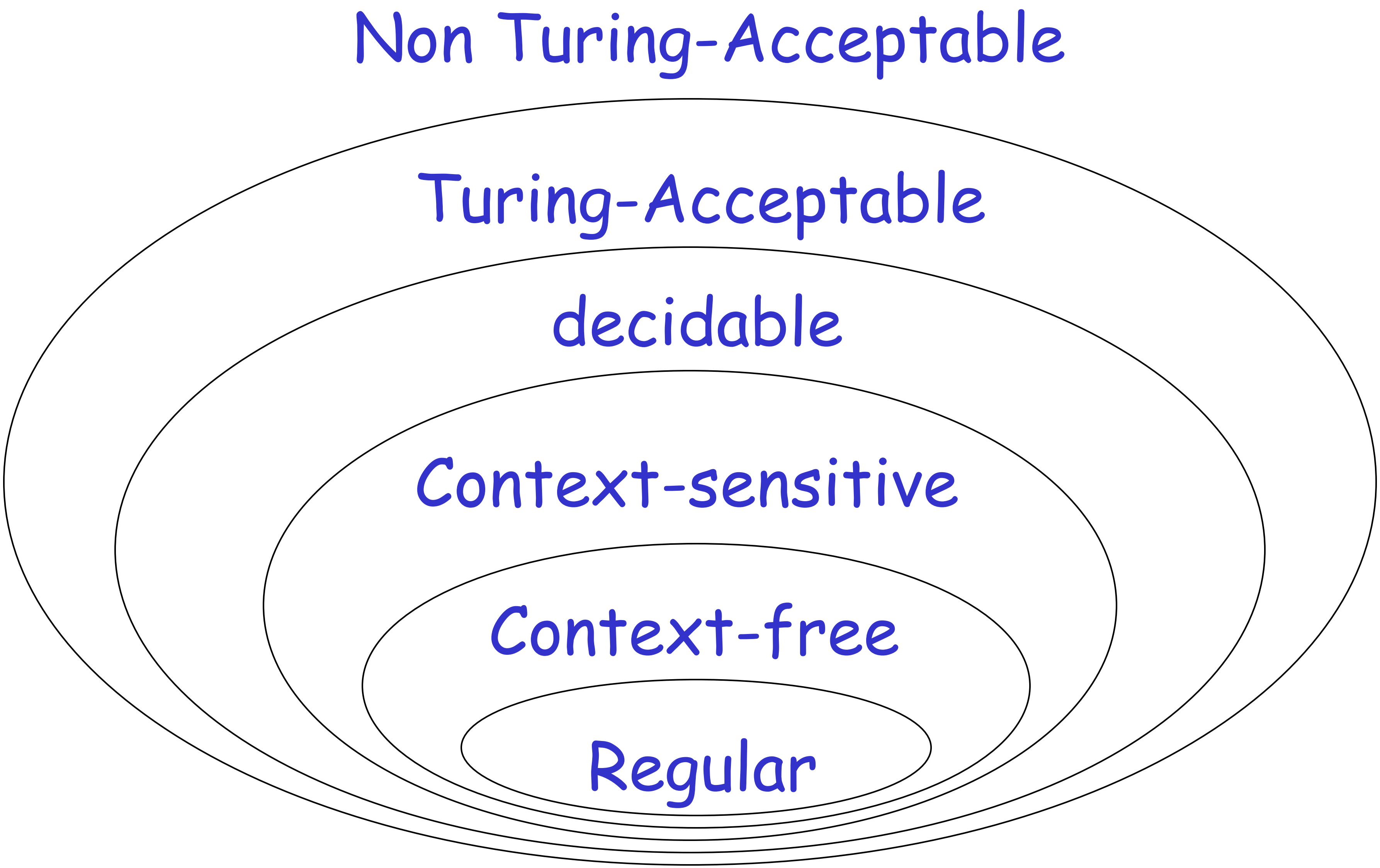
$$a^*b$$

$$b^*c + a$$

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

$$etc...$$

The Chomsky Hierarchy



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)

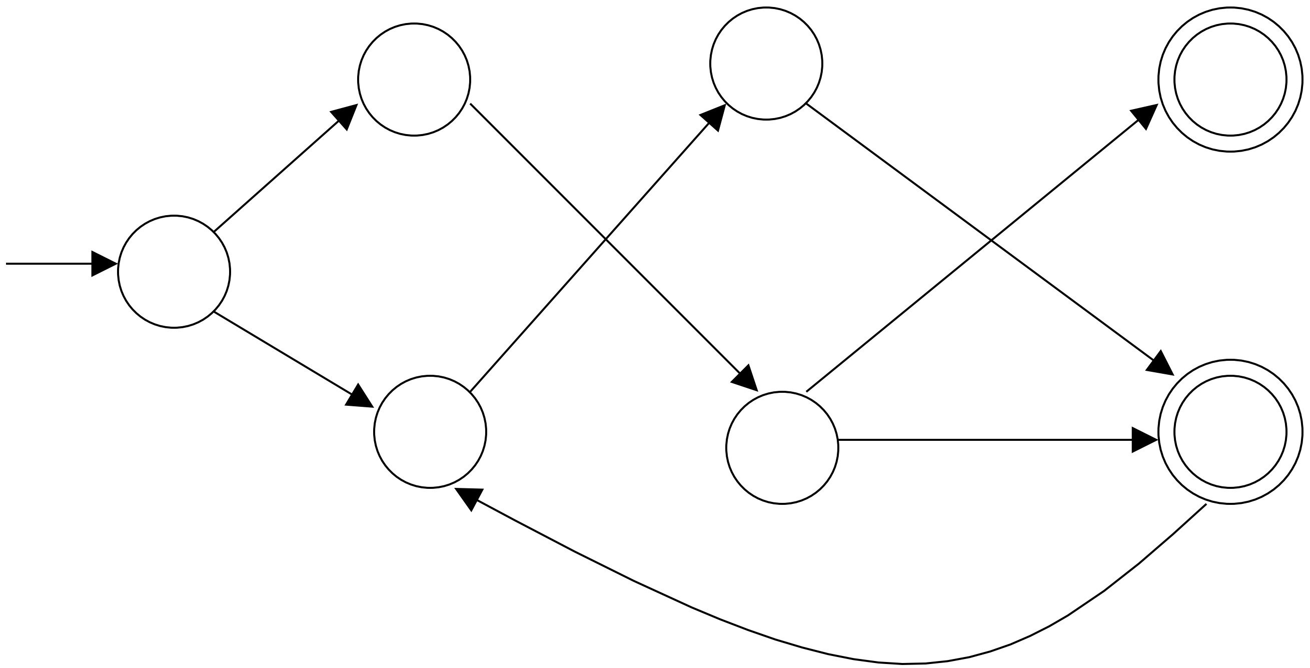
Solution: use the Pumping Lemma !!!

The Pumping Lemma

Take an infinite regular language L

(contains an infinite number of strings)

There exists a DFA that accepts L



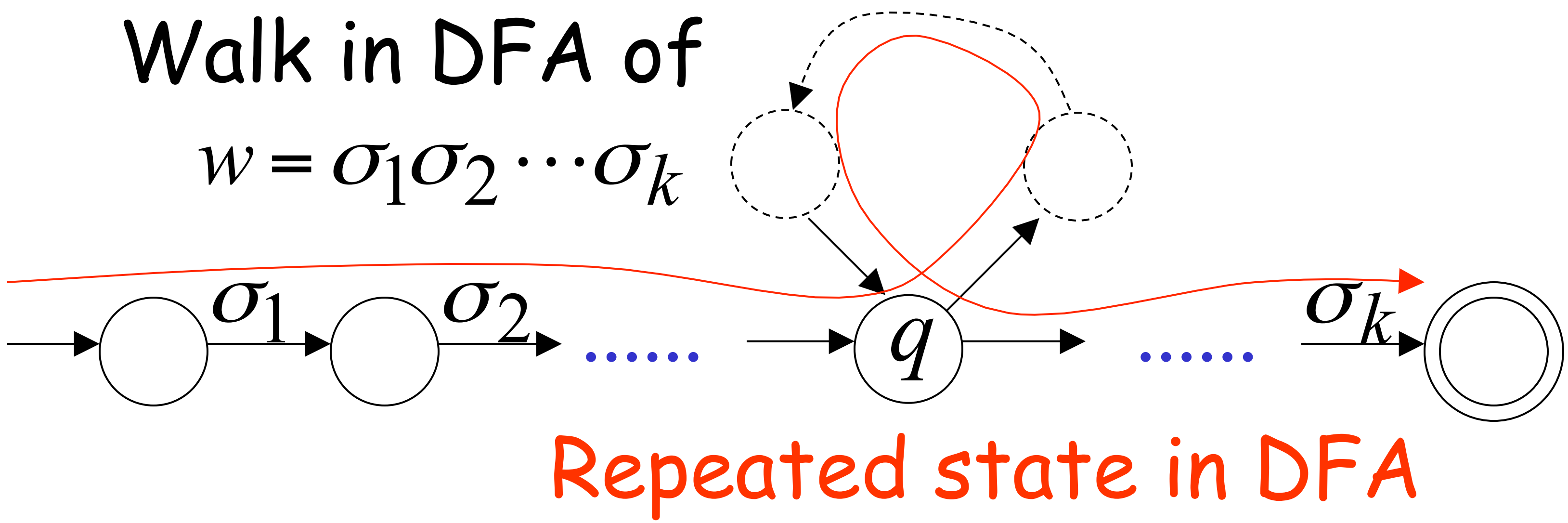
m

states

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

(number of states of DFA)

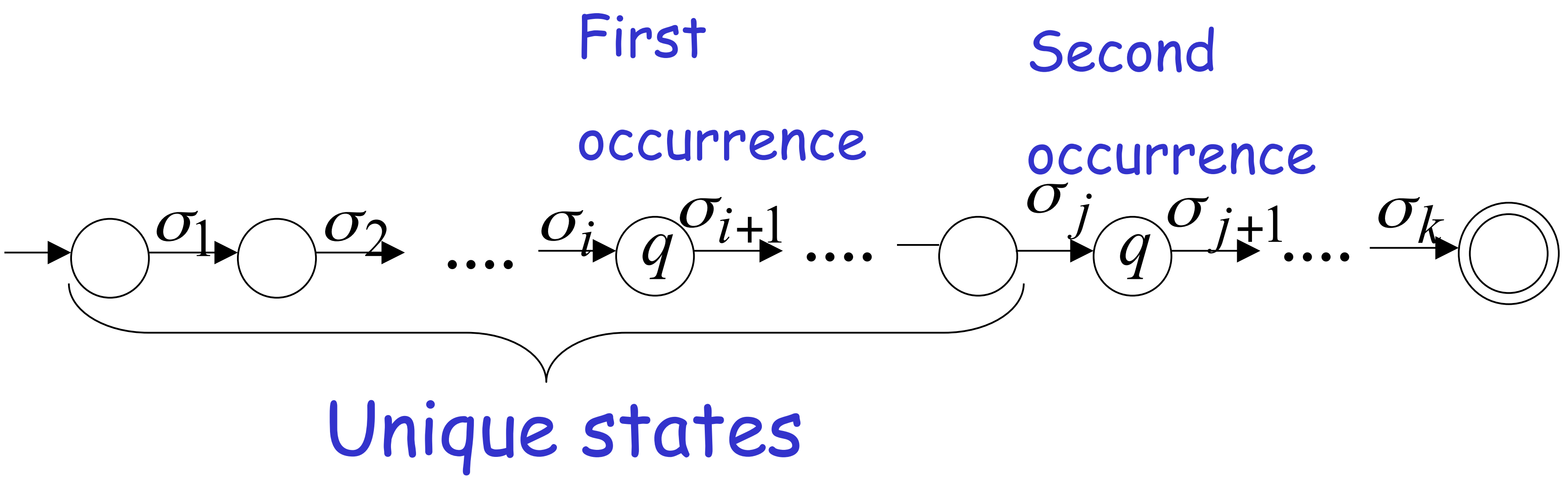
then, at least one state is repeated in the walk of w



There could be many states repeated

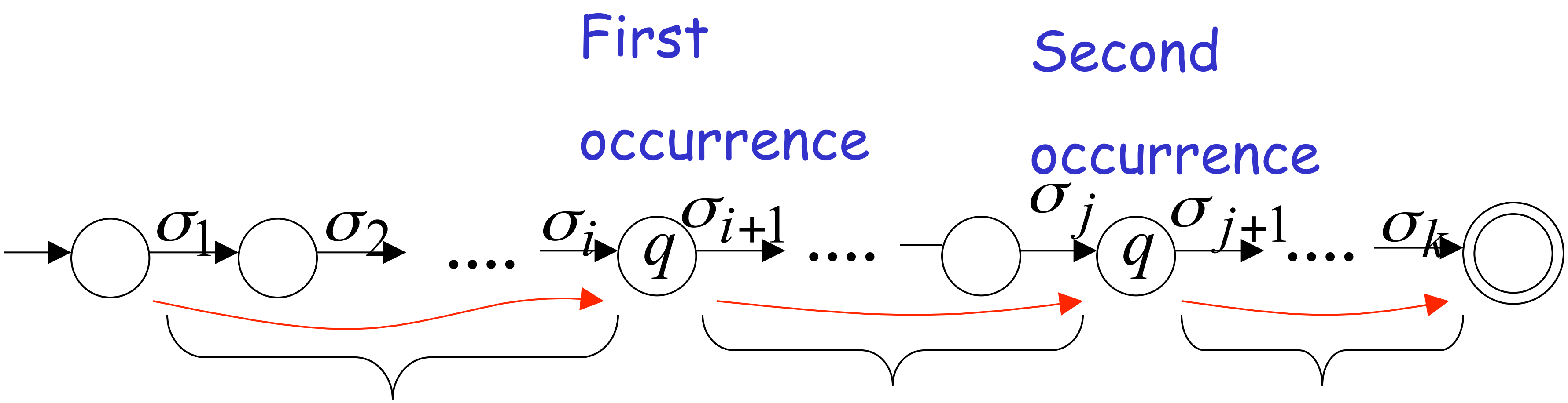
Take q to be the first state repeated

One dimensional projection of walk w:



We can write $w = xyz$

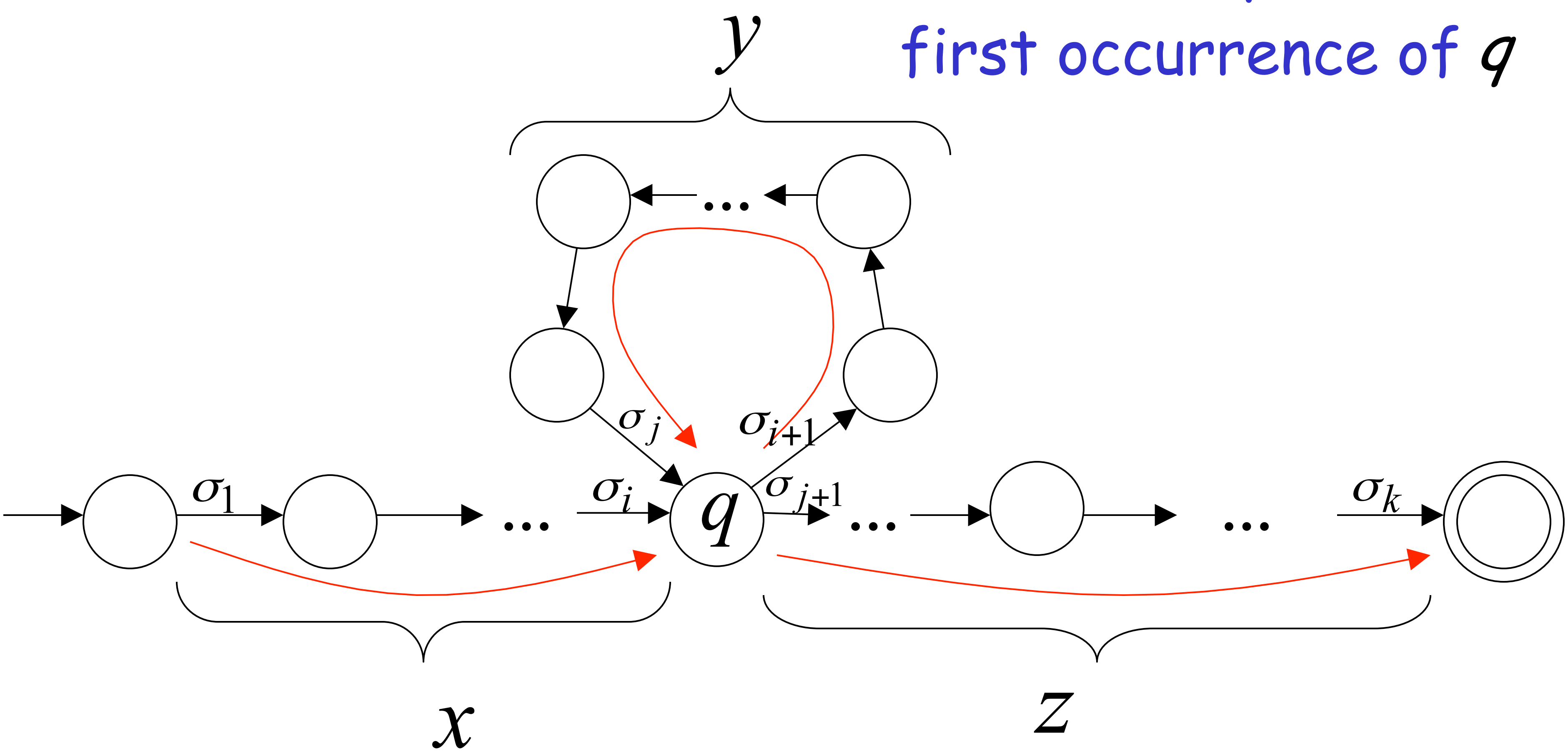
One dimensional projection of walk w:



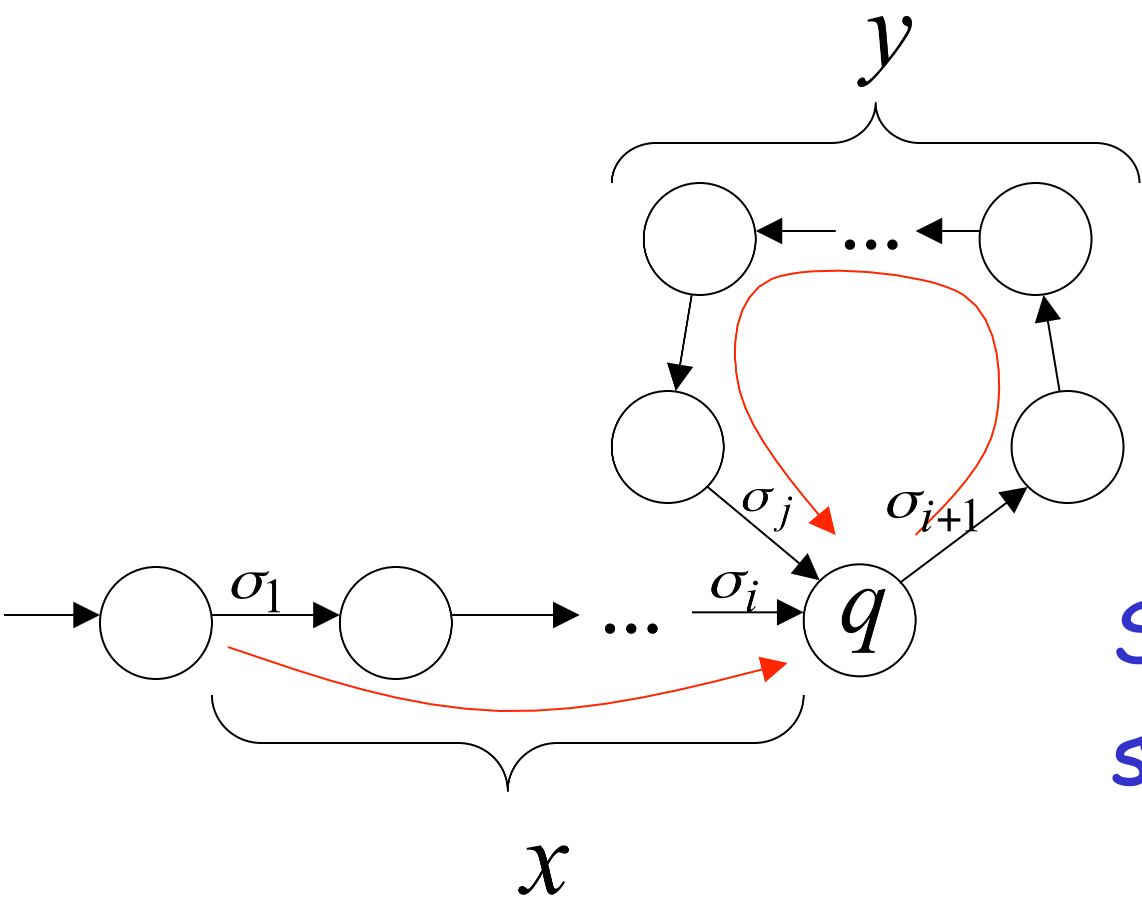
$x = \sigma_1 \cdots \sigma_i$ $\quad y = \sigma_{i+1} \cdots \sigma_j$ $\quad z = \sigma_{j+1} \cdots \sigma_k$

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

contains only first occurrence of q



Observation: length $|x\,y| \leq m$ number of states of DFA



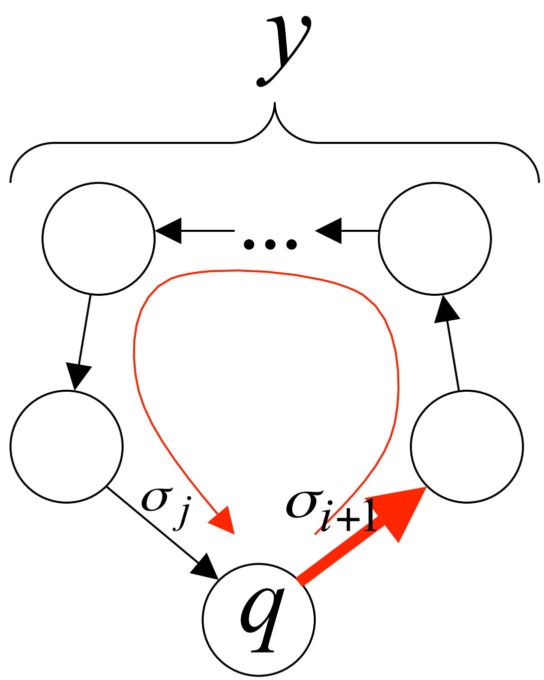
Unique States

Since, in xy no state is repeated

(except q)

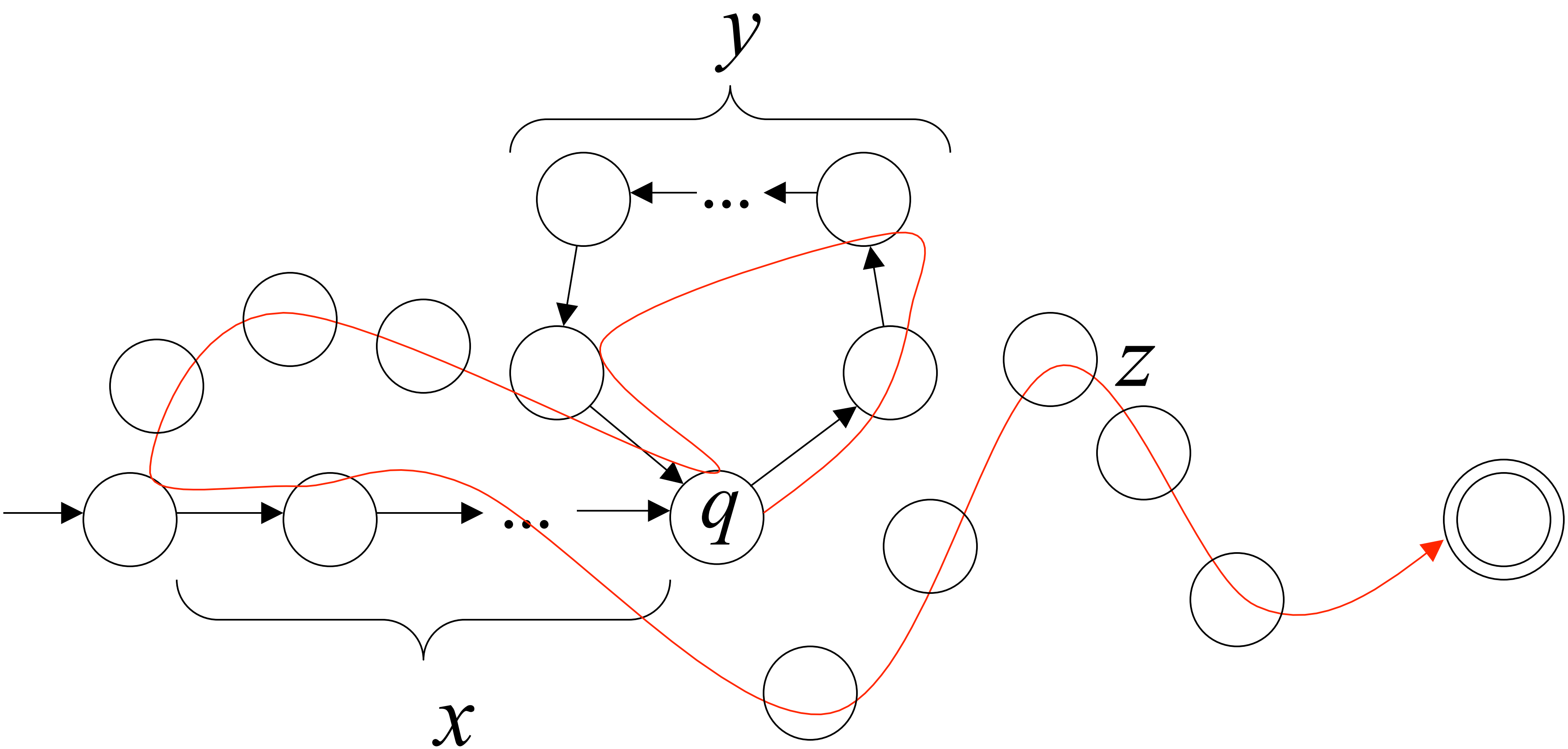
Observation: length $|y| \geq 1$

Since there is at least one transition in loop



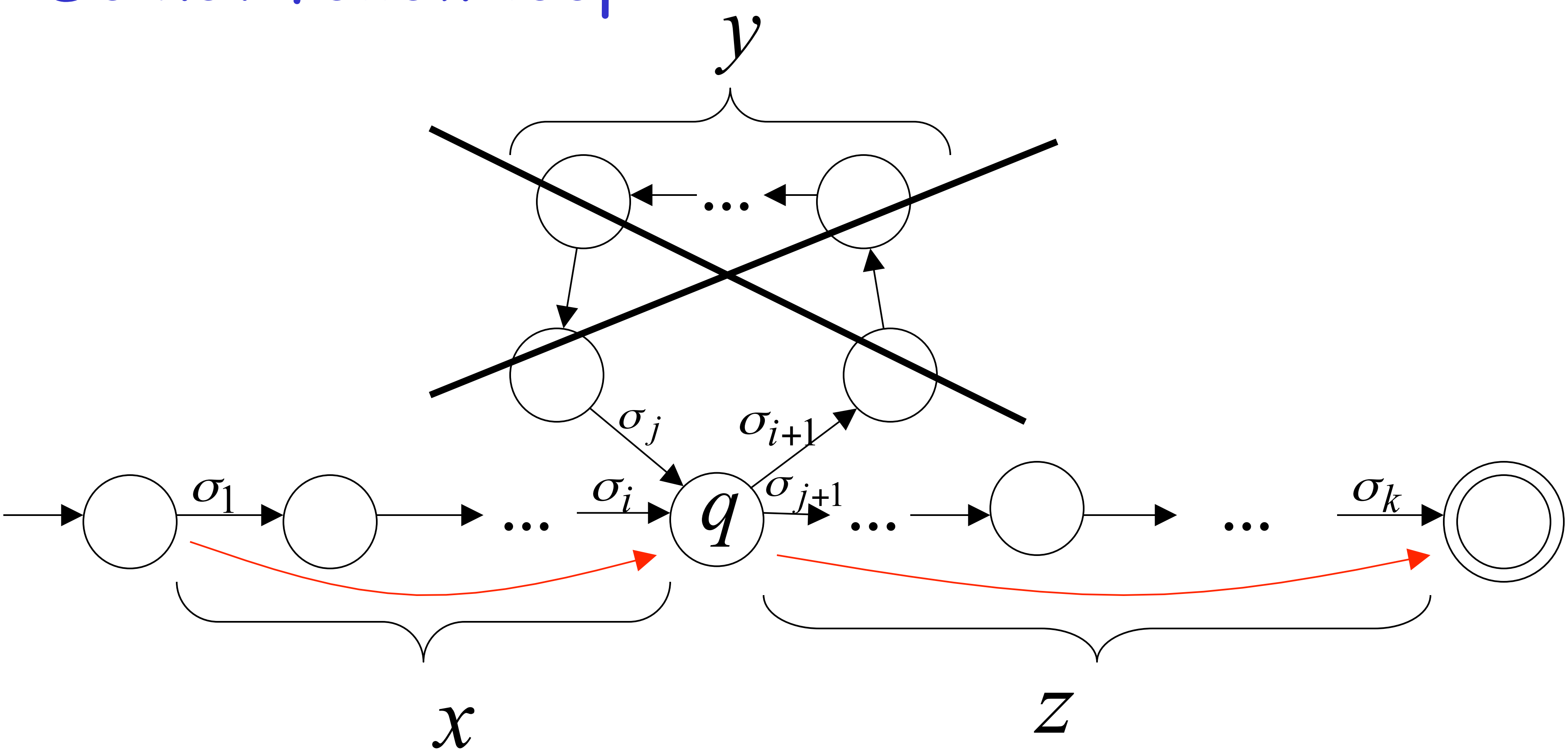
We do not care about the form of string z

z may actually overlap with the paths of x and y



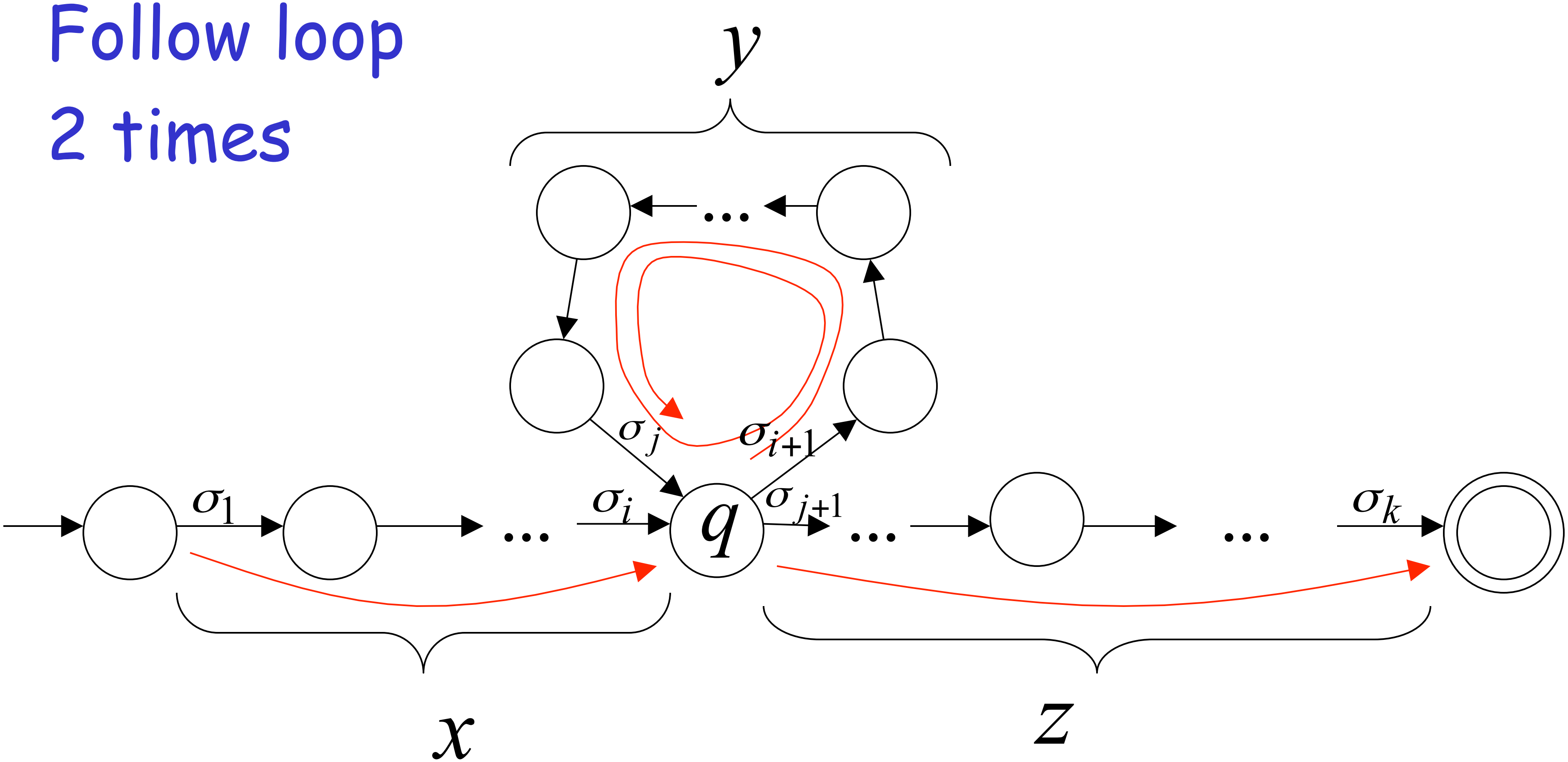
Additional string: The string $x\ z$ is accepted

Do not follow loop



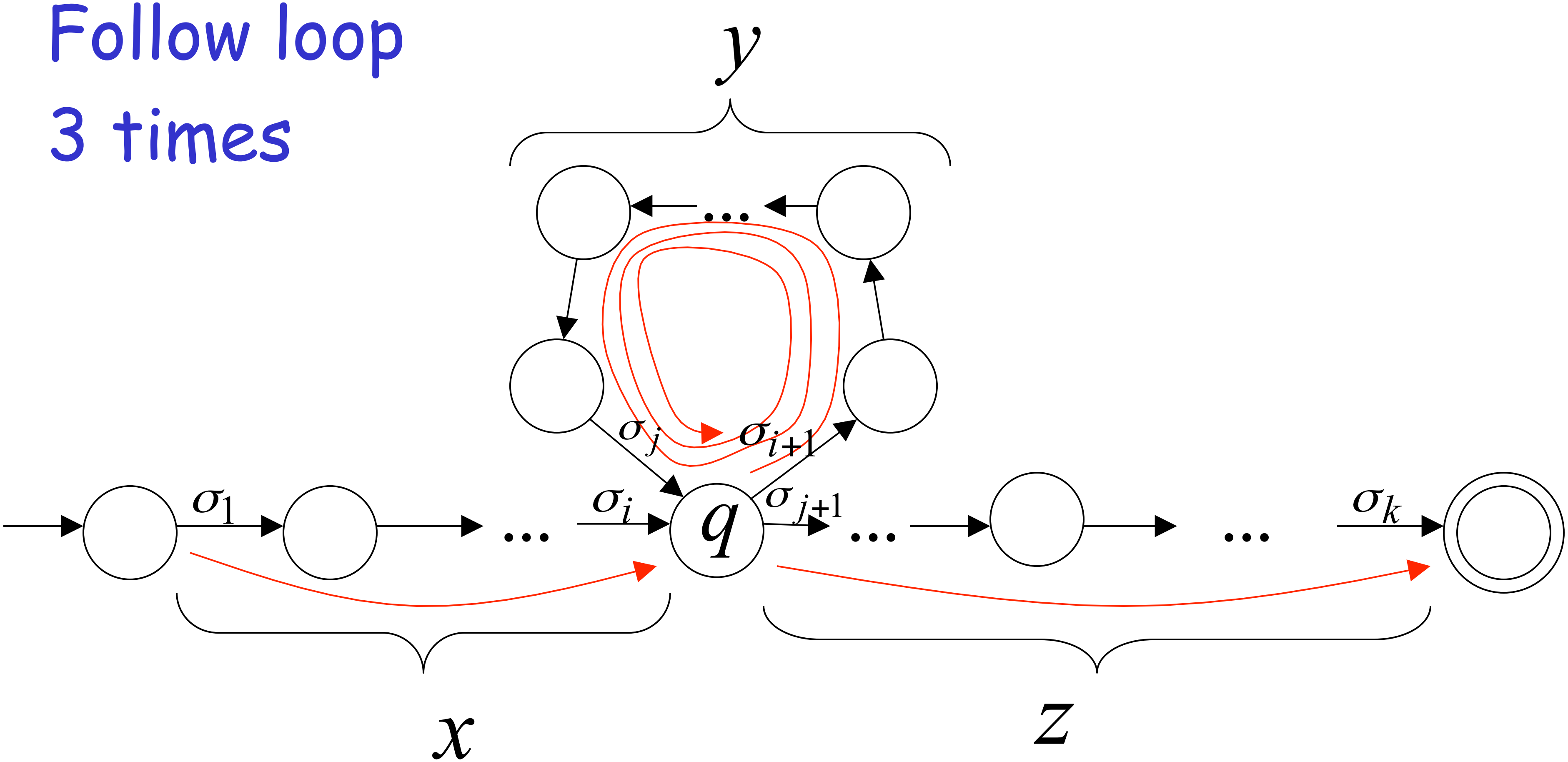
Additional string: The string $x\ y\ y\ z$ is accepted

Follow loop 2 times



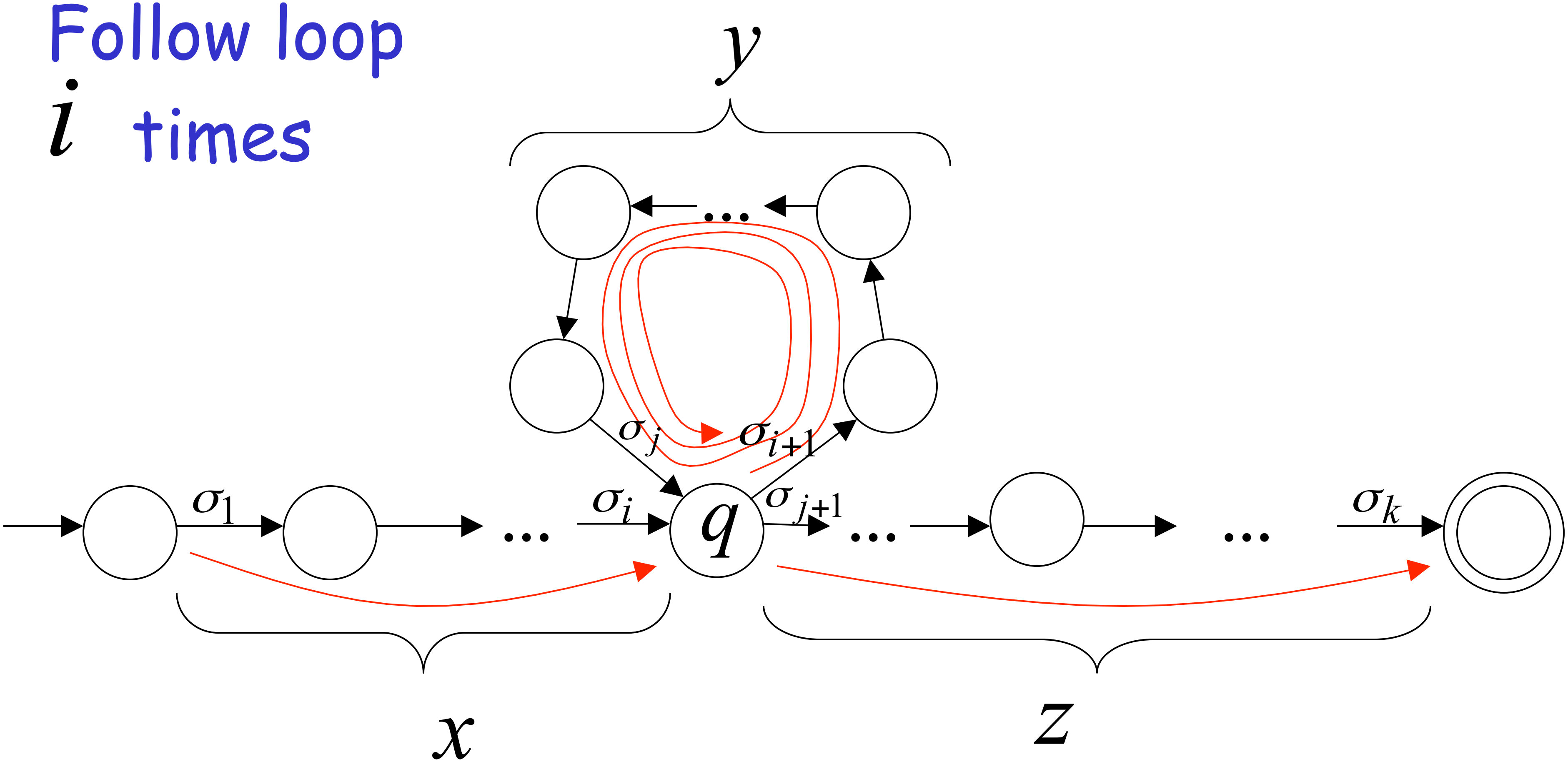
Additional string: The string $x\,y\,y\,y\,z$ is accepted

Follow loop 3 times



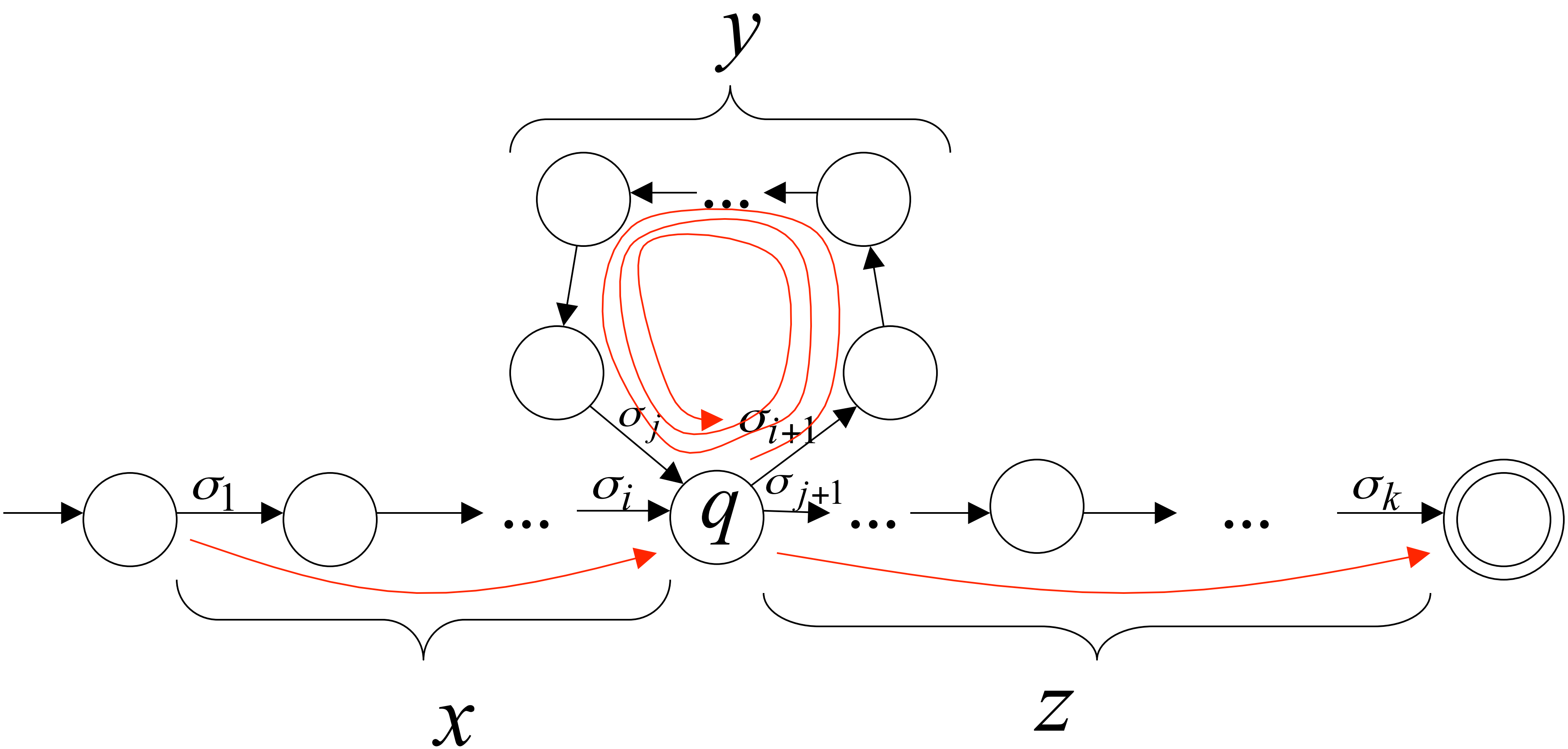
In General: The string $x\, y^i\, z$ is accepted $i = 0, 1, 2, \ldots$

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 $|w| \geq m$

$\boxed{\text{We pick } w = a^m b^m}$

From the Pumping Lemma:

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

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

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

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

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

with $z = a...ab...b$

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

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

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

$$\Downarrow$$

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

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