Properties of Regular Languages

For regular languages L_1 and L_2
we will prove that:

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

Reversal: L_1^R

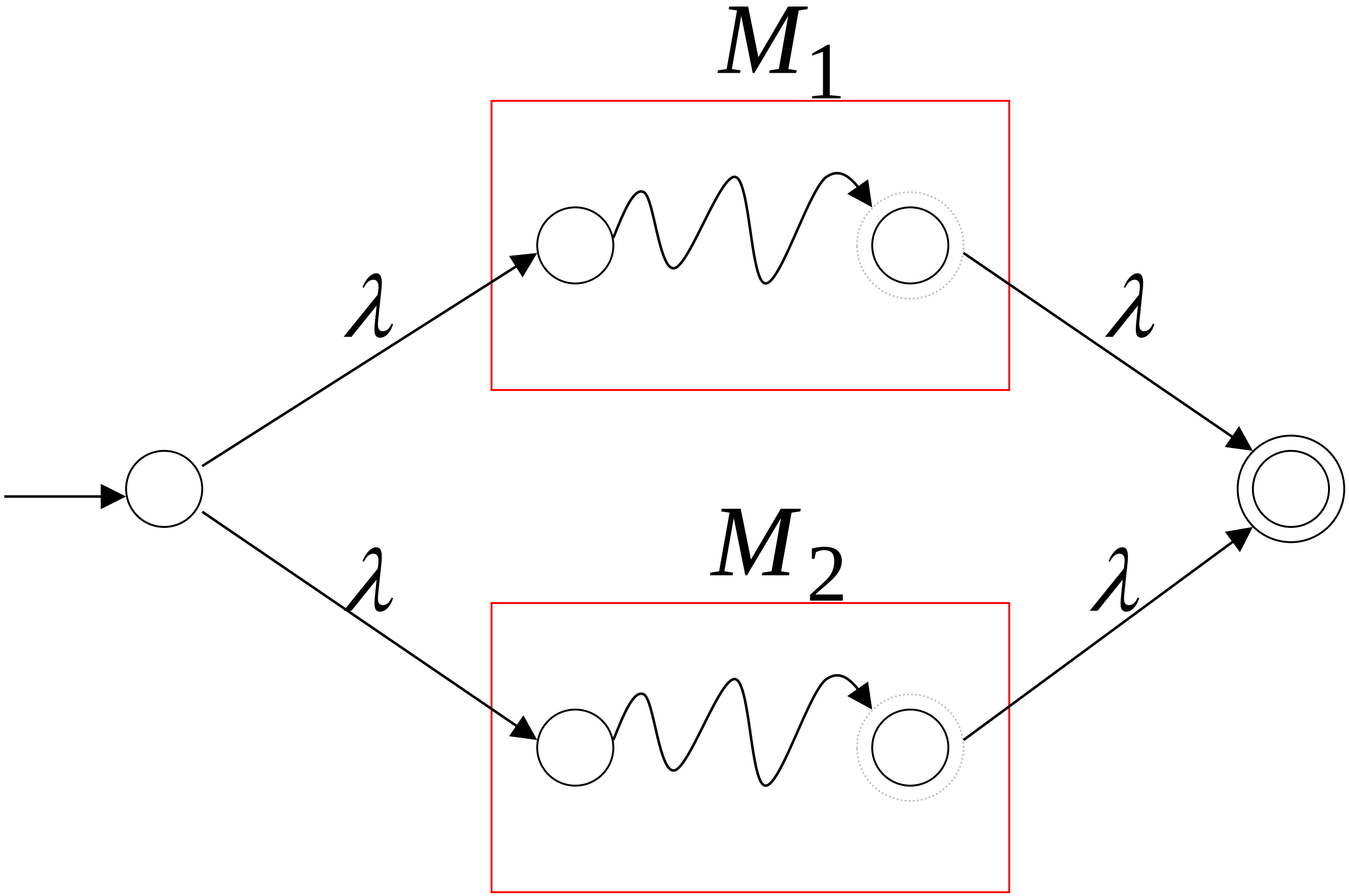
Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Regular languages are **closed** under these operations

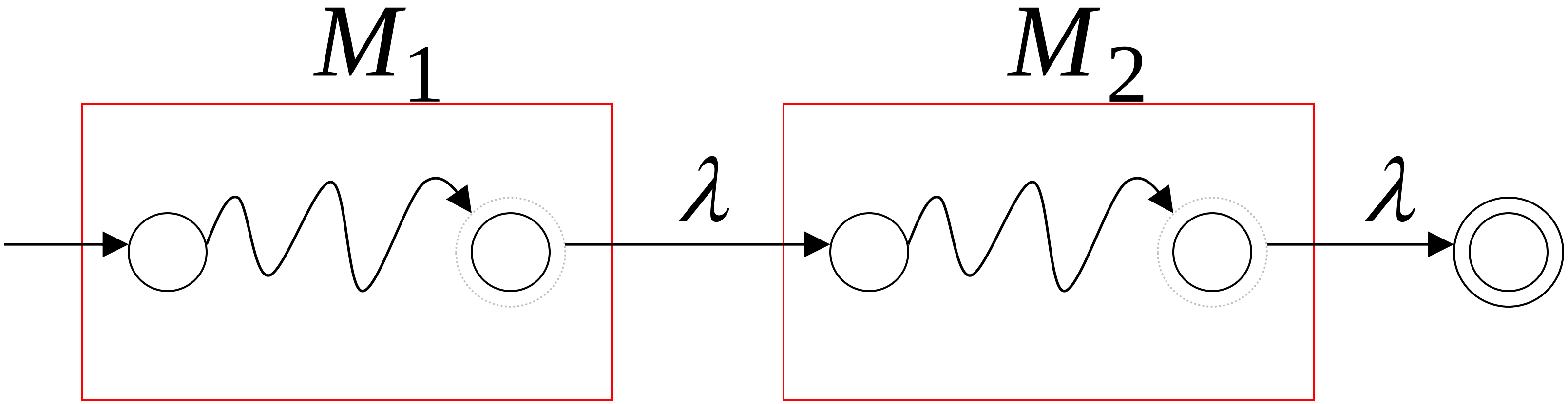
Union

NFA for $L_1 \cup L_2$



Concatenation

NFA for $L_1 L_2$

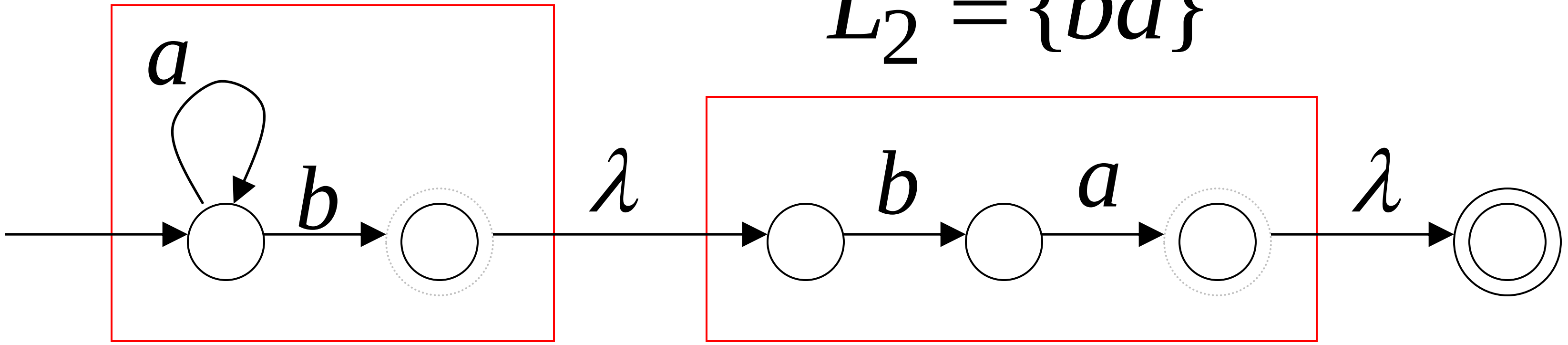


Example

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

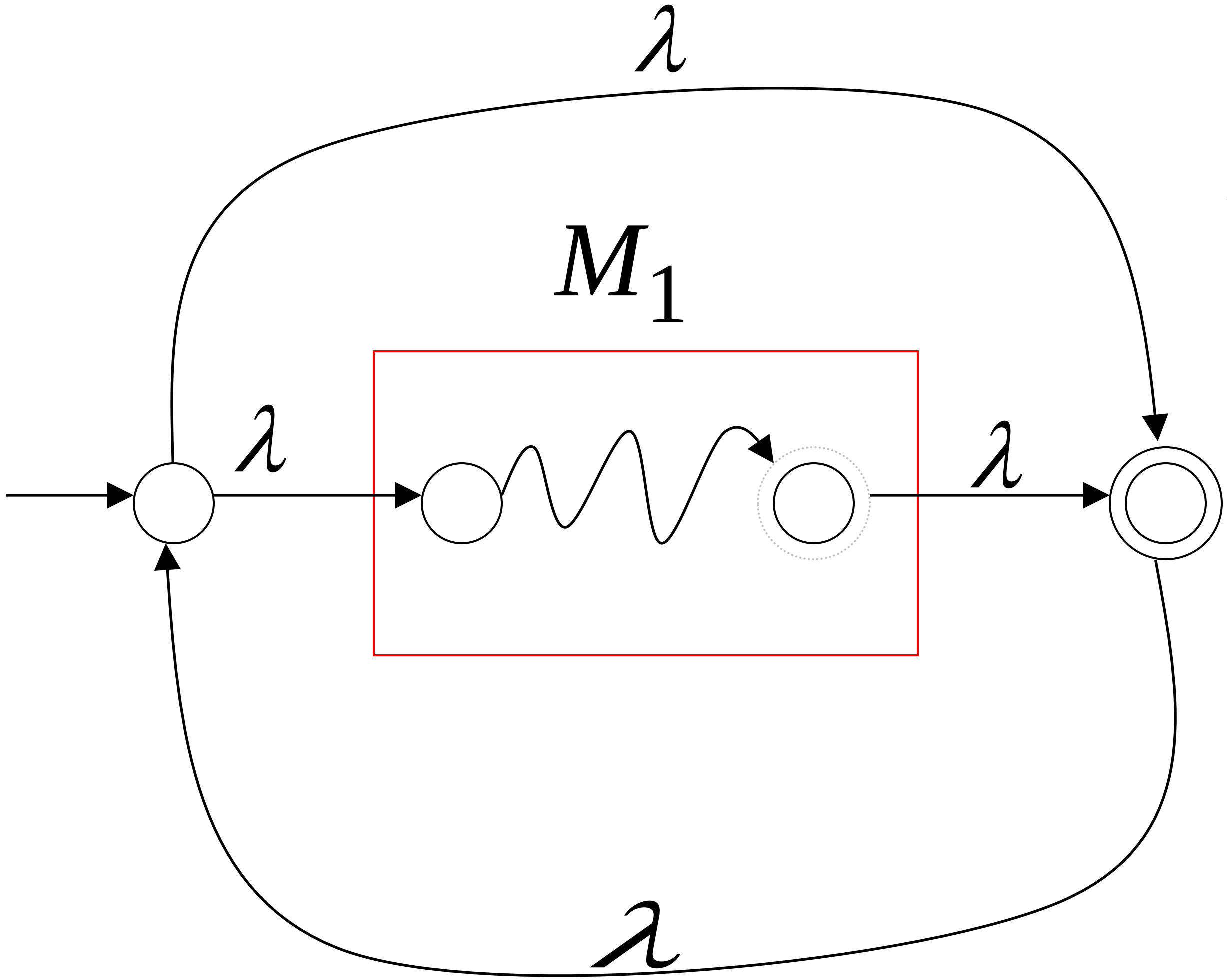
$L_1 = \{a^n b\}$

$L_2 = \{ba\}$



Star Operation

NFA for $L_1{}^*$



$\lambda \in L_1{}^*$

Reverse

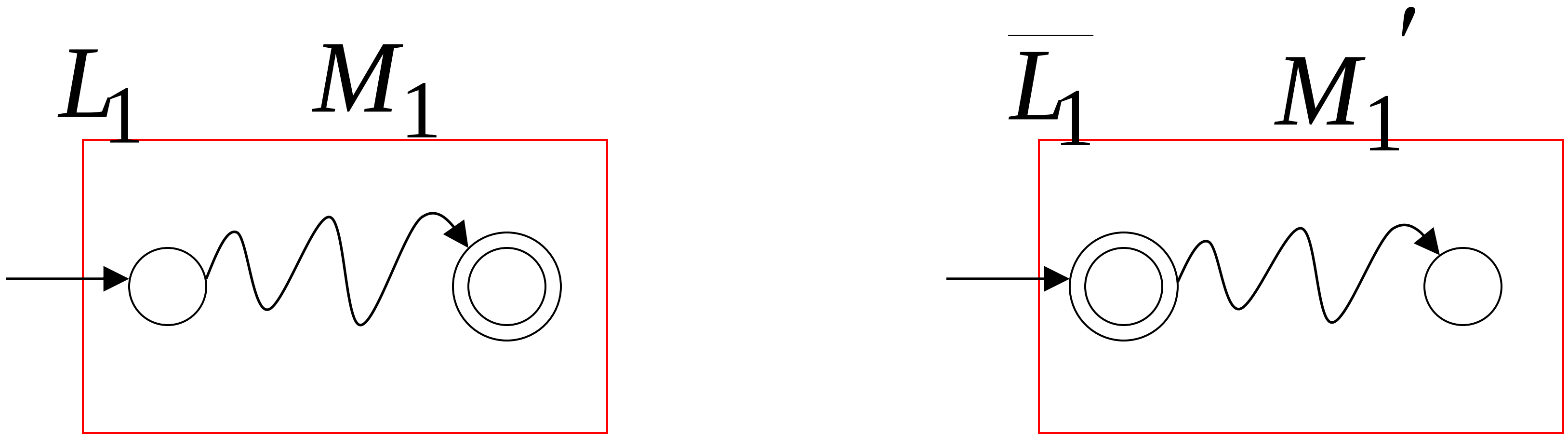
NFA for L_1^R

L_1 M_1 M_1'

1. Reverse all transitions

2. Make initial state final state
   and vice versa

Complement



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

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

Intersection

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

$L_1\,,\ L_2$ regular

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

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

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

$\Rightarrow$ $L_1 \cap L_2$ regular

Example

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

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

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

regular

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

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

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

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

Example

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

$L(r)$ = { all strings with at least two consecutive 0 }

Example

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

$L(r)$ = { all strings without two consecutive 0 }

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2

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

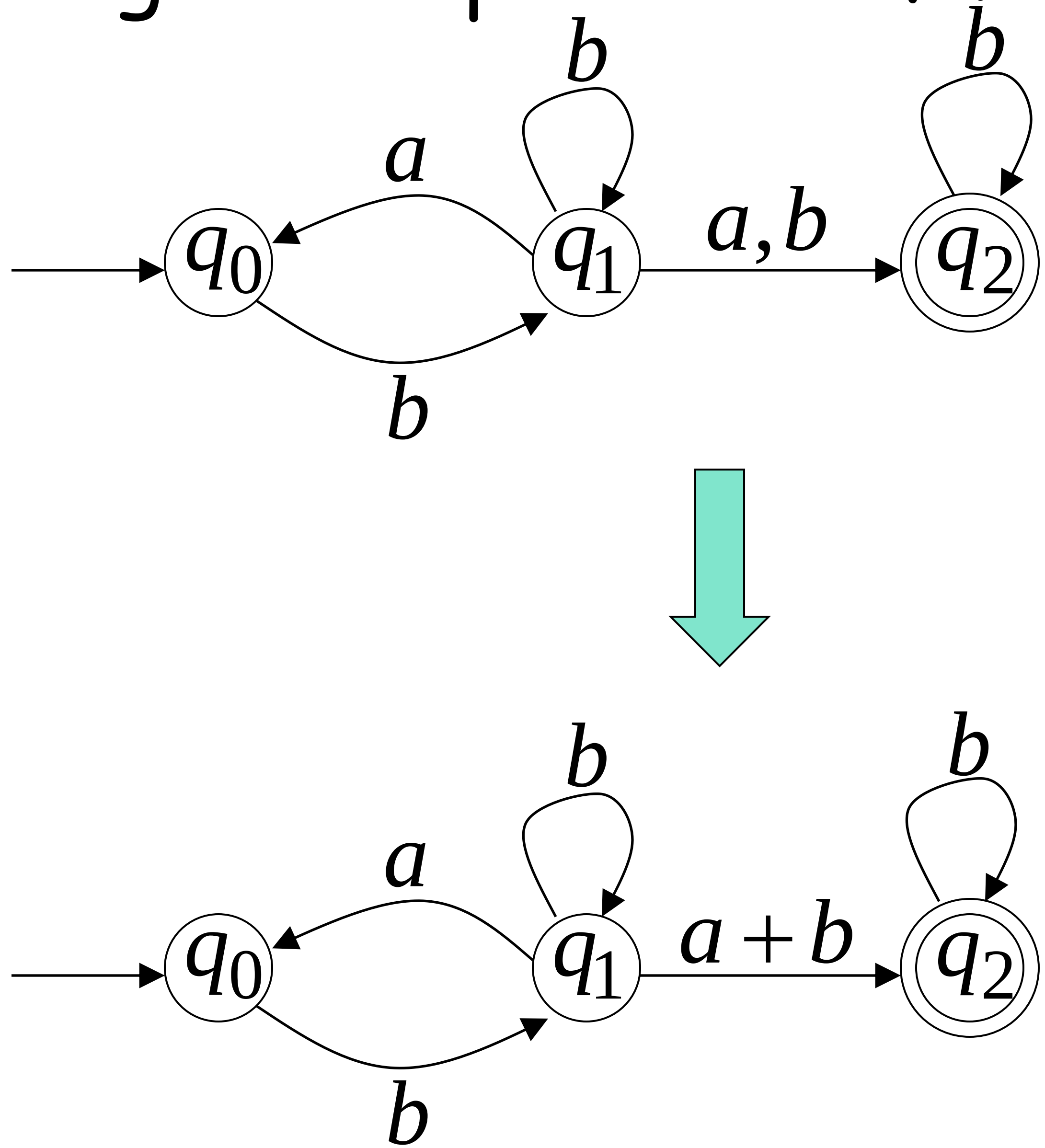
Regular Expressions and Regular Languages

Theorem

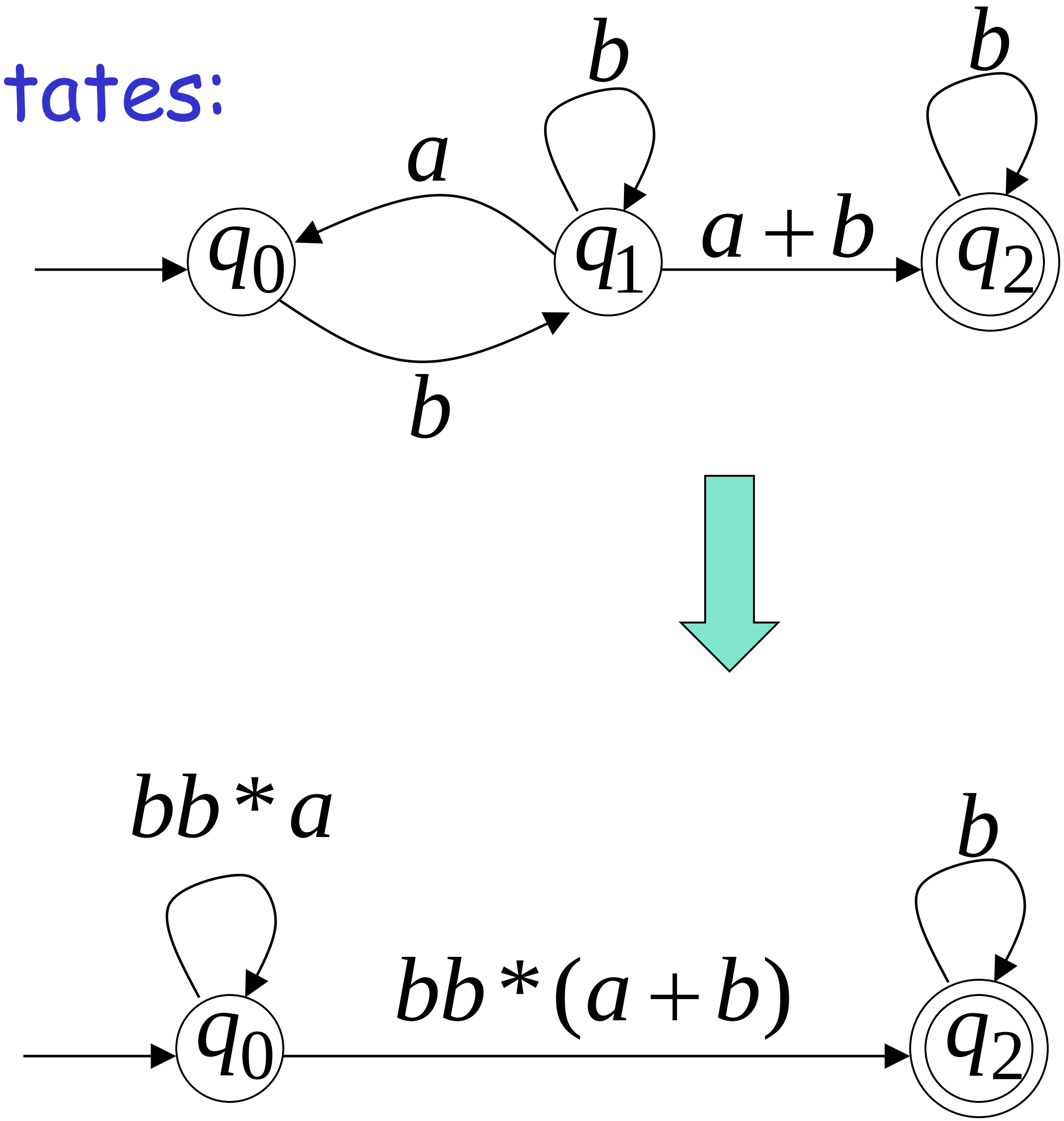
$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} = \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

See proof in the text book

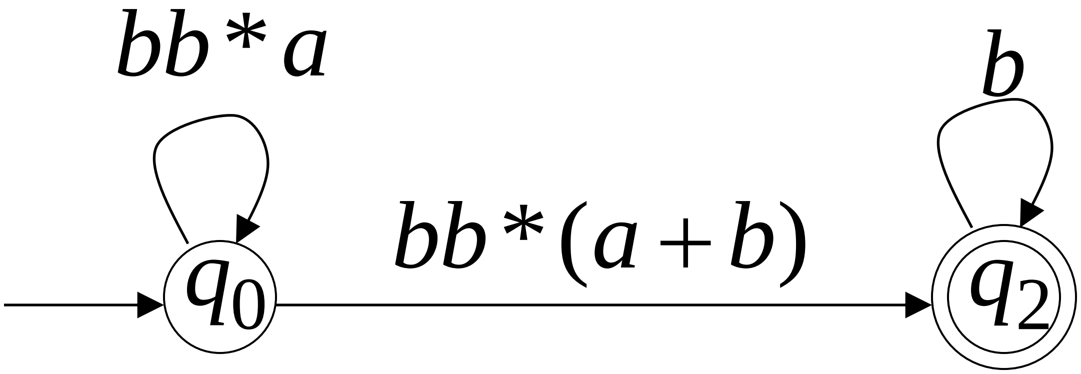
Finding the regular expression of FA



Reducing the states:



Resulting Regular Expression:

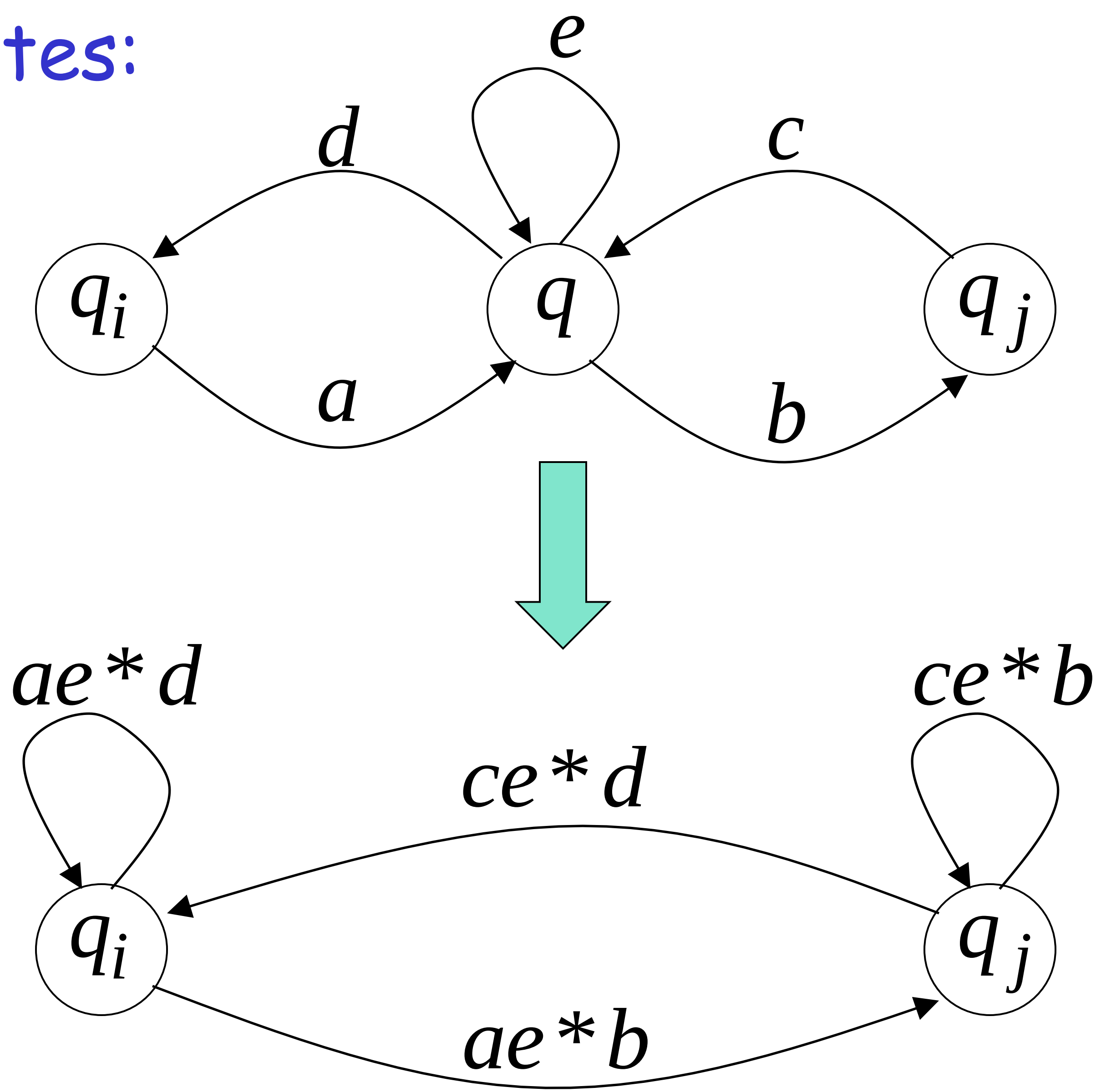


$$r = (bb^*a)^*bb^*(a+b)b^*$$

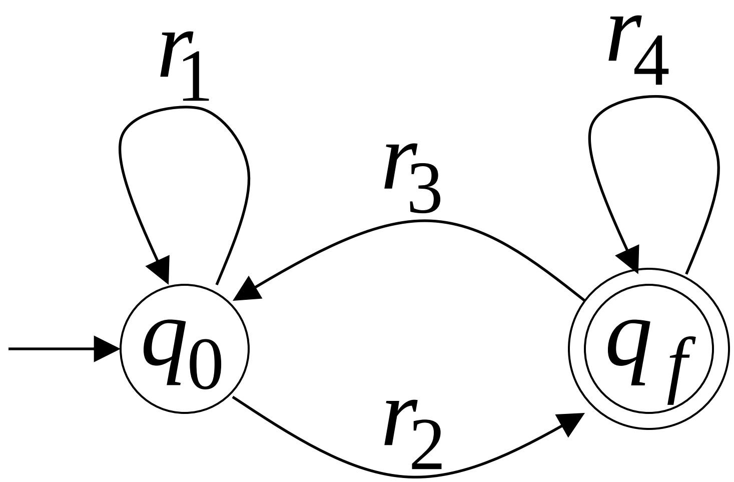
$$L(r) = L(M) = L$$

In General

Removing states:



The final transition graph:



The resulting regular expression:

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

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

Grammars

Grammars

Grammars express languages

Example: the English language

$$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle$$

$$\langle noun\_phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

$$\langle article \rangle \rightarrow a$$

$$\langle article \rangle \rightarrow the$$

$$\langle noun \rangle \rightarrow cat$$

$$\langle noun \rangle \rightarrow dog$$

$$\langle verb \rangle \rightarrow runs$$

$$\langle verb \rangle \rightarrow walks$$

Language of the grammar:

L = { "a cat runs",
"a cat walks",
"the cat runs",
"the cat walks",
"a dog runs",
"a dog walks",
"the dog runs",
"the dog walks" }

Notation

$$\overbrace{\begin{array}{l}\langle noun \rangle \rightarrow cat \\ \langle noun \rangle \rightarrow dog\end{array}}^{\text{Production Rules}}$$

Variable
(Nonterminal symbol)

Terminal symbol

Another Example

Grammar:
$$S \to aSb$$
$$S \to \lambda$$

Derivation of sentence ab:

$$S \Rightarrow aSb \Rightarrow ab$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

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

More Notation

Grammar $G=\left(V,T,S,P\right)$

$V:$ Set of variables

$T:$ Set of terminal symbols

$S:$ Start variable

$P:$ Set of Production rules

Example

Grammar G :

$$S \to aSb$$

$$S \to \lambda$$

$$G = (V, T, S, P)$$

$$V = \{S\} \qquad T = \{a, b\}$$

$$P = \{S \to aSb, \ S \to \lambda\}$$

More Notation

Sentential Form:

A sentence that contains variables and terminals

Example:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Sentential Forms   sentence

We write: $S \overset{*}{\Rightarrow} aaabbb$

Instead of:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Language of a Grammar

For a grammar G
with start variable S :

$$L(G) = \{w : \quad S \overset{*}{\Rightarrow} w\}$$

String of terminals

Example

For grammar G :
$$S \to Ab$$
$$A \to aAb$$
$$A \to \lambda$$

$$L(G) = \{a^n b^n b: \quad n \geq 0\}$$

Since: $S \overset{*}{\Rightarrow} a^n b^n b$

A Convenient Notation

$$A \to aAb$$
$$A \to \lambda$$

$$\Longrightarrow$$

$$A \to aAb \mid \lambda$$

$$\langle article \rangle \to a$$
$$\langle article \rangle \to the$$

$$\Longrightarrow$$

$$\langle article \rangle \to a \mid the$$

Linear Grammars

Grammars with
at most one variable at the right side
of a production

Examples:

$$S \to aSb$$
$$S \to \lambda$$

$$S \to Ab$$
$$A \to aAb$$
$$A \to \lambda$$

Another Linear Grammar

Grammar G :

$$S \to A$$

$$A \to aB \mid \lambda$$

$$B \to Ab$$

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

A Non-Linear Grammar

Grammar G :

$$S \to SS$$

$$S \to \lambda$$

$$S \to aSb$$

$$S \to bSa$$

$$L(G) = \{w: \quad n_a(w) = n_b(w)\}$$

Number of a in string w

Right-Linear Grammars

All productions have form: $A \rightarrow xB$

or

$A \rightarrow x$

string of terminals

Example: $S \rightarrow abS$

$S \rightarrow a$

Left-Linear Grammars

All productions have form:

$$A \to Bx$$

or

$$A \to x$$

string of terminals

Example:

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

Regular Grammars

A regular grammar is any right-linear or left-linear grammar

Examples:

G_1

$$S \to abS$$

$$S \to a$$

$$L(G_1) = (ab)^* a$$

G_2

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

$$L(G_2) = aab(ab)^*$$

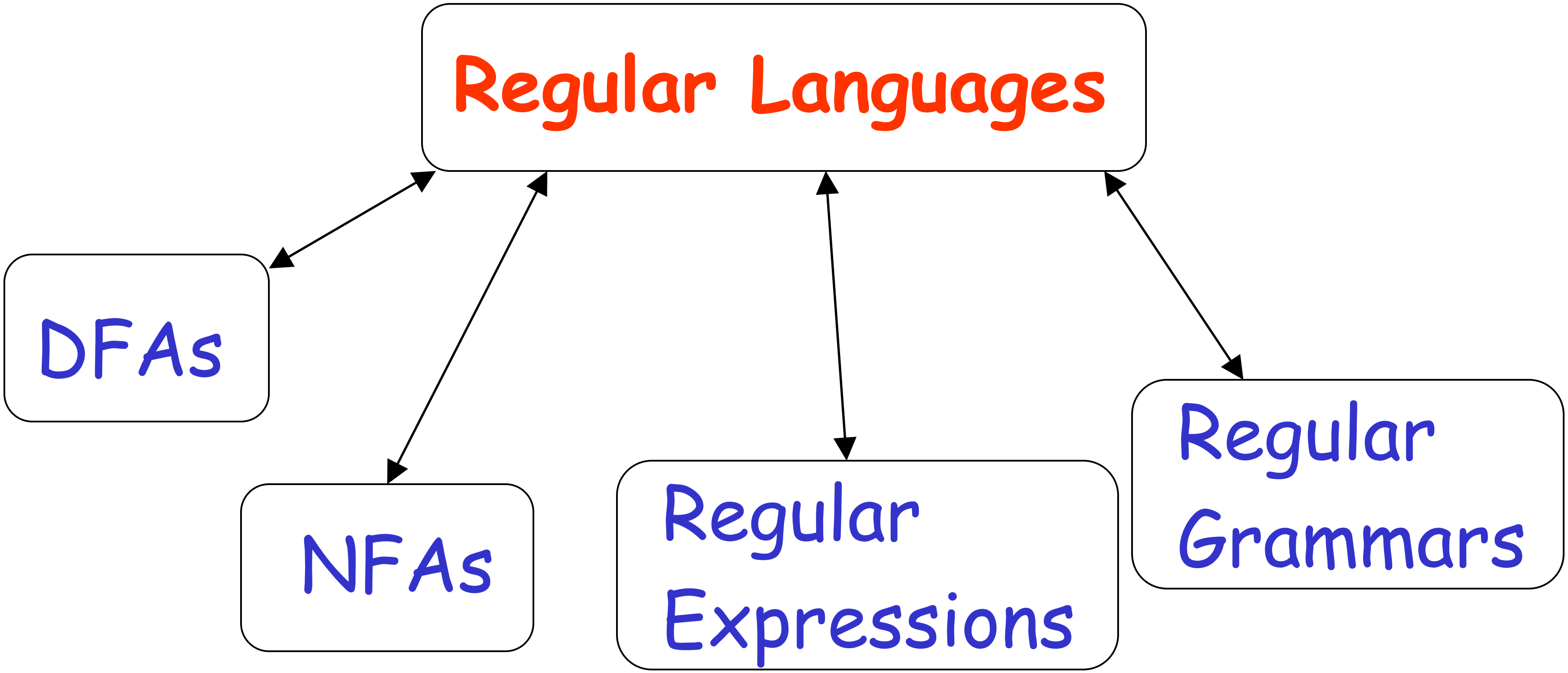
Theorem

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

"Regular grammars generate Regular Languages"

See proof in the text book

Standard Representations of Regular Languages



Regular Language can be represented in a standard representation.

Non-regular languages

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

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

Regular languages

a^*b

b^*c+a

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

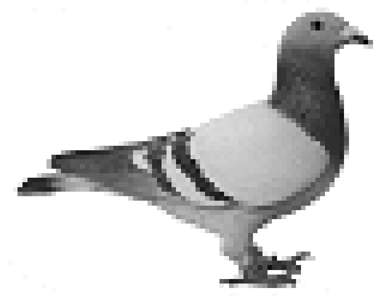
$etc...$

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

Prove that there is no DFA that accepts L

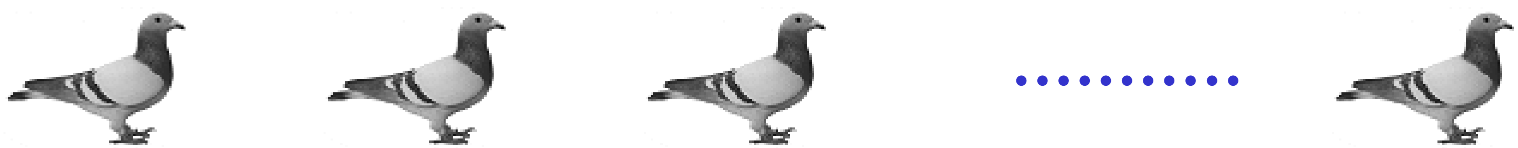
Problem: this is not easy to prove

Solution: the Pumping Lemma !!!

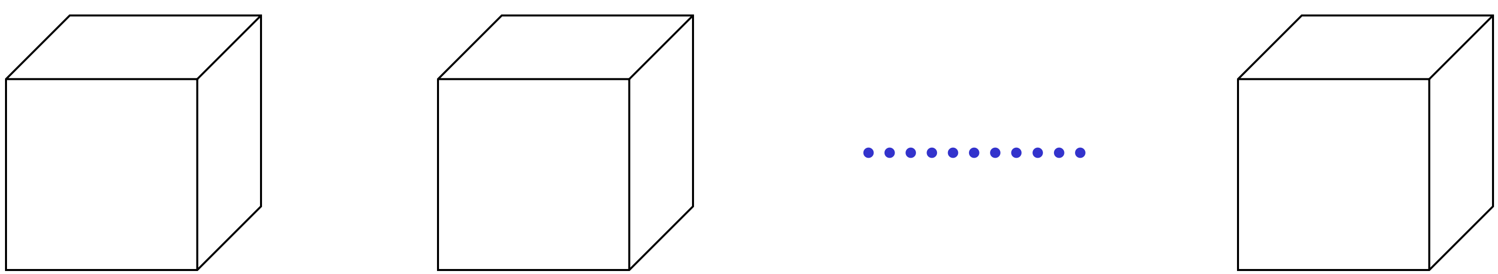


The Pigeonhole Principle

n pigeons



m pigeonholes $\quad n > m$



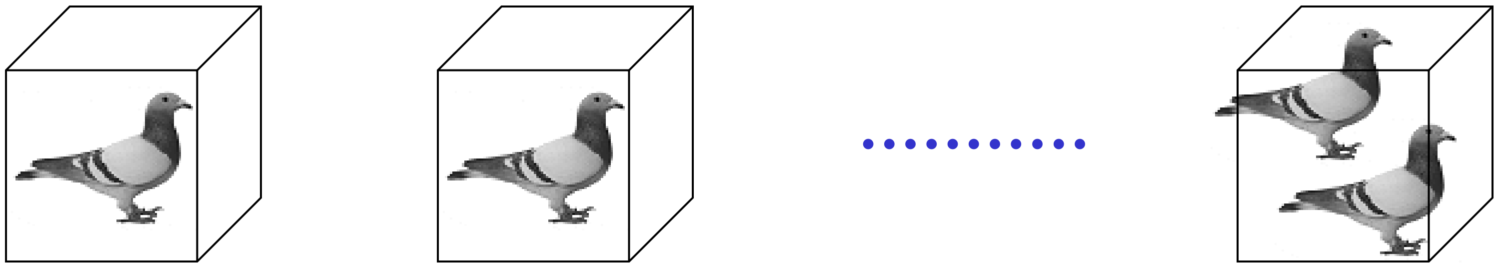
The Pigeonhole Principle

n pigeons

m pigeonholes

$$n > m$$

There is a pigeonhole with at least 2 pigeons

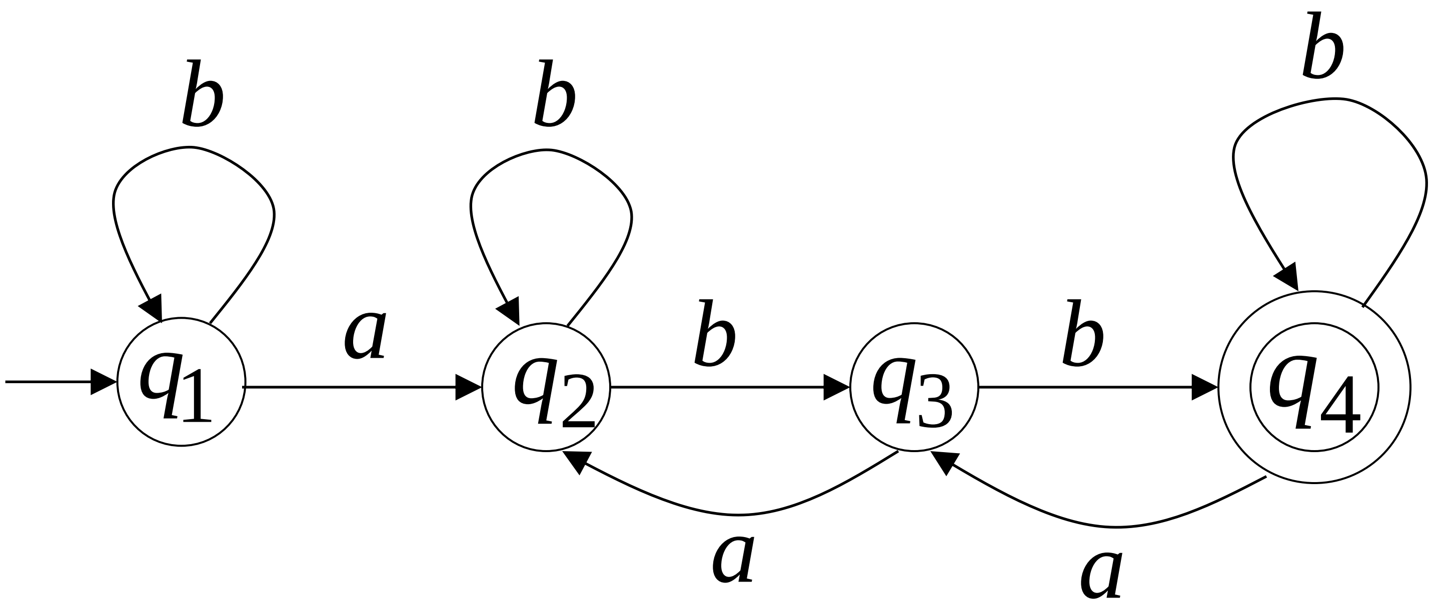


The Pigeonhole Principle

and

DFAs

DFA with 4 states

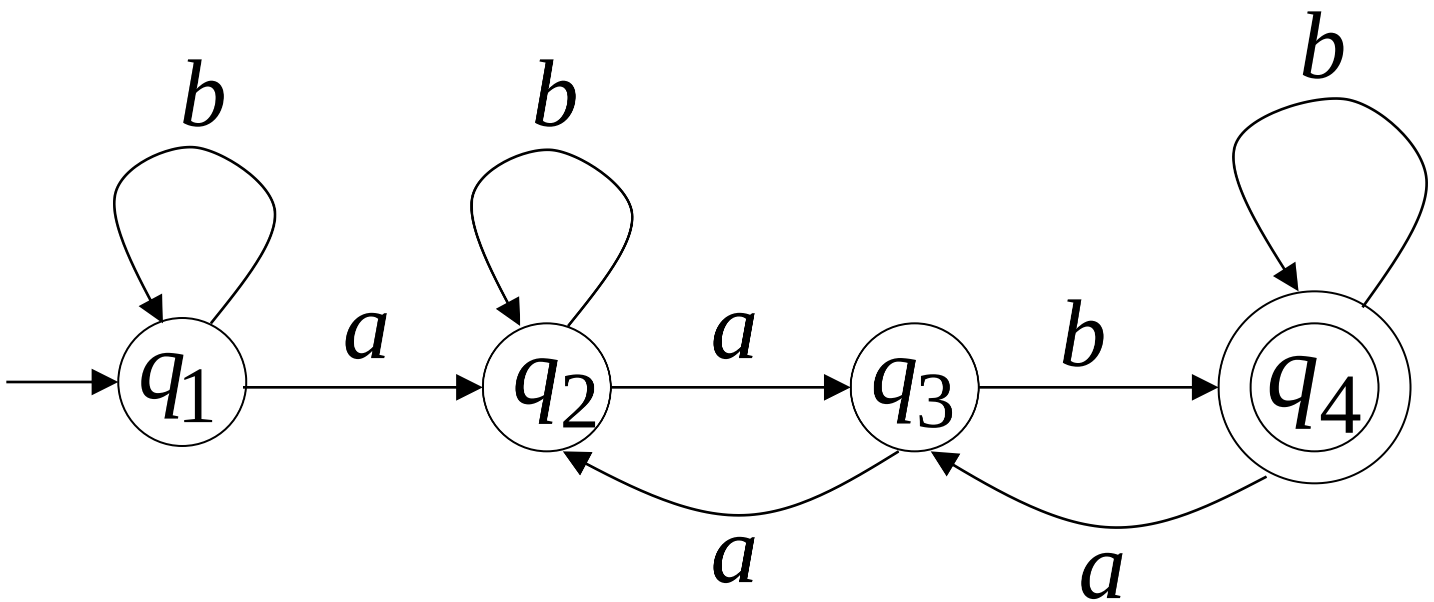


In walks of strings: a

aa

aab

no state is repeated



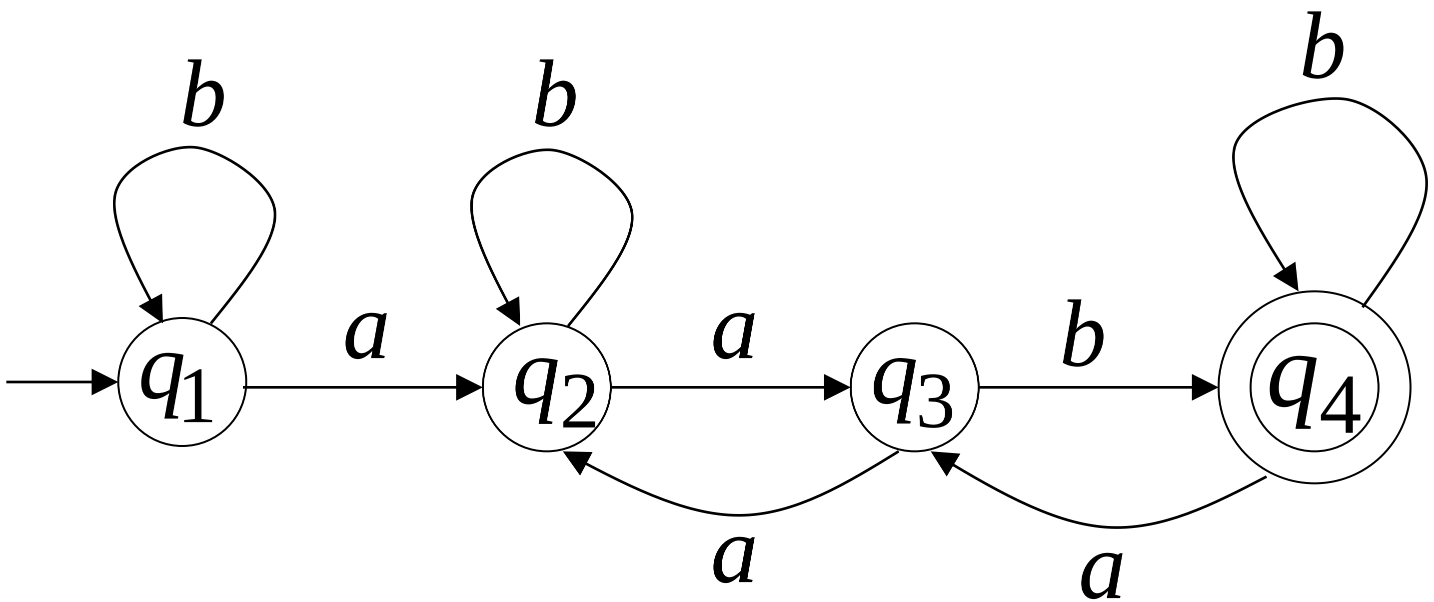
In walks of strings: $aabb$

$bbaa$

$abbabb$

$abbbabbabb...$

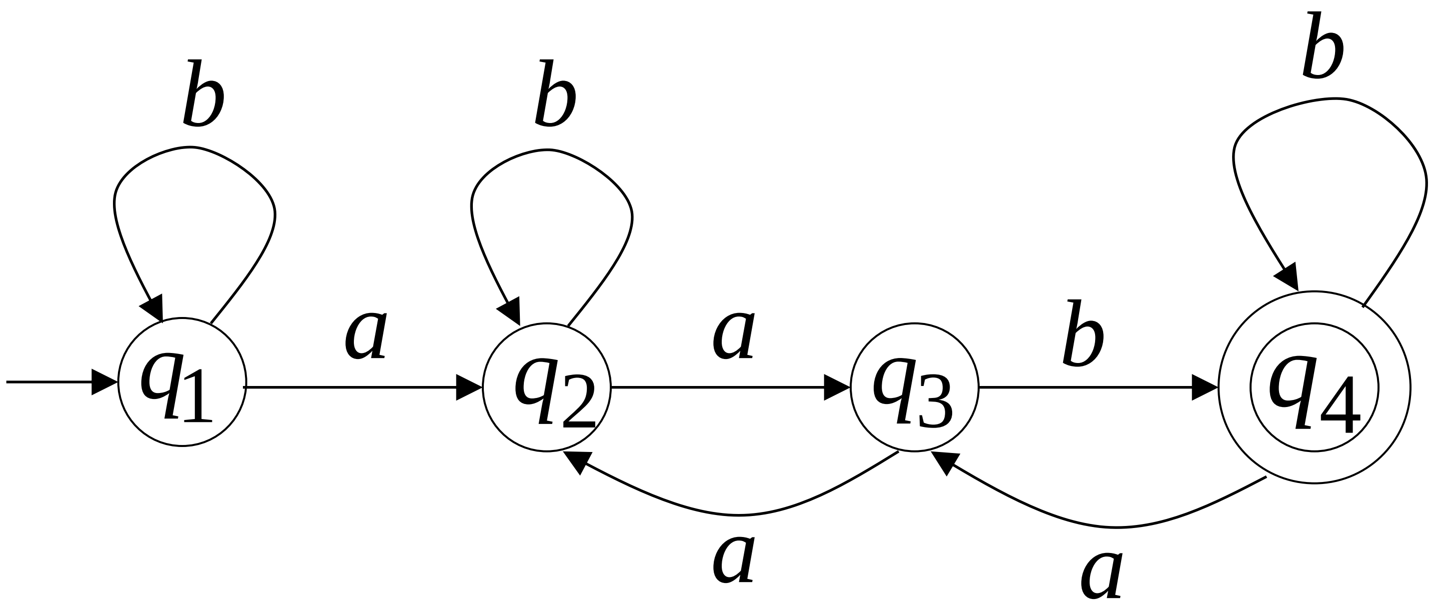
a state is repeated



If string w has length $|w| \geq 4$:

Then the transitions of string w are more than the states of the DFA

Thus, a state must be repeated

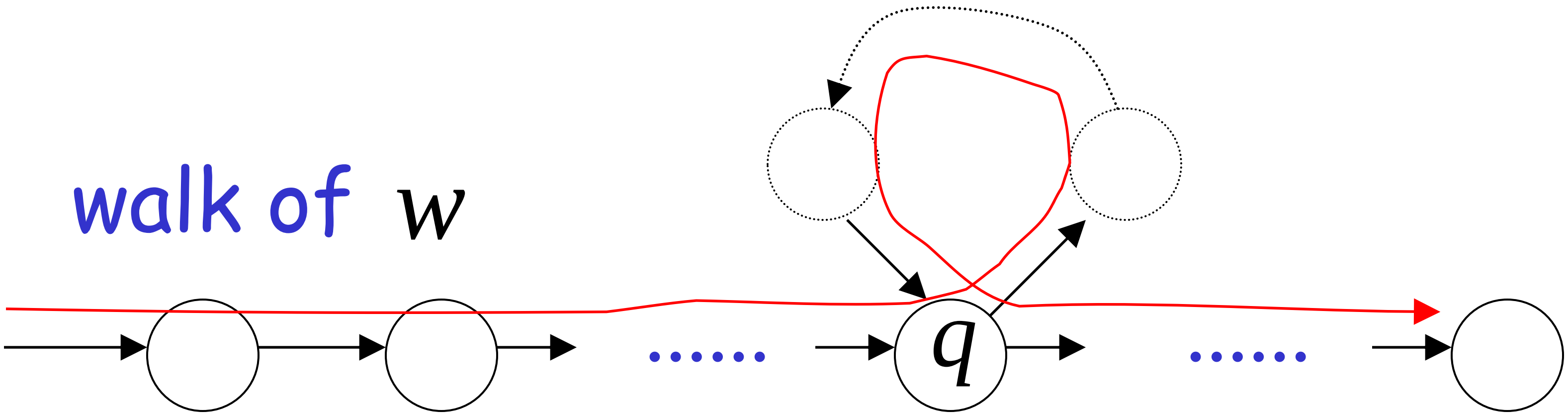


In general, for any DFA:

String w has length $\geq$ number of states

A state q must be repeated in the walk of w

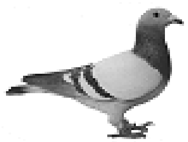
walk of w



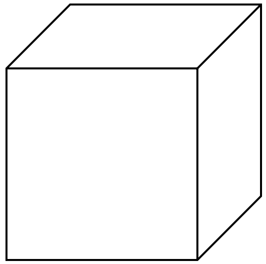
Repeated state

In other words for a string w:

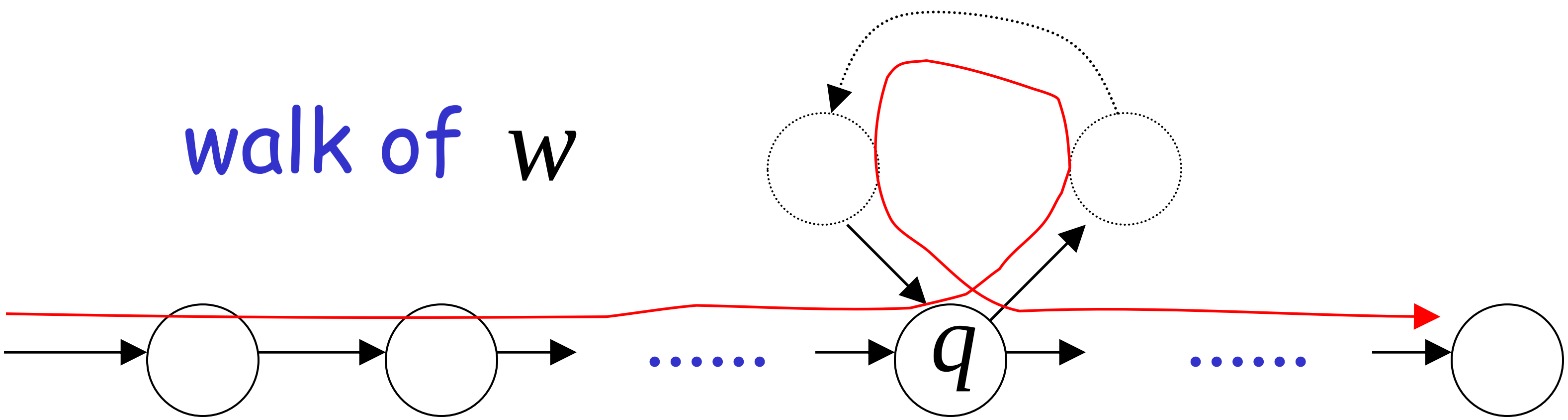
$\xrightarrow{a}$ transitions are pigeons



q states are pigeonholes



walk of w

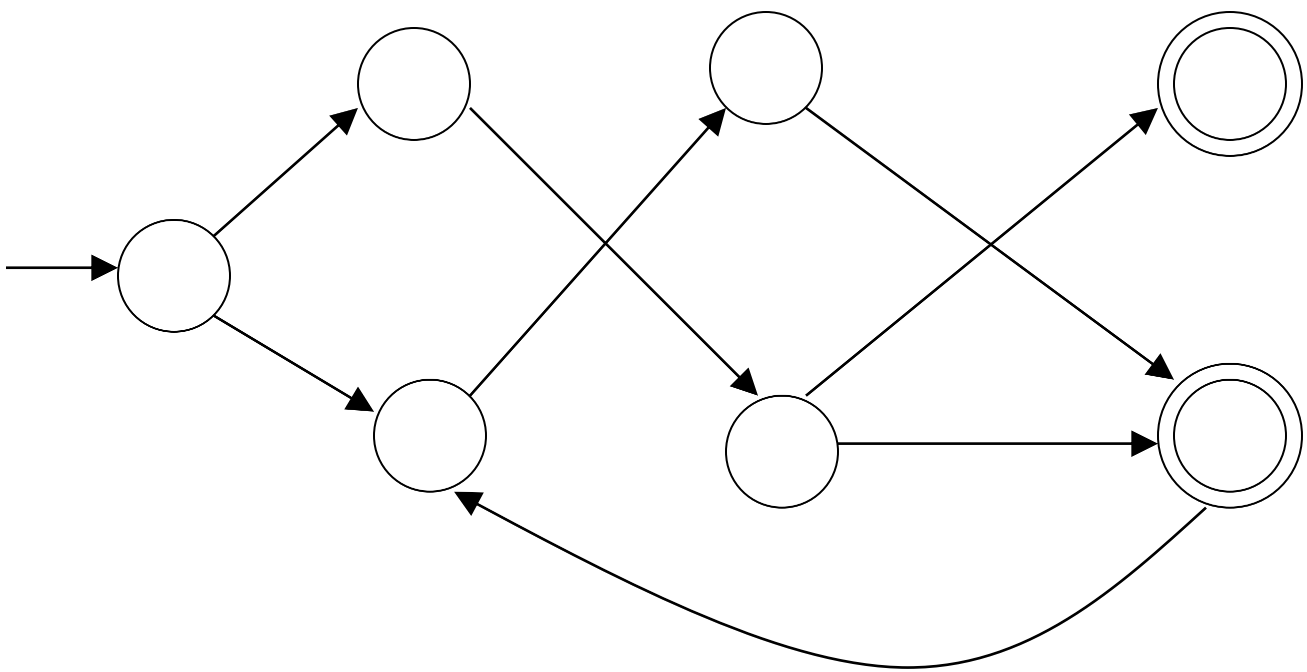


Repeated state

The Pumping Lemma

Take an infinite regular language L

There exists a DFA that accepts L

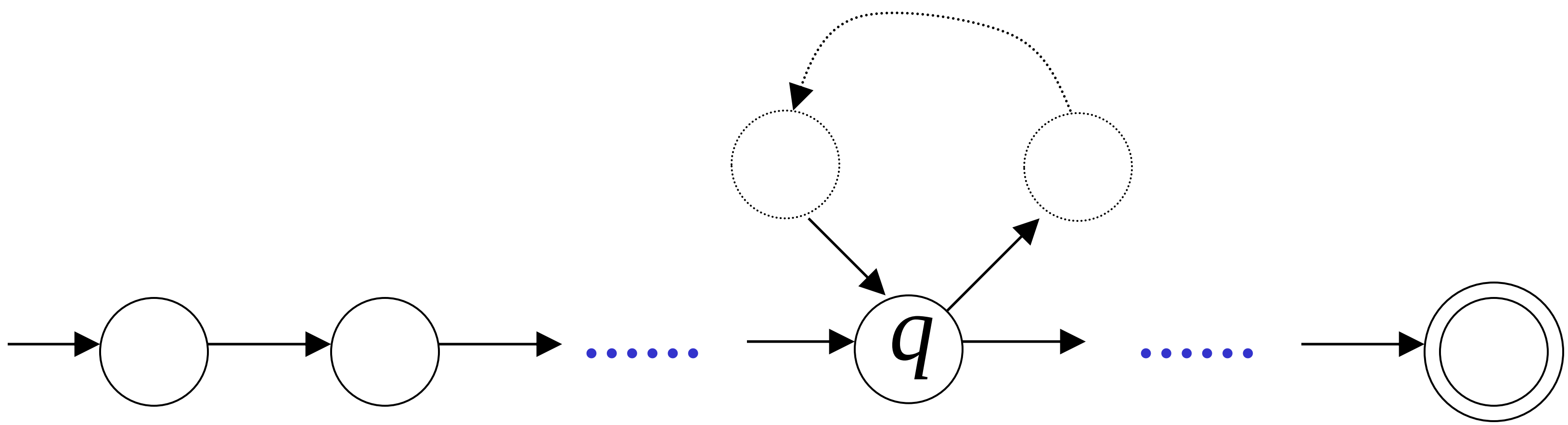


m states

If string w has length $|w| \geq m$ (number of states of DFA)

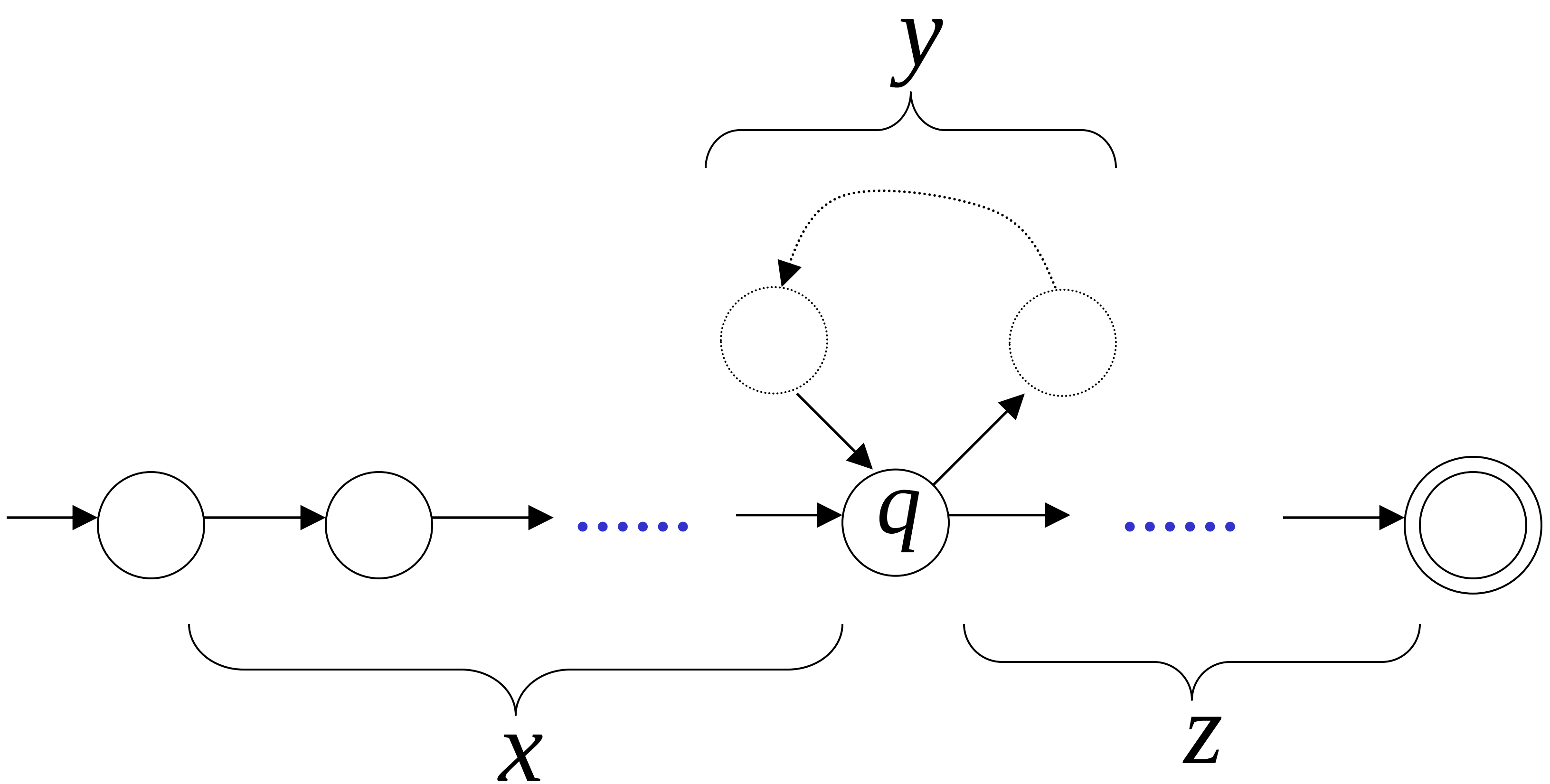
then, from the pigeonhole principle:

a state is repeated in the walk w



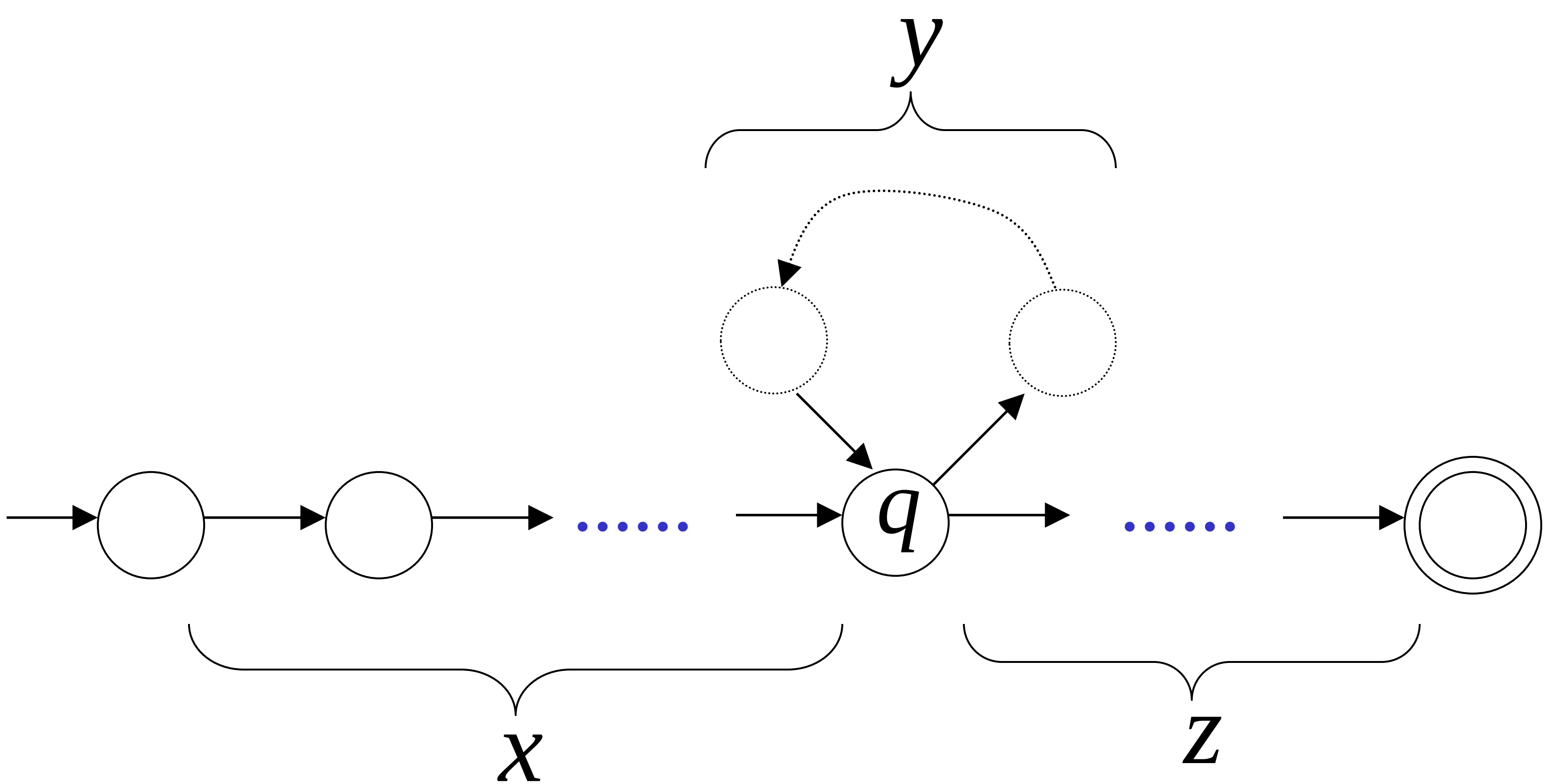
walk w

Write $w = x\ y\ z$



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

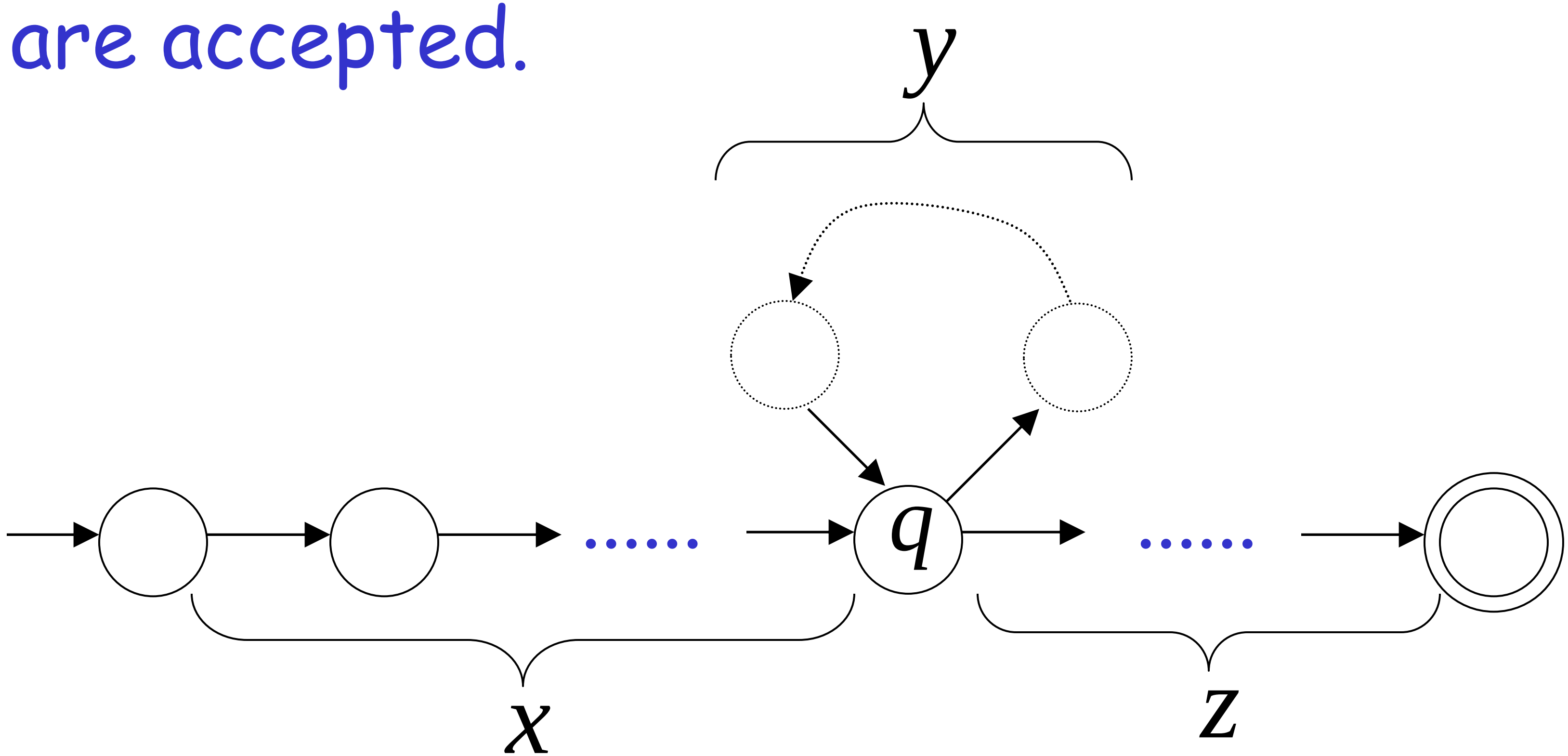
length $|y| \geq 1$



Observation:

The string $xz, \; xyz, \; xyyz, \; xyyyz, \; \ldots$

are accepted.



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

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

The Pumping Lemma:

- Given a infinite regular language L
- there exists an integer m
- 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: $x\,y^i\,z \in L \qquad i = 0, 1, 2, \ldots$

Applications

of

the Pumping Lemma

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 integer in the Pumping Lemma

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

length $|w| \geq m$

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

Write: $a^m b^m = x\, y\, z$

From the Pumping Lemma

it must be that length $|x\, y| \le m, \quad |y| \ge 1$

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

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

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

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

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

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

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

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

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

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

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

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

$$\Downarrow$$

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

CONTRADICTION!!!

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

Conclusion: L is not a regular language

The Pumping Lemma:

Reminder

- Given a infinite regular language L
- there exists an integer m
- 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: $x\, y^i\, z \in L \qquad i = 0, 1, 2, \ldots$

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 integer in the Pumping Lemma

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

length $|w| \geq m$

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

Write $a^m b^m b^m a^m = x\ y\ z$

From the Pumping Lemma

it must be that length $|\,x\ y\,| \le m, \quad |\,y\,| \ge 1$

$$xyz = \overbrace{\underbrace{a...a}_{x}\underbrace{a...a}_{y}\underbrace{...a}_{}}^{m}\underbrace{\overbrace{b...b}^{m}\overbrace{b...b}^{m}\overbrace{a...a}^{m}}_{}$$

$$\text{with } z = ...a\,b...b\,b...b\,a...a$$

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

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

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

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

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

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

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

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

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

$$\Downarrow$$

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

CONTRADICTION!!!

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

Conclusion: L is not a regular language

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 integer in the Pumping Lemma

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

length $|w| \geq m$

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

Write $a^m b^m c^{2m} = x\ y\ z$

From the Pumping Lemma

it must be that length $|x\ y| \leq m, \quad |y| \geq 1$

$$xyz = \overbrace{\underbrace{a...a}_{x}\underbrace{a...a}_{y}a...a}^{m}\overbrace{b...b}^{m}\overbrace{c...c}^{2m}$$

$$\underbrace{a...ab...bc...cc...c}_{z}$$

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

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

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

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

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

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

From the Pumping Lemma: $xz \in L$

$$xz = \underbrace{\overbrace{a...a}^{x}\overbrace{a...a}^{m-k}}\underbrace{\overbrace{b...b}^{m}\overbrace{c...cc...c}^{2m}}_{z} \in L$$

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

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

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

$$\Downarrow$$

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

CONTRADICTION!!!

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

Conclusion: L is not a regular language