2016

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1. Closure Properties of Regular Languages
2. Elementary Questions about Regular Languages
3. Identifying Nonregular 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: $L_1 \cap L_2$

Difference: $L_1 - L_2$

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: $\overline{L_1}$

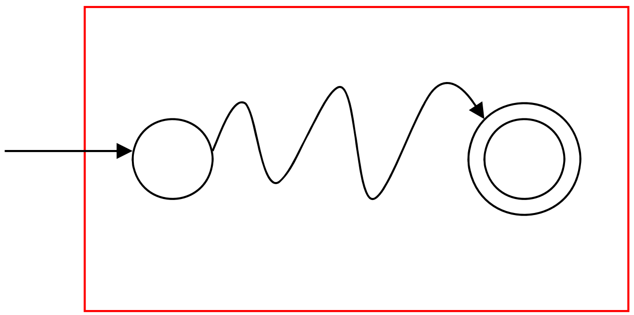
Intersection: $L_1 \cap L_2$

Difference: $L_1 - L_2$

Regular language L_1

$$L(M_1) = L_1$$

NFA M_1

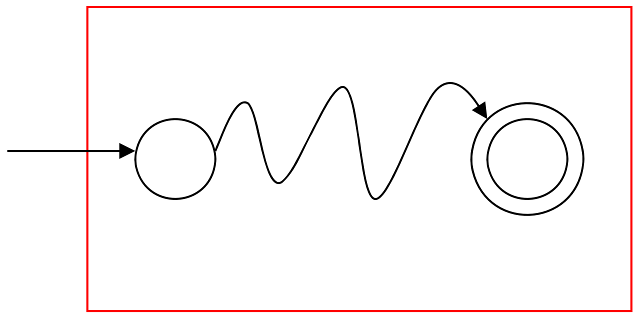


Single final state

Regular language L_2

$$L(M_2) = L_2$$

NFA M_2



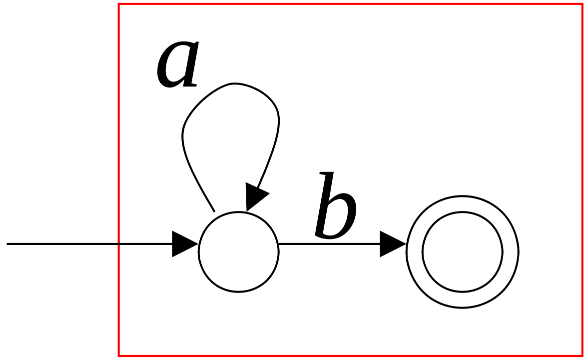
Single final state

Example

$$n \geq 0$$

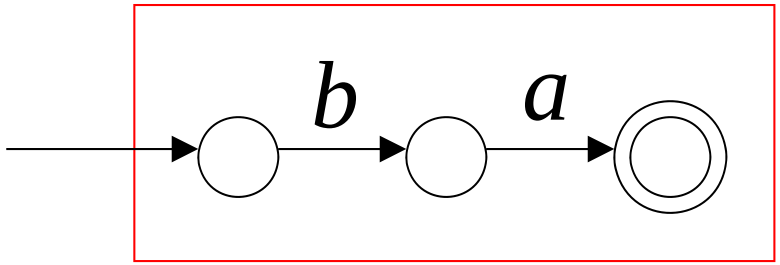
$$L_1 = \{a^n b\}$$

M_1



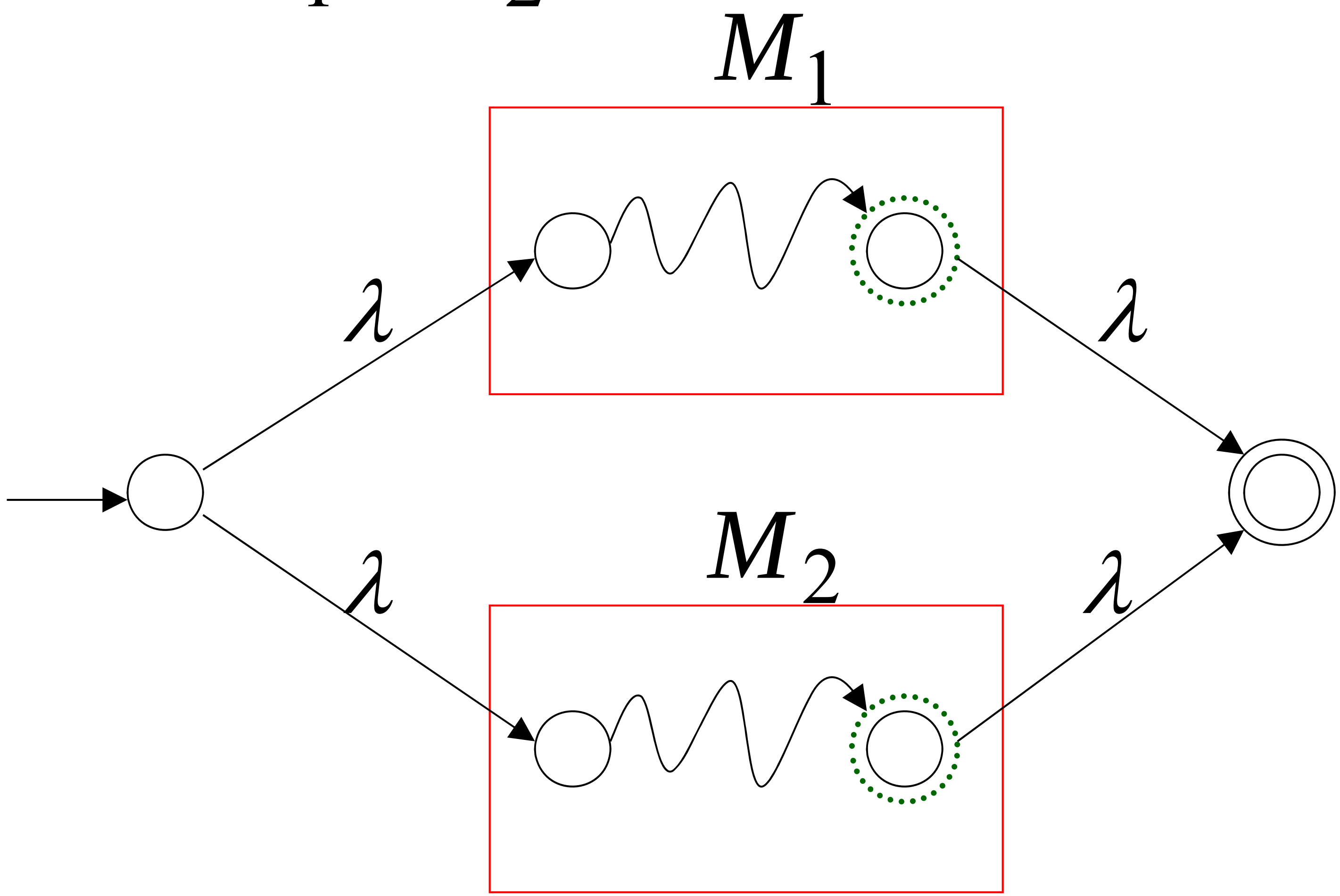
$$L_2 = \{ba\}$$

M_2



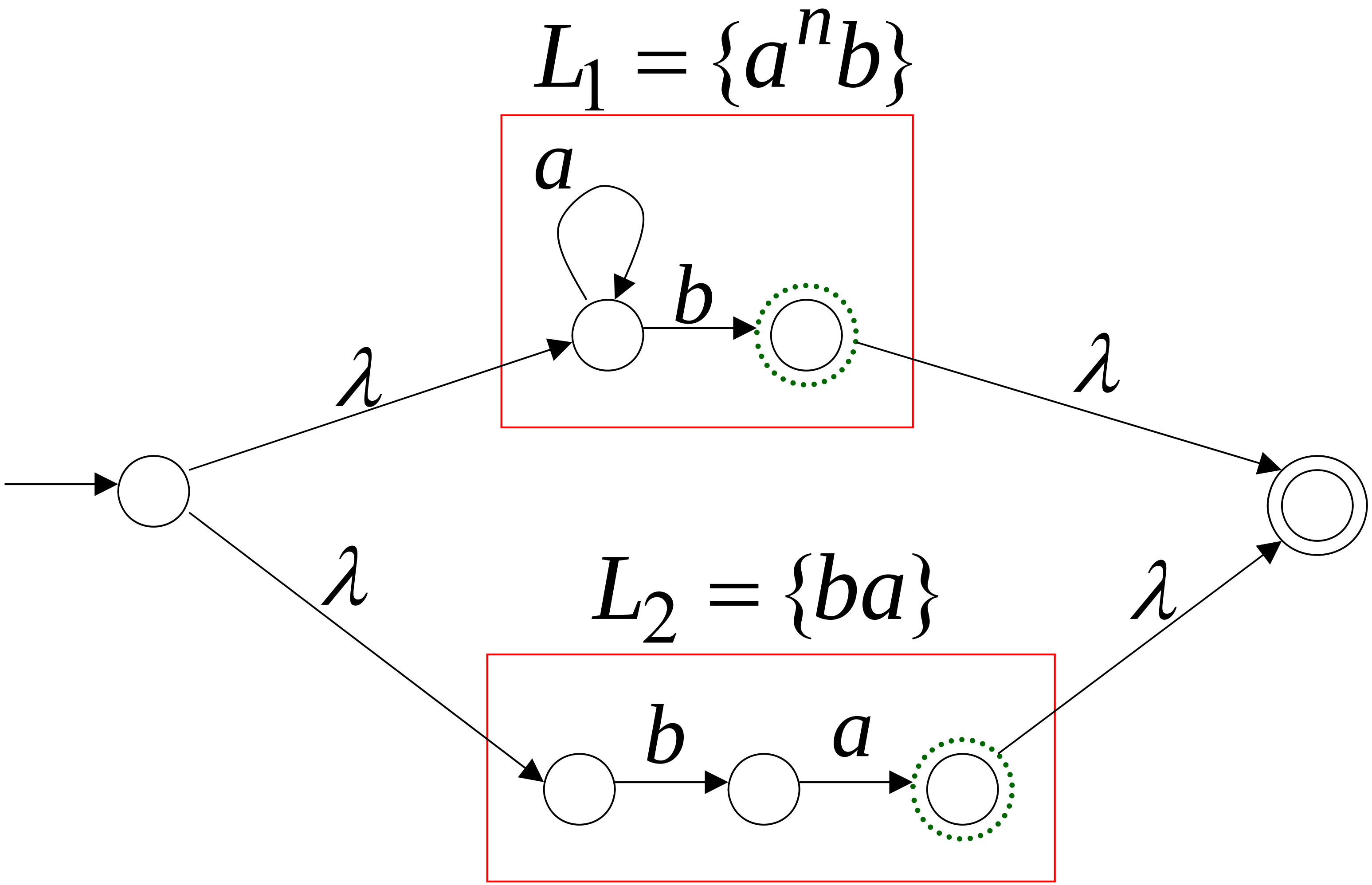
Union

NFA for $L_1 \cup L_2$



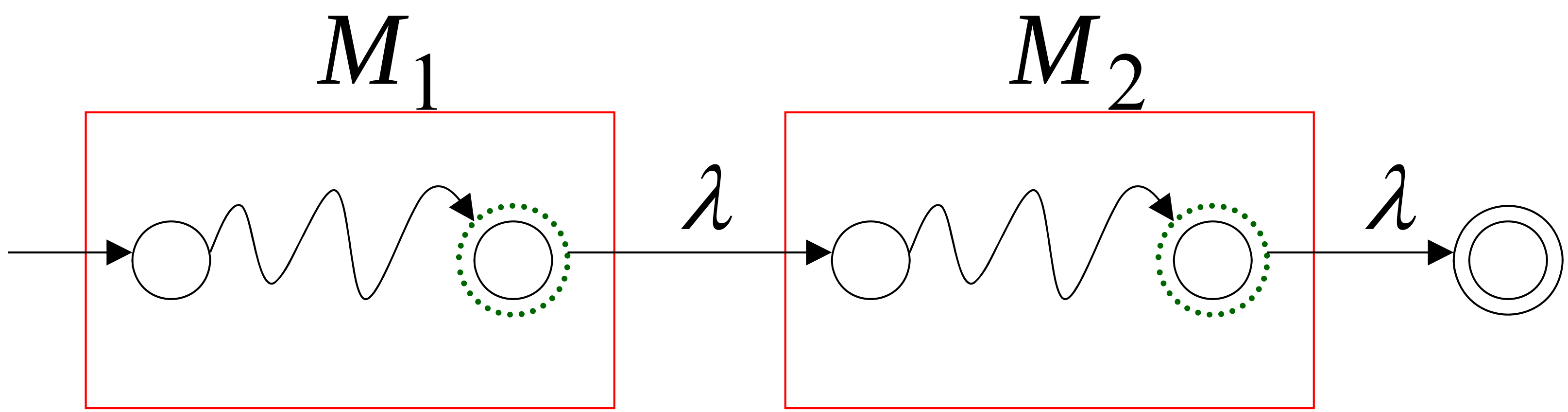
Example

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



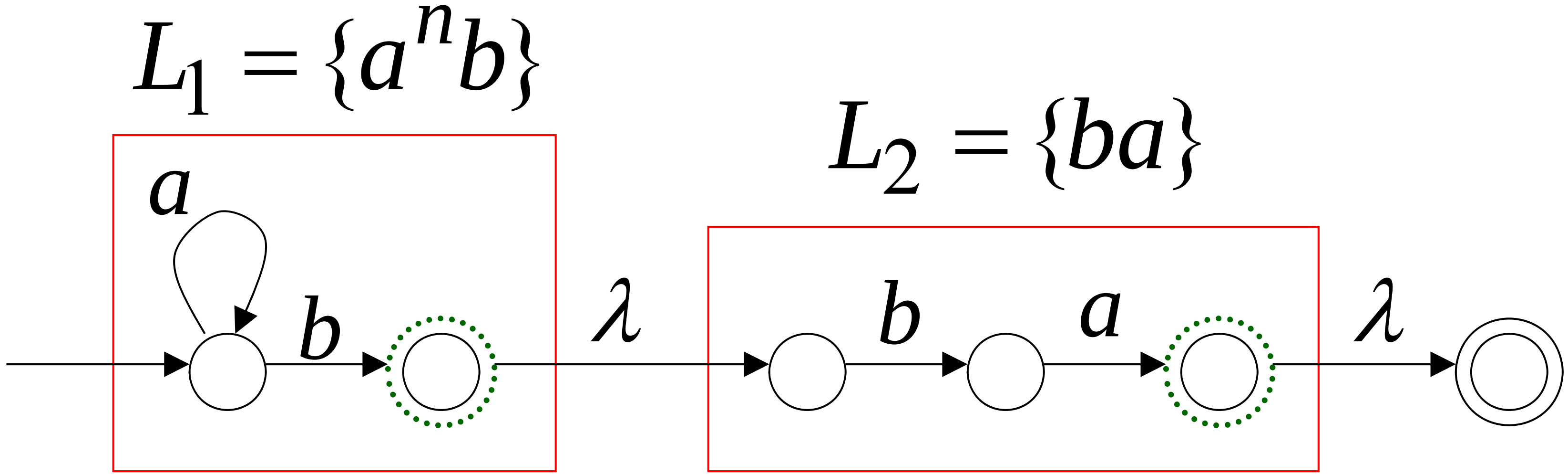
Concatenation

NFA for $L_1 L_2$



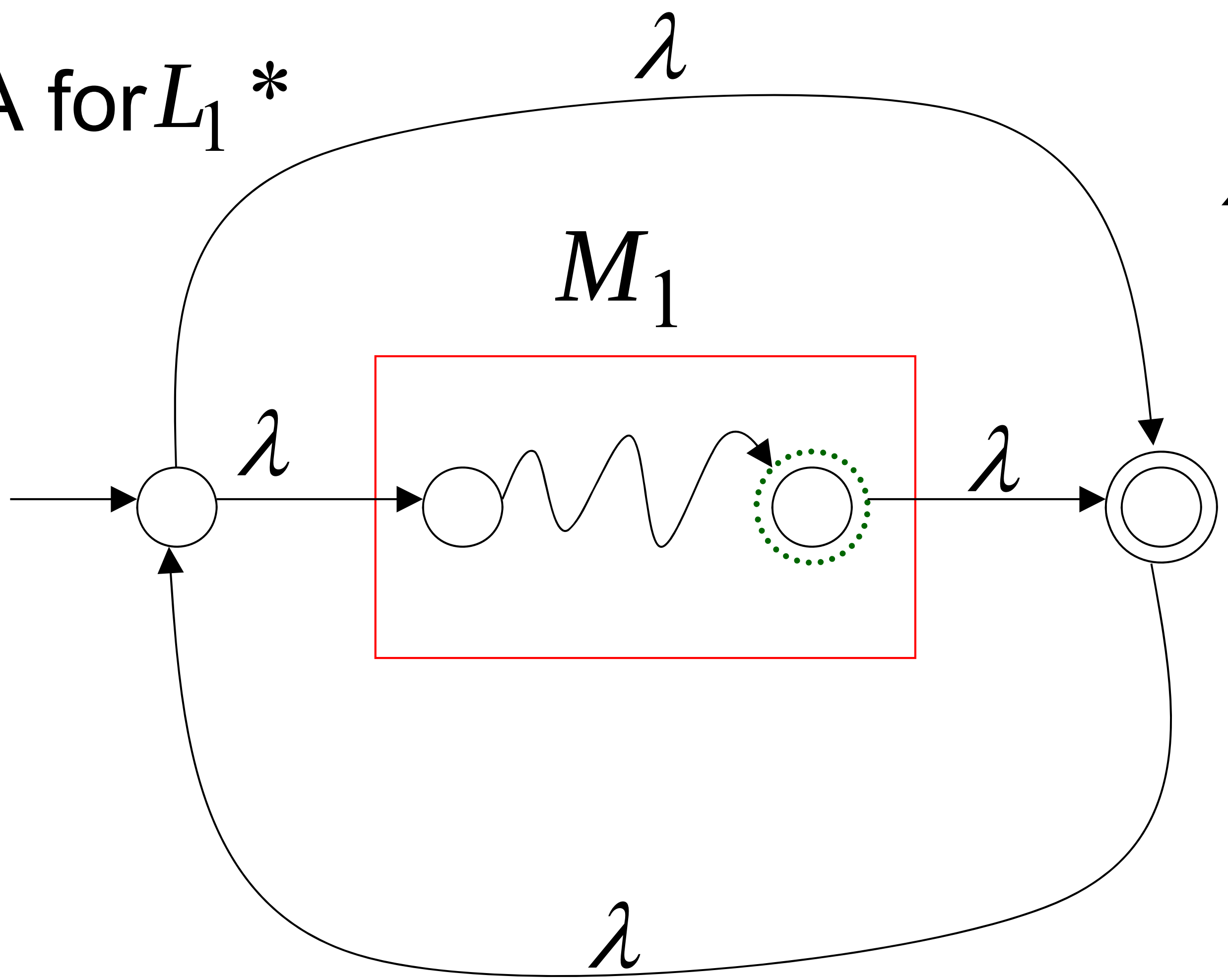
Example

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



Star Operation

NFA for $L_1{}^*$



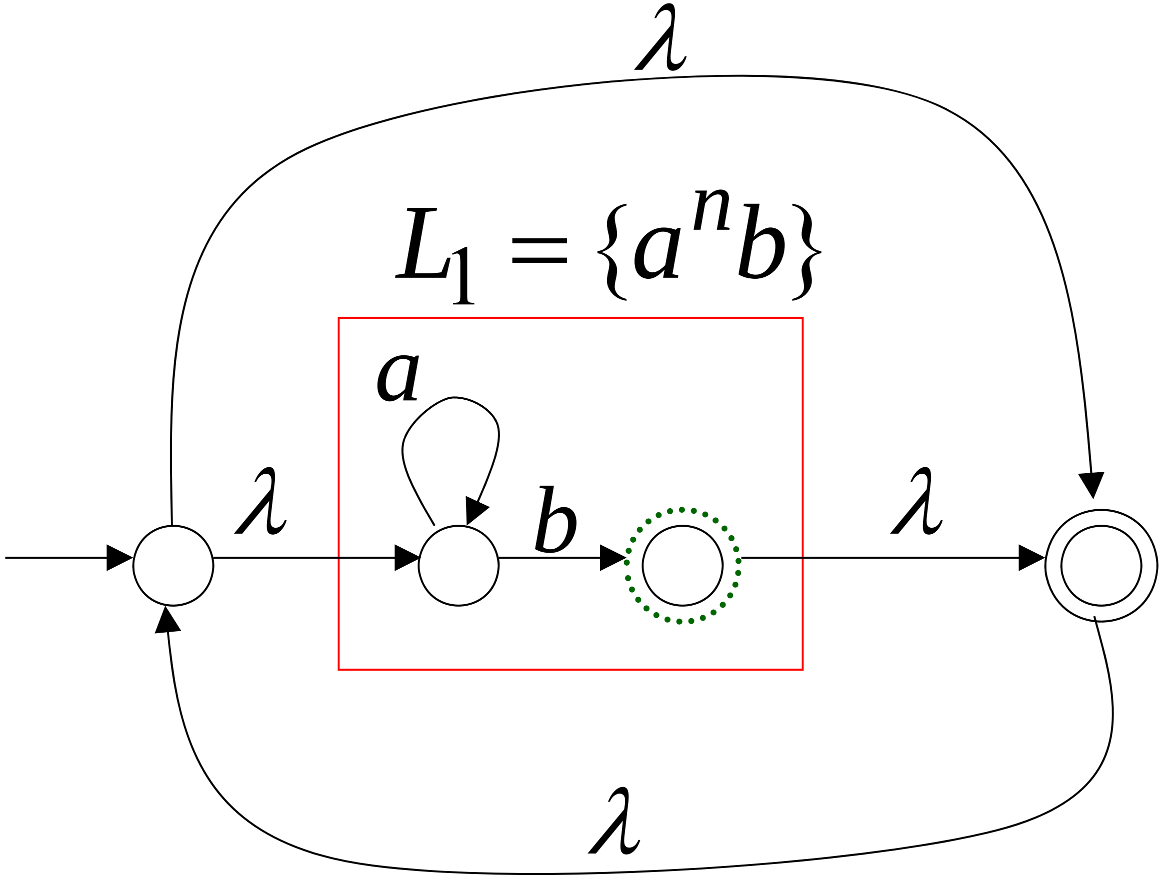
$\lambda \in L_1{}^*$

Example

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

$w = w_1 w_2 \cdots w_k$

$w_i \in L_1$



Reverse

NFA for L_1^R

L_1 M_1 M_1'

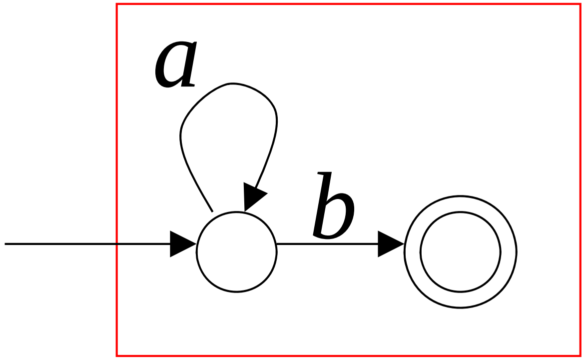
1. Reverse all transitions

2. Make initial state final state
   and vice versa

Example

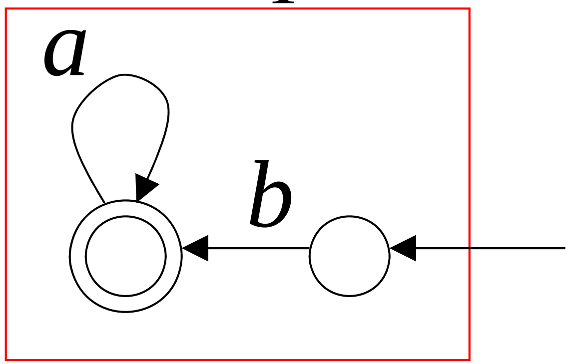
M_1

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

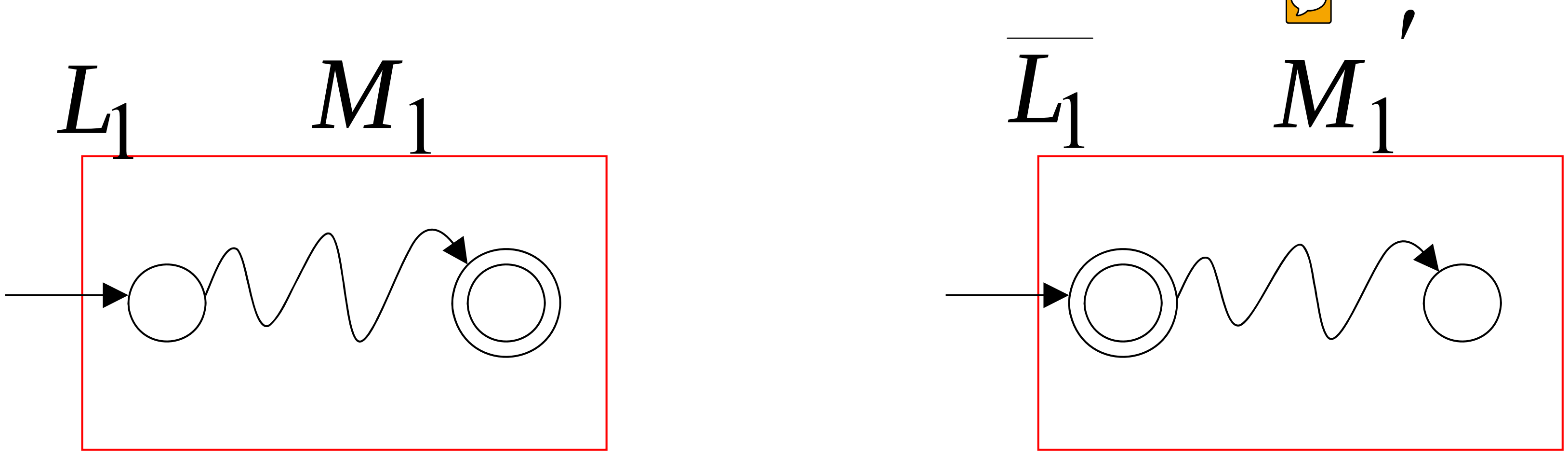


M_1'

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



Complement



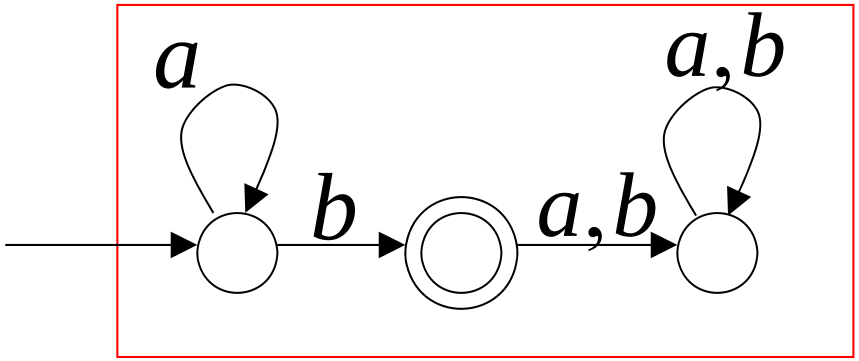
1. Take the **DFA** that accepts L_1

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

Example

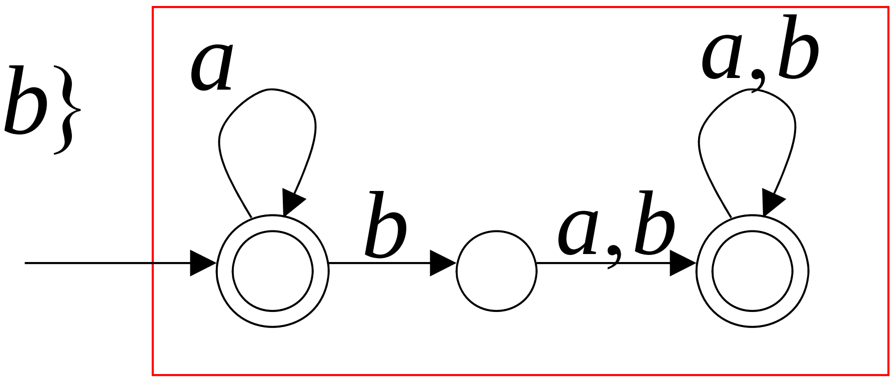
M_1

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



M_1'

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

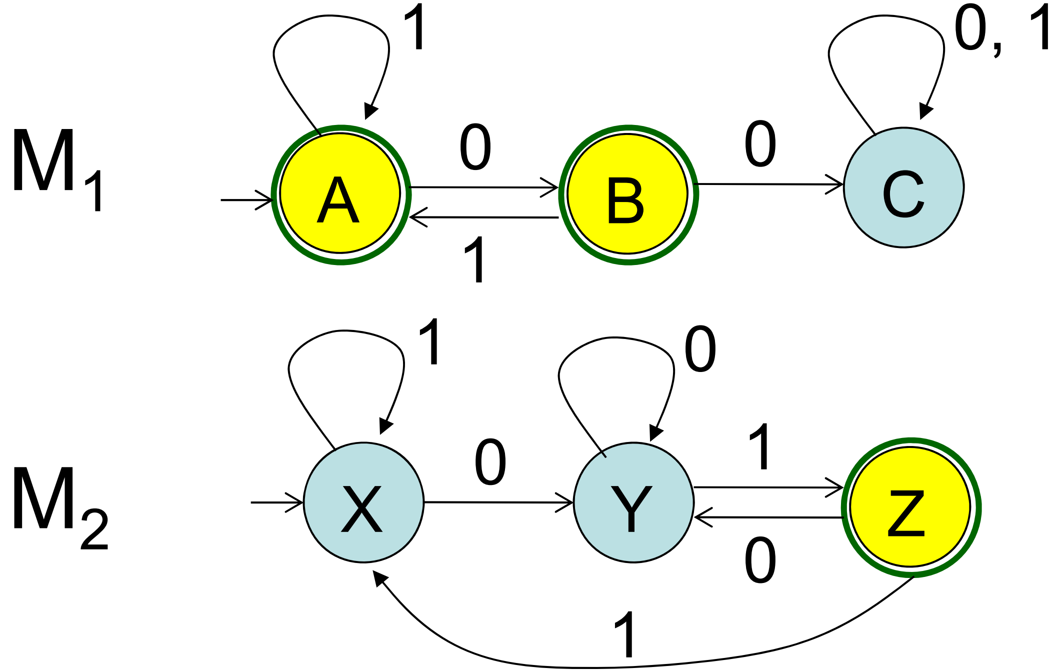


Intersection and Difference

- Let's step through an example

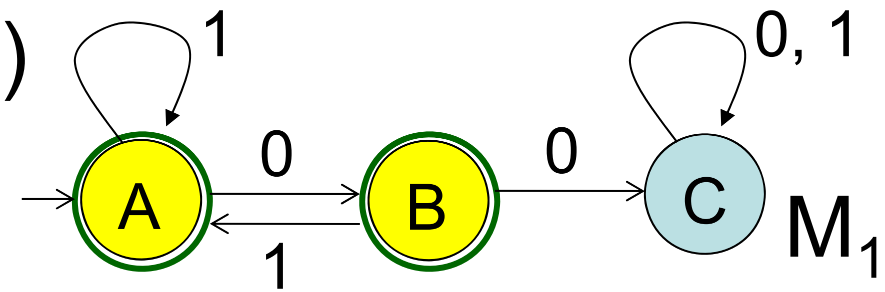
  $L_1 = \{ x \mid 00 \text{ is not a substring of } x\}$

  $L_2 = \{ x \mid x \text{ ends in } 01\}$

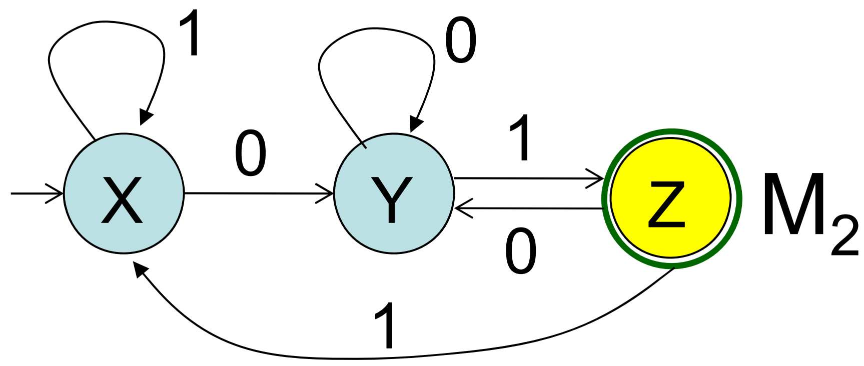


Intersection and Difference

- $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - $Q_1 = \{A, B, C\}$
  - $q_1 = A$
  - $F_1 = \{A, B\}$



- $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
  - $Q_2 = \{X, Y, Z\}$
  - $q_2 = X$
  - $F_2 = \{Z\}$

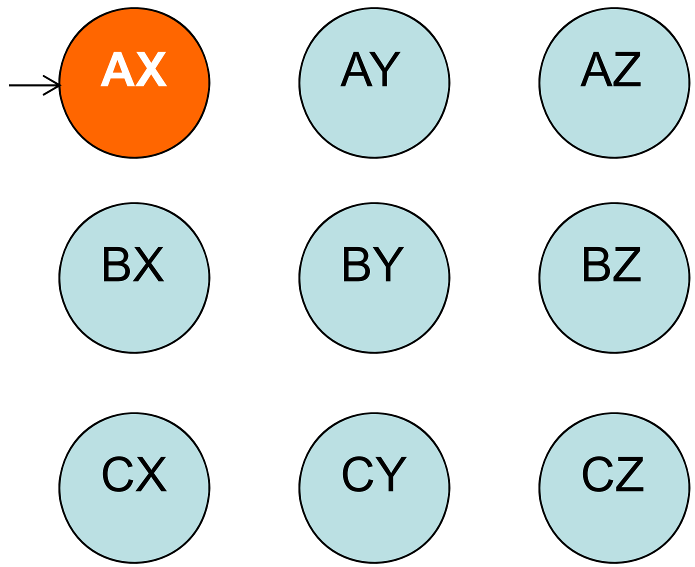


Intersection and Difference

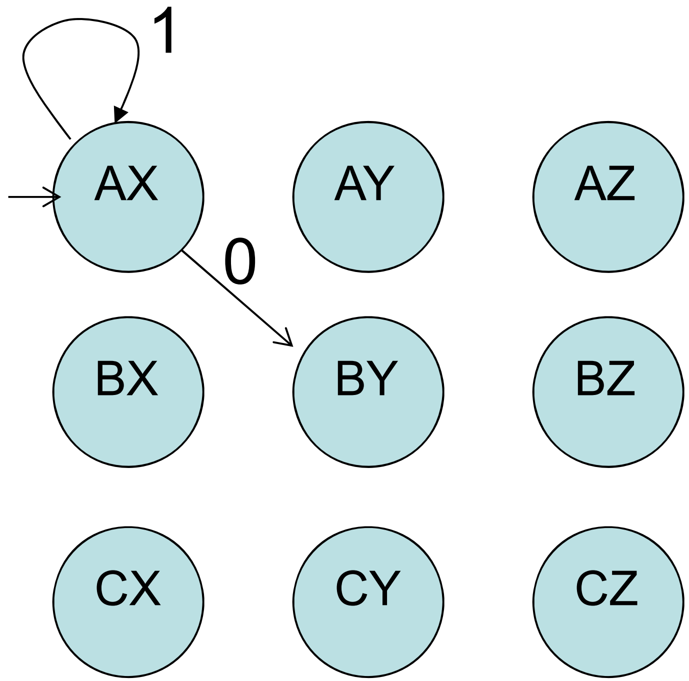
- $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{AX, AY, AZ, BX, BY, BZ, CX, CY, CZ\}$
  - $q_0 = AX$

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
- $Q_1 = \{A, B, C\}$
- $q_1 = A$
- $F_1 = \{A, B\}$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- $Q_2 = \{X, Y, Z\}$
- $q_2 = X$
- $F_2 = \{Z\}$

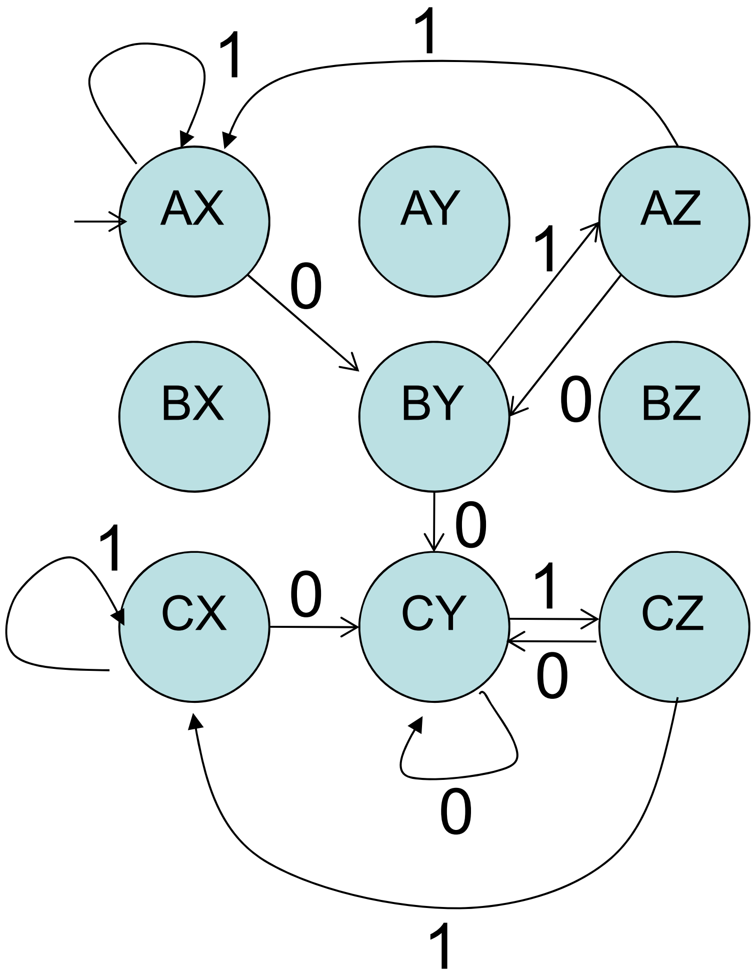


Intersection and Difference



$\delta((A,X), 1) = (\delta1\ (A,1), \delta2\ (X,1)) = (A, X)$
$\delta((A,X), 0) = (\delta1\ (A,0), \delta2\ (X,0)) = (B, Y)$

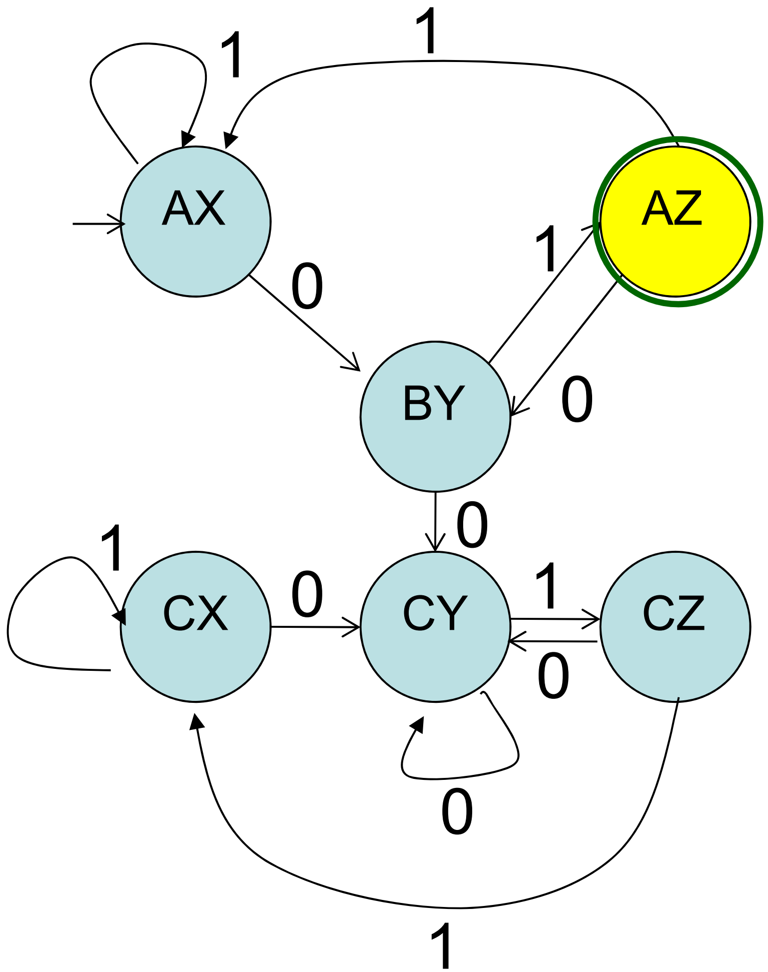
Intersection and Difference

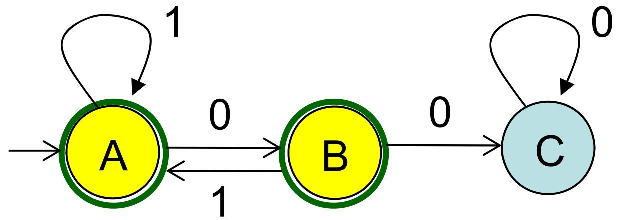


Intersection and Difference

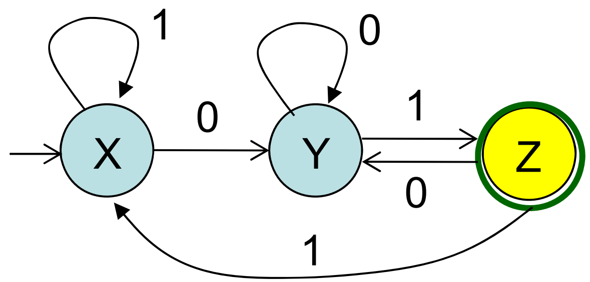
- Finally we can define F, the set of accepting states in M
- Intersection ($L_1 \cap L_2$)
  - $F = \{(p,q) \mid p \in F_1 \text{ and } q \in F_2\}$
- Difference ($L_1 - L_2$)
  - $F = \{(p,q) \mid p \in F_1 \text{ and } q \notin F_2\}$

Intersection ($L_1 \cap L_2$)



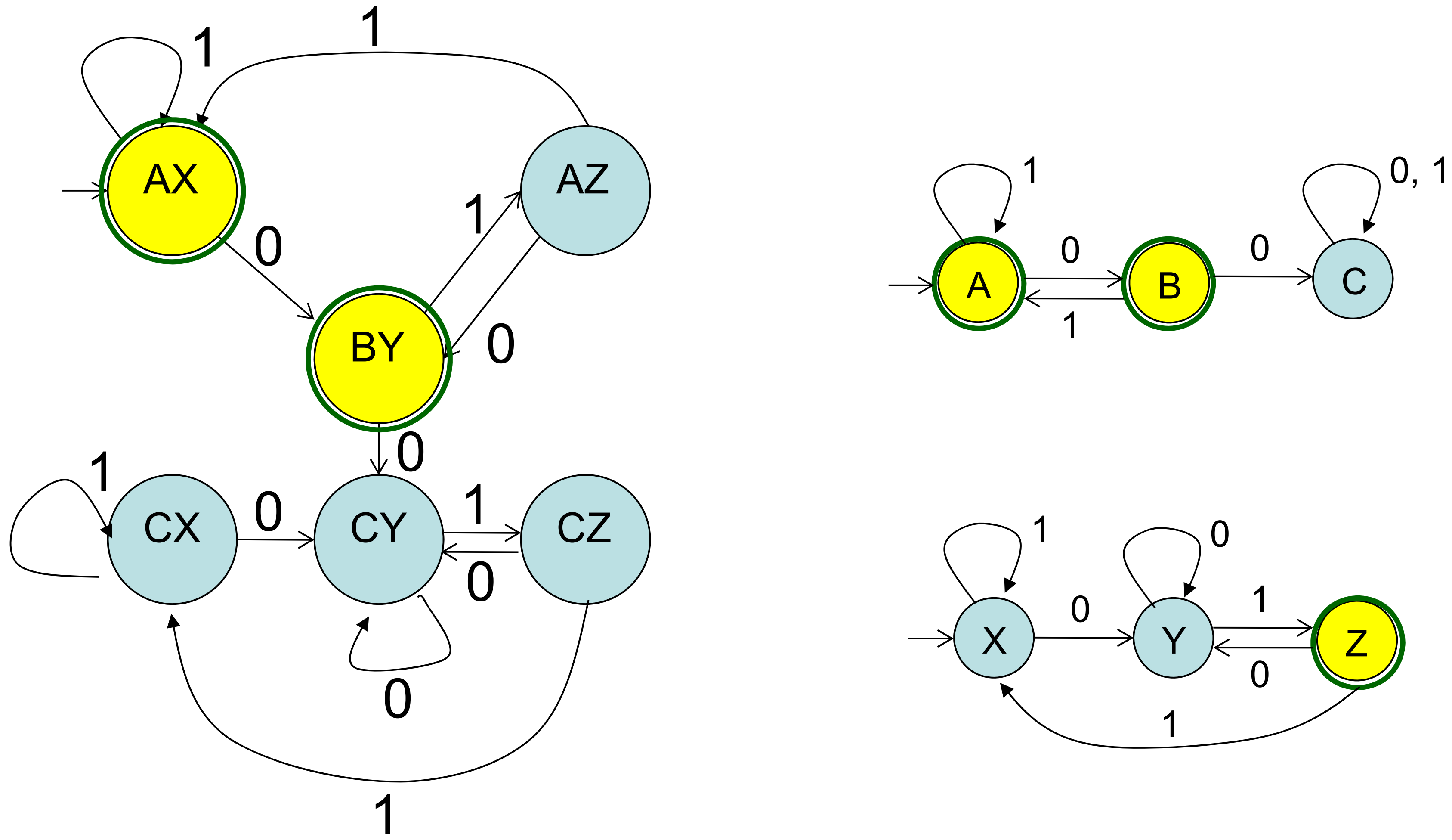


M_1

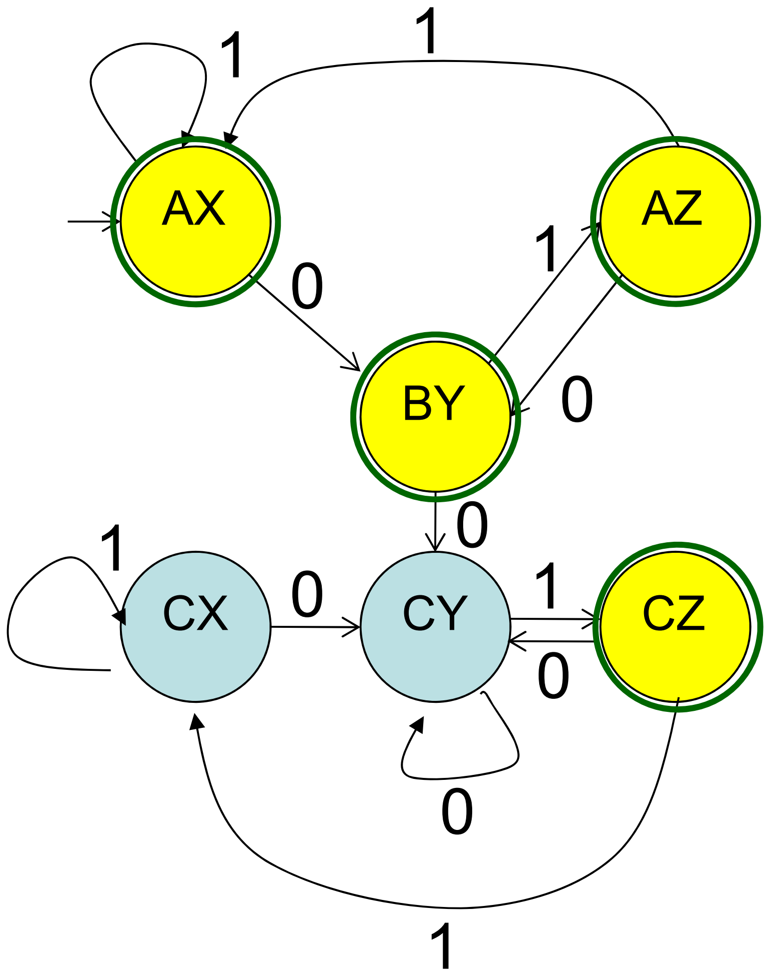


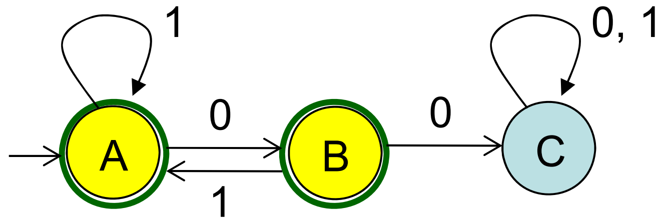
M_2

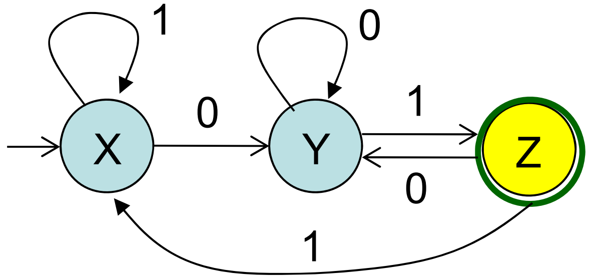
Difference (L_1 - L_2)



Union (L_1 U L_2)







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

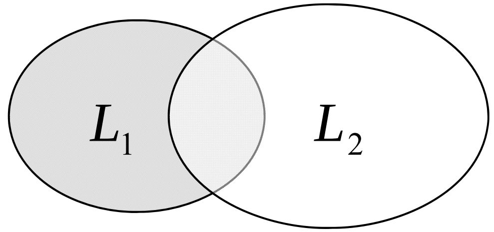
$L_1 = \{a^n b\}$ regular

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

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

regular

Difference



Venn diagrams

$$L_1 - L_2 = L_1 \cap \overline{L_2}$$

L_1 , L_2 regular

$\overline{L_2}$ regular

$L_1 \cap \overline{L_2}$ regular

Closure under Other Operations

- Definition 4.1:
  – Suppose Σ and Γ are alphabets. Then a function

$$h: \Sigma \rightarrow \Gamma^*$$

is called a homomorphism. In other words, a homomorphism is a substitution in which a single letter is replaced with a string.

If L is a language on Σ, then its homomorphic image is defined as

$$h(L) = \{h(w): w \in L\}$$

Example 4.2

- $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, c\}$ and define h by

$$h(a) = ab, h(b) = bbc.$$

$h(aba) = abbbcab$

The homomorphic image of $L = \{aa, aba\}$ is
$h(L) = \{abab, abbbcab\}$

Example 4.3

- $\Sigma = \{a, b\}$ and $\Gamma = \{b, c, d\}$ and define h by

$$h(a) = dbcc,\ h(b) = bdc.$$

If L is the regular language denoted by

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

then

$$r_1 = (dbcc + (bdc)^*)(dbccdbcc)^*$$

Theorem 4.3

- Let h be a homomorphism. If L is a regular language, then its homomorphic image $h(L)$ is also regular.

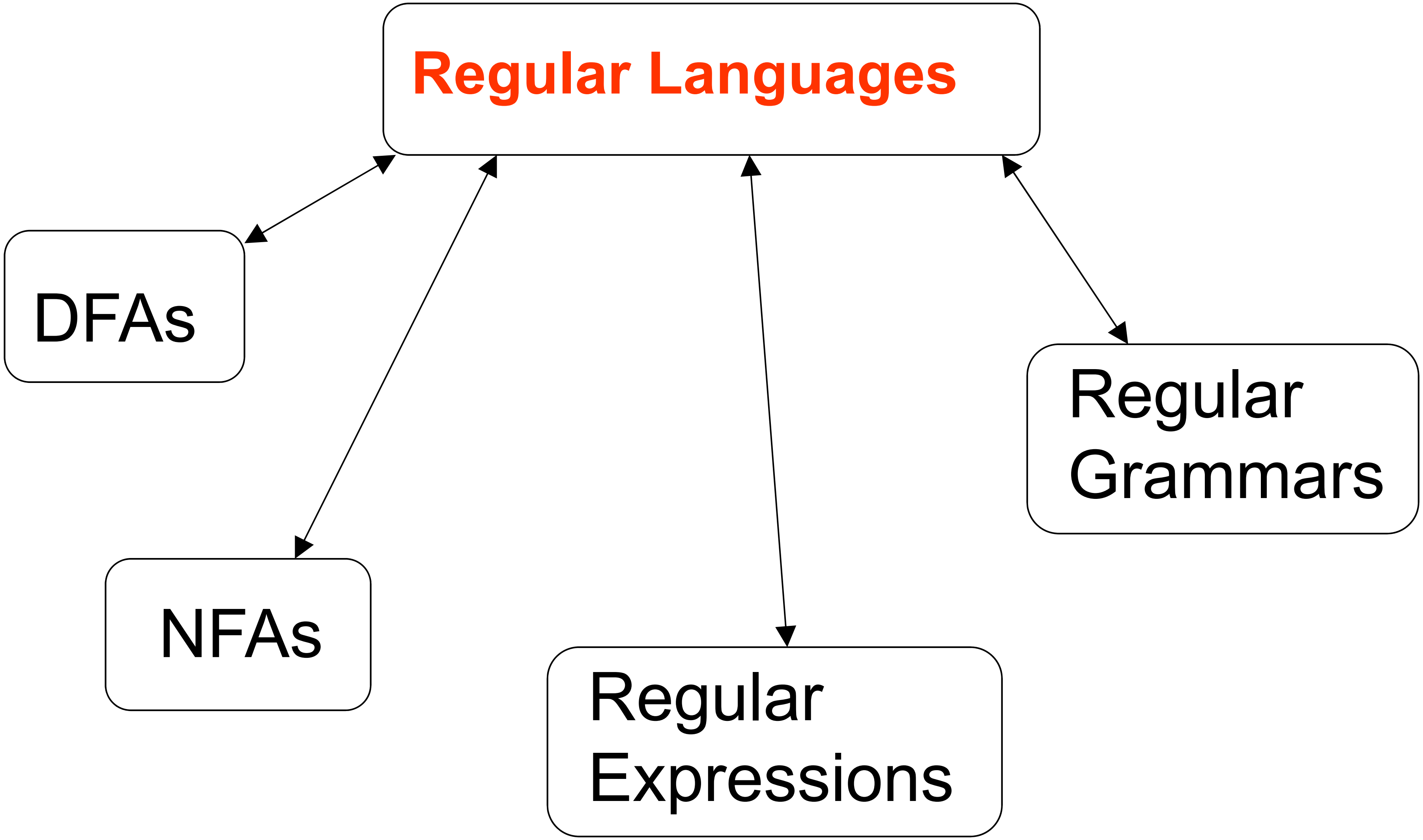
Outline

1 Closure Properties of Regular Languages

2 Elementary Questions about Regular Languages

3 Identifying Nonregular Languages

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

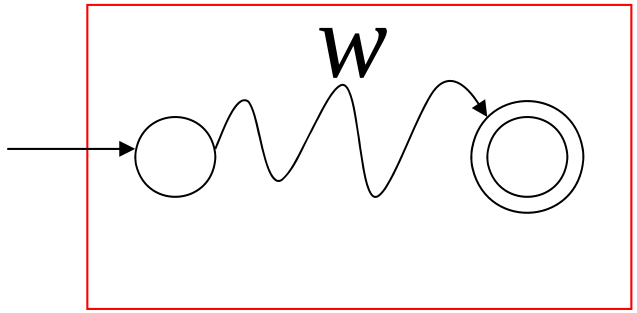
We mean: Language L is in a standard representation

Membership Question

Question: Given regular language L
and string w
how can we check if $w \in L$ **?**

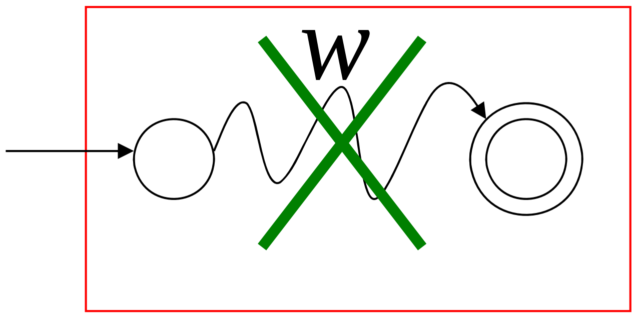
Answer: Take the DFA that accepts L
and check if w is accepted

DFA



$$w \in L$$

DFA



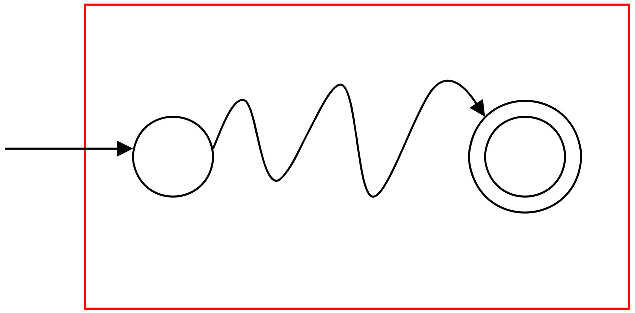
$$w \notin L$$

Question: Given regular language L how can we check if L is empty: $(L = \varnothing)$?

Answer: Take the DFA that accepts L

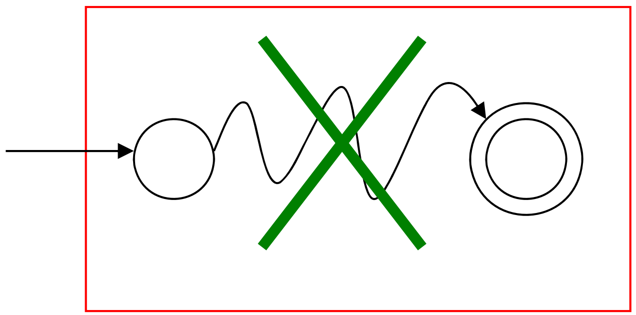
Check if there is any path from the initial state to a final state

DFA



$$L \neq \emptyset$$

DFA



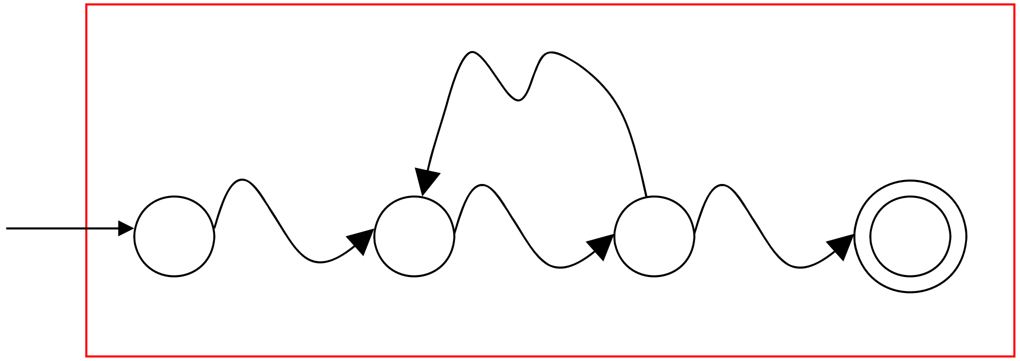
$$L = \emptyset$$

Question: Given regular language L how can we check if L is finite?

Answer: Take the DFA that accepts L

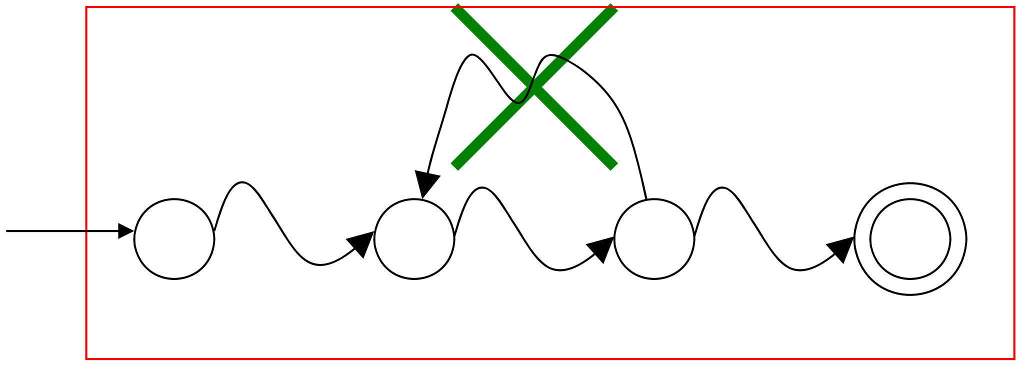
Check if there is a walk with cycle from the initial state to a final state

DFA



L is infinite

DFA



L is finite

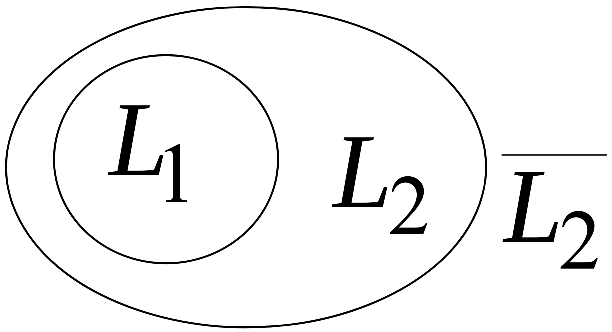
Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$ **?**

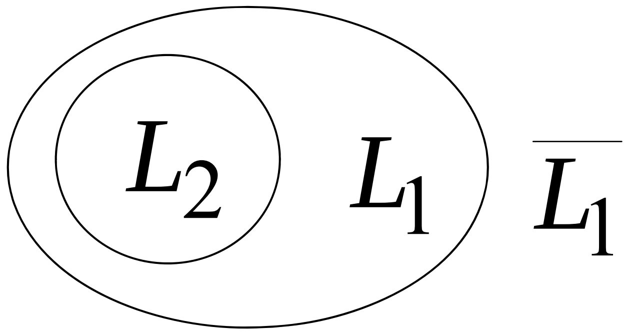
Answer: Find if $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

$$\Downarrow$$

$L_1 \cap \overline{L_2} = \varnothing$ and $\overline{L_1} \cap L_2 = \varnothing$





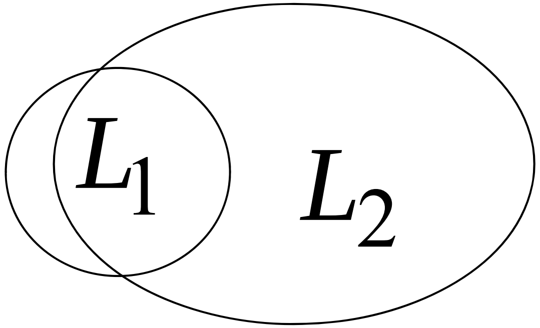
$L_1 \subseteq L_2$ $L_2 \subseteq L_1$

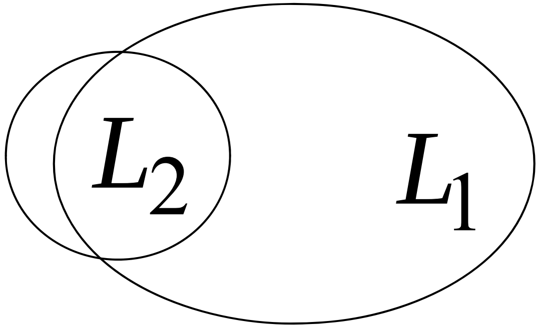
$$\Downarrow$$

$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$L_1 \cap \overline{L_2} \neq \emptyset$ or $\overline{L_1} \cap L_2 \neq \emptyset$





$L_1 \not\subset L_2$ $L_2 \not\subset L_1$

$$L_1 \neq L_2$$

Outline

1. Closure Properties of Regular Languages
2. Elementary Questions about Regular Languages
3. Identifying Nonregular Languages

Non-regular languages

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

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

Regular languages

$$a^*b \qquad b^*c + a$$

$$b + c(a+b)^* \qquad etc...$$

Finite languages

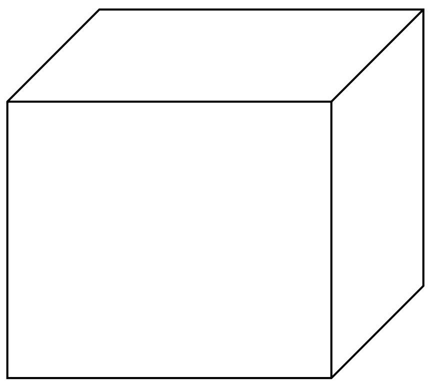
How can we prove that a language L
is not regular?

Prove that there is no DFA that accepts L

Problem: this is not easy to prove

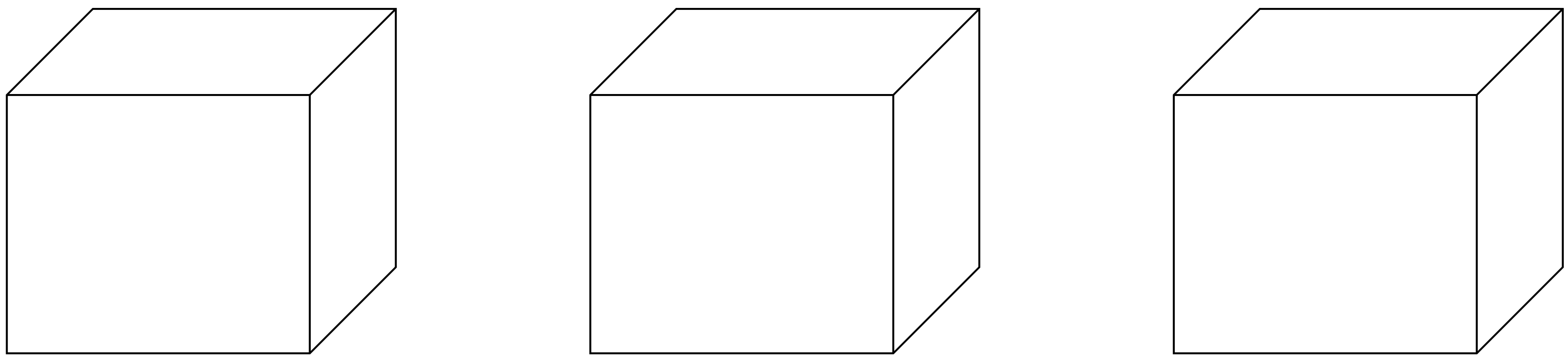
Solution: the Pumping Lemma !!!

The Pigeonhole Principle

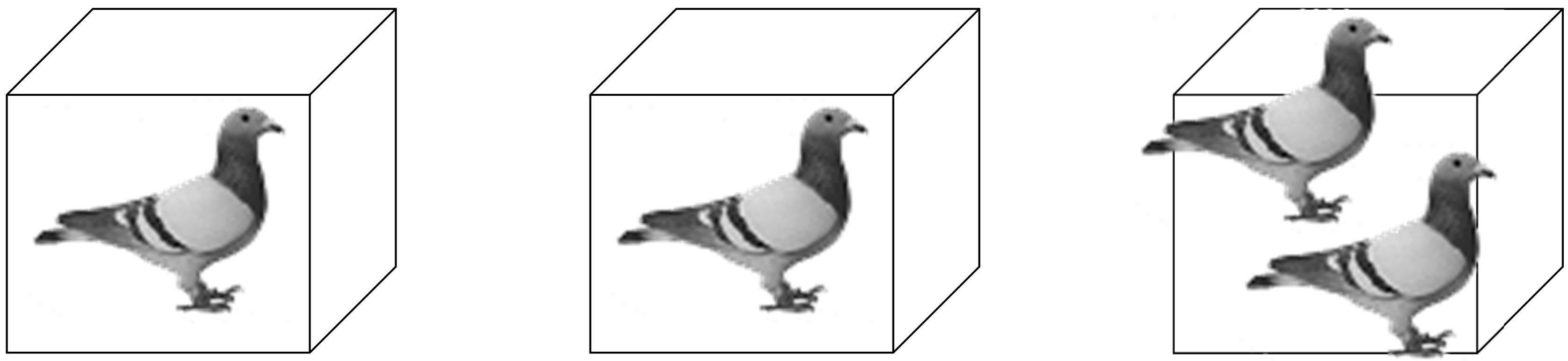


4 pigeons

3 pigeonholes

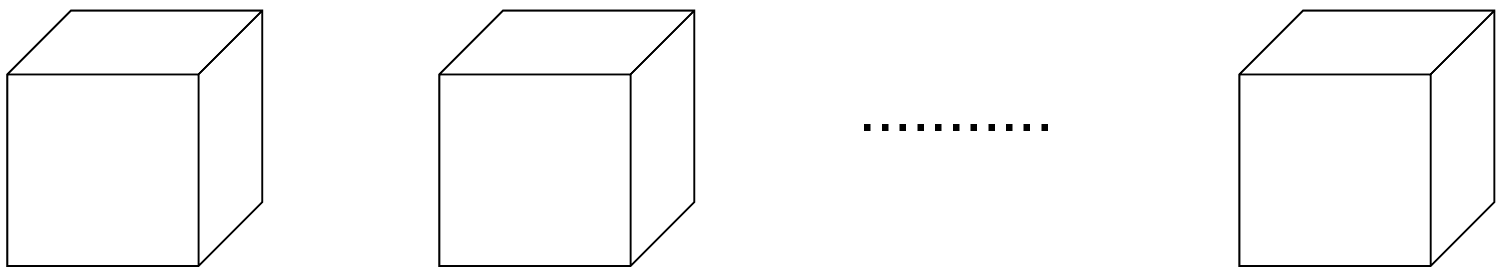


A pigeonhole must
contain at least two pigeons



n pigeons

m pigeonholes $\qquad n > m$



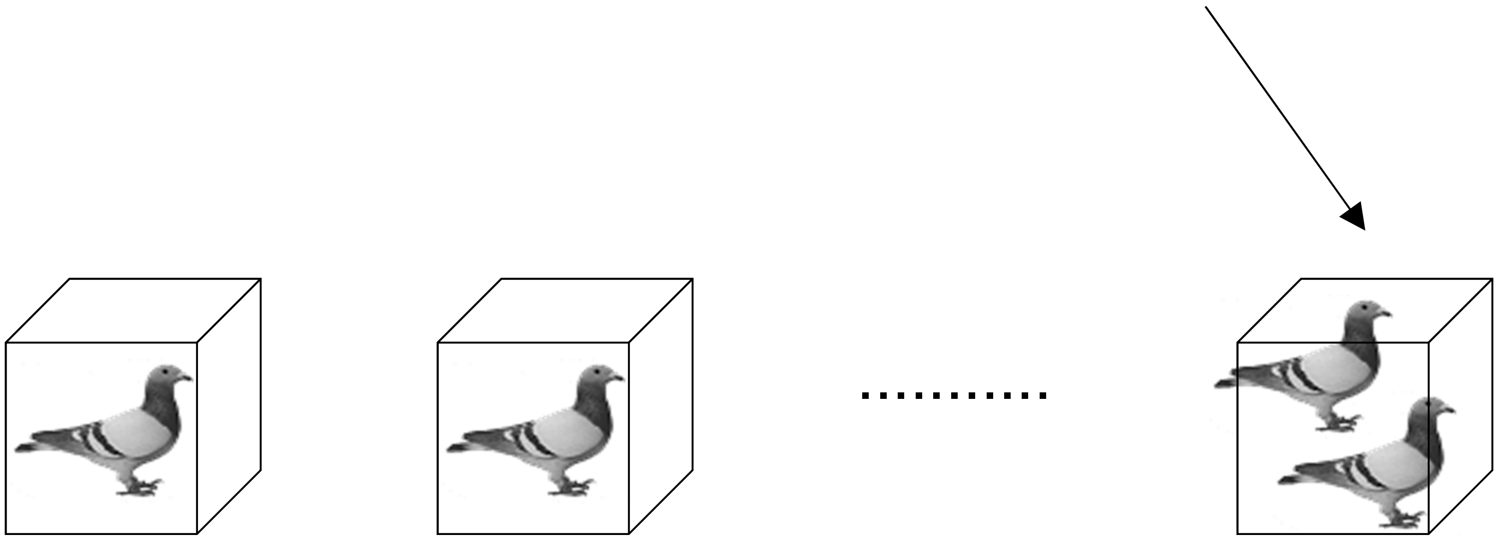
The Pigeonhole Principle

n pigeons

m pigeonholes

$$n > m$$

There is a pigeonhole with at least 2 pigeons

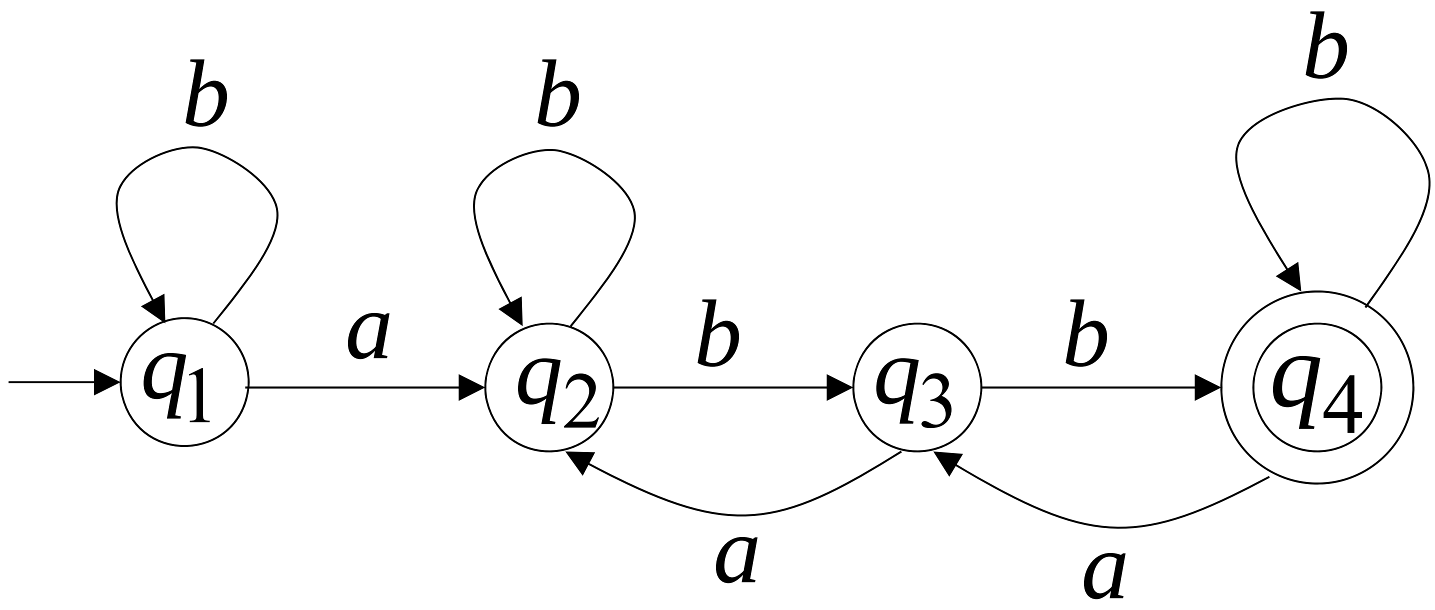


The Pigeonhole Principle

and

DFAs

DFA with 4 states

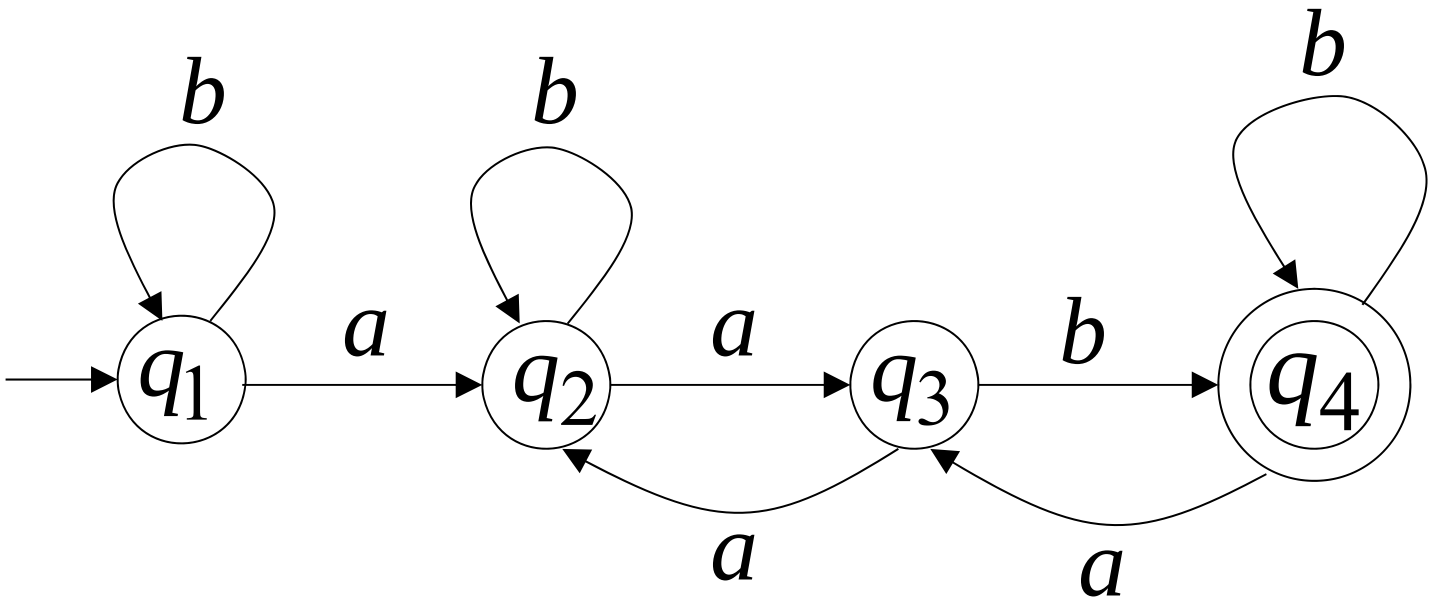


In walks of strings: a

aa

aab

no state
is repeated



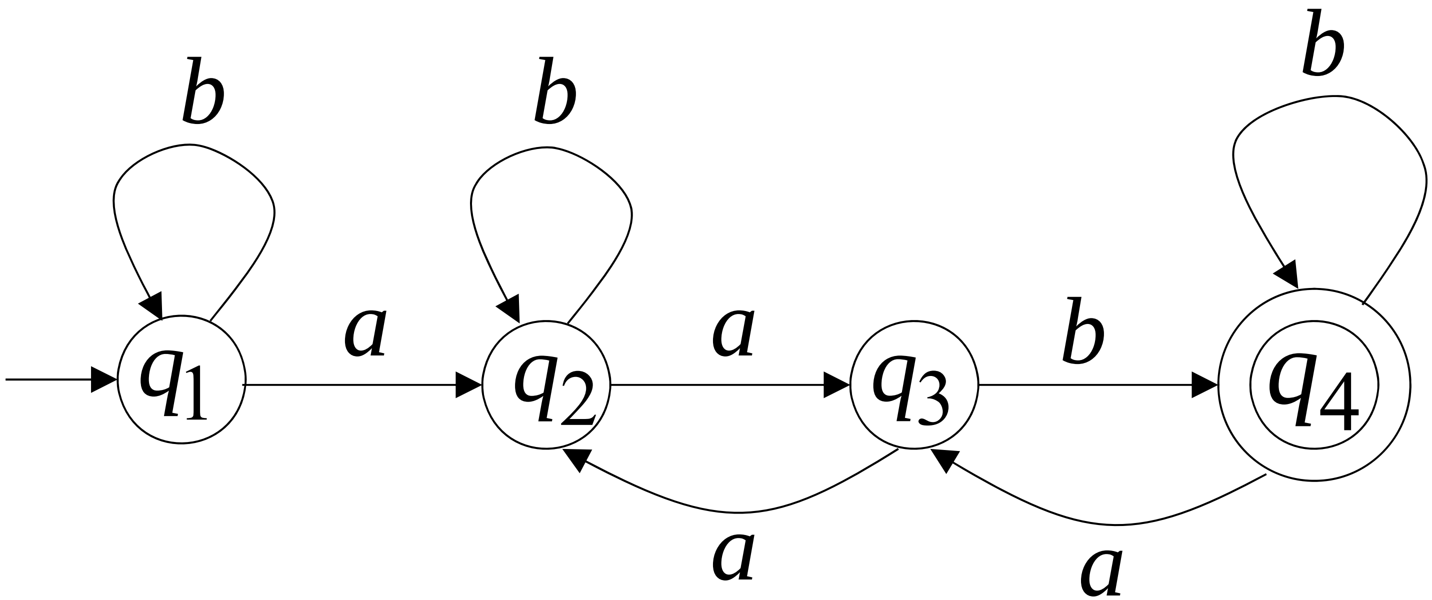
In walks of strings: *aabb*

bbaa

abbabb

abbbabbabb...

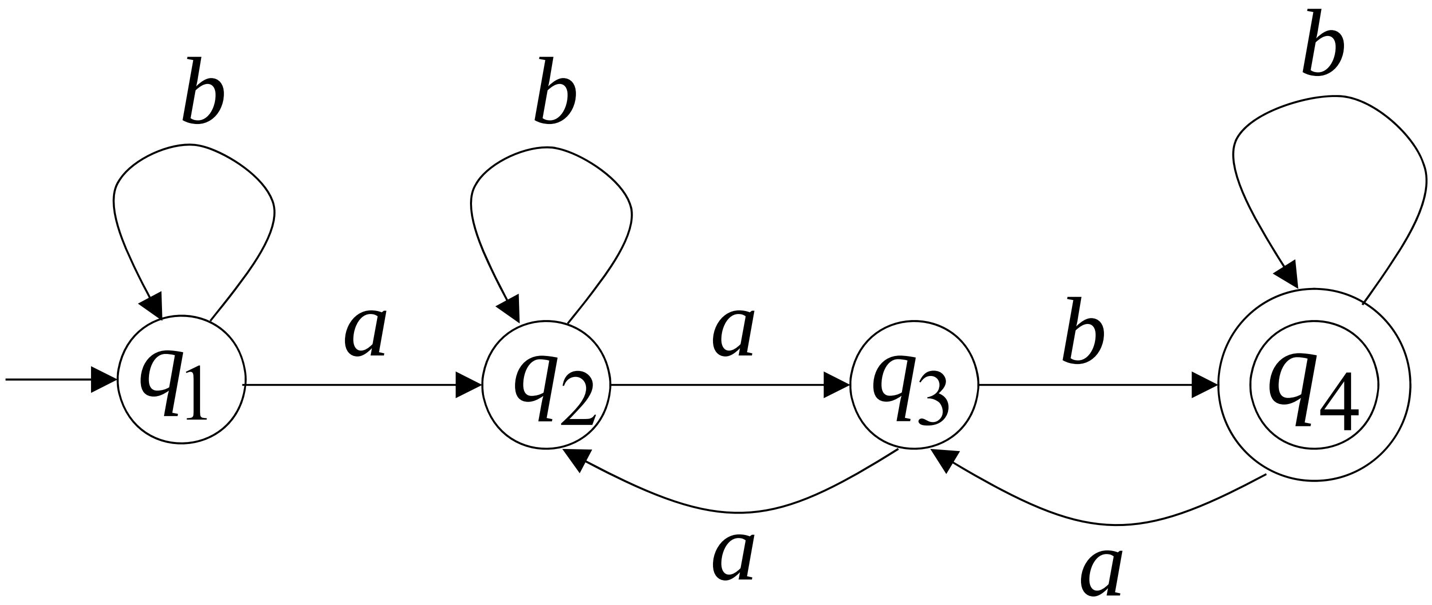
a state
is repeated



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

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

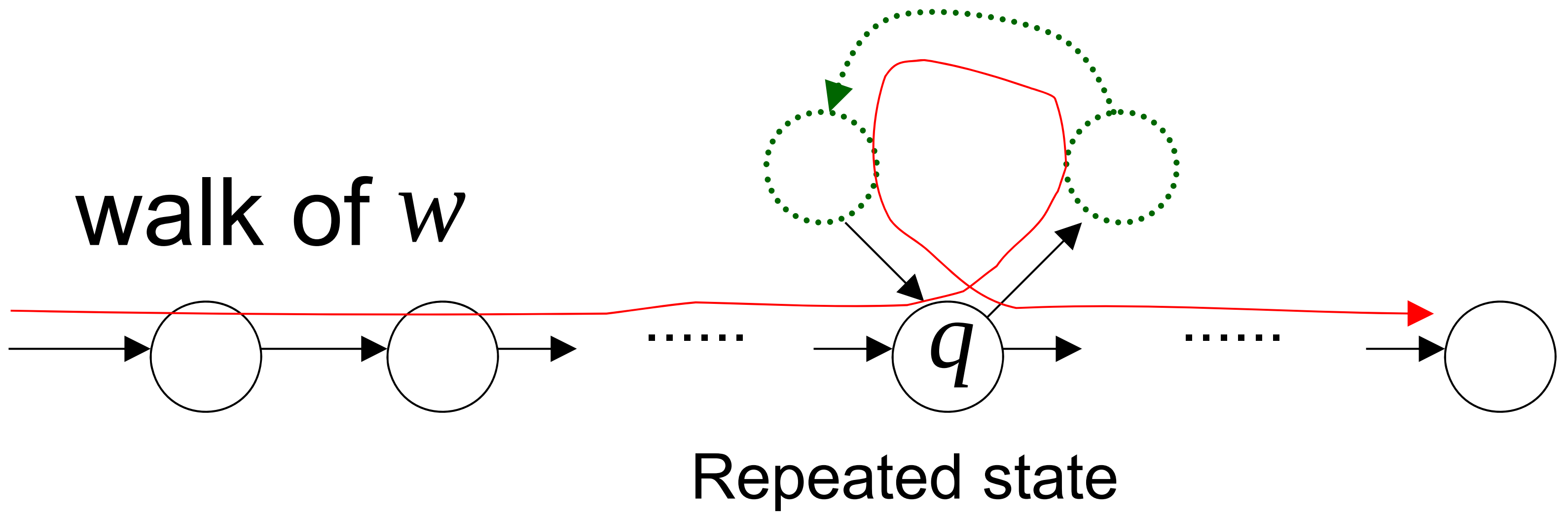
Thus, a state must be repeated



In general, for any DFA:

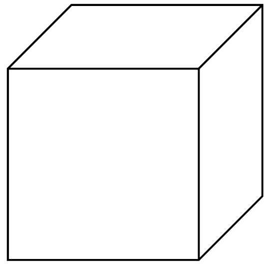
String w has length $\geq$ number of states

A state q must be repeated in the walk of w

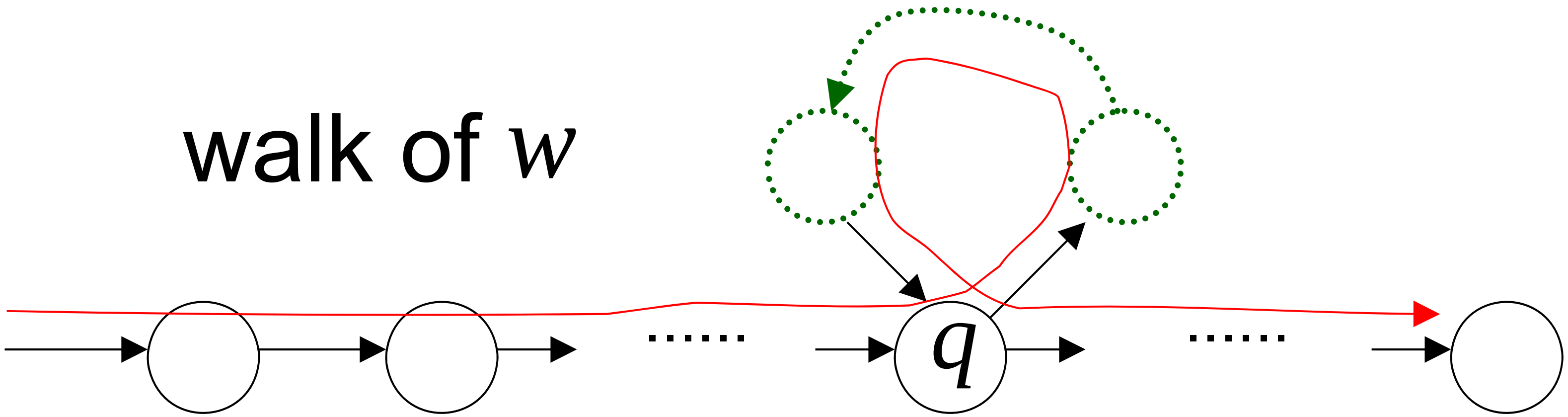


In other words for a string w :

$\xrightarrow{a}$ transitions are pigeons 

q states are pigeonholes 

walk of w



Repeated state

Example 4.6

- (Is $L = \{a^n b^n : n \geq 0\}$ regular?)
- Suppose L is regular → A DFA M exists for it
  - $\delta^*(q_0, a^i)$ for i = 1, 2, 3, … (unlimited)
  - But only a finite number of states in M
  - By pigeonhole principle, there must some state q s.t.
    $\delta^*(q_0, a^n) = q$ and $\delta^*(q_0, a^m) = q$ with $n \neq m$
  - Since M accepts $a^n b^n$ we must have
    $\delta^*(q, b^n) = q_f \in F$
    $\delta^*(q_0, a^m b^n) = q_f \in F$ (contradiction!! $\because n \neq m$)

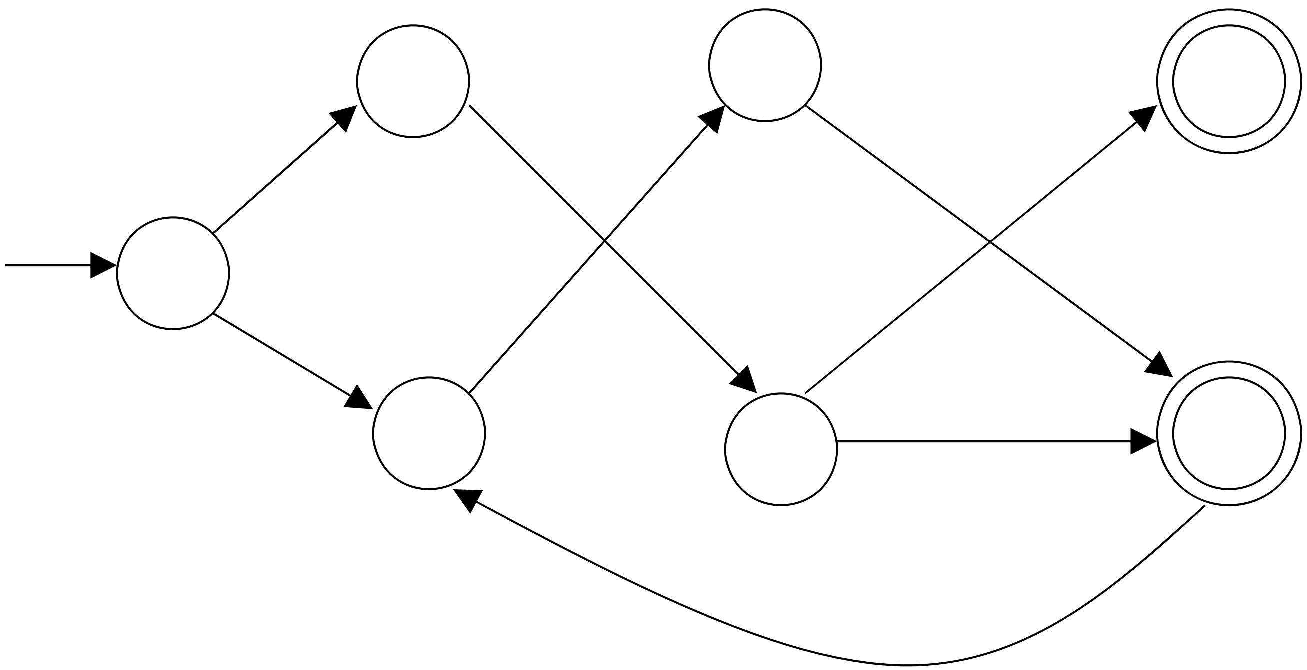
To accept all $a^n b^n$, an automaton would have to differentiate between all prefixes a^n and a^m.

But since there are only a finite number of internal states with which to do this, there are some n and m for which the distinction cannot be made.

The Pumping Lemma

Take an infinite regular language L

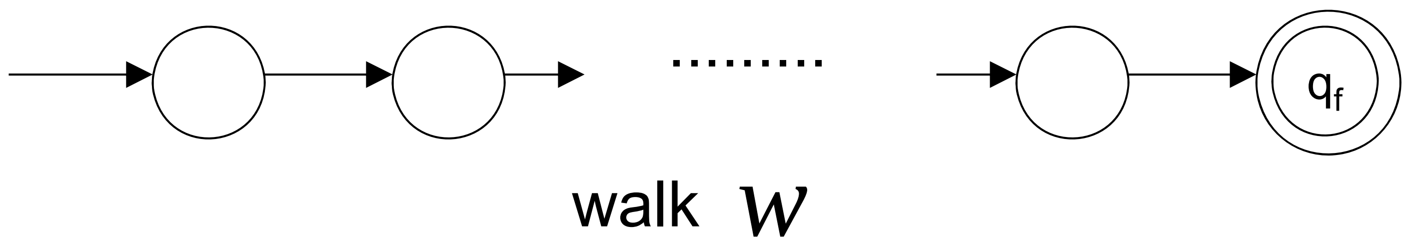
There exists a DFA M that accepts L



m
states

Take a string w with $w \in L$ (drive to q_f)

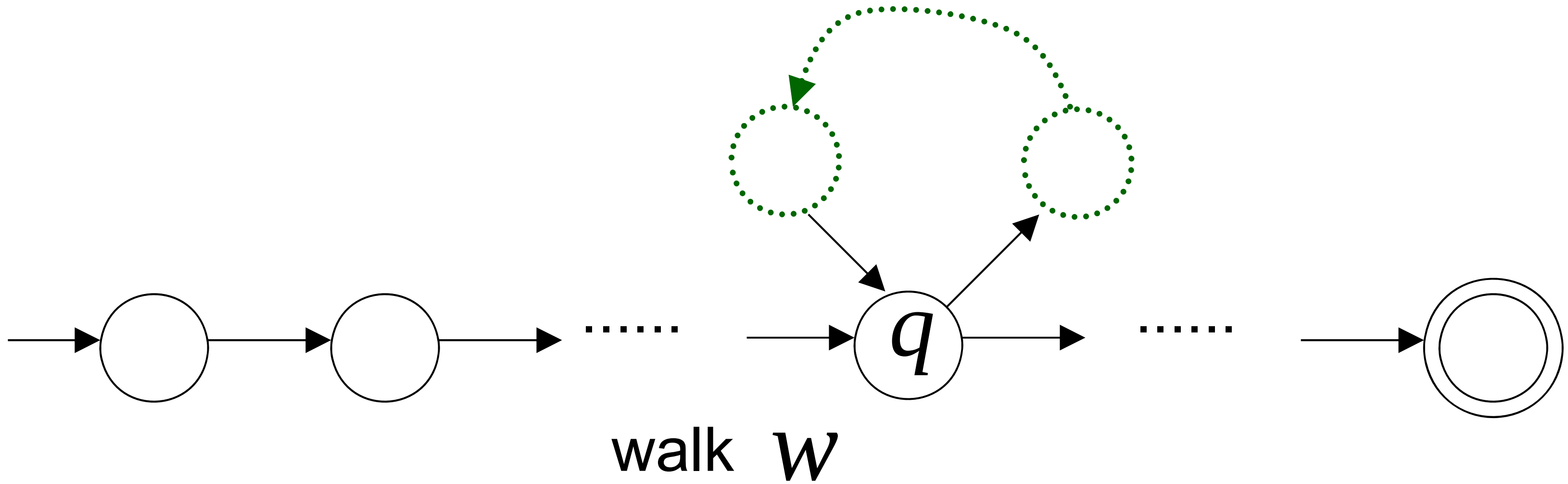
There is a walk with label w :



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

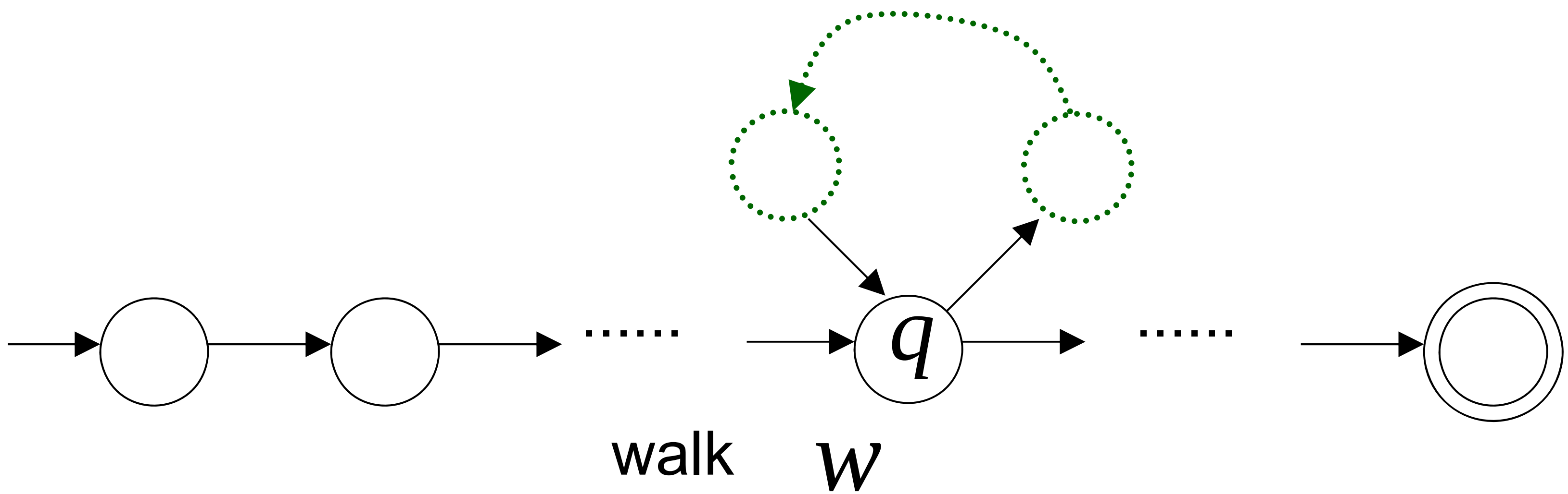
then, from the pigeonhole principle:

a state is repeated in the walk w

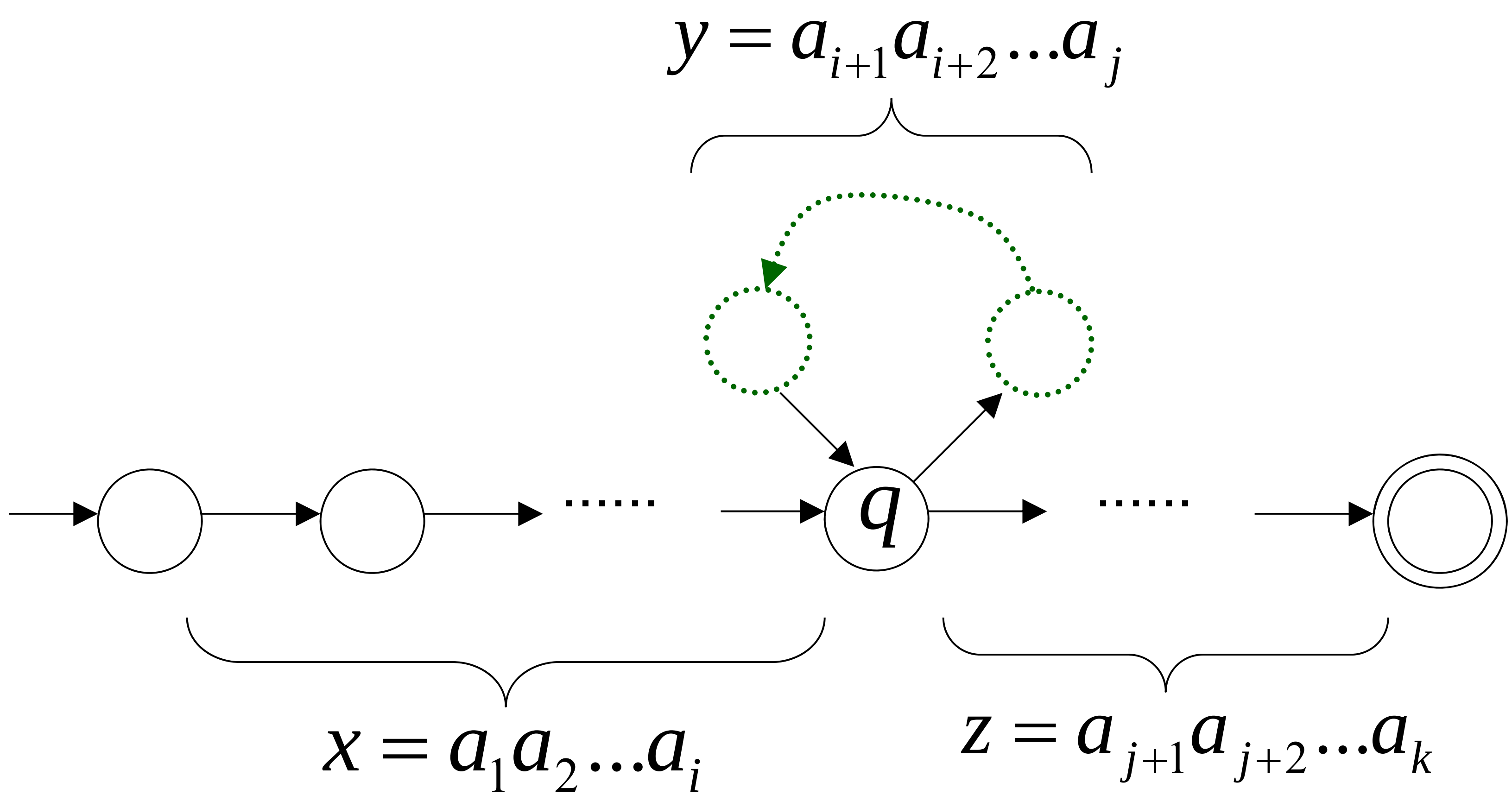


Let q be the first state repeated in the walk of w

(such a repetition must start no later than the m^{th} move)



Write $w = x\ y\ z$



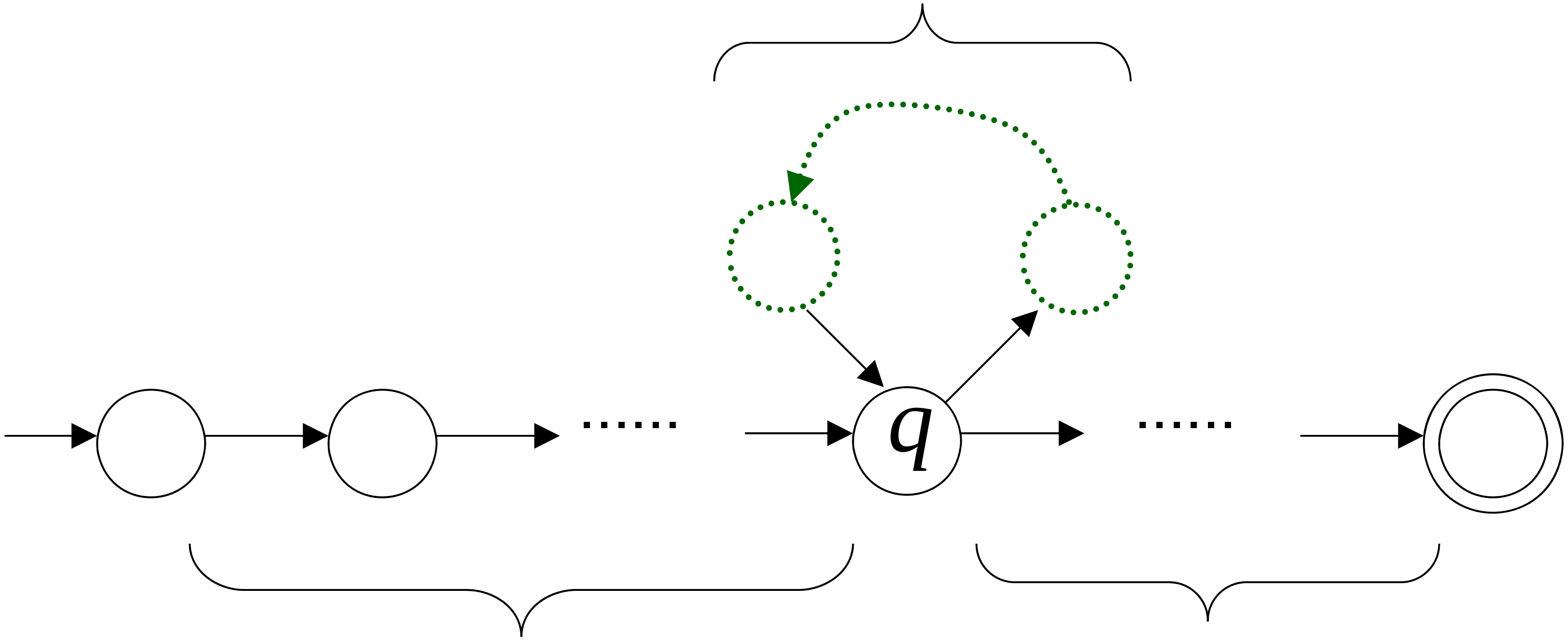
Observations:

length $|x\,y| \leq m$ Number of states of DFA

length $|y| \geq 1$

Remember 'q' is the first repeated state, meaning that a_1, a_2, …, a_i, a_{i+1} …, a_j, are passed through different states

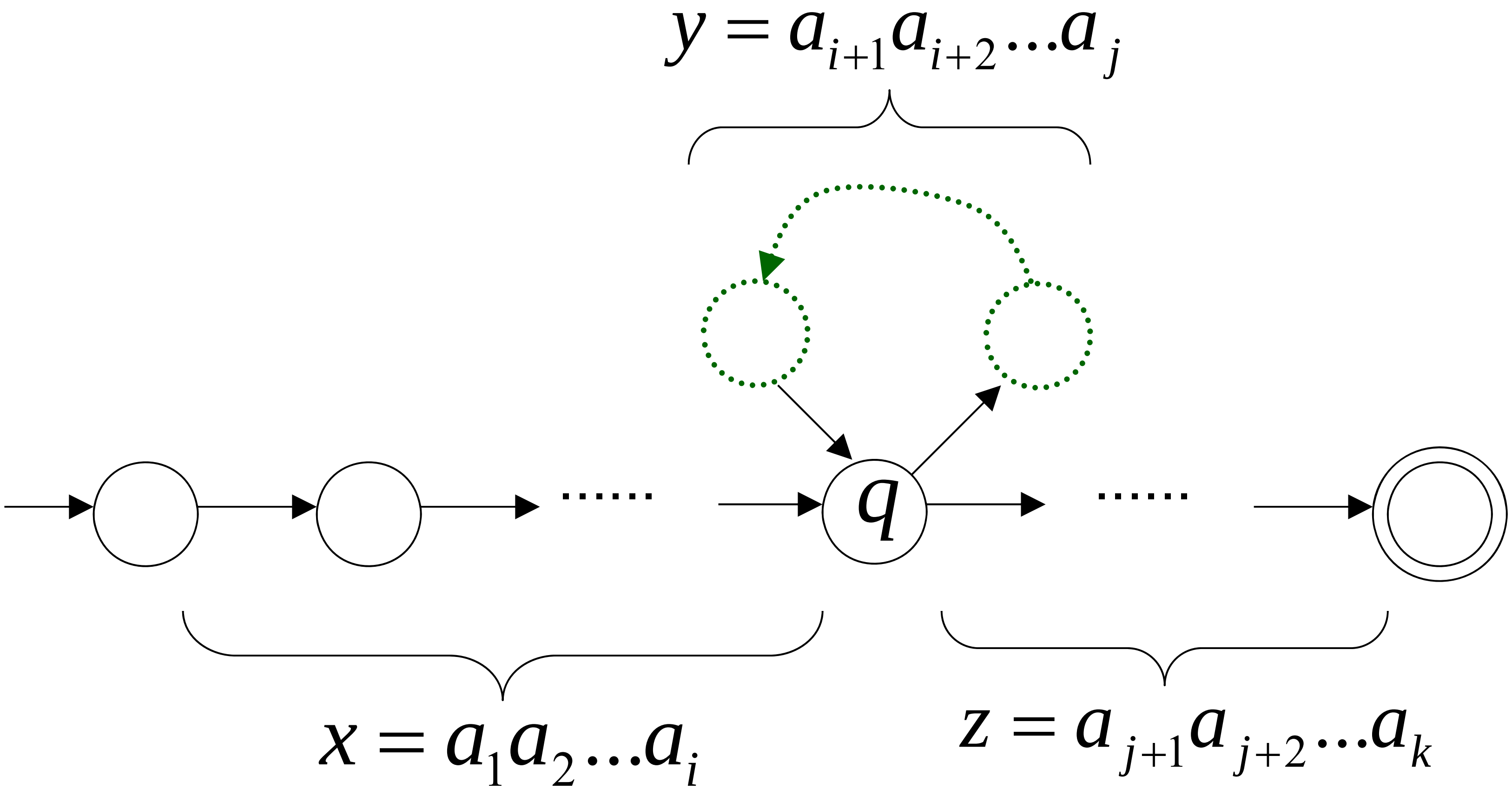
$$y = a_{i+1}a_{i+2}\ldots a_j$$



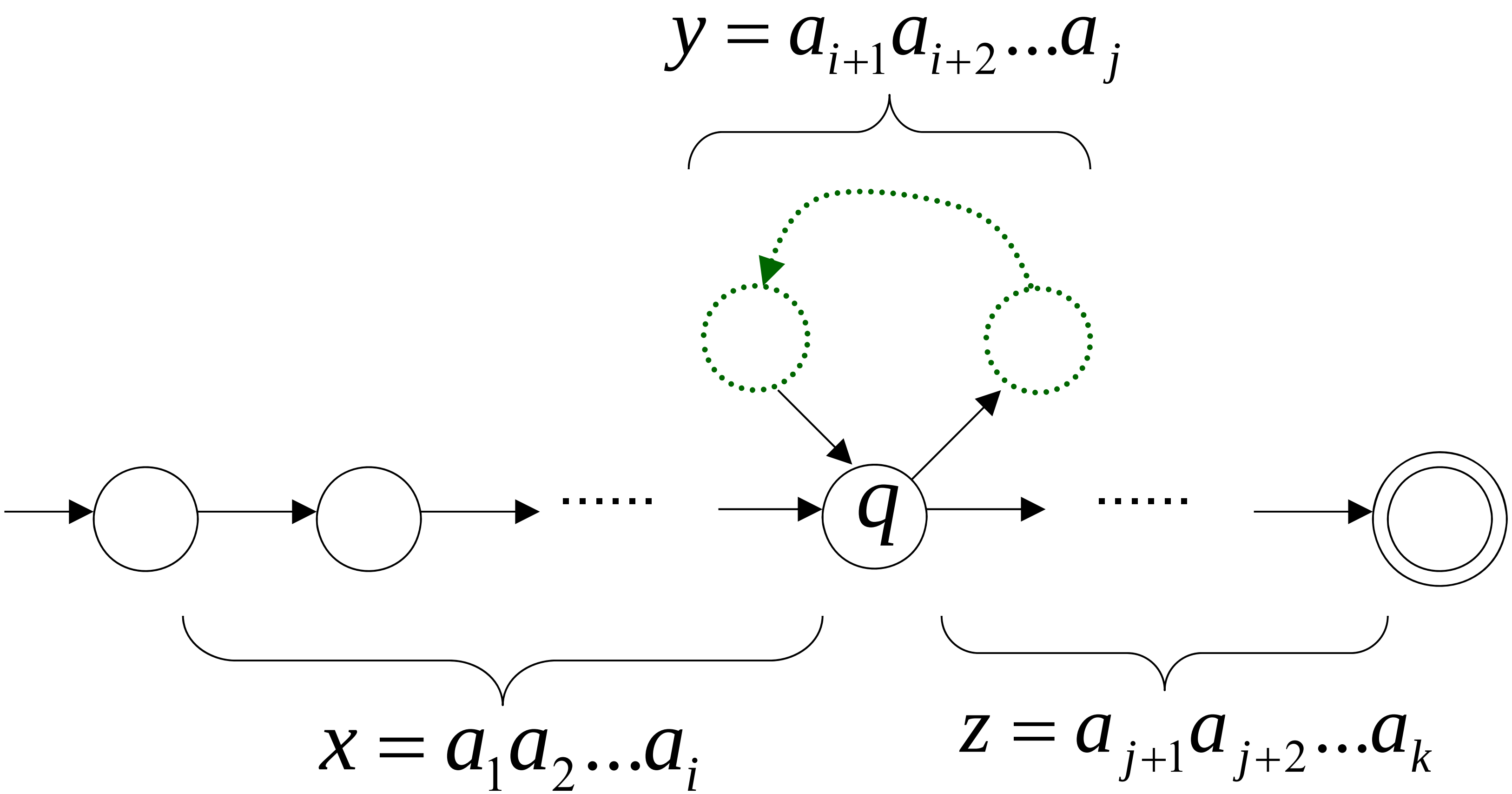
$$x = a_1 a_2 \ldots a_i$$

$$z = a_{j+1}a_{j+2}\ldots a_k$$

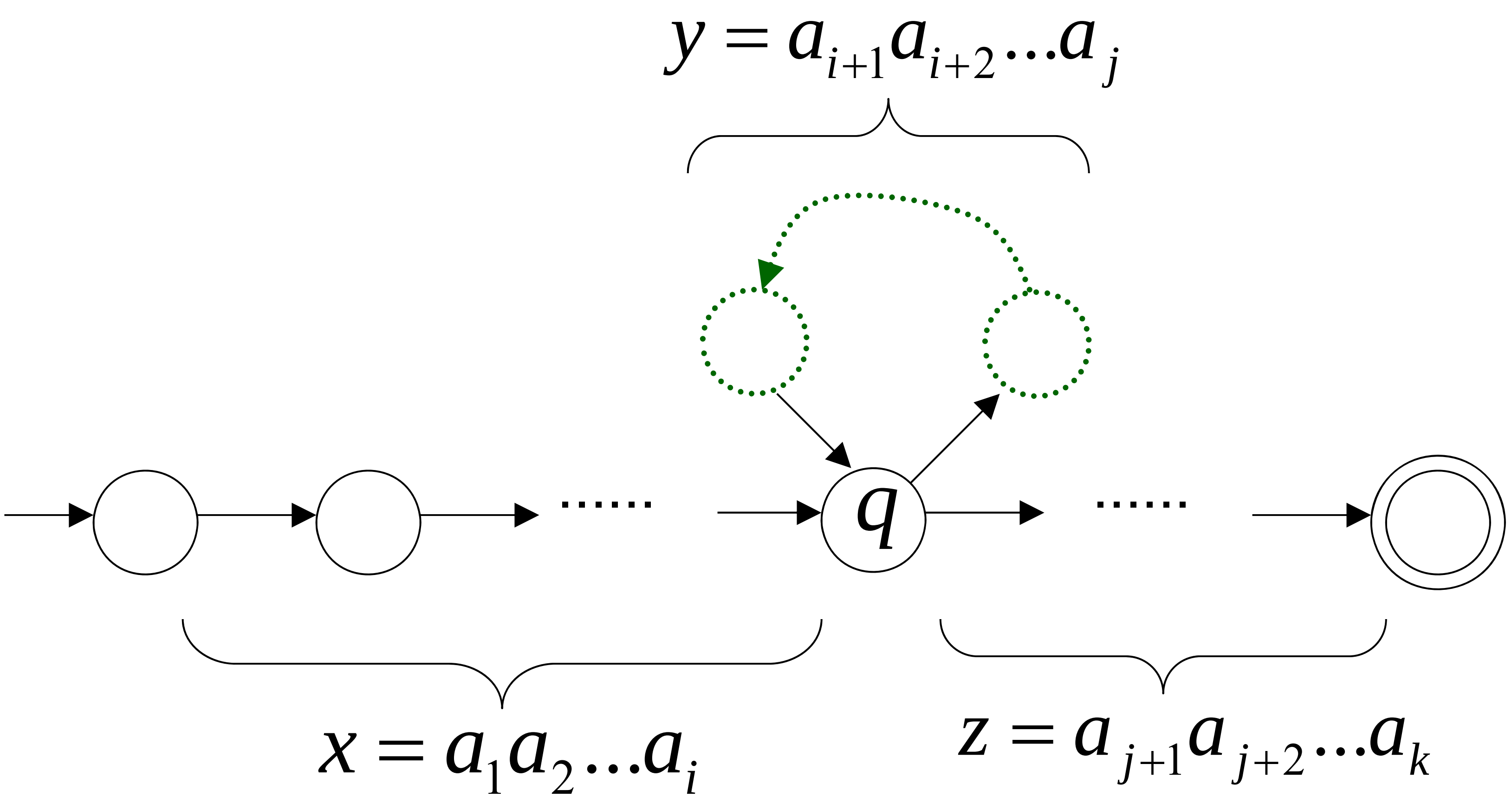
Observation: The string $x\,z$ is accepted



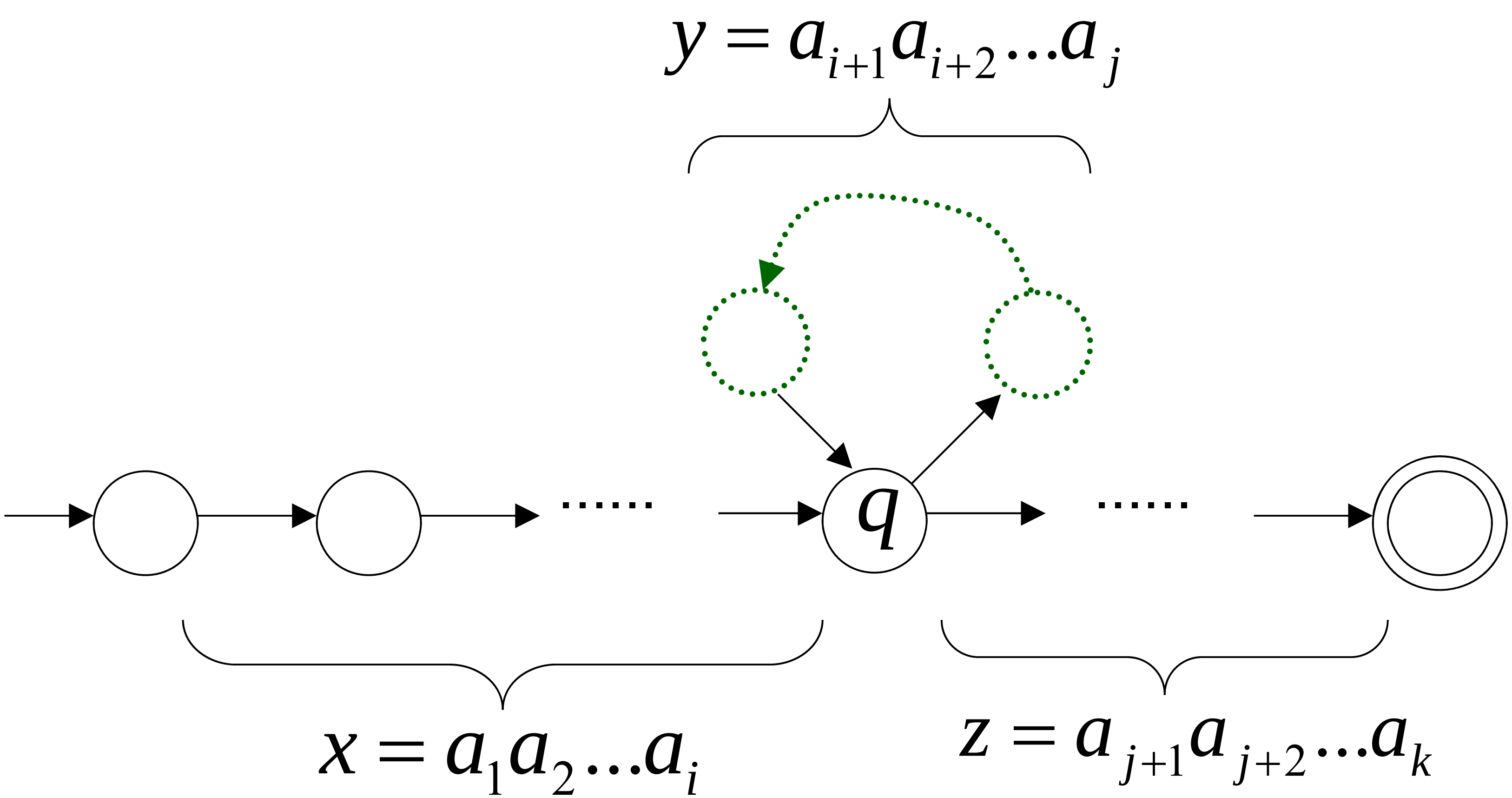
Observation: The string $x\ y\ y\ z$ is accepted



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



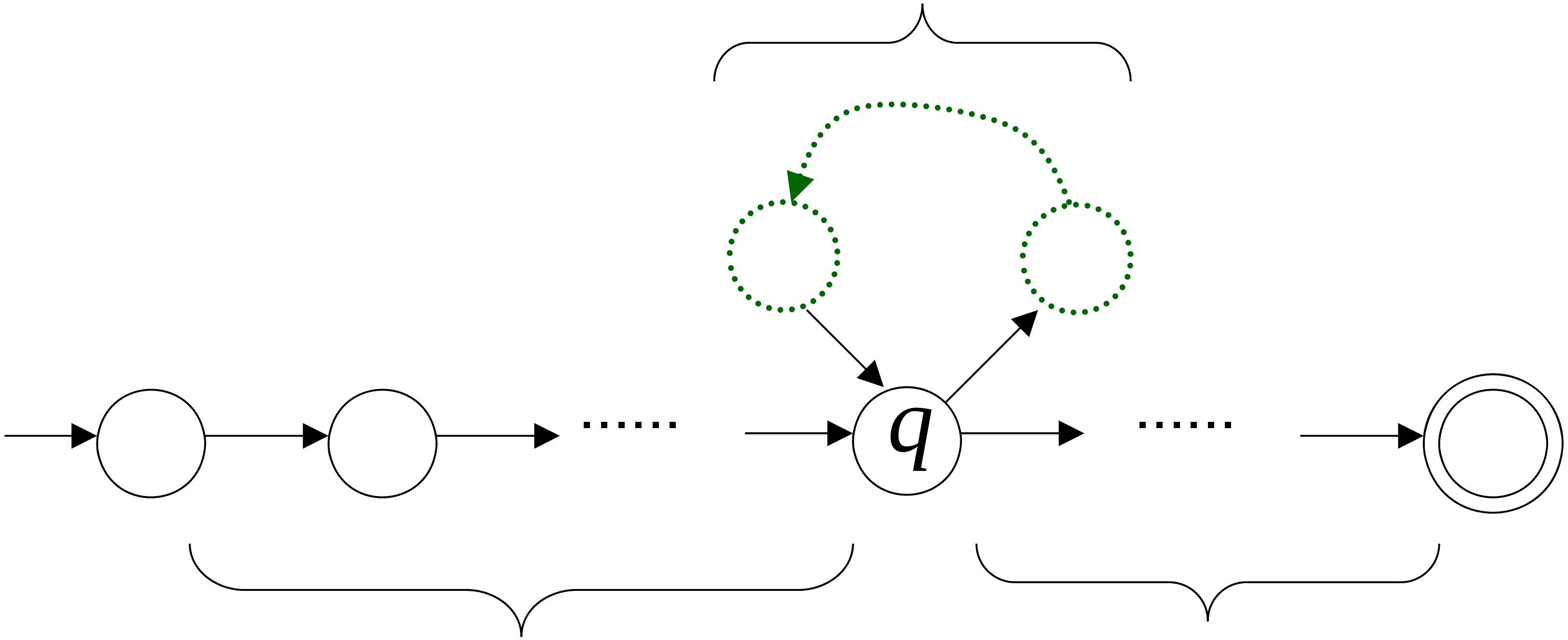
In General: The string $x\,y^i\,z$ is accepted $i = 0, 1, 2, \ldots$



In General: $x\, y^{i}\, z \in L \qquad i = 0, 1, 2, \ldots$

Language accepted by the DFA

$y = a_{i+1} a_{i+2} \ldots a_j$



$x = a_1 a_2 \ldots a_i$

$z = a_{j+1} a_{j+2} \ldots a_k$

In other words, we described:

The Pumping Lemma !!!

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

The Pumping Lemma Game

- Goal: Win the game by establishing a contradiction of the pumping lemma

O Picks m

P Picks a string w in L of length equal or greater than m. We are free to choose any w, subject to w ϵ L and |w| ≥ m.

O Chooses the decomposition xyz, subject to |xy| ≤ m, |y| ≥ 1.

P Picks i such that the pumped string w_i is not in L.

Applications

of

the Pumping Lemma

Example 4.7

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

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

O

Let m be the integer in the Pumping Lemma

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

length $|w| \geq m$

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

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

From the Pumping Lemma

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

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

$$\underbrace{a \ldots a}_{x} \underbrace{a \ldots a}_{y} \underbrace{a \ldots a b \ldots b}_{z}$$

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

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

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

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

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

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

P

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

$$\overbrace{\underbrace{a...a}_{x}\underbrace{a...a}_{y}\underbrace{a...a}_{y}a...a}^{m+k}$$

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

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!!!

Example 4.8

- Show that $L = \{ww^R : w \in \Sigma^*\}$ is not regular

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{ww^R : w \in \Sigma^*\}$$

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

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

(with the first m spanning all the leading a's)

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

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

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

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

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

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

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

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

with z the part $a...ab...bb...ba...a$ after the second y.

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: $L = \{ww^R : w \in \Sigma^*\}$

$$\Downarrow$$

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

CONTRADICTION!!!

$$L = \{ww^R : w \in \Sigma^*\}$$

If we choose $w = a^{2m} \in L$

The opponent picks $y = a^k$?

To apply the pumping lemma, we assume that the opponent will make the best move.
Ex. $y = aa$

Example 4.9

- Let Σ = {a, b}. The language

  $L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$ is not regular

O Given m

P Picks w = $a^m b^{m+1}$

O $|xy| \leq m$ → picks y with all a's → $y = a^k$, $1 \leq k \leq m$

P Picks i = 2 → $w_2 = a^{m+k} b^{m+1}$ is not in L

Example 4.10

- Let Σ = {a, b}. The language

  $L = \{(ab)^n a^k : n > k, k \geq 0\}$ is not regular

O Given m

P Picks w = $(ab)^{m+1}a^m$

O |xy| ≤ m → picks y = a (or ab)

P Picks i = 0 → $w_0 = (ab)^p b(ab)^q a^m$ is not in L

($w_0 = (ab)^m a^m$ is not in L)

Example 4.11

- Let Σ = {a}. The language
  $L = \{a^n : n \text{ is a perfect square}\}$ is not regular

O Given m

P Picks $w = a^{m^2}$

O $|xy| \leq m$ → picks $y = a^k$, $1 \leq k \leq m$

P Picks $i = 0$ → $w_0 = a^{m^2-k}$ is not in L

∵ $m^2-k > (m-1)^2$

Example 4.12

- Let $\Sigma = \{a, b, c\}$. The language $L = \{a^n b^k c^{n+k} : n \geq 0, k \geq 0\}$ is not regular

O Given m

P Picks $w = a^m b^m c^{2m}$

O $|xy| \leq m \rightarrow$ picks $y = a^k$, $1 \leq k \leq m$

P Picks $i = 0 \rightarrow w_0 = a^{m-k} b^m c^{2m}$ is not in L

Use homomorphism $h(a) = a$, $h(b) = a$, $h(c) = c$

$\rightarrow h(L) = \{a^{n+k} c^{n+k} : n+k \geq 0\}$

Example 4.13

- Let Σ = {a, b}. The language $L = \{a^n b^l : n \neq l\}$ is not regular

Set n = l + 1?

$$L_1 = \overline{L} \cap L(a^* b^*)$$

O Given m

P Picks $w = a^{m!} b^{(m+1)!}$

O |xy| ≤ m → picks $y = a^k$, $1 \leq k \leq m$

P Pumps i times → $w_i = a^{m!+(i-1)k} b^{(m+1)!}$

$$if \ \exists i \quad \text{s.t.} \quad \mathrm{m!+(i-1)k = (m+1)!}$$

$$\mathrm{i = 1 + \frac{mm!}{k}}$$

∵ k ≤ m → i is an integer

Common Pitfalls Using Pumping Lemma

- Use pumping lemma to show that a language is regular
- Start with a string not in L
- Make some assumptions about the decomposition xyz

Common Pitfalls Using Pumping Lemma

- Use pumping lemma to show that a language is regular
  - Even if you can show that no string in a language L can ever be pumped out, you cannot conclude that L is regular.
- Start with a string not in L
- Make some assumptions about the decomposition xyz

Common Pitfalls Using Pumping Lemma

- Use pumping lemma to show that a language is regular
- Start with a string not in L
  - EX. L = {a^n: n is a prime number}
  - Given m, let w = a^m (incorrect)
  - Given m, let w = a^P, where P is a prime number larger than m
- Make some assumptions about the decomposition xyz

Common Pitfalls Using Pumping Lemma

- Use pumping lemma to show that a language is regular
- Start with a string not in L
- Make some assumptions about the decomposition xyz
  - EX. L = {a^n: n is a prime number}
  - y = a^k, with k odd. Then w = xz is an even-length string and thus not in L (incorrect)

More Example

- Let $\Sigma = \{a\}$. The language

  $L = \{a^n$: n is a prime number$\}$ is not regular

O Take p to be the smallest prime # $\geq m$

P Picks $w = a^p$

O $|xy| \leq m \rightarrow$ picks y with all a's $\rightarrow$ $y = a^k, 1 \leq k \leq m$

P Pumps i times$\rightarrow$ $w_i = a^{p+(i-1)k}$

if we take $i-1=p$, then $p+(i-1)k=p(k+1)$ is composite and w_{p+1} is not in L

Short Quiz

Please use the pumping lemma to show that each of these languages is nonregular:

$$A = \{x \in \{0,1\}^* \ : \ \text{the length of } x \text{ is odd, and its middle symbol is } 1\}$$

$$\{a^n b^n a^m \text{ where } n = 0, 1, 2, \ldots \text{ and } m = 0, 1, 2, \ldots\}$$

Questions?