Automata Formal Languages & Logic

Preet Kanwal

Department of Computer Science & Engineering

Automata Formal Languages & Logic

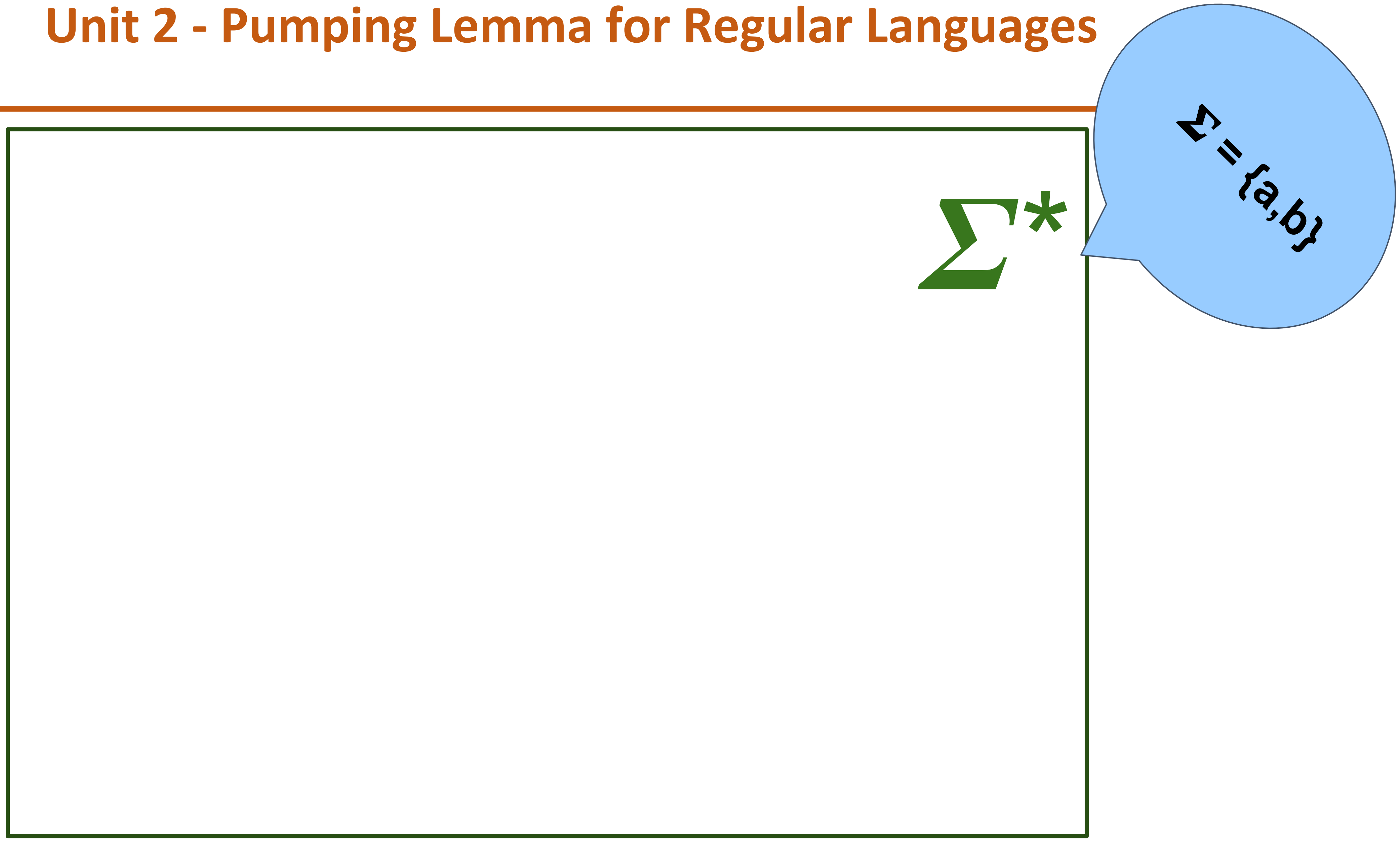
Unit 2

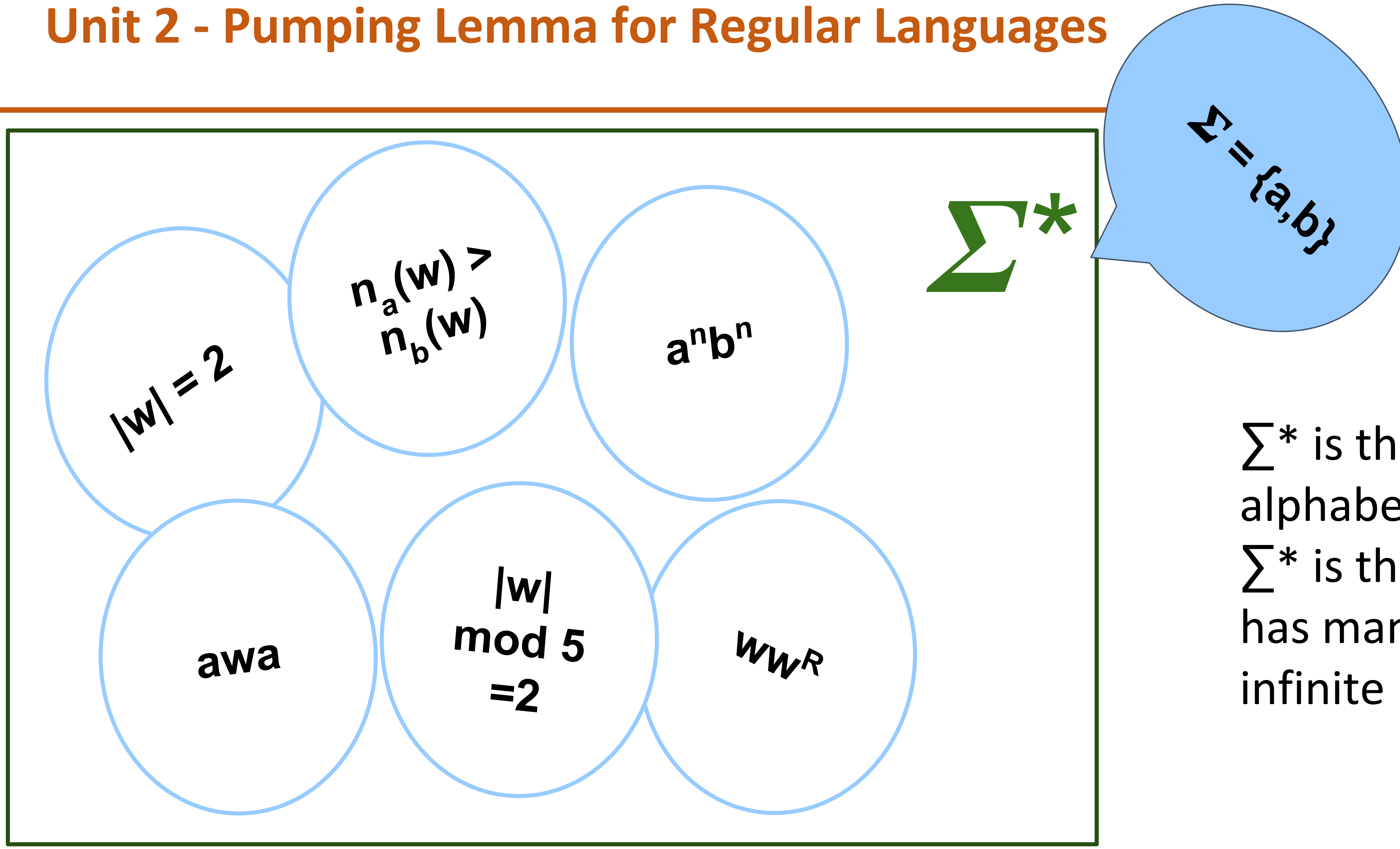
Preet Kanwal

Department of Computer Science Engineering

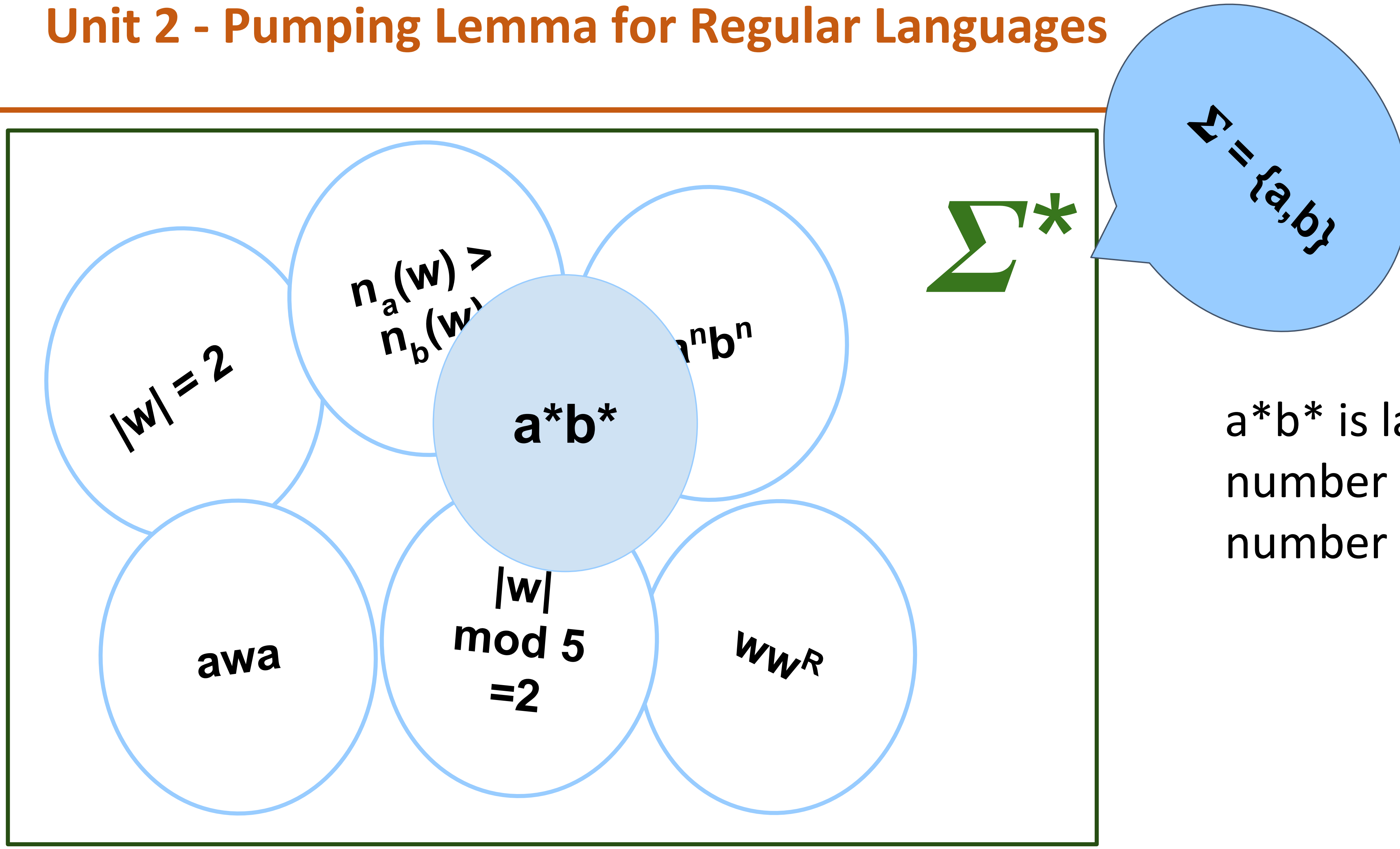
Σ^*

Unit 2 - Pumping Lemma for Regular Languages

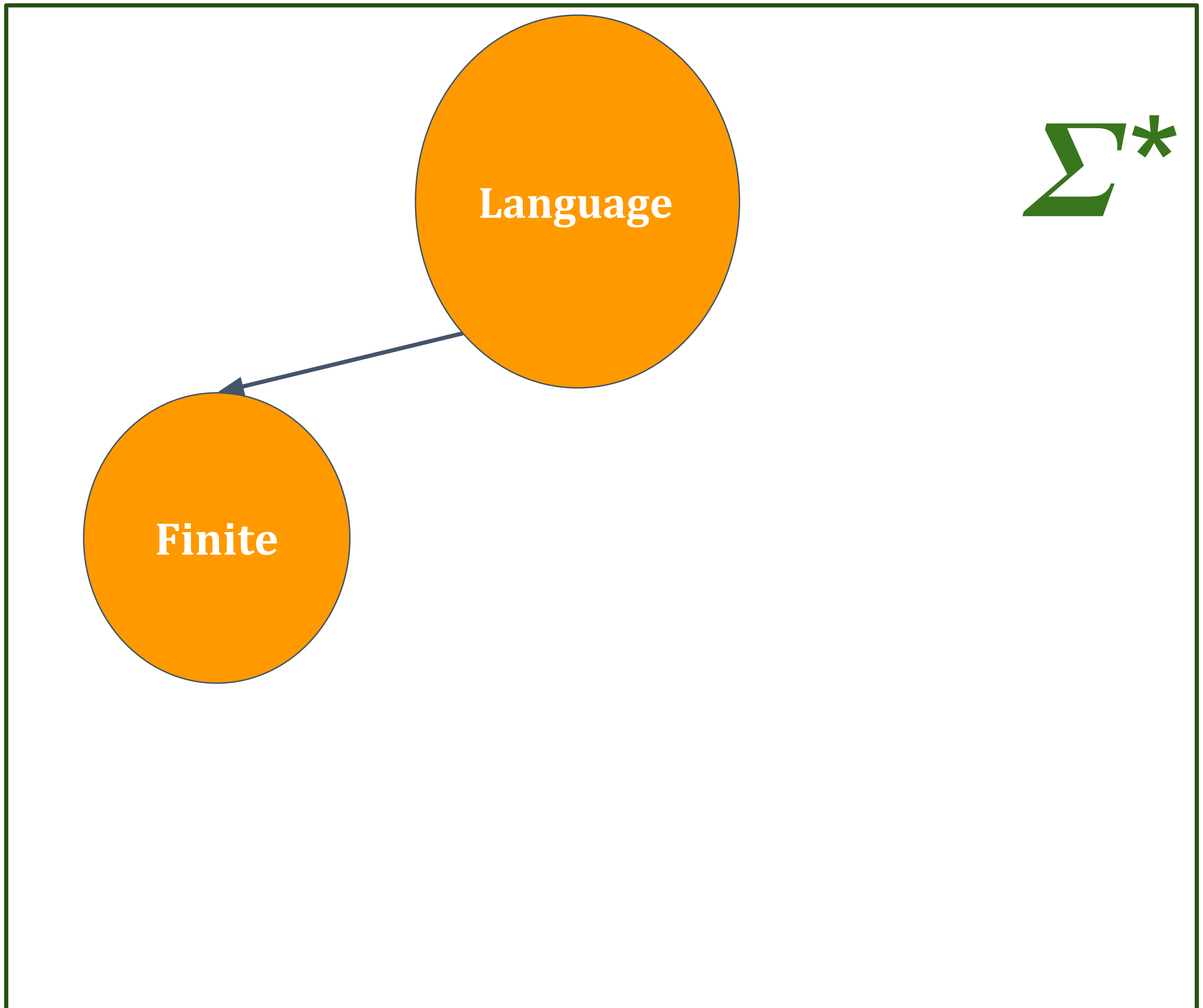




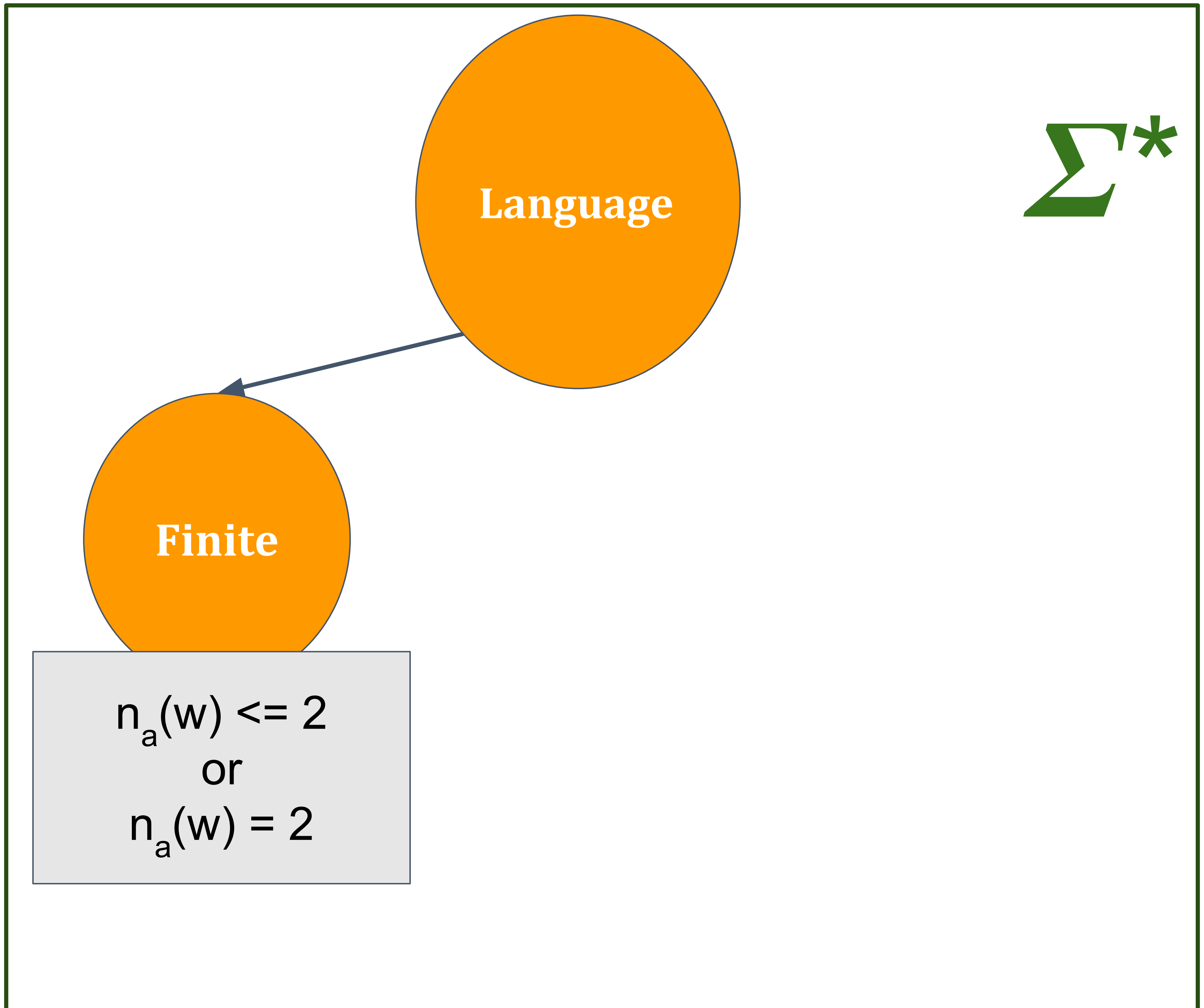
∑* is the set of all strings over the alphabet ∑={a,b}
∑* is the universal language where it has many finite languages and infinite languages as shown in figure.

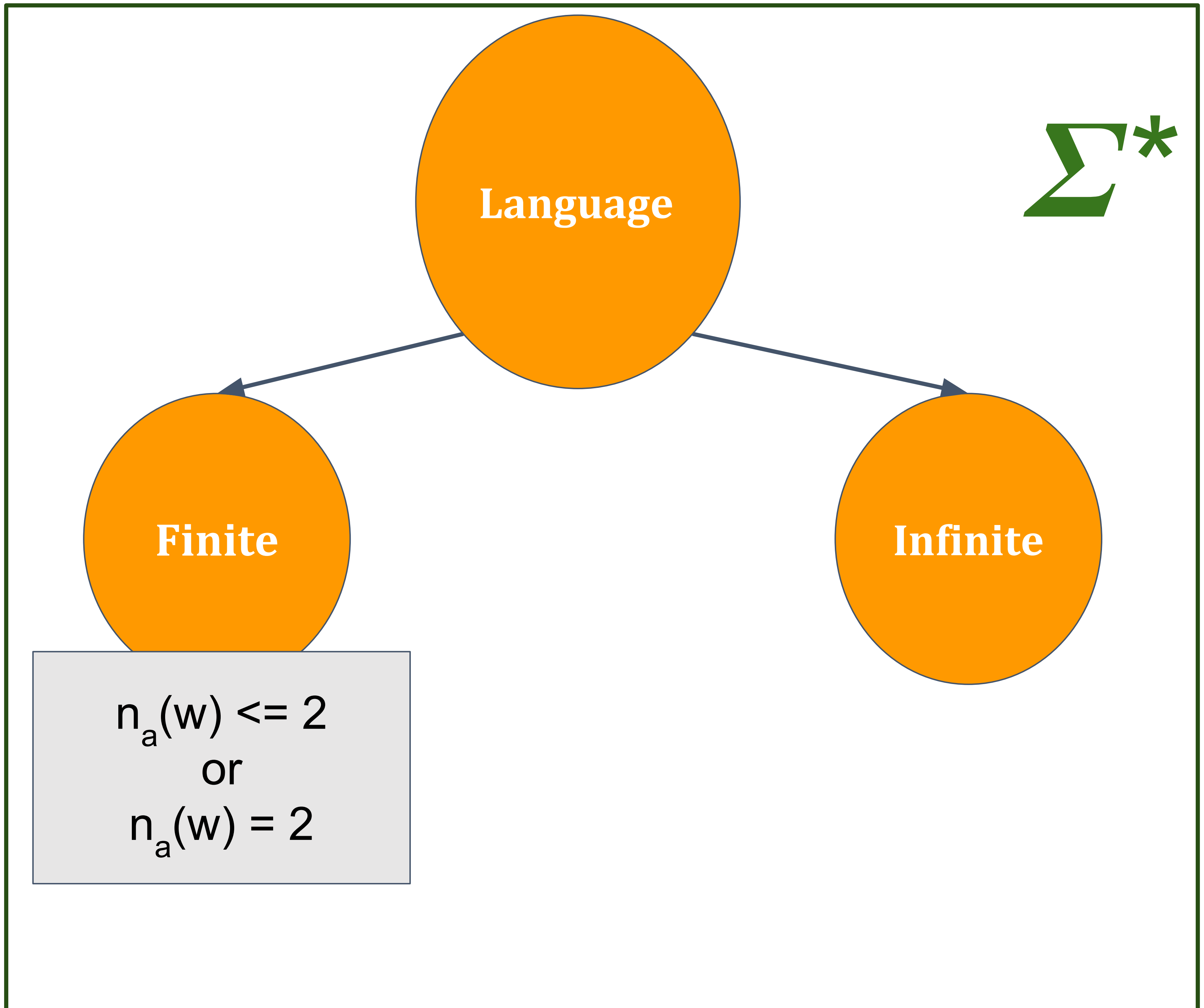


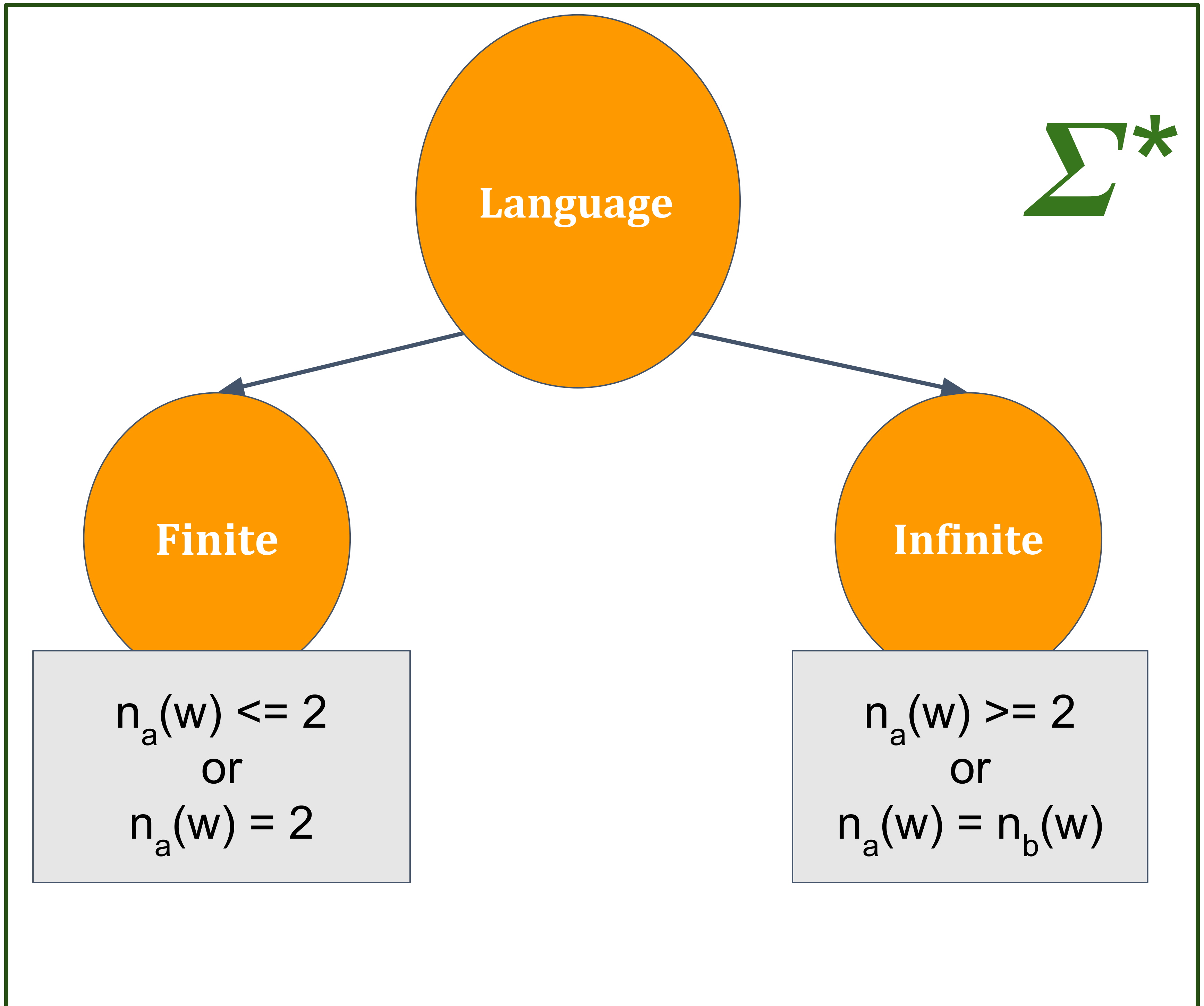
a*b* is language which contains any number of a's followed by any number of b's.


Language
Finite
Σ^*

Unit 2 - Pumping Lemma for Regular Languages




Σ^*
Language
Finite
Infinite
$n_a(w) <= 2$
or
$n_a(w) = 2$



Formal language is of two types.

1. finite language
2. infinite language

   with example shown in figure.

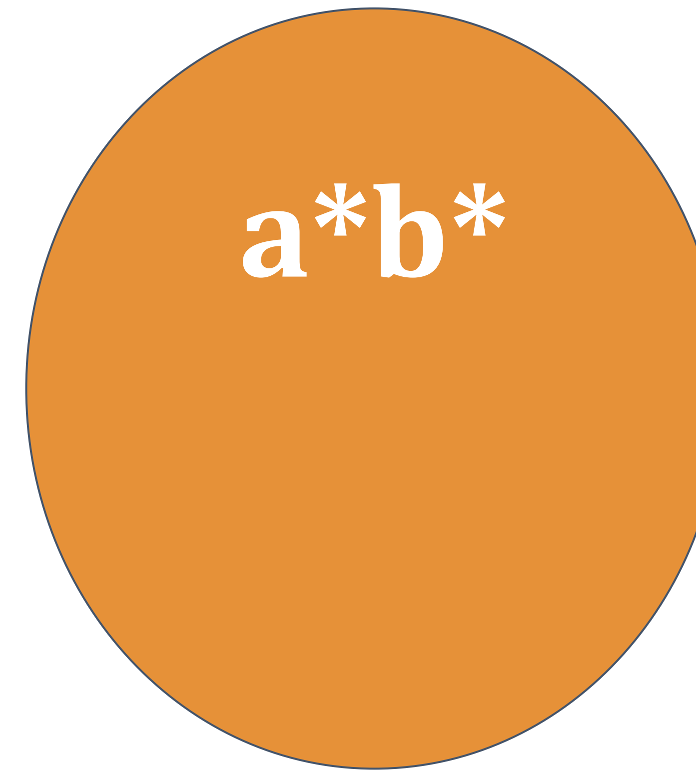
The regular languages can be finite or infinite languages and regular languages corresponds to

- DFA/NFA
- Regular grammar
- Regular Expression

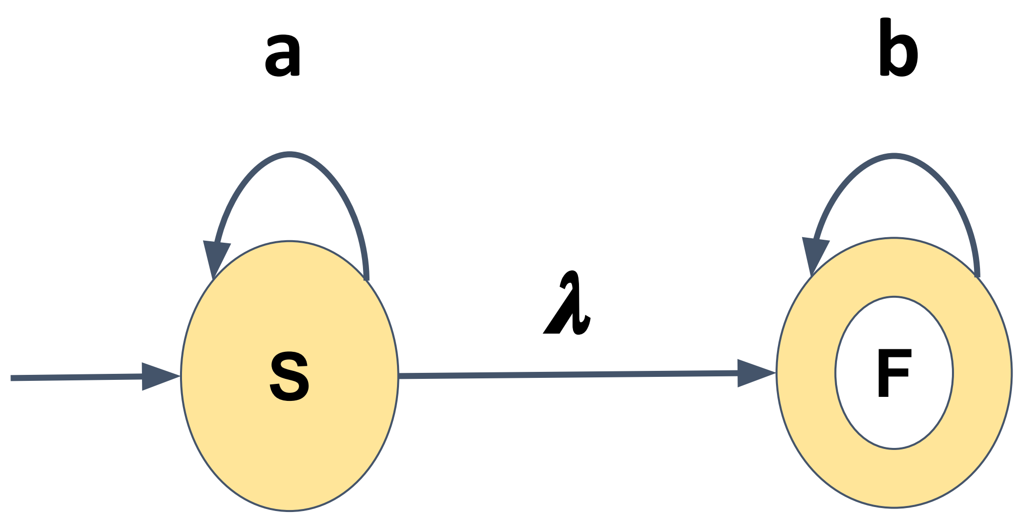
Is there any infinite language for which we cannot construct a Finite Automata?

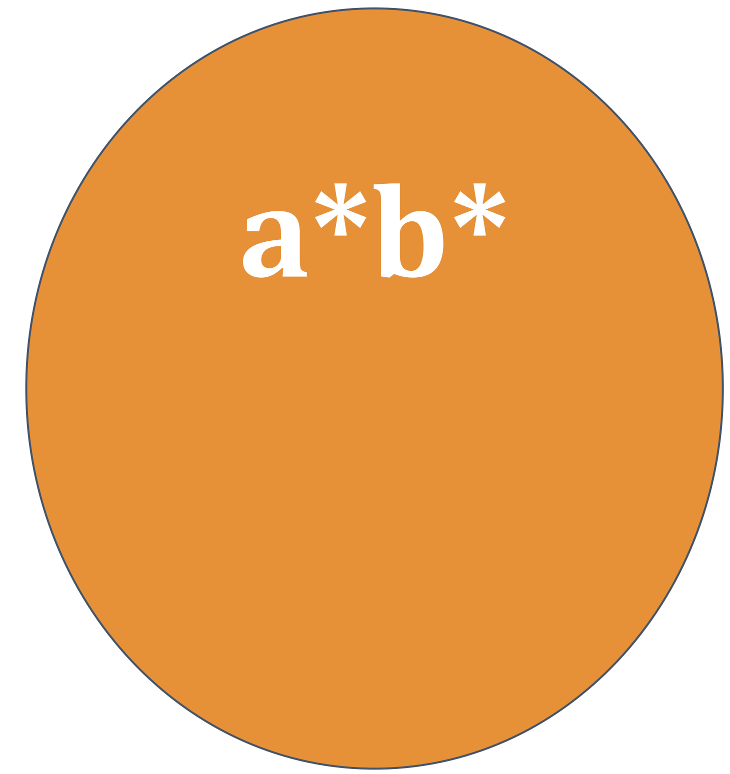
That means, a language which is not regular?

Let's look at an example :

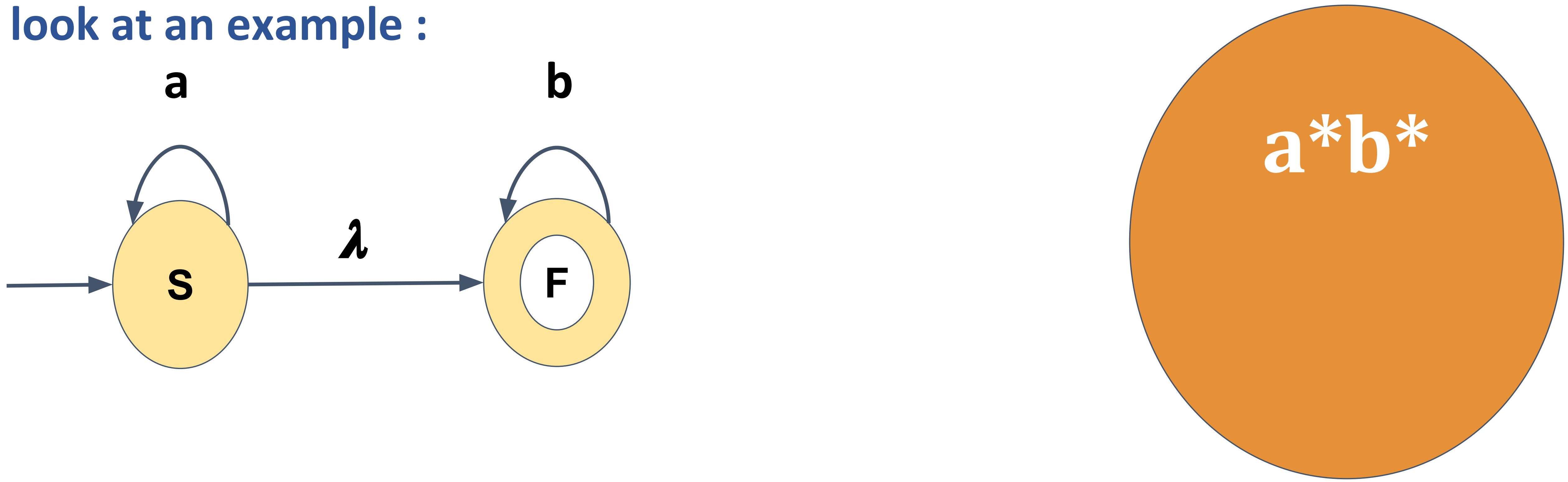


Let's look at an example :





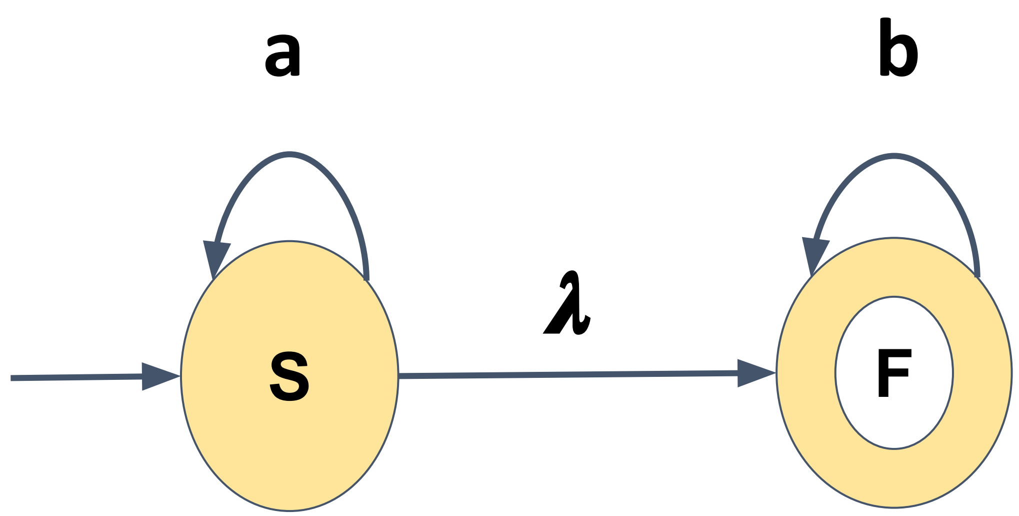
Let's look at an example :



S -> aS | F

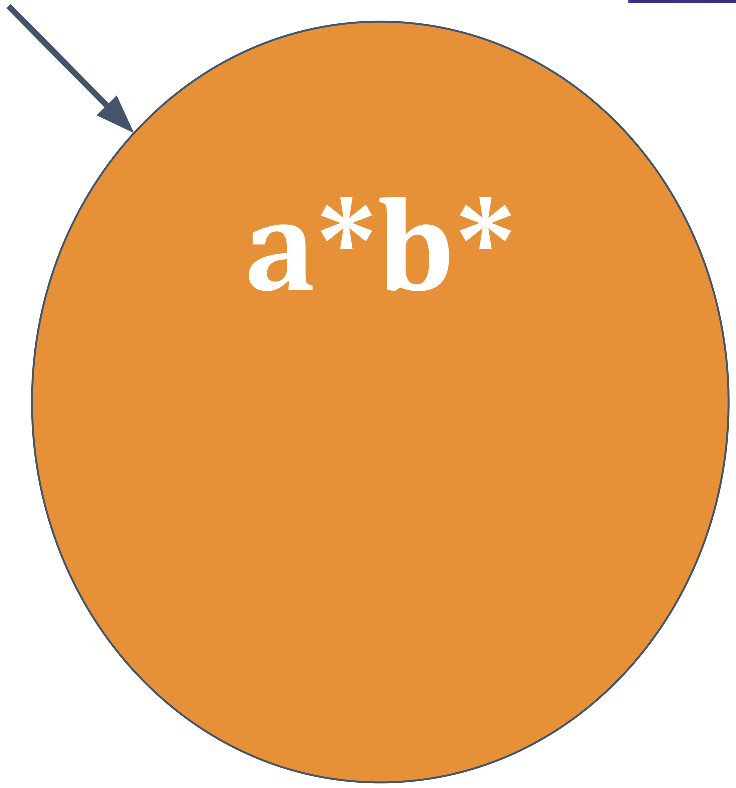
F -> bF | λ

Let's look at an example :



FA

Regex



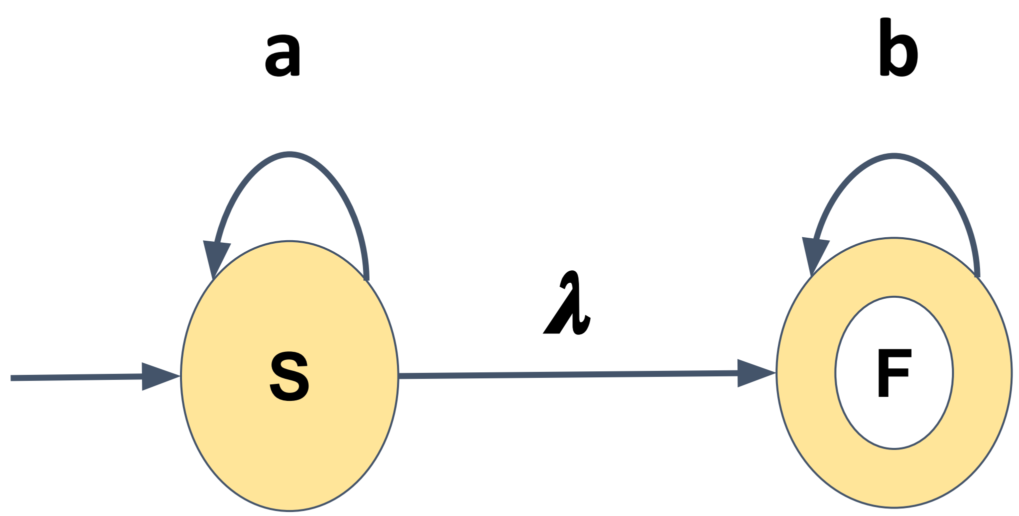
S -> aS | F

F -> bF | λ

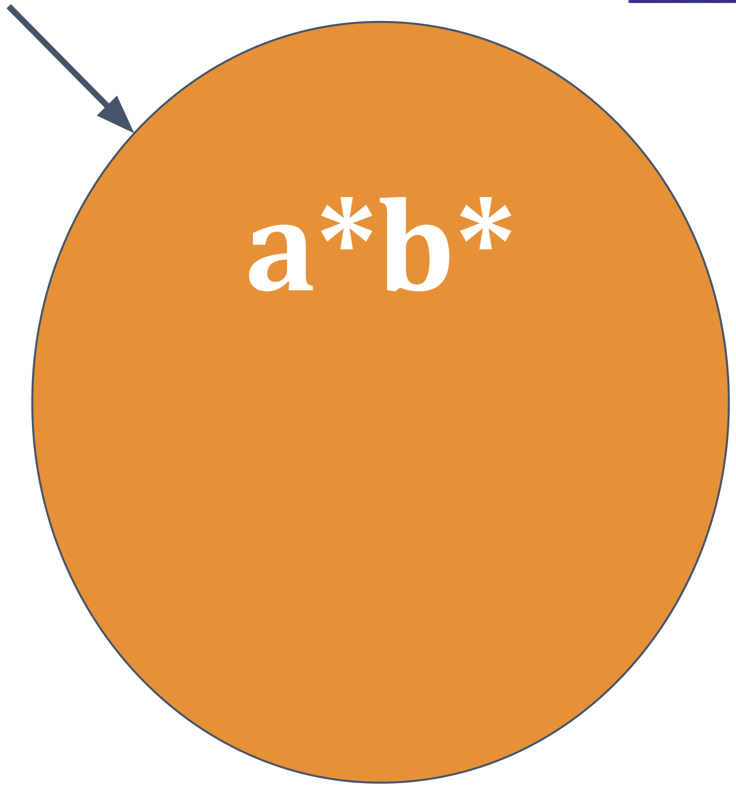
Regular Grammar

Let's look at an example :

FA



Regex



Definitely Regular !

S -> aS | F

F -> bF | λ

Regular Grammar

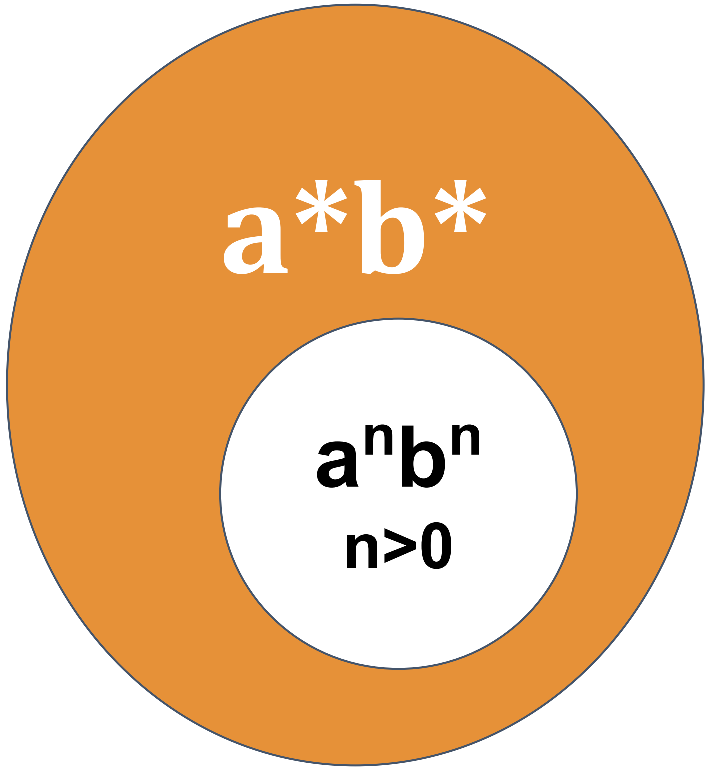
For the language a^nb^n Can we construct either of :

Finite Acceptor or

Regular Grammar or

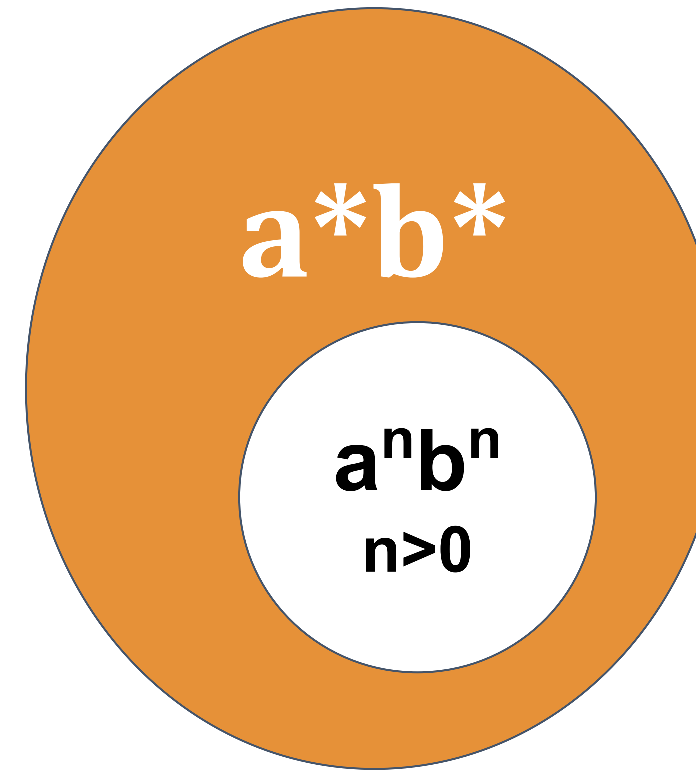
Regular Expression

????????????

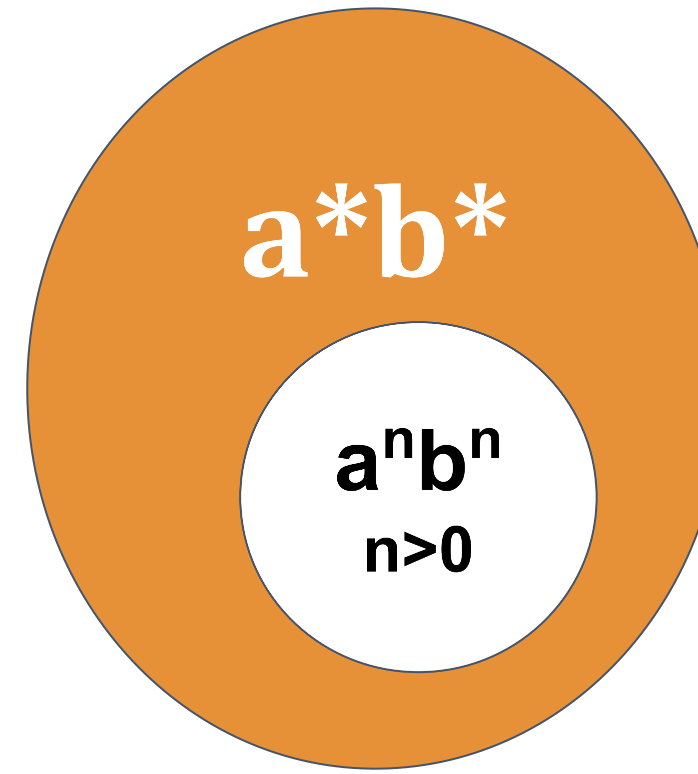


aaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbb

state q



Basically,



There is no way to remember how many a’s you have seen to compare with the upcoming b’s !

The value of n could be anything!

We cannot come up with a FA that takes care of all n!

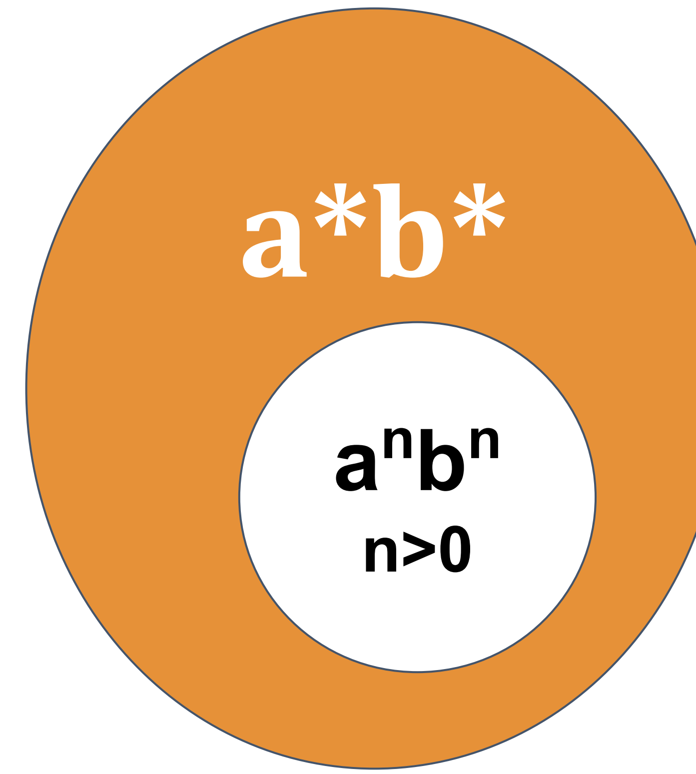
Basically,

There is no way to remember ho[illegible] you have seen to compare with th[illegible]

Definitely Non-Regular !

The value o[illegible] could be anything!

We cannot come up with a FA that takes care of all n!



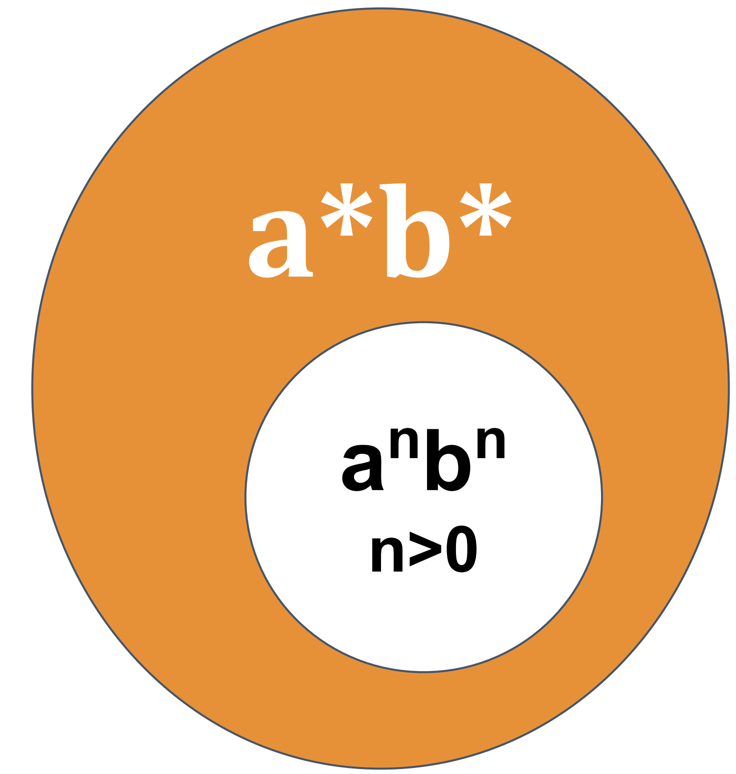
Basically,

There is no way to re... you have seen to compa...

The value ... n cou...

We cannot come up w...h a FA that takes care of all n!

Finite Automata has limits !



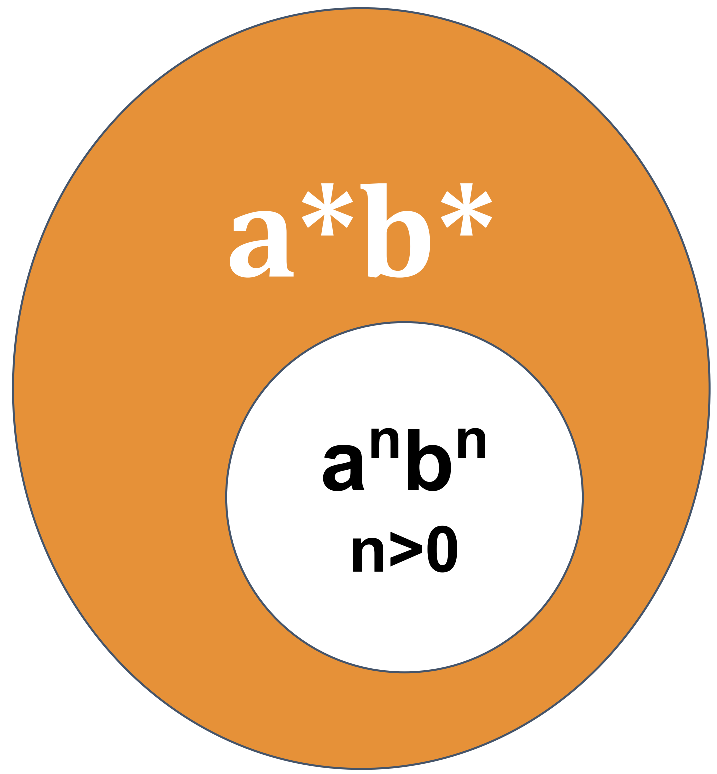
Limits of Finite Automata :

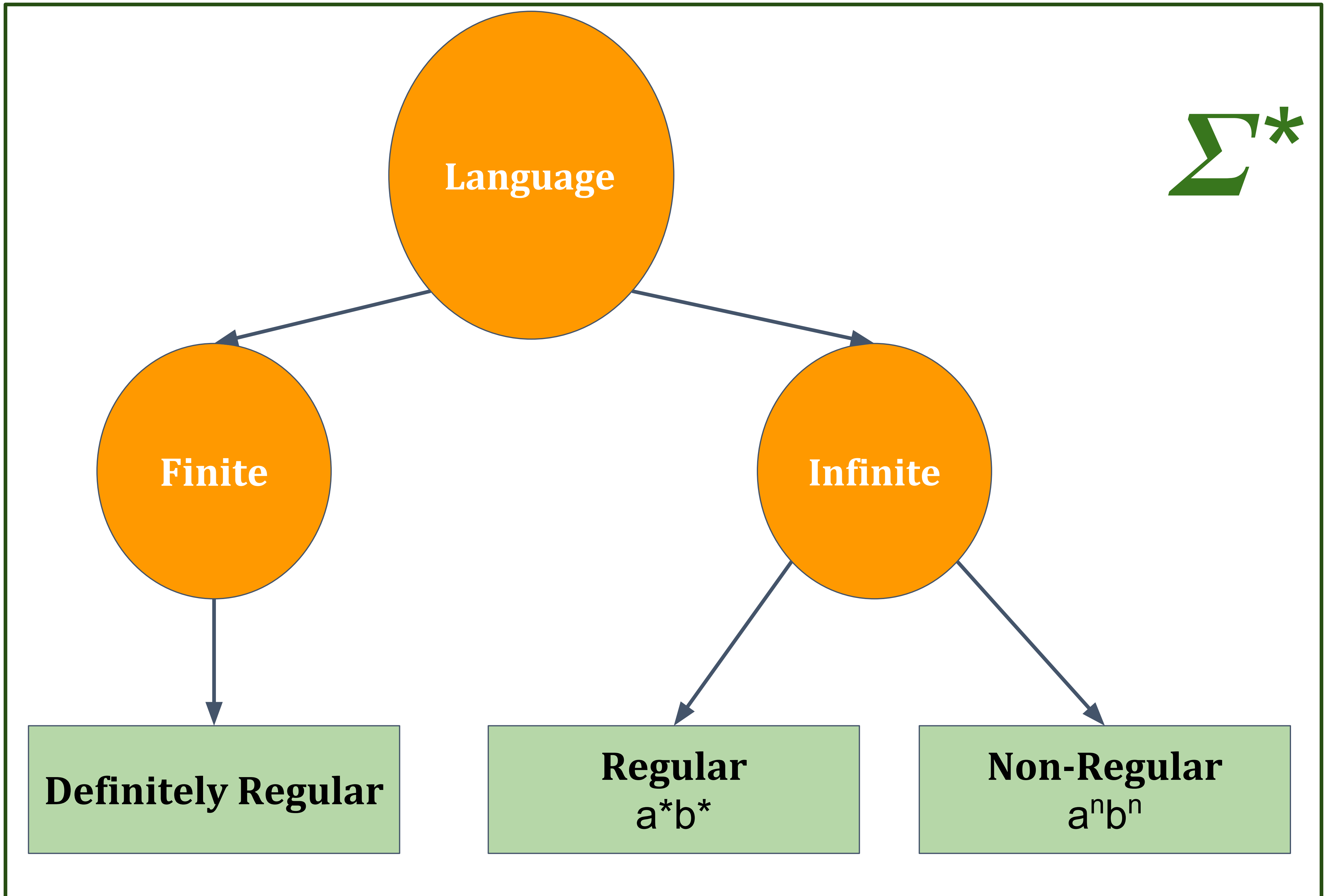
Finite States !

A finite automata can only "count"

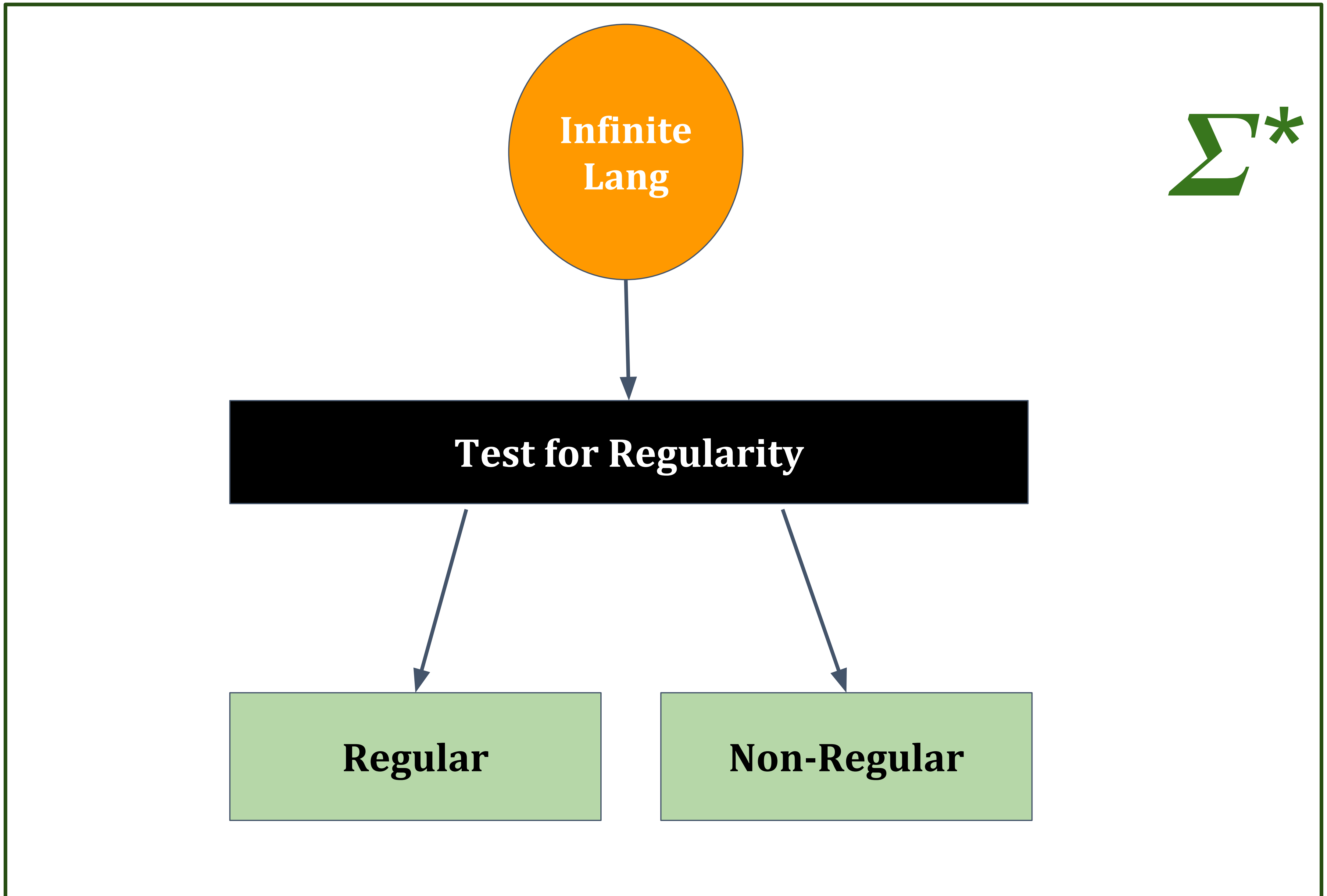
that is,

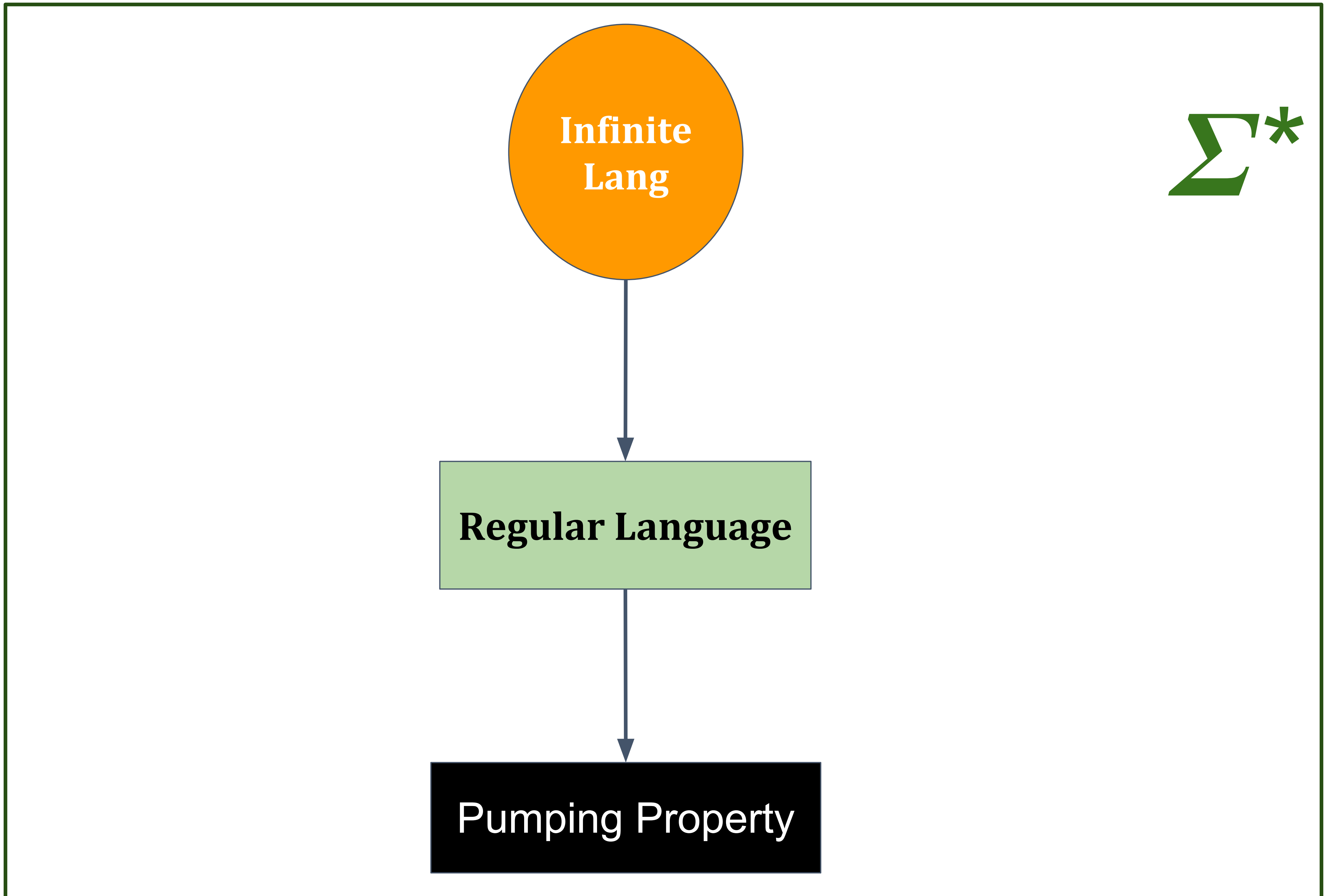
It can maintain a counter, where different states correspond to different values of the counter.

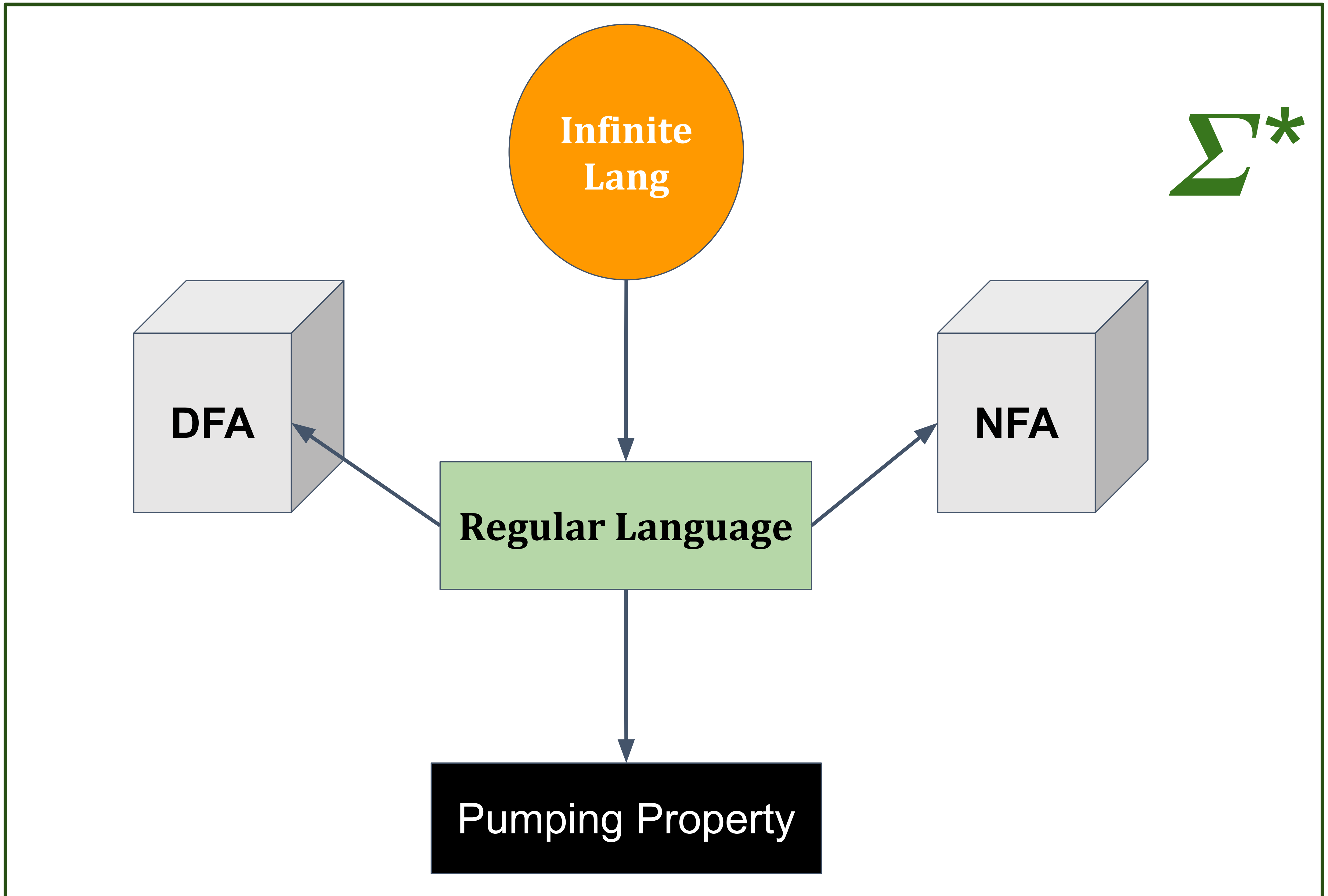



Σ*
Language
Finite
Infinite
Definitely Regular
Regular
a*b*
Non-Regular
a^n b^n

Languages which are not regular do not have a simple repeating pattern. Either they have no pattern at all or they have more than one pattern, with multiple patterns being correlated with each other.All of our mechanisms for dealing with regular languages - finite automata,RegEx and regular grammars are unable to deal with such languages.

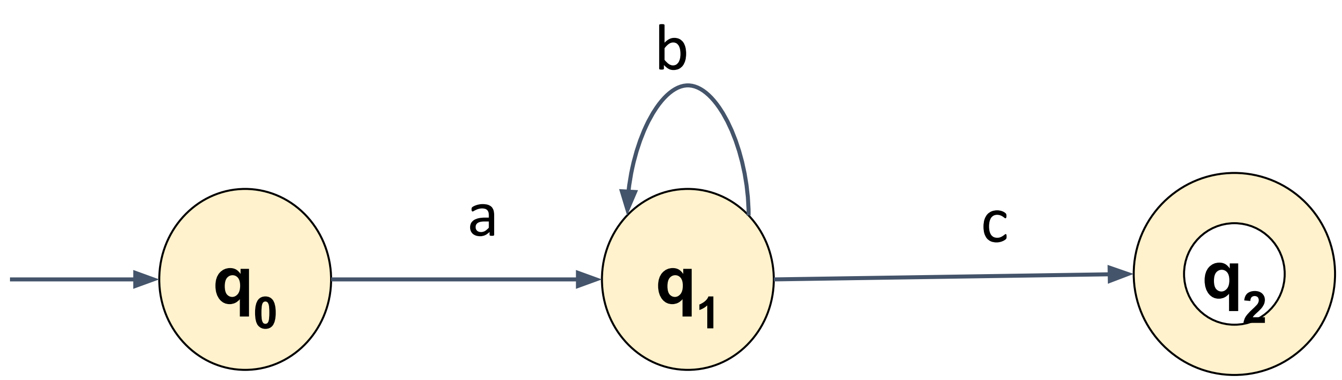

Infinite Lang
Test for Regularity
Regular
Non-Regular
Σ^*


Infinite Lang
Regular Language
Pumping Property
Σ^*

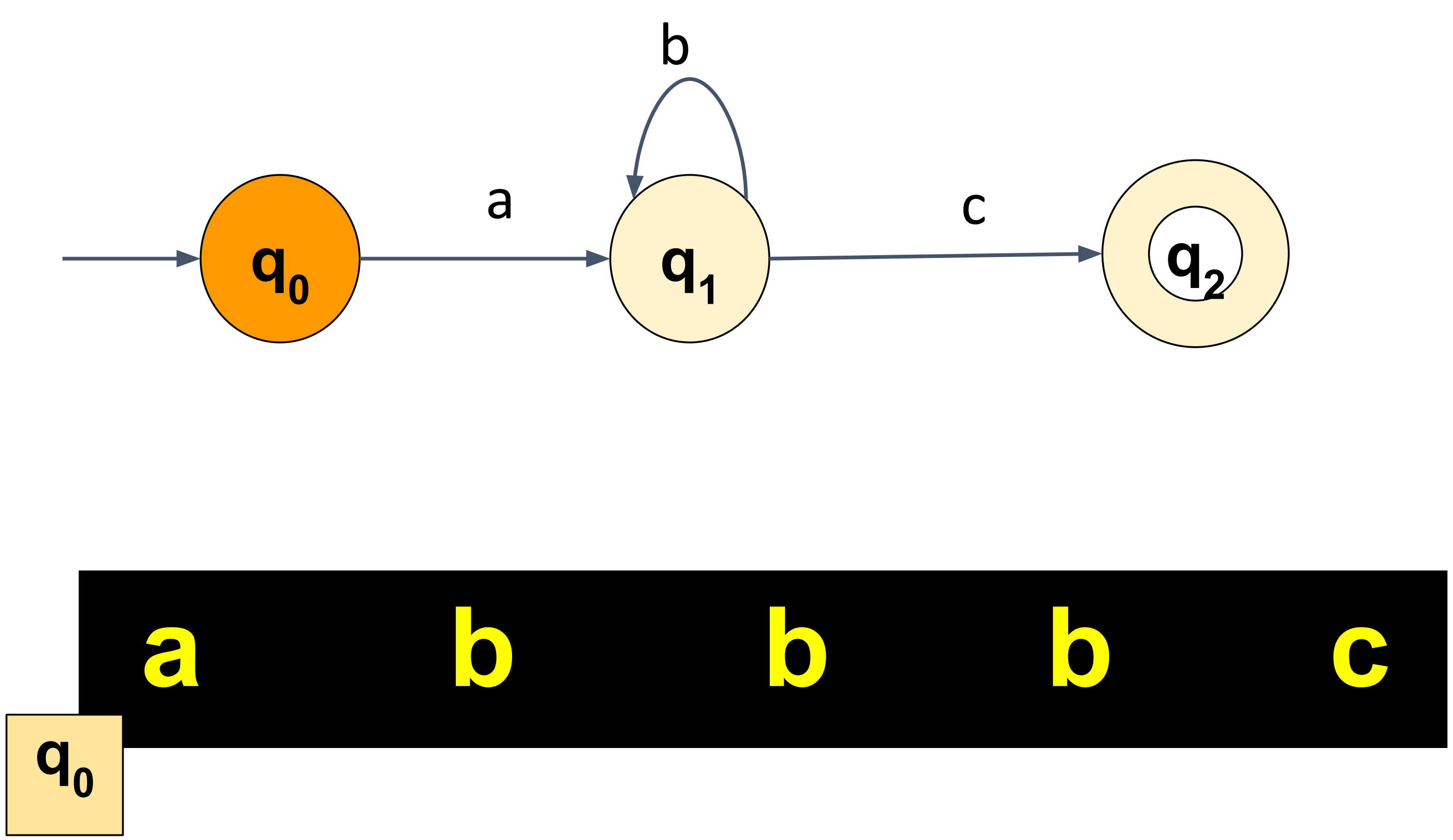

Infinite Lang
DFA
NFA
Regular Language
Pumping Property
Σ*

To understand the limitation of a finite number of states being able to handle only regular languages with their simple repeating pattern.
We use the pigeonhole principle to show the languages are not regular.
Pigeonhole principle says that only n things at most can fit into n slots or holes.
The repeating pattern is handled by the loop in the finite automaton for the language.

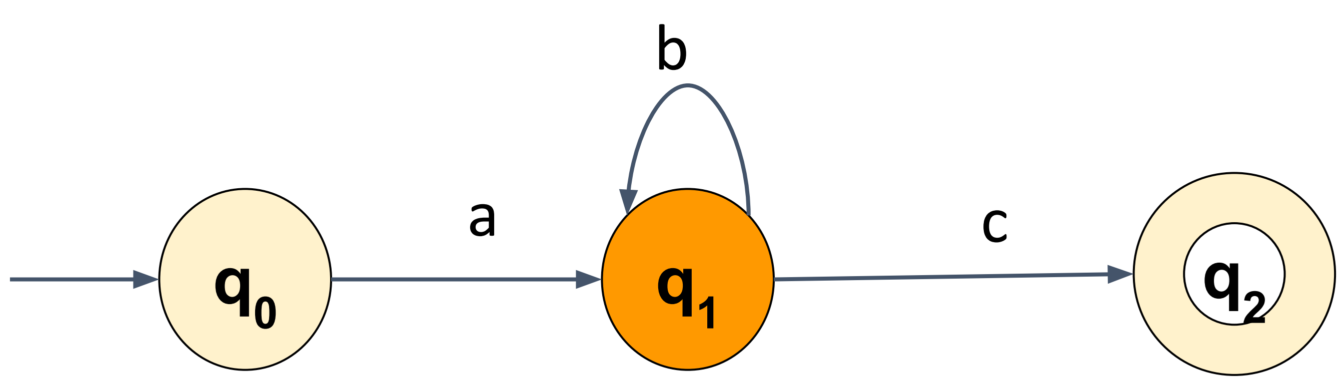
Let's take an example of infinite regular language ab*c

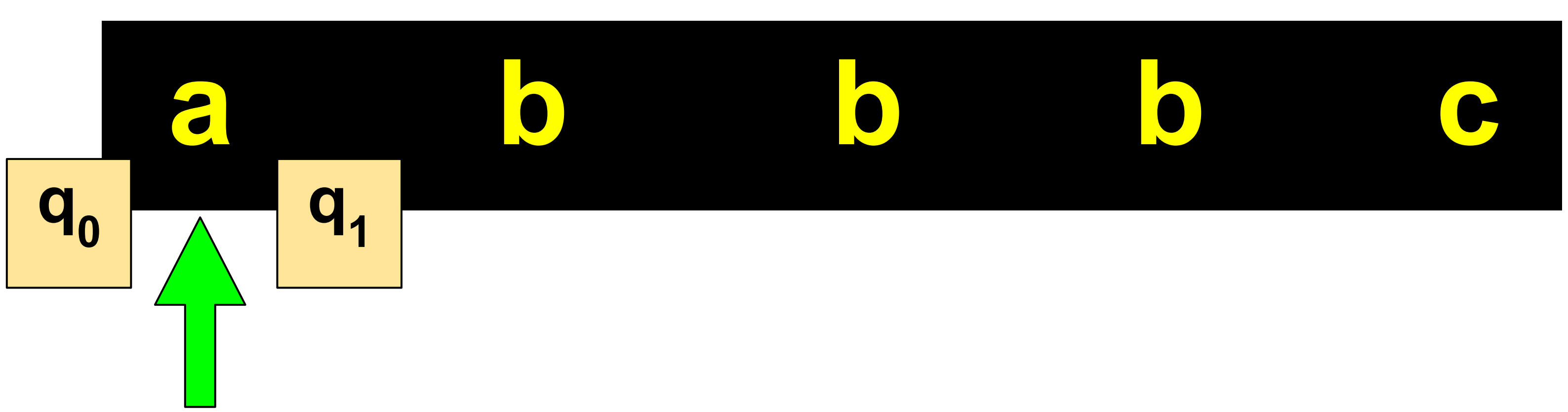


Let’s take an example of infinite regular language ab*c

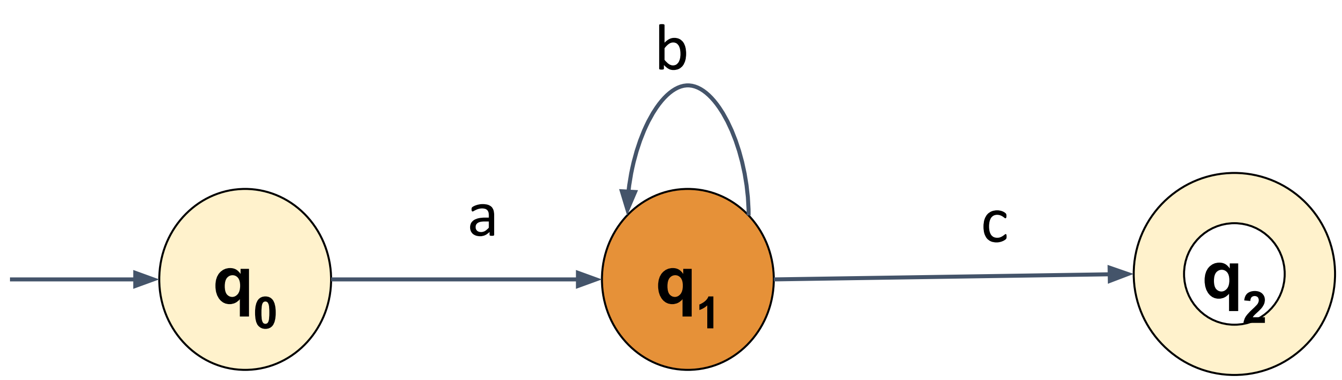


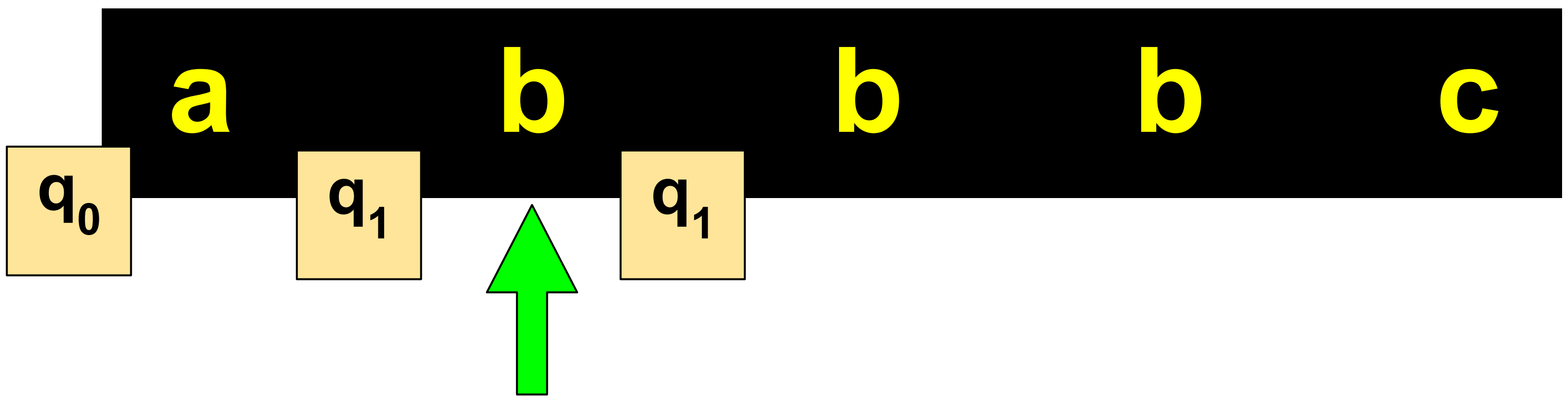
Let's take an example of infinite regular language ab*c



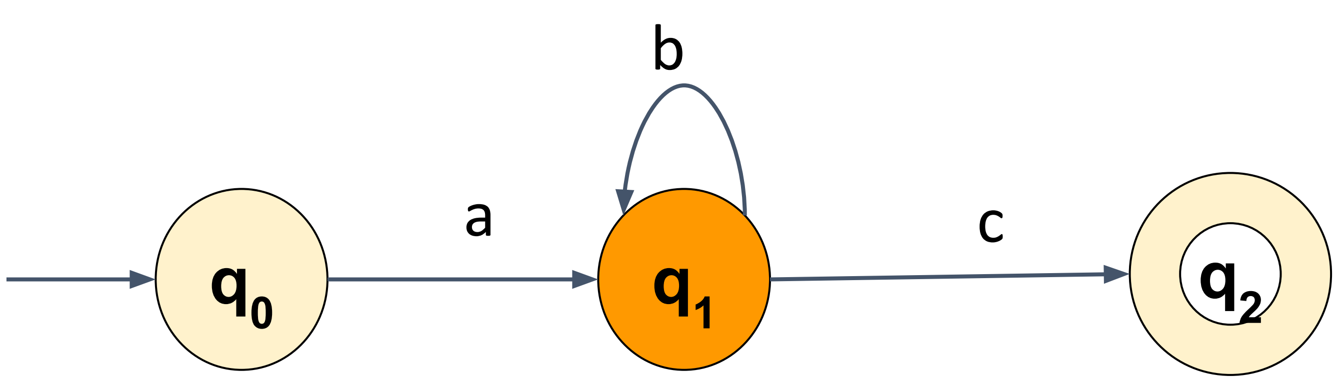


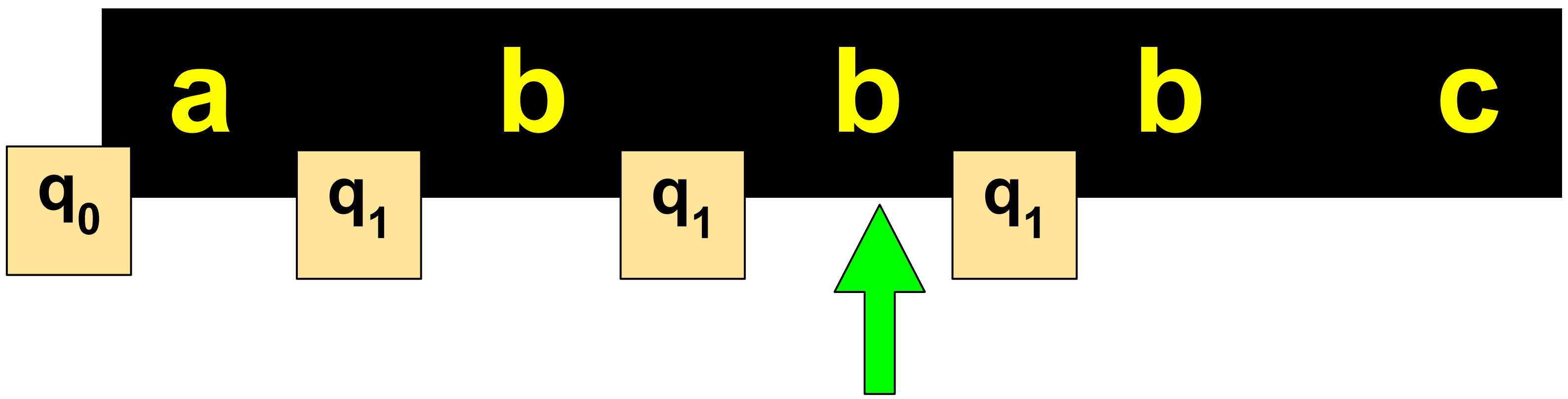
Let's take an example of infinite regular language ab*c



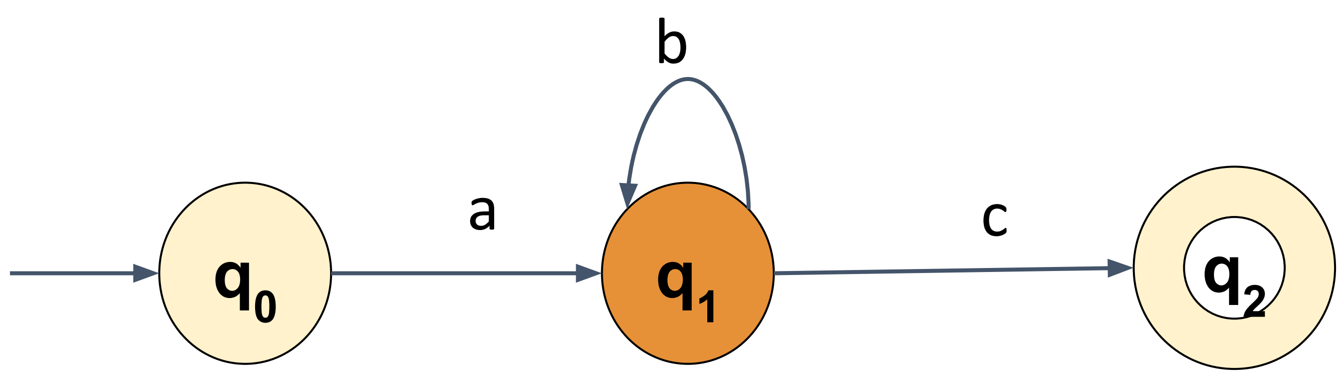


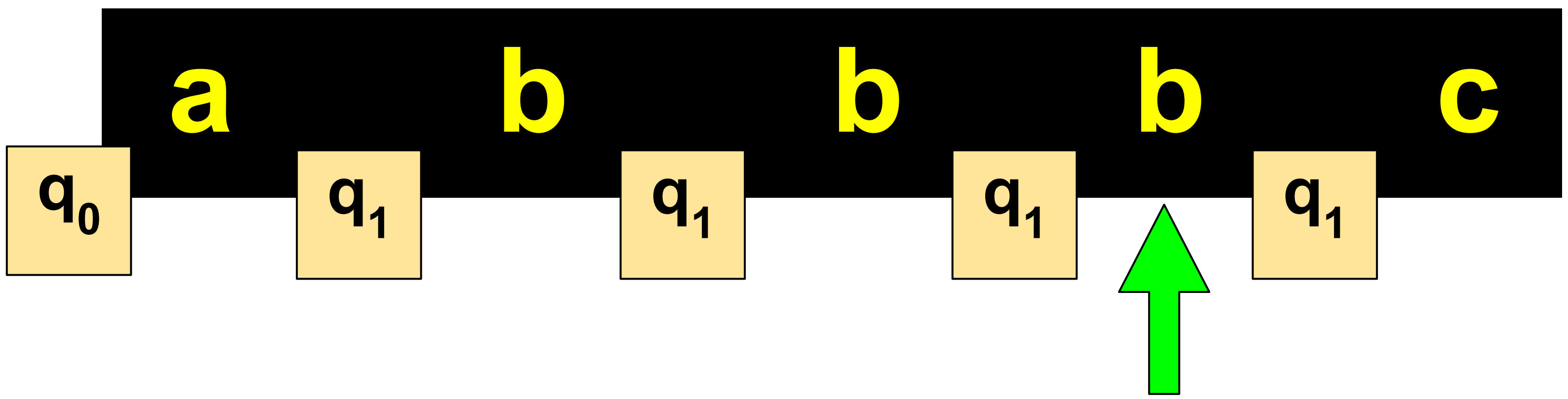
Let's take an example of infinite regular language ab*c





Let’s take an example of infinite regular language ab*c

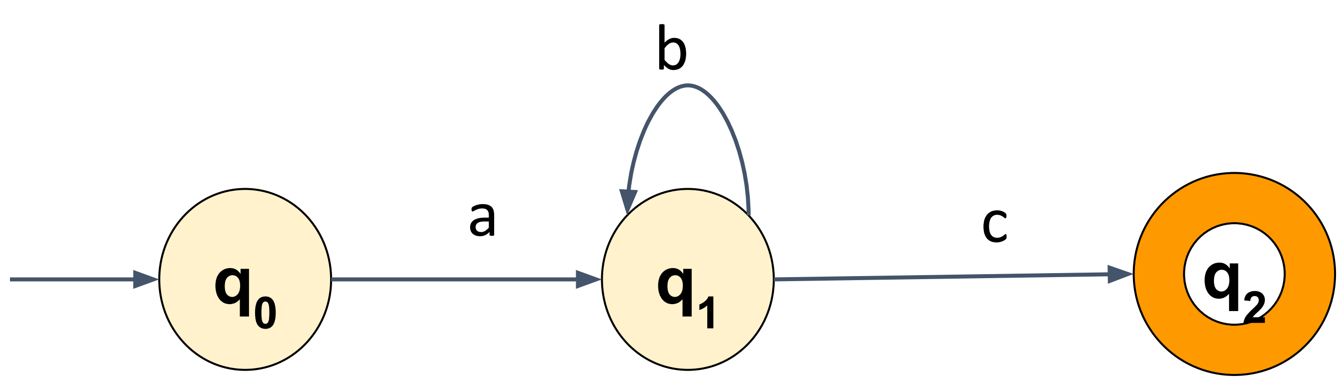


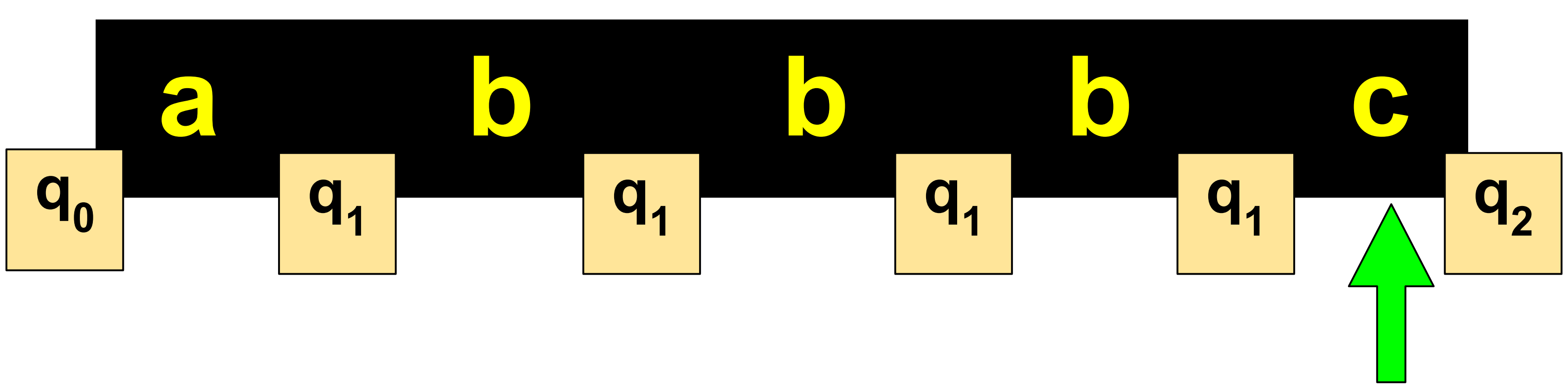


Let’s take an example of infinite regular language ab*c

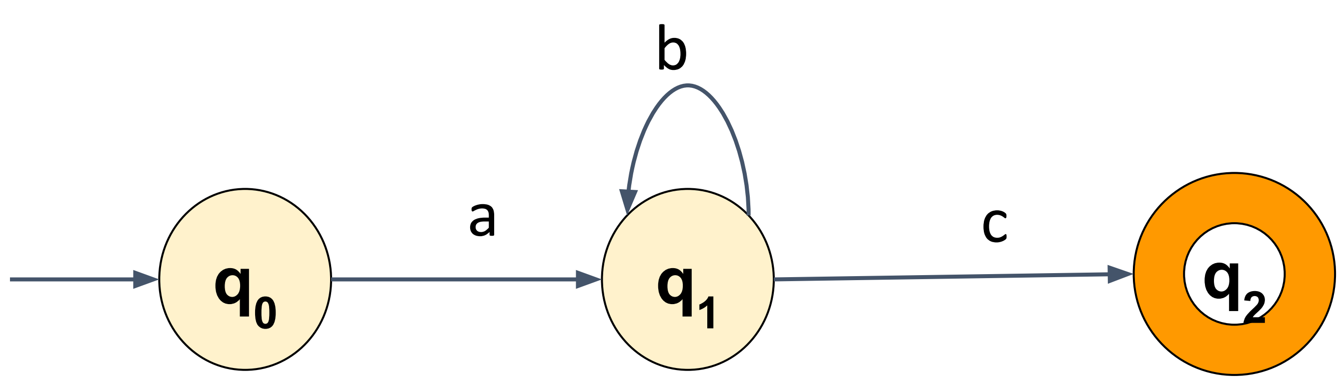
if |w| >= n

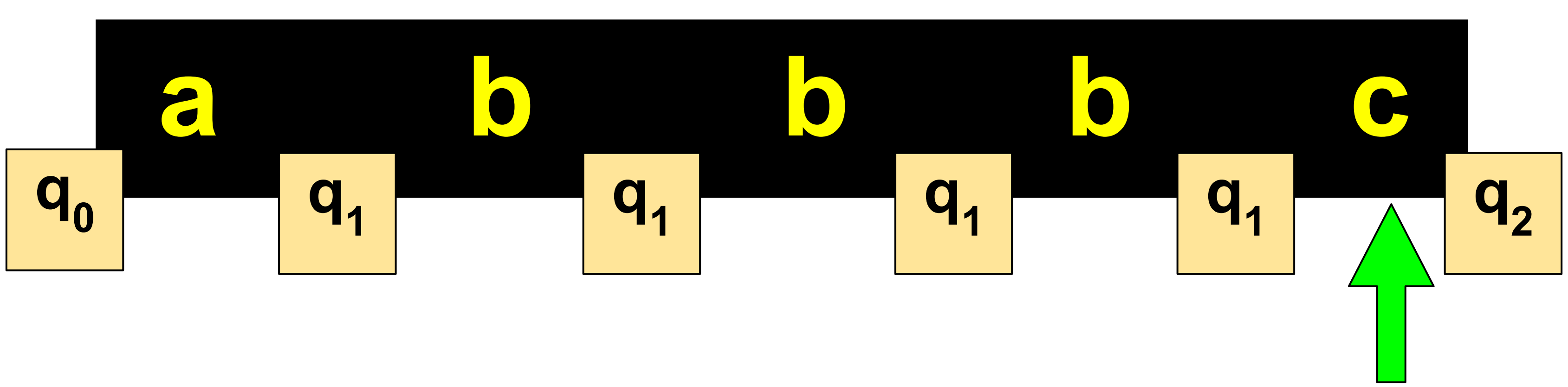
we visit a set of states more than once





Let’s take an example of infinite regular language ab*c





if |w| >= n

we visit a set of states more than once

The Pigeonhole Principle

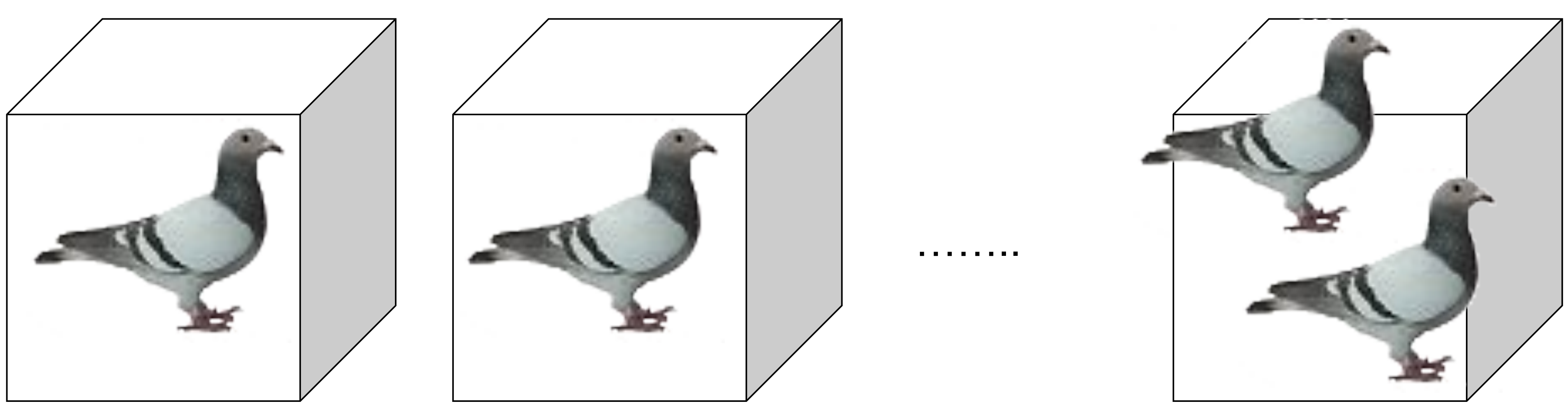
The Pigeonhole Principle

$$n > m$$

pigeons

pigeonholes

There is a pigeonhole
with more than 1 pigeon



The Pigeonhole Principle

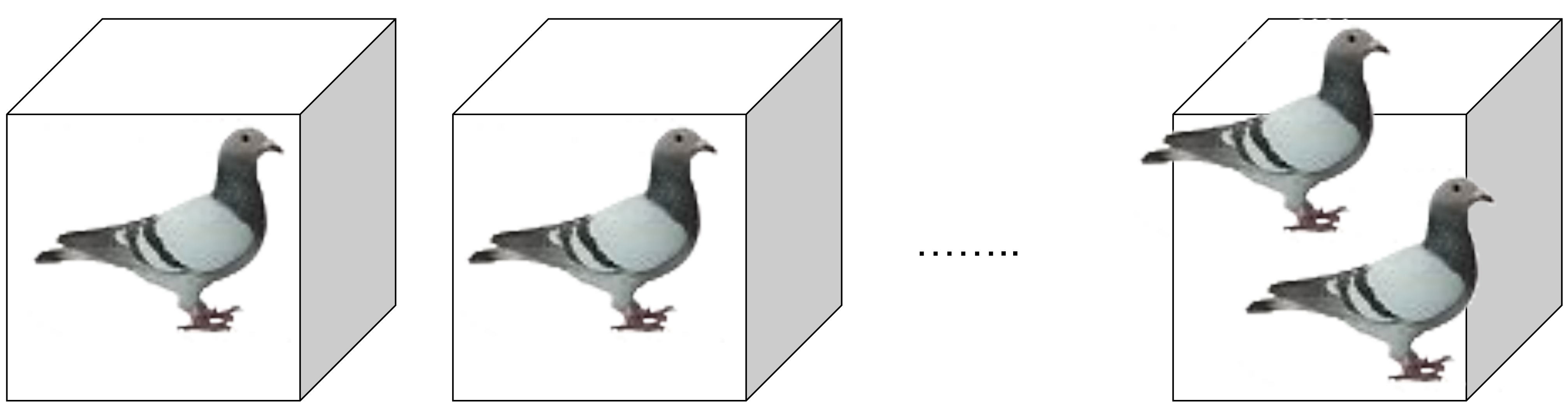
$$n > m$$

pigeons | pigeonholes

| String | | **# States n**

if |w| > n
A set of states will be visited more than once

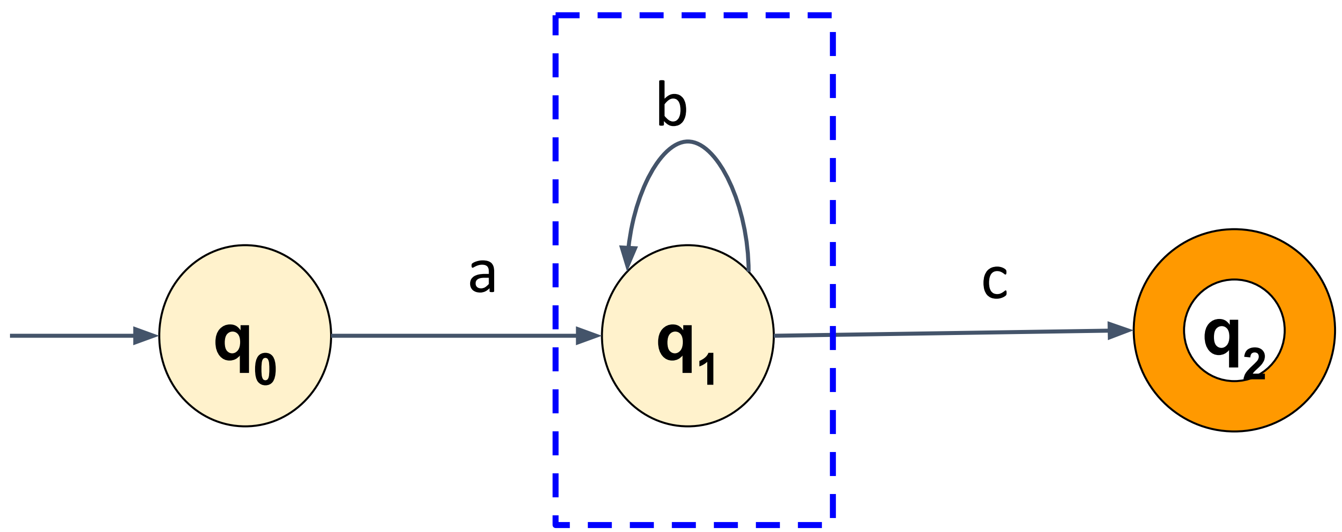
There is a pigeonhole with more than 1 pigeon

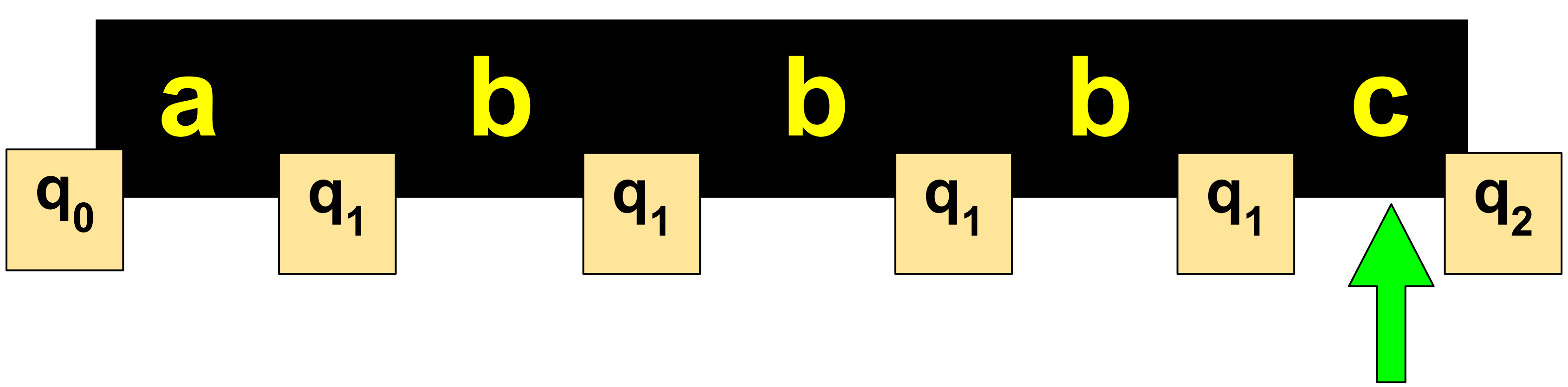


Visiting Multiple States :

- Let D be a DFA with n states.
- Any string w accepted by D that has length at least n must visit some state twice.
- Number of states visited is equal to the length of the string plus one.
- By the pigeonhole principle, some state is duplicated.
- The substring of w between those revisited states can be removed, duplicated, tripled, etc. without changing the fact that D accepts w.

Let’s take an example of infinite regular language ab*c





if |w| >= n

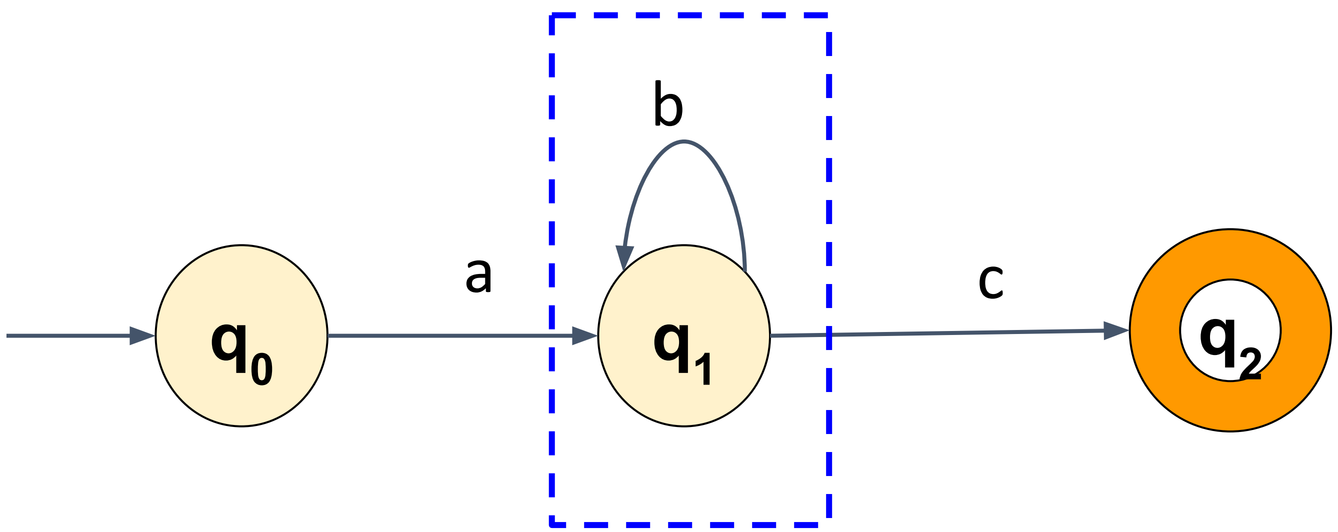
we visit a set of states more than once which means,

there exists a loop in our Automata (within these n states)

if we pump that loop 0 or more no. of times,

the resultant string will always be in the language

Let's take an example of infinite regular language ab*c



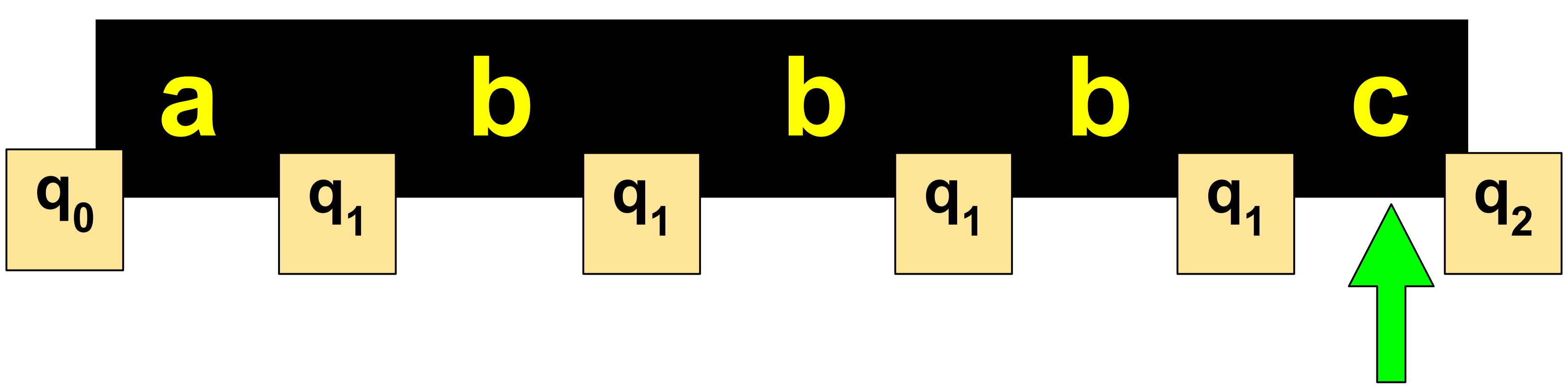
There exists 3 parts to a string w:

w = x y z

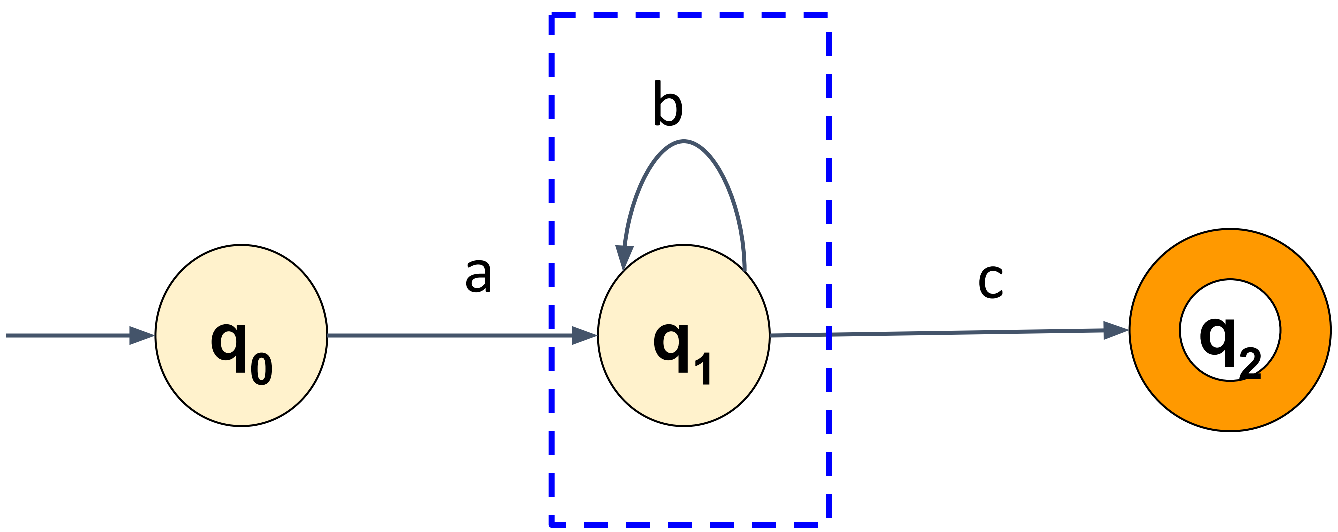
Part before the loop

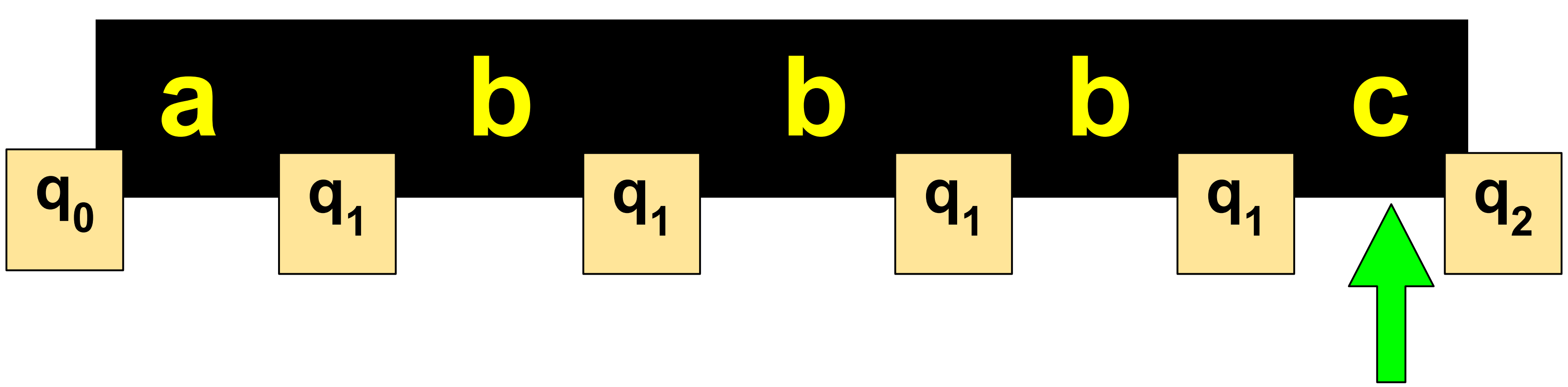
The loop

Part after the loop



Let’s take an example of infinite regular language ab*c





There exists 3 parts to a string w:

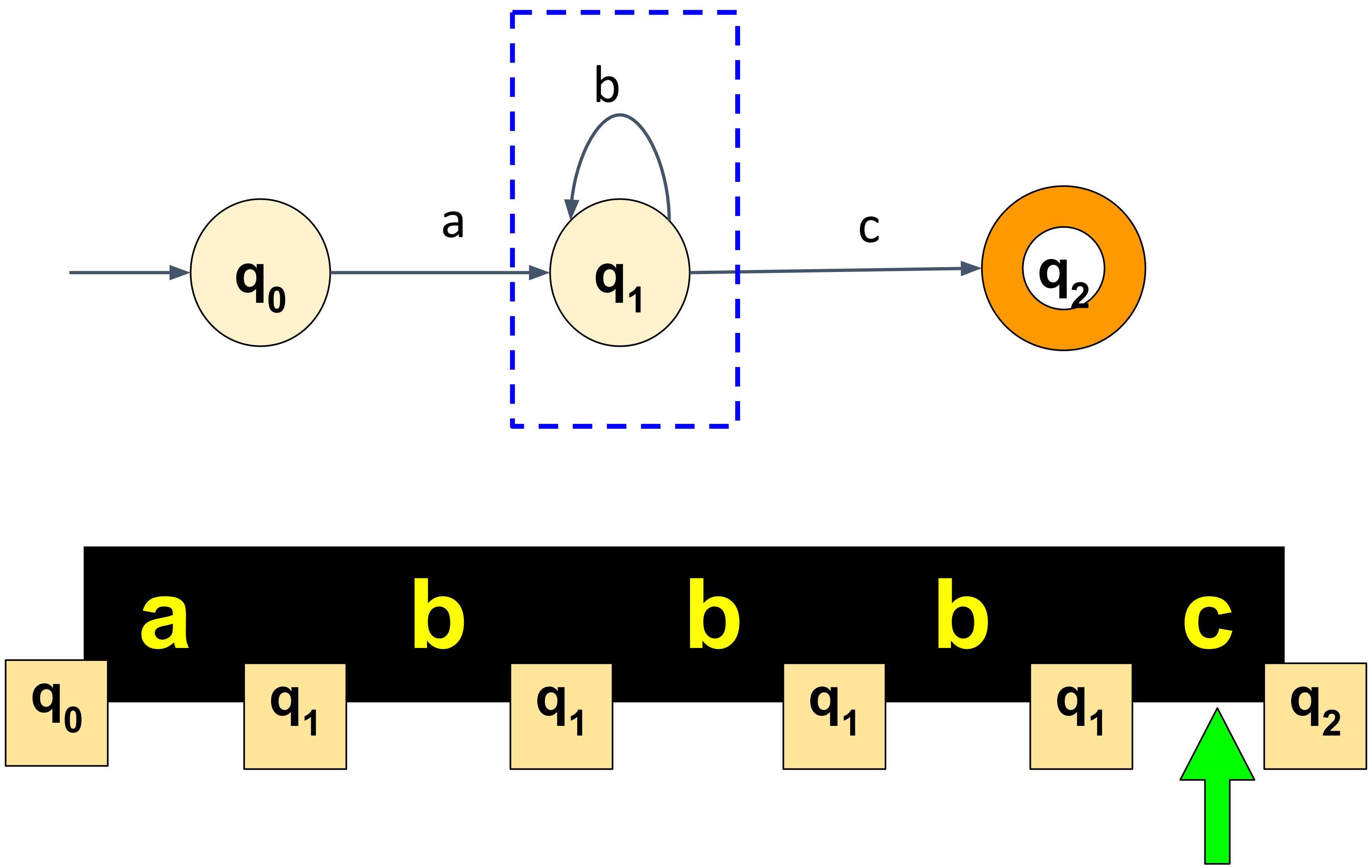
w = x y z

Part before the loop: a

The loop: b

Part after the loop: c

Let’s take an example of infinite regular language ab*c



There exists 3 parts to a string w:

w = x y z

Part before the loop: a

The loop: b

Part after the loop: c

$y \neq \epsilon$ that is $|y| >= 1$

The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

In our example of infinite regular language ab*c

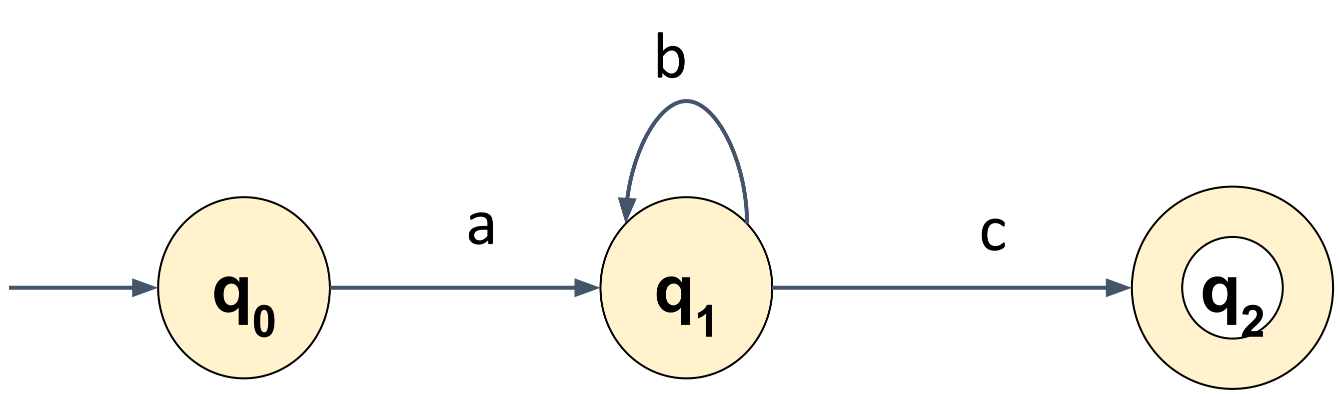
The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

n =3

In our example of infinite regular language ab*c



The Pumping property States,

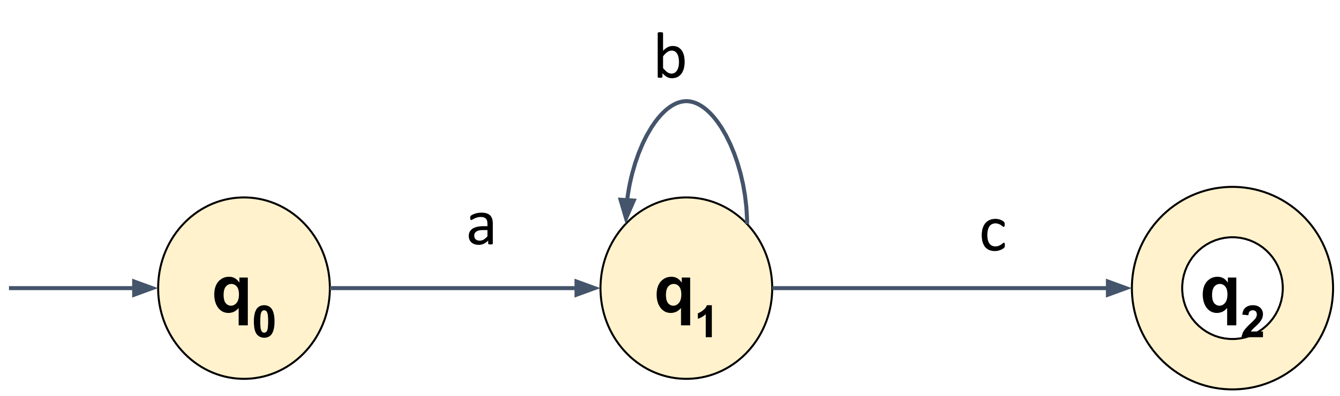
For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

n =3

n is also called Pumping Constant

In our example of infinite regular language ab*c



The Pumping property States,

For every Regular language L, (infinite)

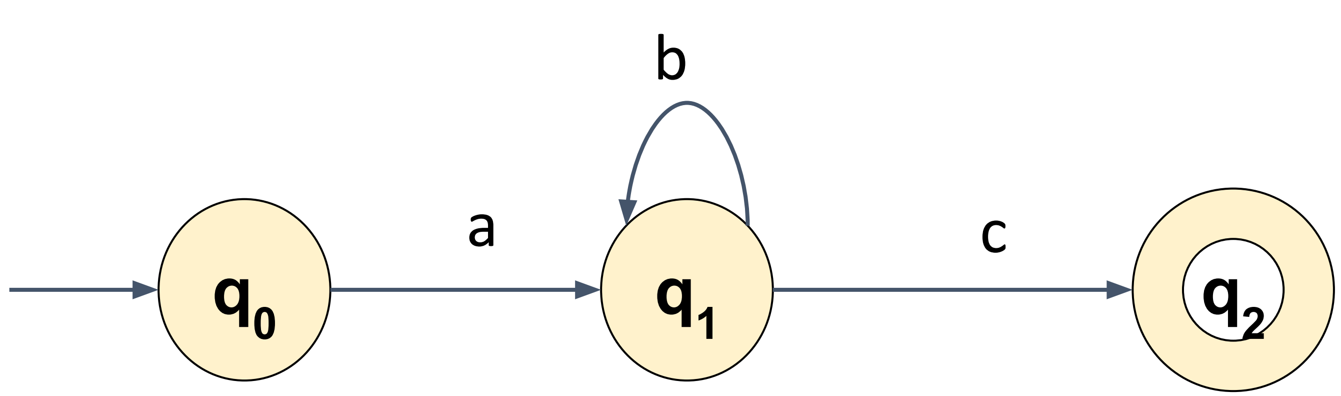
there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

|w| >= n

n =3

In our example of infinite regular language ab*c



The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

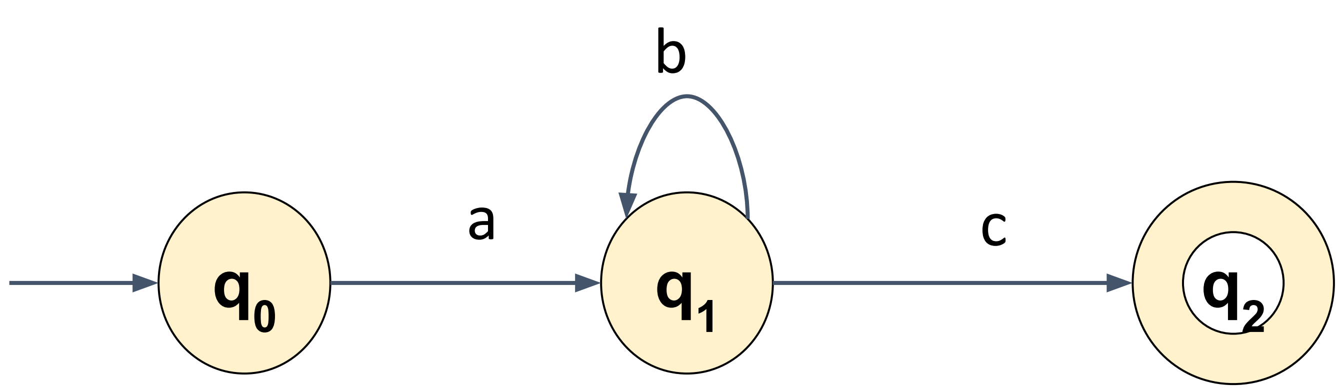
For every string w that belongs to L such that,

|w| >= n

n =3

w = abbbc
|w| = 5 > n

In our example of infinite regular language ab*c



The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

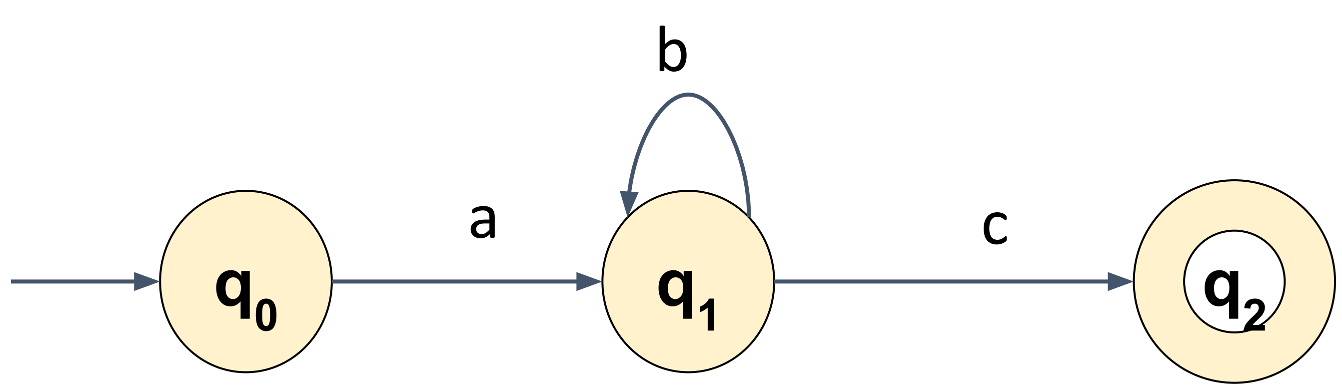
$|w| >= n$

There exists a break up of the string in three parts w = xyz such that $|y| >=1$ and $|xy| <= n$

n =3

w = abbbc
$|w| = 5 > n$

In our example of infinite regular language ab*c



The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

$|w| >= n$

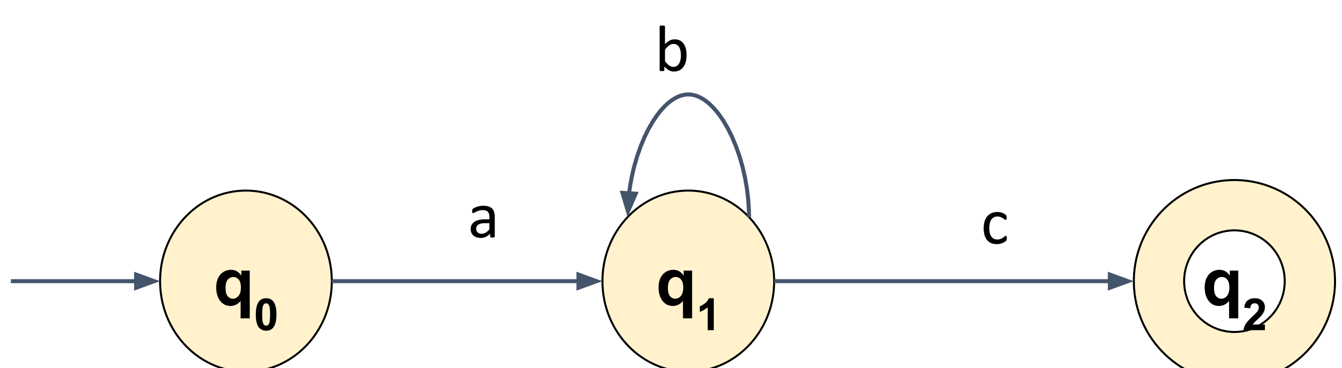
There exists a break up of the string in three parts w = xyz such that $|y| >=1$ and $|xy| <= n$

w = abbbc
$|w| = 5 > n$

n =3

w = abc
x = a
y = b
z = c

In our example of infinite regular language ab*c



The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

|w| >= n

There exists a break up of the string in three parts w = xyz
such that |y| >=1 and |xy|<=n,
for every i >= 0,

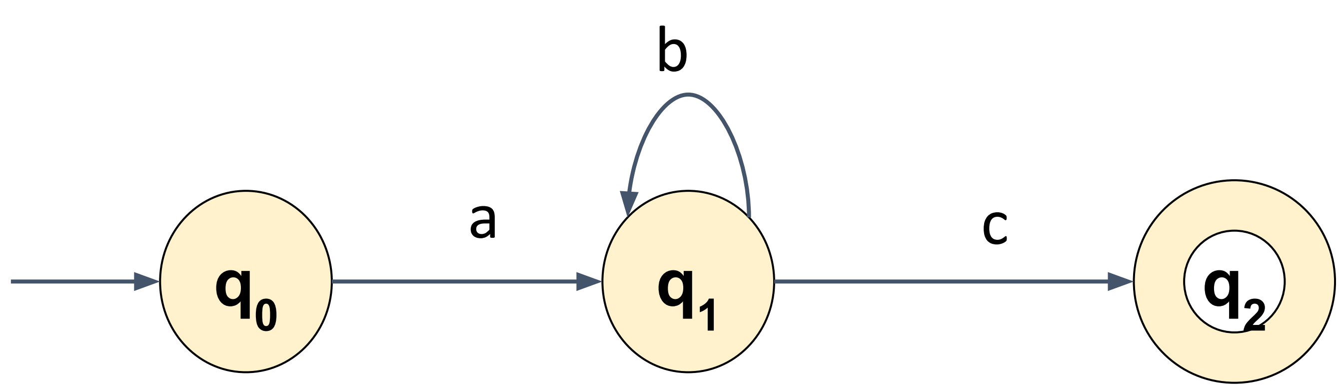
xy^iz belongs to L

w = abbbc
|w| = 5 > n

n =3

w = abc
x = a
y = b
z = c

In our example of infinite regular language ab*c



Unit 2 - Pumping Lemma for Regular Languages

The Pumping property States,

For every Regular language L, (infinite)

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

|w| >= n

There exists a break up of the string in three parts w = xyz s
uch that |y| >=1 and |xy|<=n,
for every i >= 0,

xy^iz belongs to L

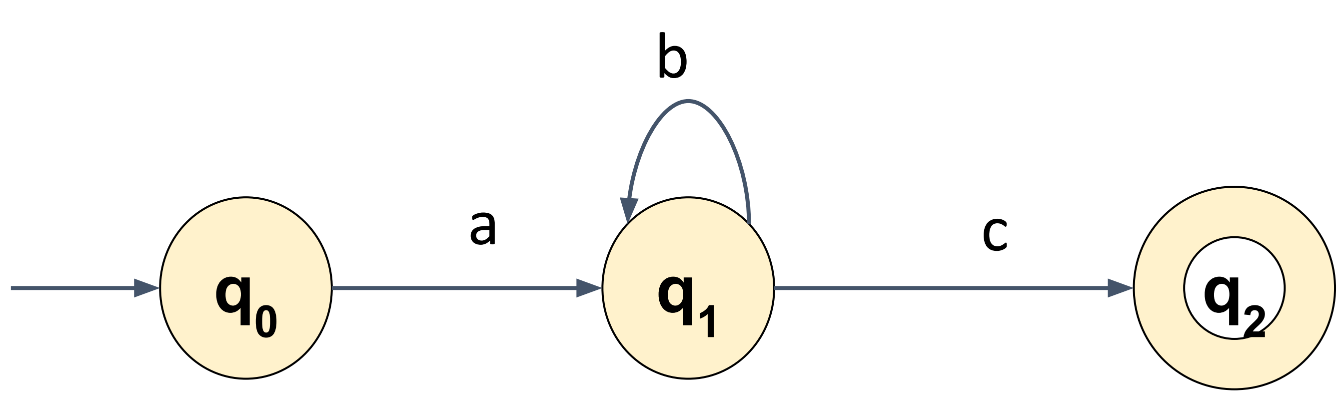
n =3

w = abbbc
|w| = 5 > n

w = abc
x = a
y = b
z = c

for i>=0,
ab^ic is in lang ab*c

In our example of infinite regular language ab*c



PES UNIVERSITY ONLINE

For Regular Languages (infinite)

Pumping Property

For every Regular language L,

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

$|w| >= n$

There exists a break up of the string in three parts $w = xyz$ such that $|y| >= 1$ and $|xy| <= n$,

for every $i >= 0$,

xy^iz belongs to L

PES UNIVERSITY ONLINE

For Regular Languages (infinite)

Pumping Property

For every Regular language L,

there exists n where n is the # states in Finite Automata for L

For every string w that belongs to L such that,

|w| >= n

There exists a break up of the string in three parts w = xyz such that |y| >= 1 and |xy|<=n,

for every i >= 0,

xy^iz belongs to L

Replace

For every ----> ∀

There exists ----> ∃

belongs to ----> ∈

PES UNIVERSITY ONLINE

For Regular Languages (infinite)

Pumping Property

$\forall$ Regular language L,

$\exists$ n where n is the # states in Finite Automata for L

$\forall$ string w $\in$ L such that,

$|w| >= n$

$\exists$ w = xyz such that $|y| >= 1$ and $|xy| <= n$,

$\forall$ $i >= 0$,

$xy^iz \in L$

Replace

For every ----> $\forall$

There exists ----> $\exists$

belongs to ----> $\in$

For Regular Languages (infinite)

Pumping Property

$\forall$ Regular language L,

$\exists$ n where n is the # states in Finite Automata for L

$\forall$ string w $\in$ L such that,

|w| >= n

$\exists$ w = xyz such that |y| >= 1 and |xy|<=n,

$\forall$ i >= 0,

$xy^iz \in L$

To Prove a lang is Non-Regular

~Pumping Property

For Regular Languages (infinite)

Pumping Property

$\forall$ Regular language L,

$\exists$ n where n is the # states in Finite Automata for L

$\forall$ string w $\in$ L such that,

|w| >= n

$\exists$ w = xyz such that |y| >=1 and |xy|<=n,

$\forall$ i >= 0,

$xy^iz \in L$

~ $\forall$ = $\exists$

~ $\exists$ = $\forall$

To Prove a lang is Non-Regular

~Pumping Property

For Regular Languages (infinite)

Pumping Property

$\forall$ Regular language L,

$\exists$ n where n is the # states in Finite Automata for L

$\forall$ string w $\in$ L such that,

|w| >= n

$\exists$ w = xyz such that |y| >= 1 and |xy|<=n,

$\forall$ i >= 0,

$xy^iz \in L$

~ $\forall$ = $\exists$

~ $\exists$ = $\forall$

To Prove a lang is Non-Regular

~Pumping Property

$\exists$ a language L which is claimed to be regular,

$\forall$ n where n is the # states in Finite Automata for L

$\exists$ string w $\in$ L such that,

|w| >= n

$\forall$ w = xyz such that |y| >= 1 and |xy|<=n,

$\exists$ i >= 0,

$xy^iz \notin L$

This contradicts the claim made, hence proving that the language is not regular

For Regular Languages (infinite)

Pumping Property

∀ Regular language L,

∃ n where n is the # states in Finite Automata for L

∀ string w ∈ L such that,

|w| >= n

∃ w = xyz such that |y| >= 1 and |xy|<=n,

∀ i >= 0,

xy^iz ∈ L

~ ∀ = ∃

~ ∃ = ∀

To Prove a lang is Non-Regular

~Pumping Property

∃ a language L … be regular,

∀ n whe… …ata for L

∃ stri…

∀ w = …

∃ i >= 0, …

This contradic… … proving that the language is not …

Proof by Contradiction

For Regular Languages (infinite)

Pumping Property

∀ Regular language L,

∃ n where n is the # states ... L

∀ string **w** ∈ L such that

|w| >= n

∃ w = xyz such that |y| >= 1 and |xy| <= n,

∀ i >= 0,

xy^iz ∈ L

Pumping Lemma

~ ∀ =

~ ∃ = ∀

To Prove a lang is Non-Regular

~Pumping Property

∃ a language L ... be regular,

∀ n whe... ata for L

∃ stri...

∀ w = ...

∃ i >= 0...

This contradic... e proving that the language is not ...

Proof by Contradiction

Procedure to prove a language is Not regular :

1. Assume the opposite: L is regular

2. Use Pumping Lemma to obtain a contradiction

   It suffices to show that only one string gives a contradiction

3. Thereby proving L is not regular

Procedure to prove a language is Not regular :

1. Assume the opposite: L is regular

2. Use Pumping Lemma to obtain a contradiction

! String must be chosen appropriately

It suffices to show that only one *string* gives a contradiction

3. Thereby proving L is not regular

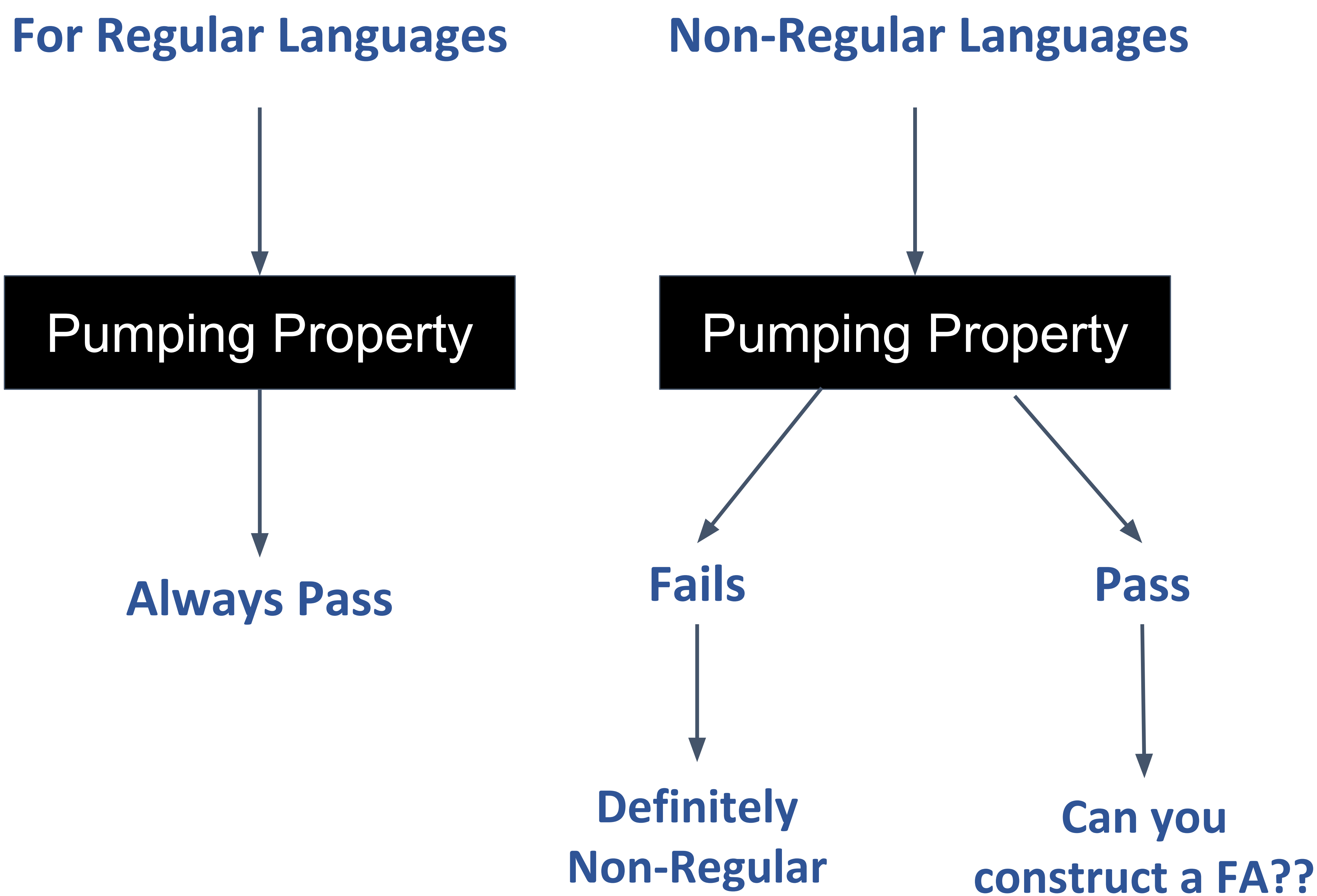
For Regular Languages

$\downarrow$

Pumping Property

$\downarrow$

Always Pass


For Regular Languages
Pumping Property
Always Pass
Non-Regular Languages
Pumping Property
Fails
Pass
Definitely Non-Regular
Can you construct a FA??

For Regular Languages

N n-Regular Languages

Pump

roperty

Pumping lemma is a necessary but not sufficient condition for a language to be regular

Always Pas

is

Pass

Definitely Non-Regular

Can you construct a FA??

Pumping lemma as a game

Pumping lemma is a game between

You vs. Adversary

The role

You

Adversary

Claims L is regular

The role

You

Okay! Gimme the no. of states in your machine for L

Adversary

Claims L is regular

The role

You

Okay! Gimme the no. of states in your machine for L

Adversary

Claims L is regular

There are n states in my automata for L

The role

You

Okay! Gimme the no. of states in your machine for L

Okay! here is the string w from L such that
|w| >= n
Could you tell me where is the loop in your machine?

Adversary

Claims L is regular

There are n states in my automata for L

The role

You

Adversary

Claims L is regular

Okay! Gimme the no. of states in your machine for L

There are n states in my automata for L

Okay! here is the string w from L such that
|w| >= n
Could you tell me where is the loop in your machine?

The loop is xy^iz

The role

You

Okay! Gimme the no. of states in your machine for L

Okay! here is the string w from L such that
|w| >= n
Could you tell me where is the loop in your machine?

Find some i, so that the resultant string is not in L

Adversary

Claims L is regular

The are n states in my automata for L

The loop is xy^iz

The role

You

Okay! Gimme the no. of states in your machine for L

Okay! here is the string w from L such that
|w| >= n
Could you tell me where is the loop in your machine?

Find some i, so that the resultant string is not in L

Adversary

Claims L is regular

The are n states in my automata for L

The loop is xy^iz

Oh no! I lost!!!!
L is not regular

The role

You

Adversary

laims L is regular

Okay! Gimme the
states in your machi

e are n states in my
automata for L

Okay! here is the st
from L such th
|w| >= n
Could you tell me w
the loop in your machine?

The loop is xy^iz

Okay! but for some i, the
resultant string is not in L

Oh no! I lost!!!!
L is not regular

Using Pumping lemma prove that the language $a^n b^n$ is not regular

The role

You

Adversary

Claims $L = a^n b^n$ is regular

The role

You

Okay! Gimme the no. of states in your machine for L

Adversary

Claims L = $a^n b^n$ is regular

The role

You

Okay! Gimme the no. of states in your machine for L

Adversary

Claims L = $a^n b^n$ is regular

There are 10 states in my automata for L

The role

You

Okay! Gimme the no. of states in your machine for L

okay! I'll choose the string a^6b^6
$|w| >= 10$
Now tell me where is the loop in your automata?

Adversary

Claims $L = a^nb^n$ is regular

There are 10 states in my automata for L

Where is the loop ?

aaaaaabbbbbb

Where is the loop ?

aaaaabbbbbb

Loop must
be within first
10 symbols

Where is the loop ?

aaaaaabbbbbb

$a^5(a)^ib^6$

Where is the loop ?

aaaaaabbbbbb

$a^5(a)^0b^6$

Pump down, i=0 → $a^5b^6 \notin L$

Where is the loop ?

aaaaaabbbbbb

$a^5(a)^2b^6$

Pump down, i=0 $\rightarrow a^5b^6 \notin L$

Pump up, i=2 $\rightarrow a^7b^6 \notin L$

Where is the loop ?

aaaaaabbbbbb

$a^5(a)^ib^6$

Pump do... L

i = 1 doesn't help !!

..., i=2 → $a^7b^6 \notin L$

Where is the loop ?

aaaaaabbbbbb

Pump up, i=2 → $a^5ababb^5 \notin L$

$a^5(ab)^ib^5$

Where is the loop ?

aaaaaabbbbbb

Pump up, i=2 → $a^5ababb^5 \notin L$

i = 0 or i = 1 doesn't help !!

$a^5(ab)^ib^5$

Where is the loop ?

aaaaaabbbbbb

Pump down, i=0 $\rightarrow a^6b^5 \notin L$

Pump up, i=2 $\rightarrow a^6b^7 \notin L$

$a^6(b)^ib^5$

Where is the loop ?

aaaaaabbbbbb

i = 1 doesn't help !!

Pump down, i=0 → a^6b^5

Pump up, i=2 → $a^6b^7 \notin L$

$a^6(b)^ib^5$

Where is the loop ?

aaaaaabbbbbb

For every break up possible we got some i that will result in a string ∉ to L

The role

You

Okay! Gimme the no. of states in your machine for L

okay! I'll choose the string
a^6b^6
$|w| >= 10$
Now tell me where is the loop in your automata?

Adversary

Claims $L = a^nb^n$ is regular

There are 10 states in my automata for L

We saw and explored different possibilities where the loop could be

The role

You

Okay! Gimme the no. of states in your machine for L

okay! I'll choose the string
a^6b^6
$|w| >= 10$
Now tell me where is the loop in your automata?

Okay! but for some i, nothing worked out!

Adversary

Claims $L = a^nb^n$ is regular

There are 10 states in my automata for L

We saw and explored different possibilities where the loop could be

The role

You

Okay! Gimme the no. of states in your machine for L

okay! I'll choose the string
a^6b^6
$|w| >= 10$
Now tell me where is the loop in your automata?

Okay! but for some i, nothing worked out!

Adversary

Claims $L = a^nb^n$ is regular

There are 10 states in my automata for L

We saw and explored different possibilities where the loop could be

Oh no! I lost!!!!
L is not regular

The role

You

Adversary

ms L = a^nb^n is regular

Okay! Gimme the
states in your machi

e are 10 states in my
automata for L

okay! I'll choose the
a^6b^6
$|w| >= 10$
Now tell me where is the
loop in your automata?

e saw and explored
different possibilities where
the loop could be

Okay! but for some i, nothing
worked out!

Oh no! I lost!!!!
L is not regular

Using Pumping lemma prove that the language of palindromes ww over $\{a,b\}^*$ is not regular

The role

You

Adversary

Claims L = ww is regular

The role

You

Adversary

Claims L = ww is regular

Okay! Gimme the no. of states in your machine for L

The role

You

Okay! Gimme the no. of states in your machine for L

Adversary

Claims L = ww is regular

There are n states in my automata for L

The role

You

Adversary

Claims L = ww is regular

Okay! Gimme the no. of states in your machine for L

There are n states in my automata for L

okay! I'll choose the string a^na^n
$|w| >= n$
Now tell me where is the loop in your automata?

Where is the loop ?

a…a..aa..aa…a..aa..a

Where is the loop ?

a...a.aa..aa...a.aa..a

Loop must exist within first n symbols

Where is the loop ?

a…a..aa..aa…a..aa..a

$a^{n-2}(aa)^i a^n$

Where is the loop ?

a...a..aa..aa...a..aa..a

$$a^{n-2}(aa)^i a^n$$

Let's Pump down, i=0

$= a^{n-2}a^n$
$= a^{n-2}a^2a^{n-2}$
$= a^{n-1}a^{n-1}$

Where is the loop ?

a...a..aa..aa...a..aa..a

$$a^{n-2}(aa)^i a^n$$

Let's Pump up, i=3

$$= a^{n-2}(aa)^3 a^n$$
$$= a^{n-2}(a^2)^3 a^n$$
$$= a^{n-2}a^6 a^n$$
$$= a^{n-2}a^4 a^2 a^n$$
$$= a^{n-2+4}a^{n+2}$$
$$= a^{n+2}a^{n+2}$$

Where is the loop ?

a...a..aa..aa...a..aa..a

$a^{n-2}(aa)^i a^n$

Pump up or Pump down,
resultant string will always belong to L

Where is the loop ?

a...a a..a

$a^{n-2}(aa)^{i}a^{n}$

Pump up or Pump down,
resultant string will always belong to L

Where is the loop ?

a...a a..a

$a^{n-2}(aa)^i a^n$

You chose a wrong string

r Pump down,
ring will always belong to L

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu

+91 80 6666 3333 Extn 724