Theory of Computation

Introduced By:

Prof. Safia Abbas

Text Books:

"***An Introduction to the theory of computation***" by Michael Sipser, 2nd Edition, PWS Publishing Company, Inc. 2006; ISBN: 0-534-95097-3

"***An Introduction to Formal Languages & Automata***" by Peter Linz, 5th Edition, Jones & Bartlett Publishers, Inc. January 2012; ISBN:978-1-4496-1552-9 .

Agenda

- How to prove or disprove that the language is regular?
- Pumping lemma for disproving
- Practice on Pumping Lemma
- Context free grammer

Non-regular languages and pumping lemma

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

Prove that there is no FA that accepts L

Problem: this is not easy to prove

Solution: the Pumping Lemma !!!

Is the language

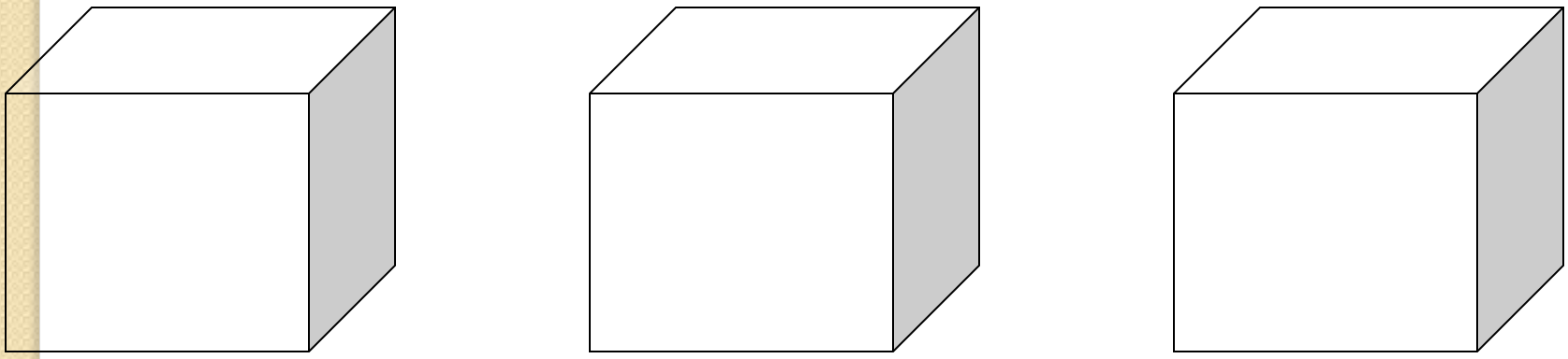
$$L = \{a^n b^n : n \geq 0\}$$

is regular???

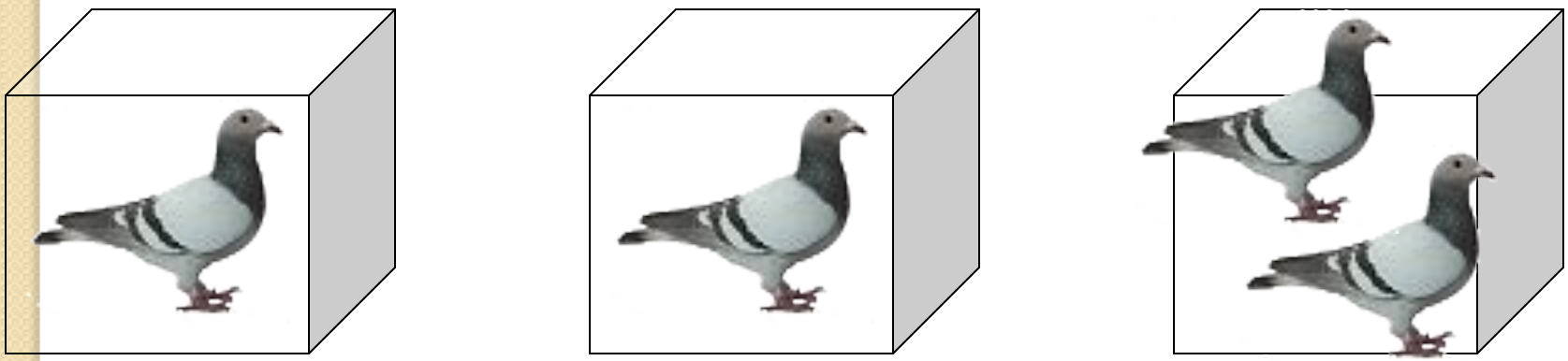
The Pigeonhole Principle

4 pigeons

3 pigeonholes

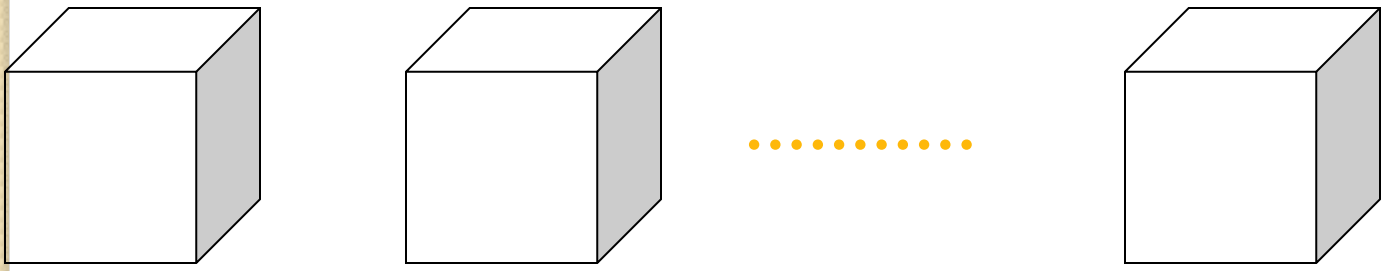


A pigeonhole must
contain at least two pigeons



n pigeons

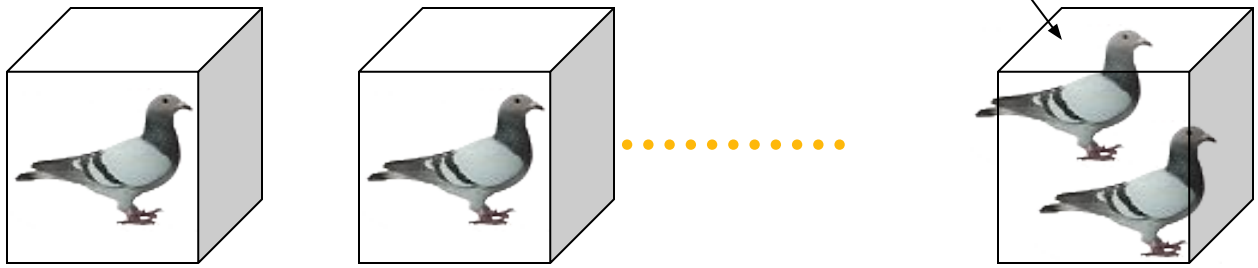
m pigeonholes $n > m$



The Pigeonhole Principle

If there are n pigeons and m pigeonholes such that $n > m$

There is a pigeonhole with at least 2 pigeons

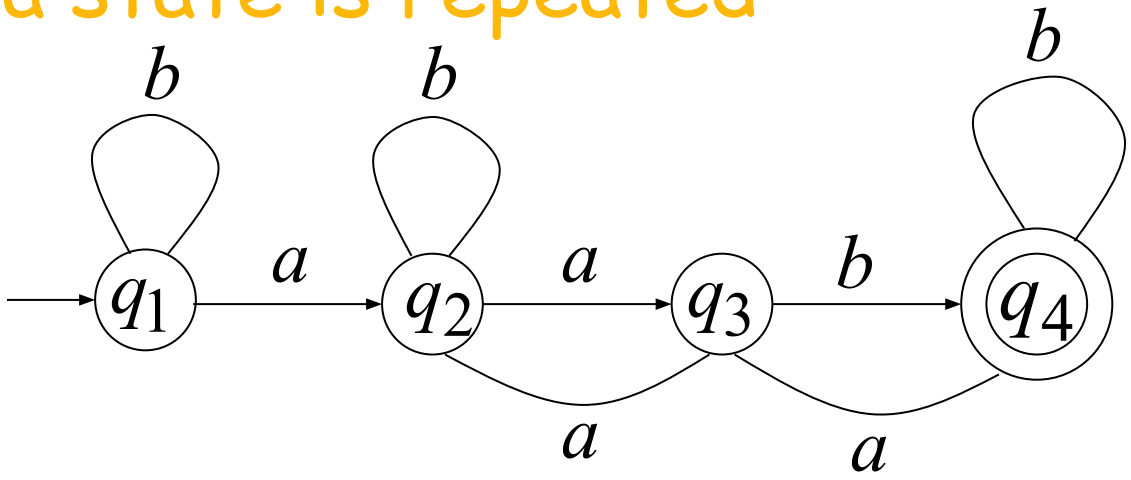


Pigeonhole principle for any FA:

If in a walk of a string w

transitions $\geq$ states of FA

then a state is repeated

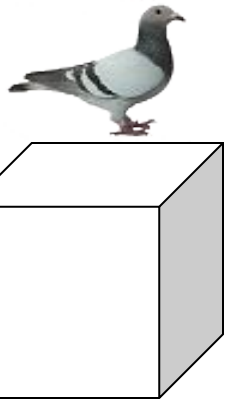


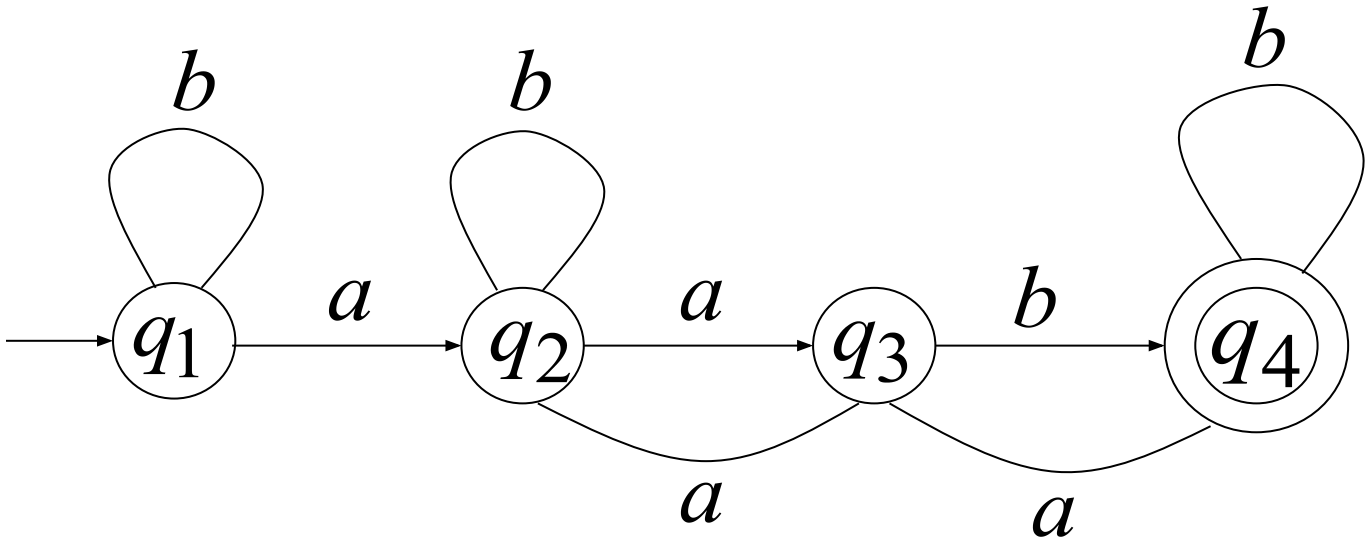
If the walk of string w has length ≥4 Then a state must be repeated

In other words for a string w :

a → transitions are pigeons

q states are pigeonholes

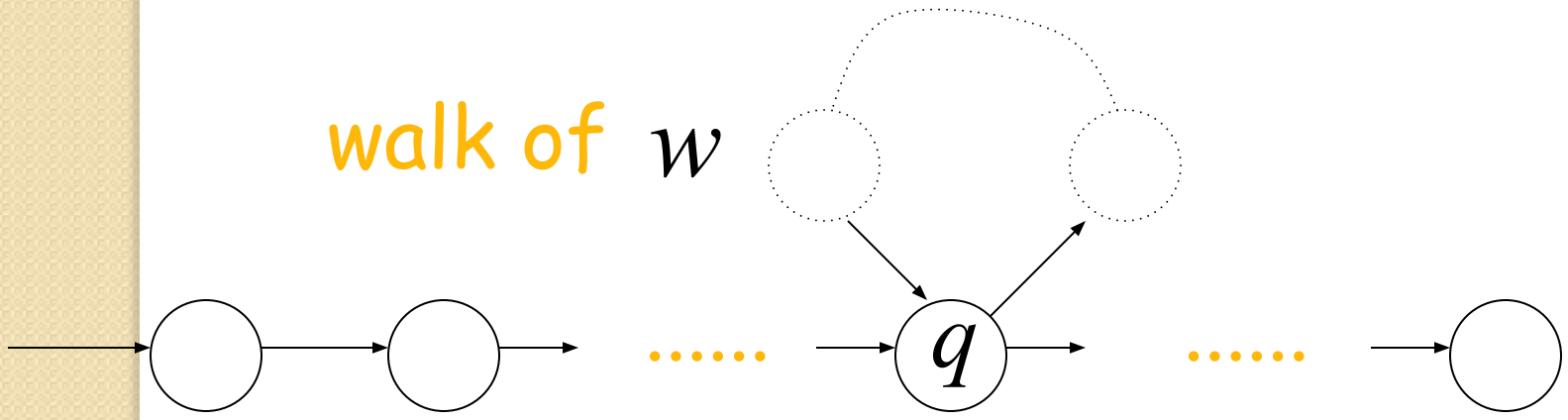




In general:

A string w has length $\geq$ number of states

A state q must be repeated in the walk w



Is the language $L = \{a^n b^n : n \geq 0\}$ is regular???

Suppose that L is regular then there is a FA M that accepts L

Now look at $\delta^*(q_0, a^i)$ for $i = 1, 2, 3, \ldots$.

there will be unlimited number of i's and finite numbers of states in the FA M

From the pigeonholes principle we will have

$$\delta^*(q_0, a^n) = q \quad \text{and} \quad \delta^*(q_0, a^m) = q,$$

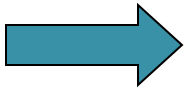
where $n \neq m$.

But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F.$$

From this we conclude that

$$\begin{aligned}\delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) \\ &= \delta^*(q, b^n) \\ &= q_f.\end{aligned}$$



contradiction

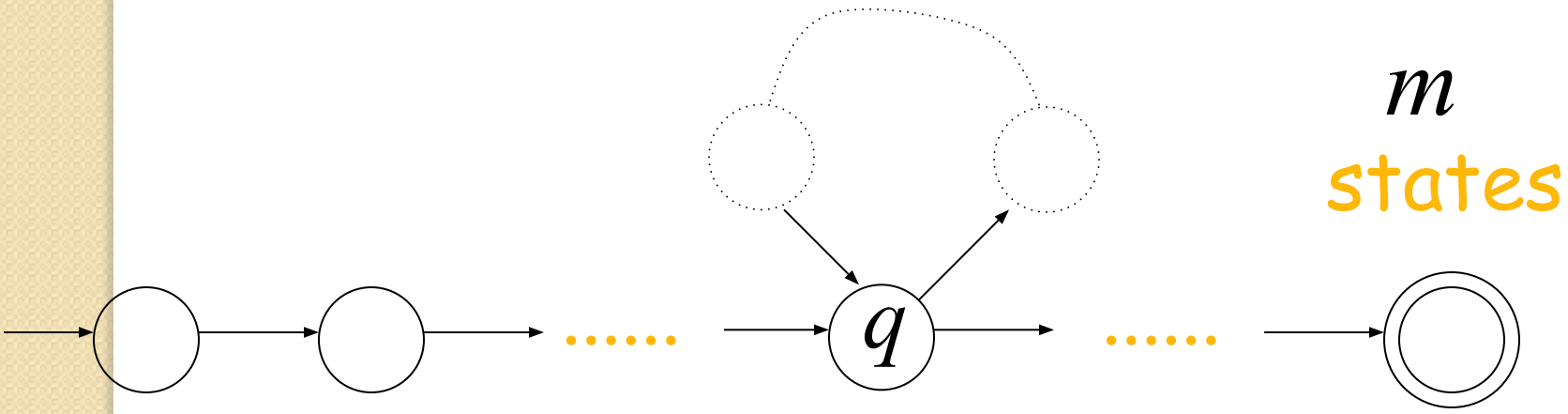
Pumping lemma for regular languages

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

Pumping lemma for regular languages

Take an infinite regular language L

FA that accepts L



Pumping lemma for regular languages

Take string w such that $w \in L$

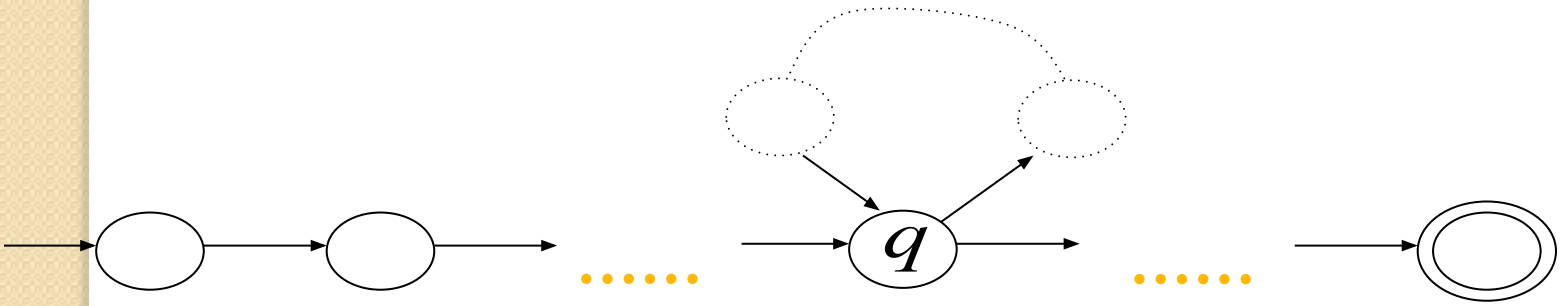
Then there is a walk with label w :

walk w

Pumping lemma for regular languages

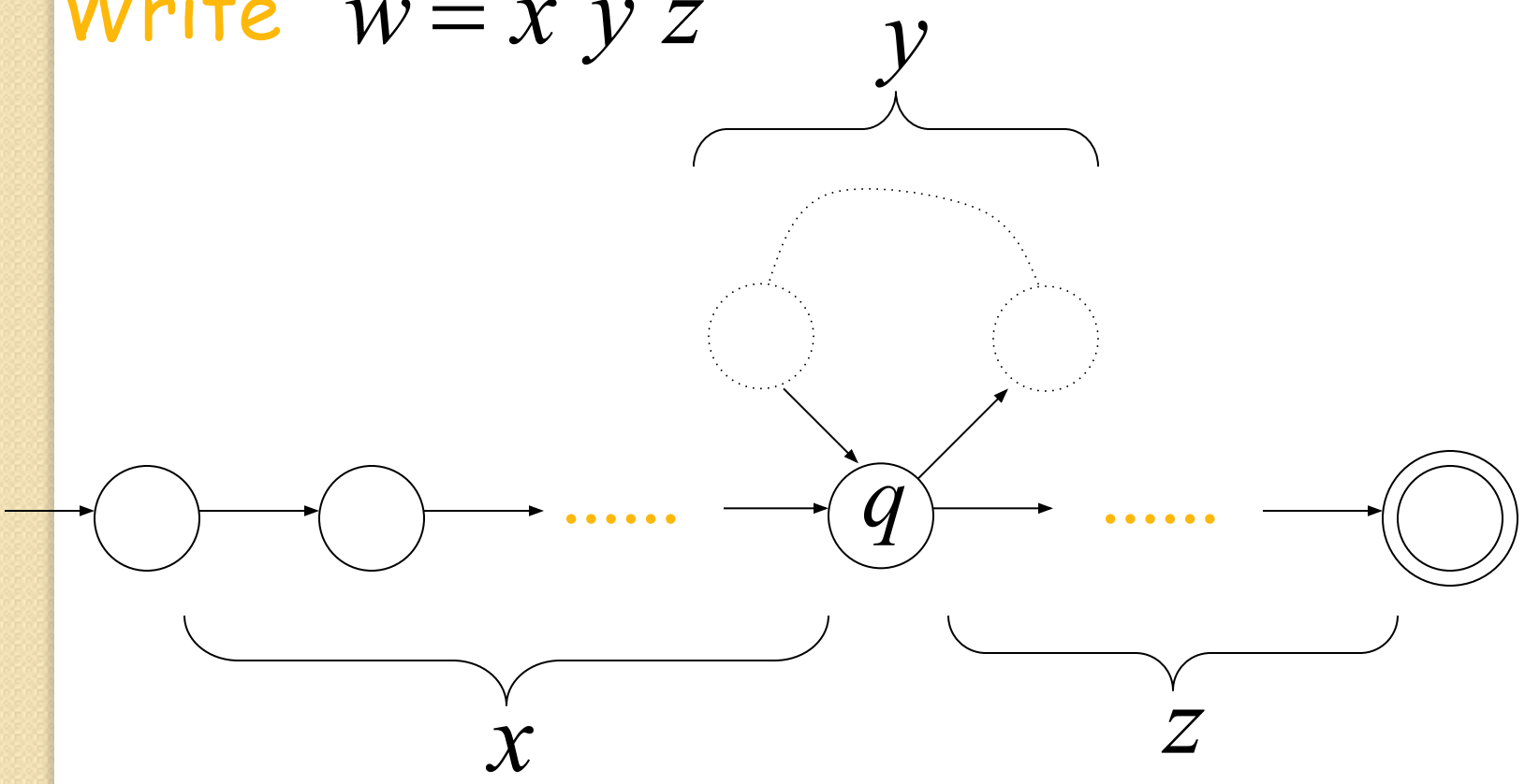
If string w has length $|w| \geq m$ (number of states)

then, from the pigeonhole principle:
a state q is repeated in the walk w



walk w

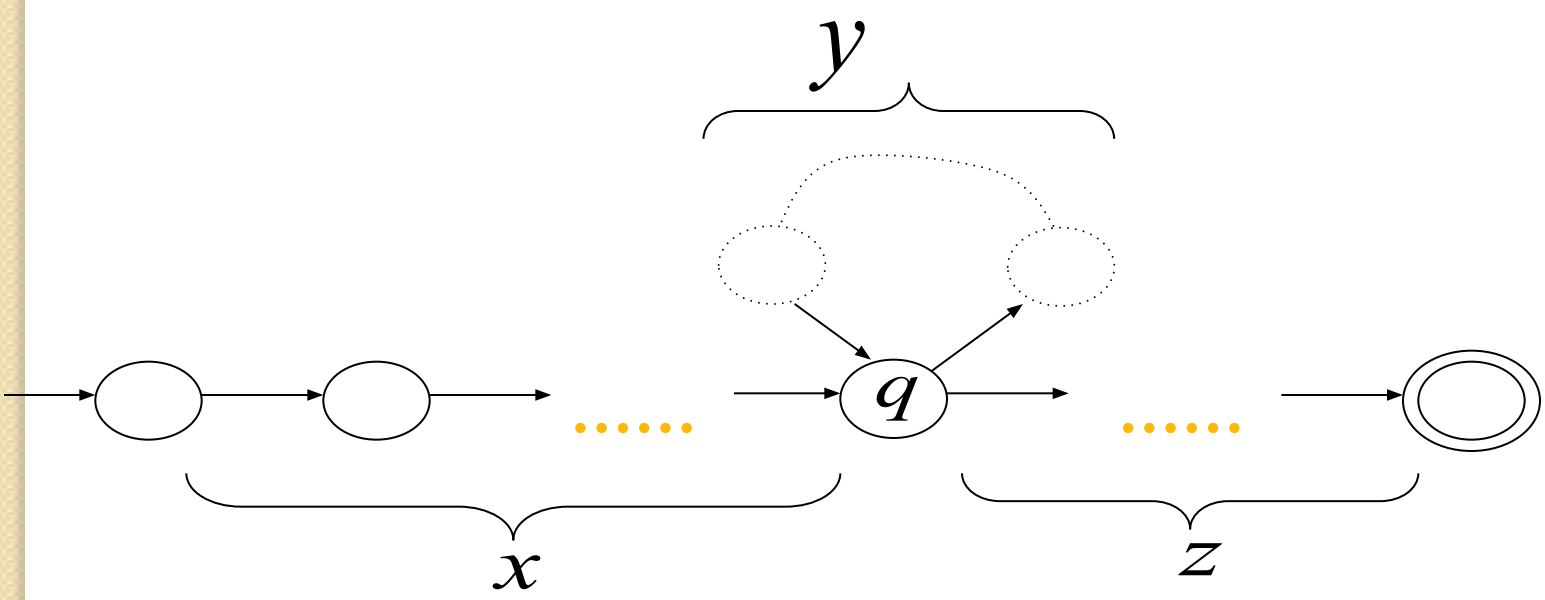
Write $w = x\ y\ z$



Observations:

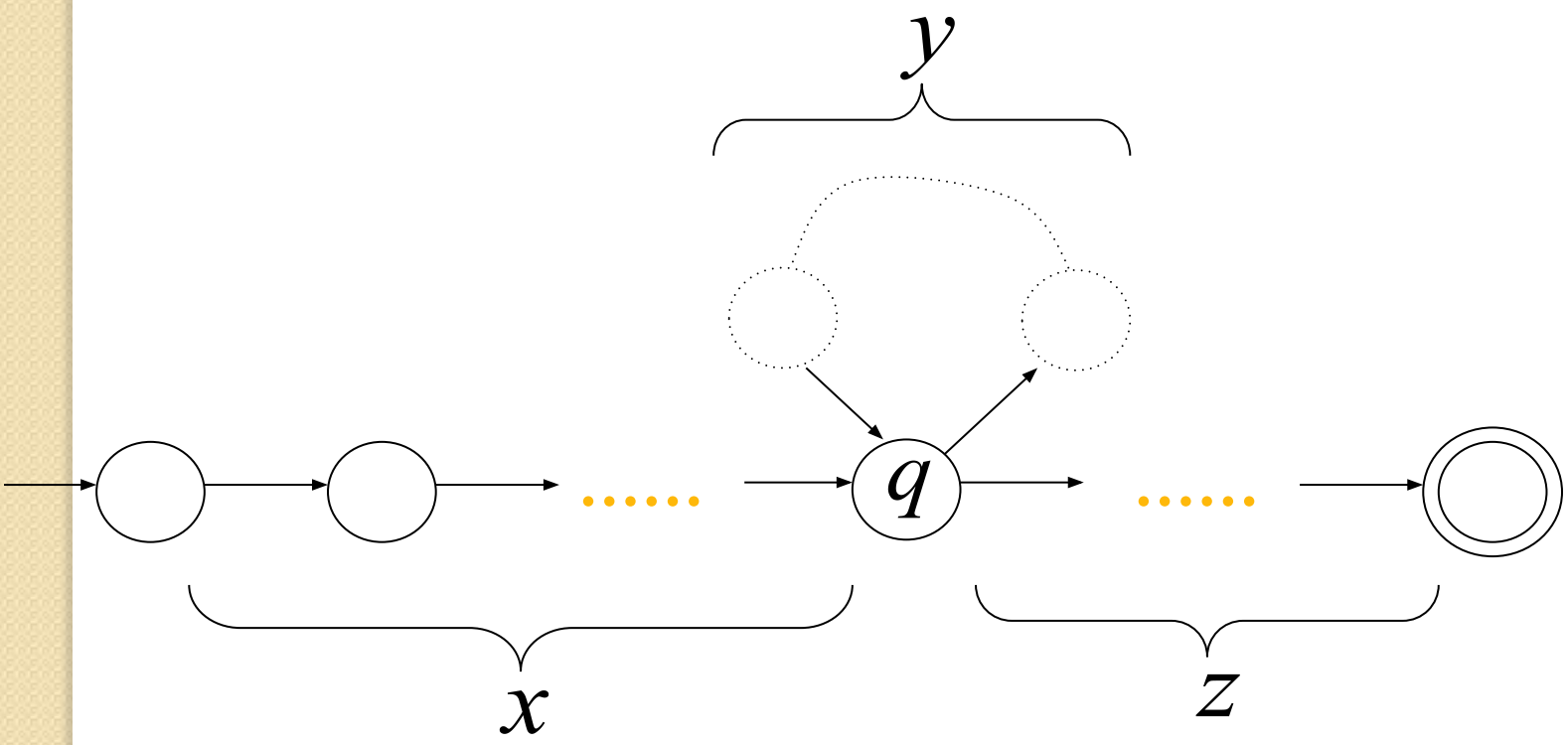
length $|xy| \leq m$ Number of states

length $|y| \geq 1$

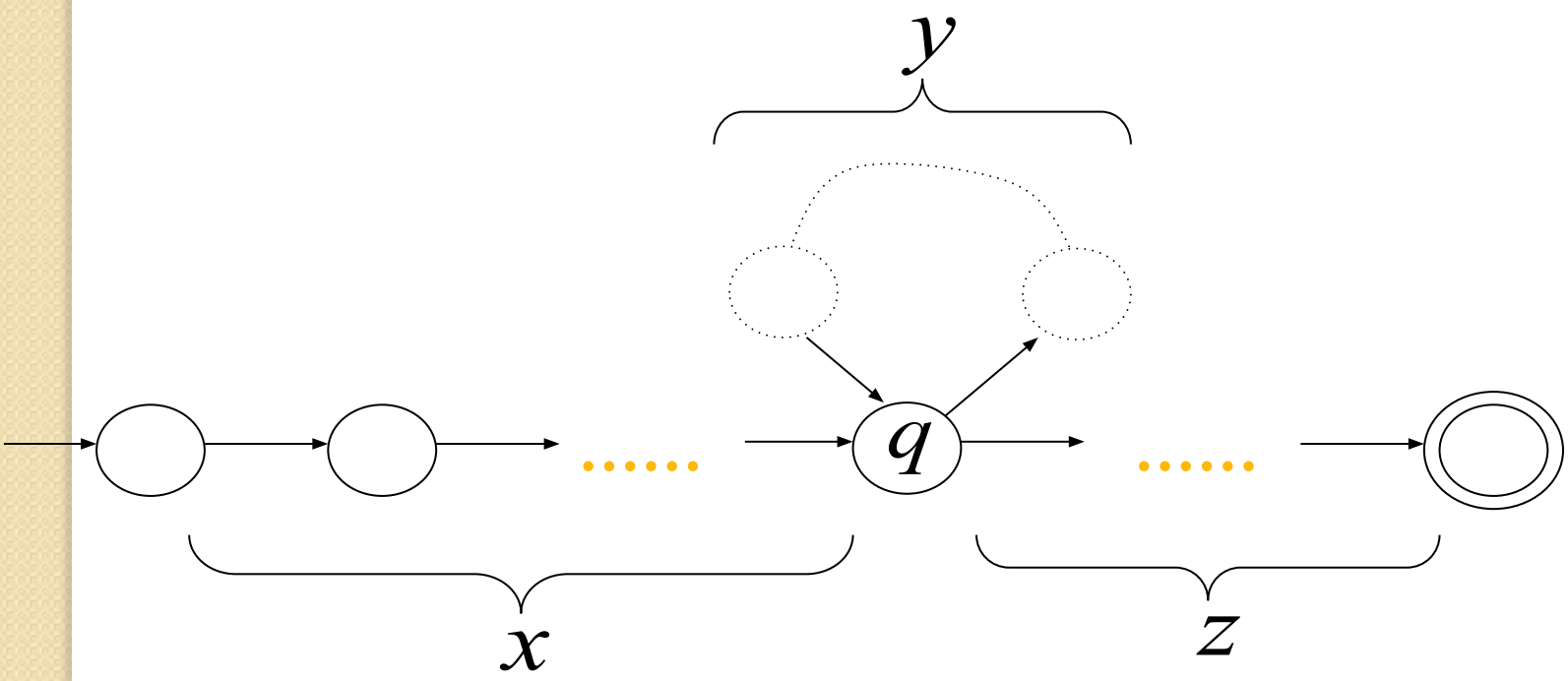


Observation:

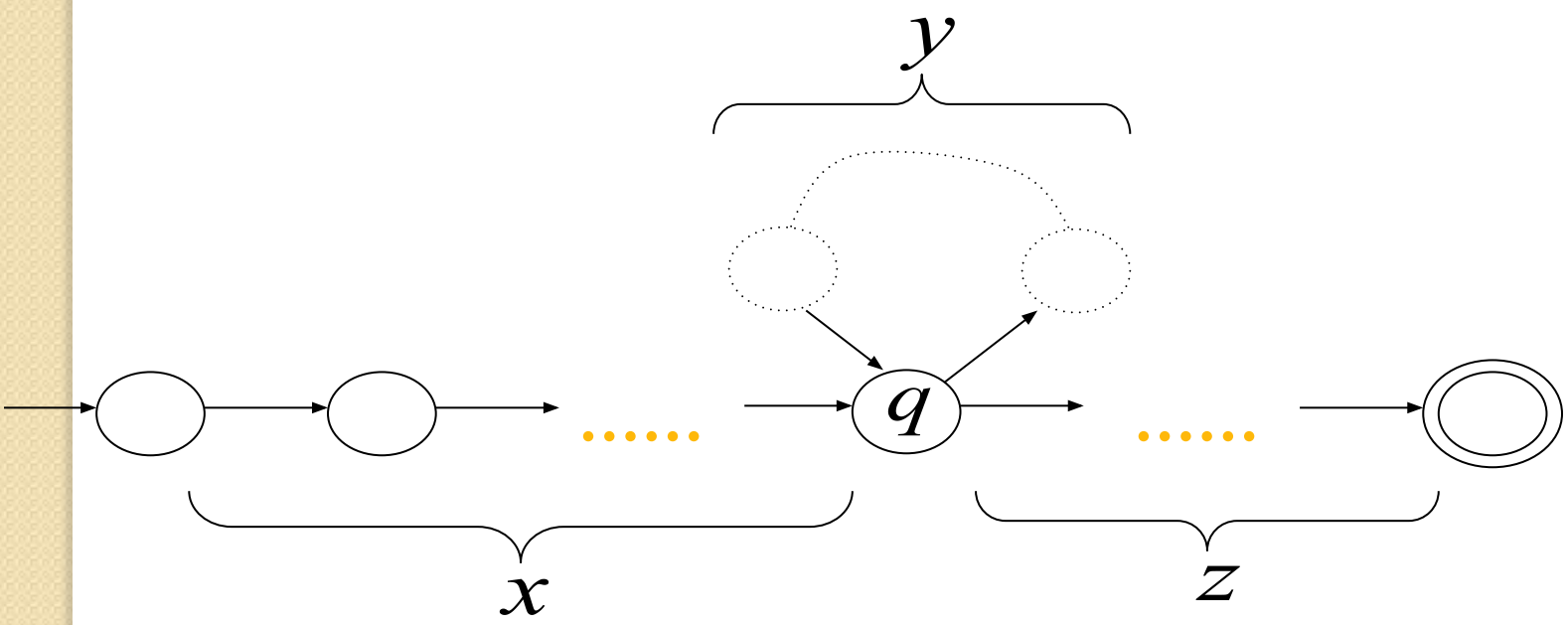
The string $x\,z$ is accepted



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



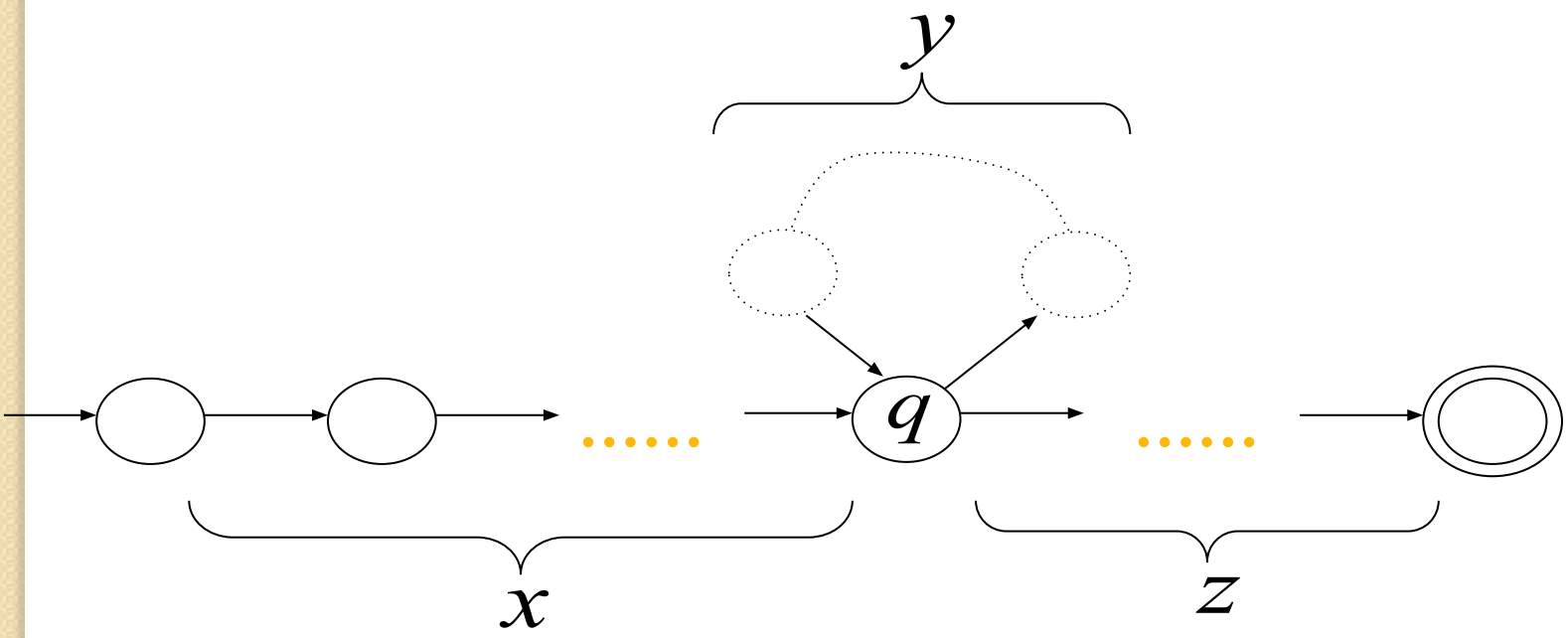
Observation: The string $x\ y\ y\ y\ z$ is accepted



In General:

The string $x\, y^i\, z$

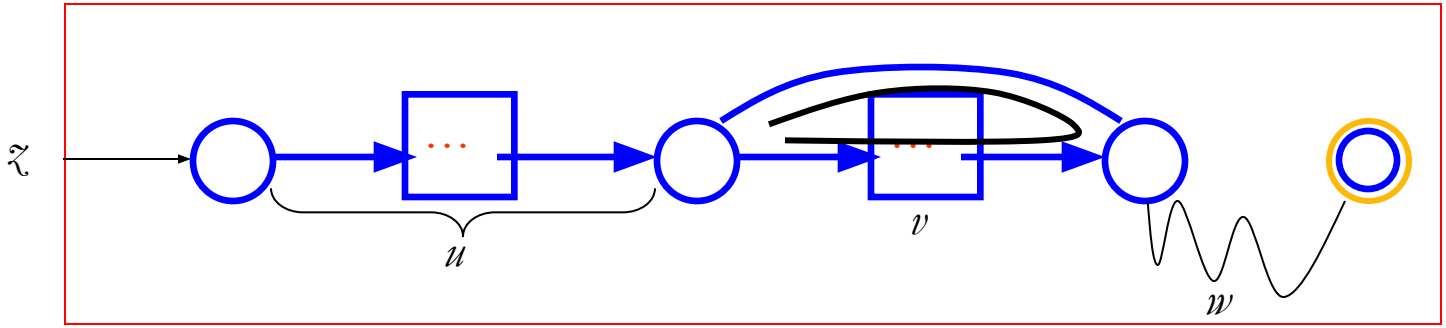
is accepted $i = 0, 1, 2, \ldots$



Pumping lemma for regular languages

- Pumping lemma: For every regular language L

  There exists a number m such that for every string z in L, we can write $z = u\,v\,w$ where
  ① $|uv| \leq m$
  ② $|v| \geq 1$
  ③ For every $i \geq 0$, the string $u\,v^i\,w$ is in L.



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

is not regular

Proof: Use the Pumping Lemma

Assume for contradiction

that L is a regular language

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

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: $w \in L$

And length $|w| \geq m$

As Example:

pick $w = a^m b^m$

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

From the Pumping Lemma

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

Therefore:

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

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

We have: $x\ y\ z = a^m b^m\ \ y = a^k, \quad k \geq 1$

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

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

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

$$x\ y^2\ z \ = \ x\ y\ y\ z \ = \ a^{m+k} b^m \ \in\ L$$

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

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

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

CONTRADICTION!!!

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

Conclusion: L is not a regular language

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

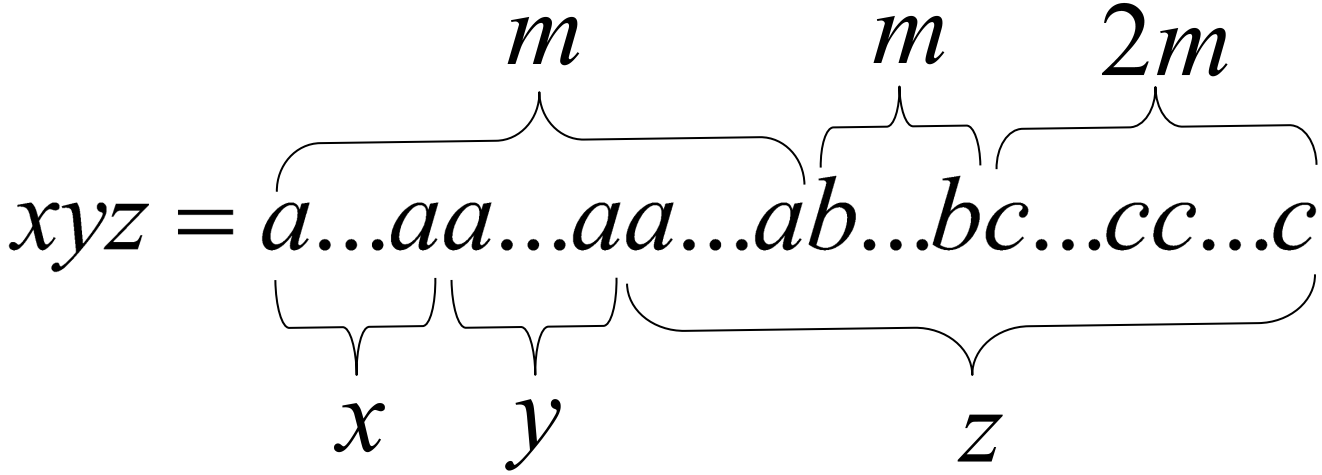
length $|w| \geq m$

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



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

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

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

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

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

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

From the Pumping Lemma: $xz \in L$

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

x z

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

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

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

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

CONTRADICTION!!!

Therefore:

Our assumption that L is a regular language is not true

Conclusion:

L is not a regular language

How can we describe non-regular languages??

Both regular and non regular can have grammar as set of production rules to generate its words

What is the grammar?

Basic Concepts

The grammar consists of the following four parts:

- An alphabet N of grammar symbols called nonterminals.
- An alphabet T of symbols called terminals, which must be distincit from nonterminals.
- A specific nonterminal symbol called the start state.
- A finite set of production rules of the form $\propto \rightarrow ß$, where $\propto$, ß are strings over the alphabet $N \cup T$ and $\propto$ is not the empty string **λ**
- we can define the language L over the grammar G as:

$$\mathbf{L(G)=\{w|\ w \in T^* \text{ and } S \Rightarrow^+ w\}..}$$

Basic Concepts

- **Grammar:** is a set of production rules used to define the structure of the strings in a language.
- If L is a language over the alphabet A, then a grammar for L consists of a set of grammar rules of the form:

$$\propto \rightarrow ß,$$

Where $\propto$ *and* $ß$ *denotes strings of symbols taken from A and from a set of grammar symbols denoted as the non-terminals set N.*

- Every grammar has a special grammar symbol called a start symbol S, and at least one production with the left side consisting of only the start symbol.
- For example if the start symbol is S, then the production rule will be:

$$S \rightarrow ß,$$

Basic Concepts

- **Example**

  Let A={a,b,c}, then the grammar for the A* language can be described by the following production rules:

$$S \rightarrow \lambda$$
$$S \rightarrow aS$$
$$S \rightarrow bS$$
$$S \rightarrow cS.$$

- **How do we know that this grammar describes the language A*?**

  We must be able to describe each string of the language in terms of the grammar rules.

- **Prove that the string aacb is in A*???**

Basic Concepts

- **Example**

  If A={a,b,c}, and the production rules is the set P the grammar G=<N,T,S,P> ≡ <{S,A,B}, {a,b,c}, S, P>,

  where P ≡ S→AB A→ **λ** |aA B→ **λ** |bB.

- Let us derive the string aab:

  S⇒AB⇒aAB⇒aaAB⇒aaB⇒aabB⇒aab.

- Note: that the language can have more than one grammar. So we should not be surprised when two people come up with two different grammars for the same language.

Basic Concepts

What is the grammar of the following languages?

- {a,ab,abb,abbb} …?
- {**λ**,a,aa,…,a^n,…} …?
- {b, bbb, …, b^{2n+1},…}…?
- {b,abc,aabcc,….,a^nbc^n,…}…?
- {ac,abc,abbc,…,ab^nc,….}…?

Context-free Grammar

- A grammar G = (N, T, S, P) is context-free if all productions in P have the form $A \rightarrow x$, where $A \in N$ and $x \in (N \cup T)^*$
- A language is said to be context-free if L=L(G) and G is context-free
- Example 1: G = ({S}, {a, b}, S, P), $S \rightarrow aSa \mid bSb \mid \lambda$.
  - is context-free. $L(G) = \{ww^R : w \ \{a, b\}^*\}$
- Example 2: G with $S \rightarrow abB, A \rightarrow aaBb \mid \lambda, B \rightarrow bbA$,
  - is context-free.
  - $L(G) = \{ab(bbaa)^n bba(ba)^n : n >= 0\}$
- Both examples 1 and 2 are context-free and linear.

Context-Free Languages

- Example 3: Show that $L = \{a^n b^m : n \neq m\}$ is context-free
  - $S \rightarrow AS_1, S_1 \rightarrow aS_1b \mid \lambda, A \rightarrow aA \mid a$, or not linear
  - $S \rightarrow AS_1 \mid S_1B, S_1 \rightarrow aS_1b \mid \lambda, A \rightarrow aA \mid a, B \rightarrow bB \mid b$ not linear

Leftmost and rightmost derivations

- Definition: The derivation is said to be leftmost if in each step the leftmost variable in the sentential form is replaced. If in each step the rightmost variable is replaced, we call the derivation rightmost.

  - $S \rightarrow AB, A \rightarrow aaA \mid \lambda, B \rightarrow Bb \mid \lambda$, $L(G) = \{a^{2n}b^m : n>=0, m>=0\}$
  - $S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$ leftmost derivation
  - $S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$ rightmost derivation

Ambiguity in Grammars and Languages

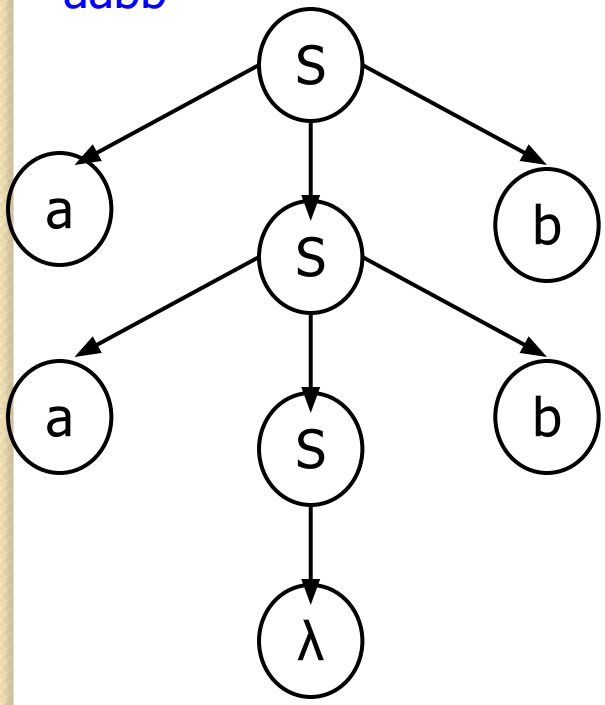
- **Definition:**
  A context-free grammar G is said to be ambiguous if there exits some w ∈ L(G) which has at least two distinct derivation or two distinct parse trees.

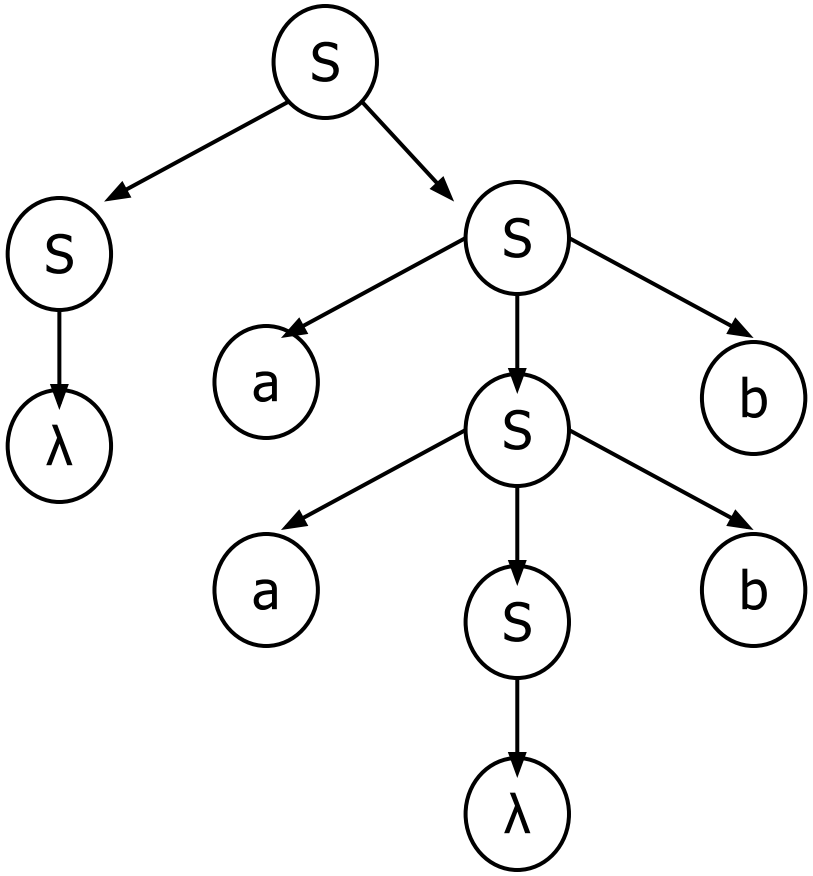
- **Example : G = (N, T, E, P), with N= {E, I}, T = {a, b, c, +, *, (,)},**
  E → I | E + E | E * E | (E), I → a | b | c
  is ambiguous since there are two derivations or parse trees for a + b * c

Ambiguity

- **Example : S → aSb | SS | λ is ambiguous. aabb has two different parse trees**

w = aabb





Ambiguity in Grammars and Languages

- **Example: G = (N, T, E, P), with N= {E, T, F, I}, T = {a, b, c, +, *, (,)},**
  $E \rightarrow T \mid E + T \mid T * E$, $T \rightarrow F \mid T * F$, $F \rightarrow I \mid (E)$, $I \rightarrow a \mid b \mid c$
  is ambiguous since there is only one derivation tree for a + b * c, however there are more than one derivation for the string a*b*c.

- **Definition: If L is a context-free language for which there exits an unambiguous grammar, then L is said to be unambiguous. If every grammar that generates L is ambiguous, then the language is called inherently ambiguous.**

- **Example : The language $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$ is inherently ambiguous. $L = L_1 \cup L_2$. L_1 is generated by $S_1 \rightarrow S_1 c \mid A, A \rightarrow aBb \mid \lambda$, L_1 is generated by $S_2 \rightarrow aS_2 \mid B, B \rightarrow bBc \mid \lambda$, and L is generated by $S \rightarrow S_1 \mid S_2$**

Note that the union, product and closure operations is done on the CFL as well

Thanks for Listening