CS372
FORMAL LANGUAGES & THE THEORY OF COMPUTATION

Dr.Nguyen Thi Thu Huong
Phone: +84 24 38696121, Mobi: +84 903253796
Email: huongnt@soict.hust.edu.vn,
huong.nguyenthithu@hust.edu.vn

Unit 4

Regular Pumping Lemma, Context Free Languages

The regular pumping lemma

- All regular languages have a special property.
- If a language does not have this property, it is not regular.
- The property states that all strings in the language can be “pumped” if they are at least as long as a certain special value, called the *pumping length*.

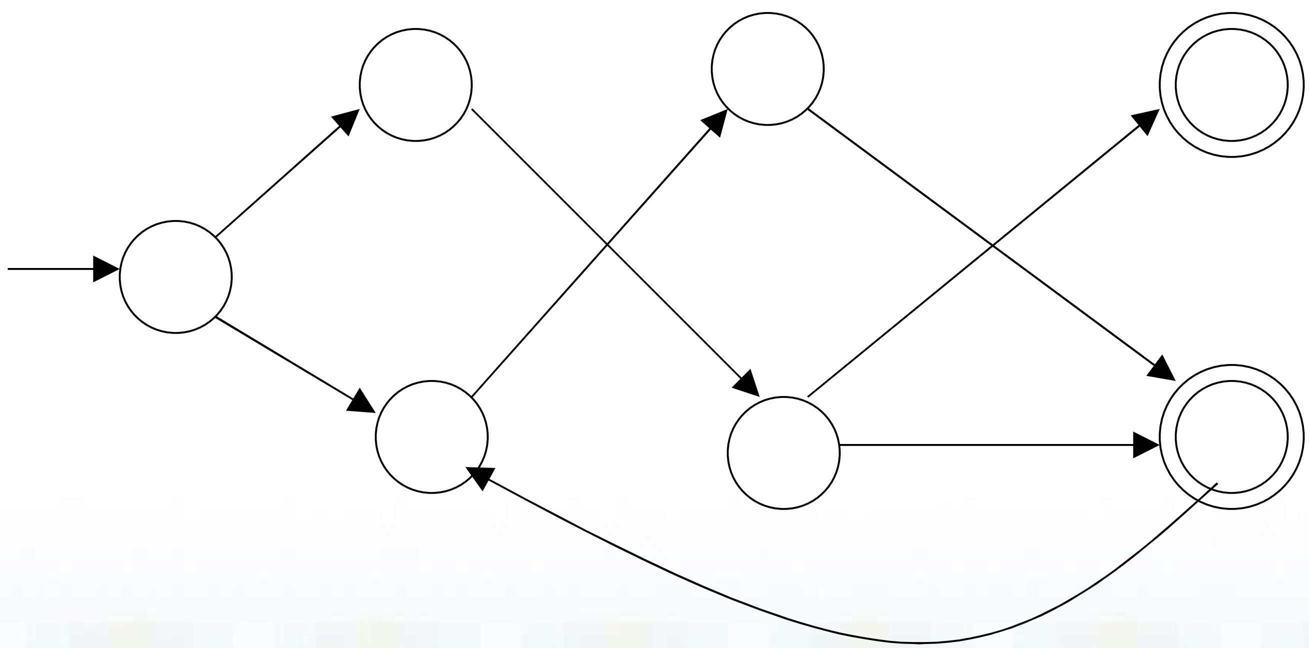
Theorem 4.1

If L is an infinite regular language, then there is a number p (the pumping length) where

- if $w \in L$, $|w| \geq p$, then w may be divided into three pieces, $w = xyz$, satisfying the following conditions:
  1. for each $i \geq 0$, $xy^iz \in L$,
  2. $|y| > 0$, and
  3. $|xy| \leq p$.

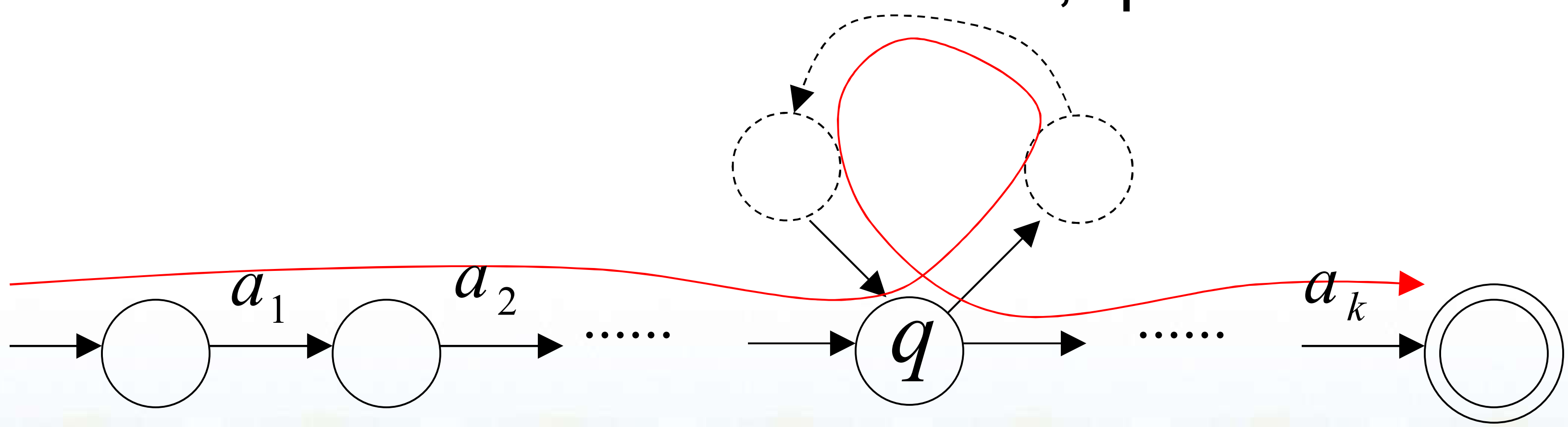
Proof of theorem 4.1

- Assume L is an infinite regular language
- There exists a minimum DFA recognize L



p states

- Consider string $w \in L$ with $|w| \geq p$
- Assume $w=a_1a_2\ldots a_k$. $|w|=k \geq p$
- To accept w, the sequence of states that M entered has k+1 elements . Because M has only p states, by pigeonhole (Dirichlet) principle, two elements must be the same state, q.

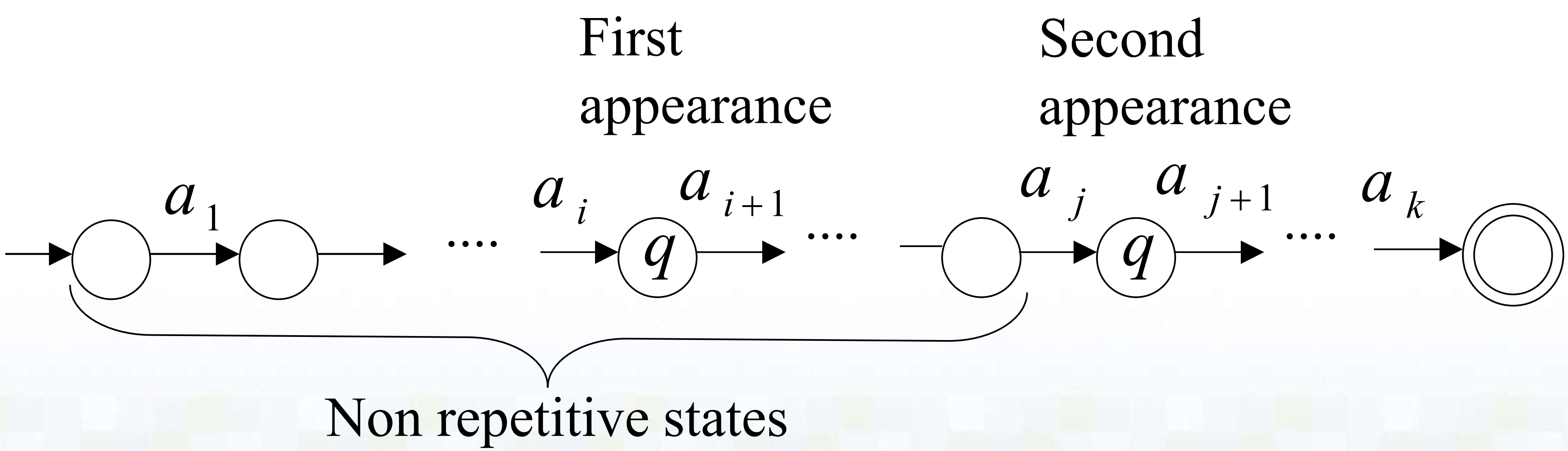


First repeated elements: the same state q

There may be many states visited twice or more

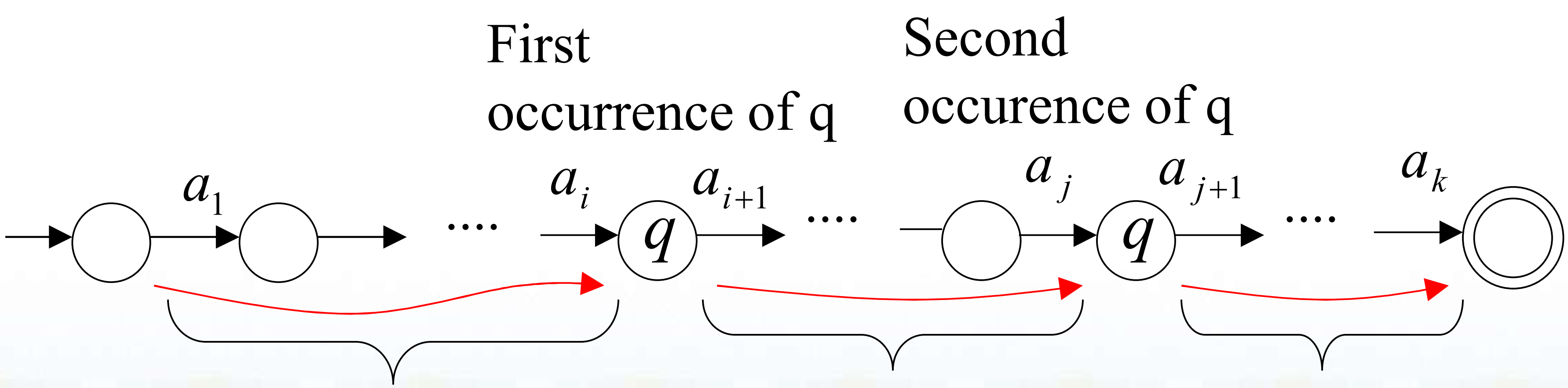
But q is the first repeated state

When M accept w



w can be divided in to 3 substrings $w = xyz$

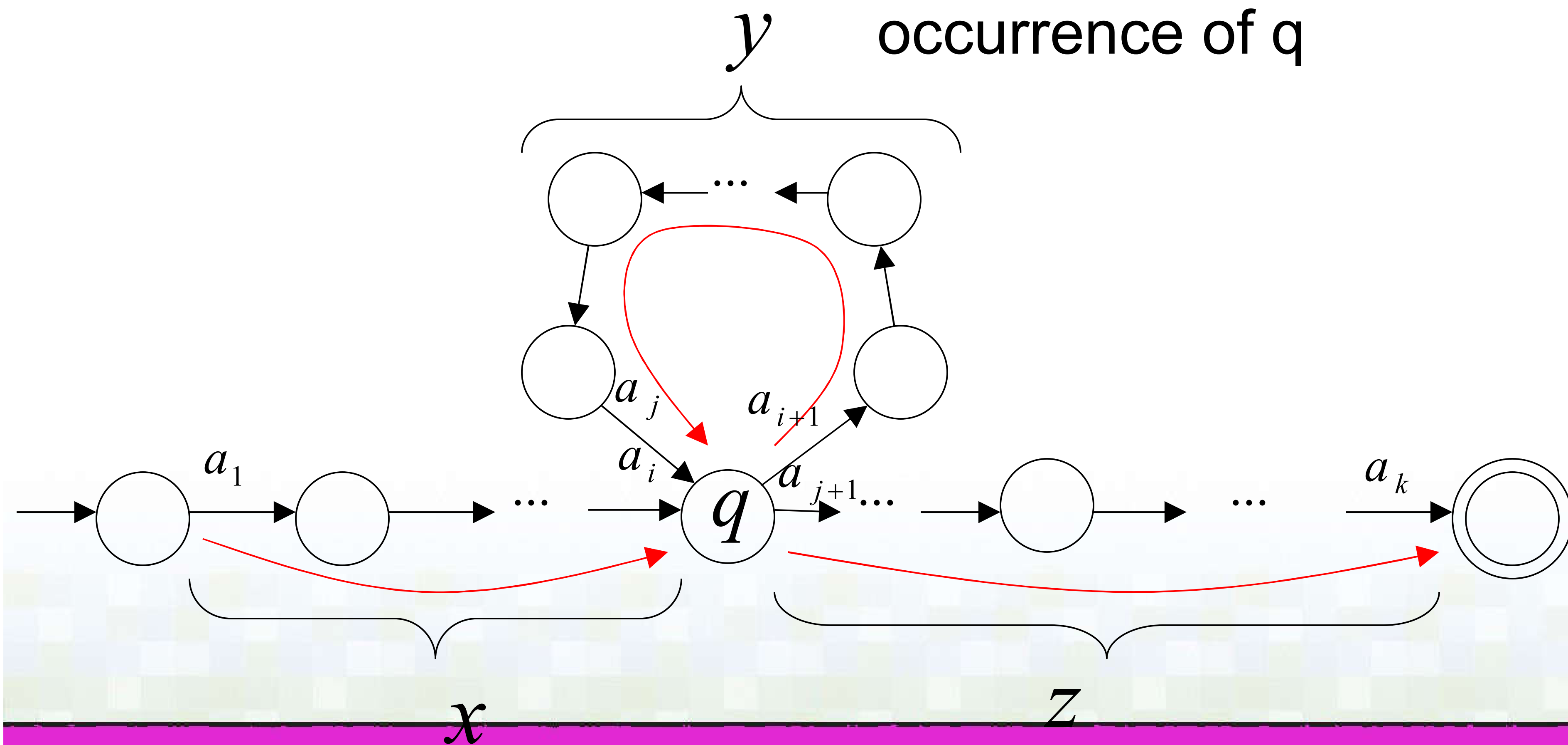
Once w is considered



$x = a_1 \cdots a_i$ $\quad$ $y = a_{i+1} \cdots a_j$ $\quad$ $z = a_{j+1} \cdots a_k$

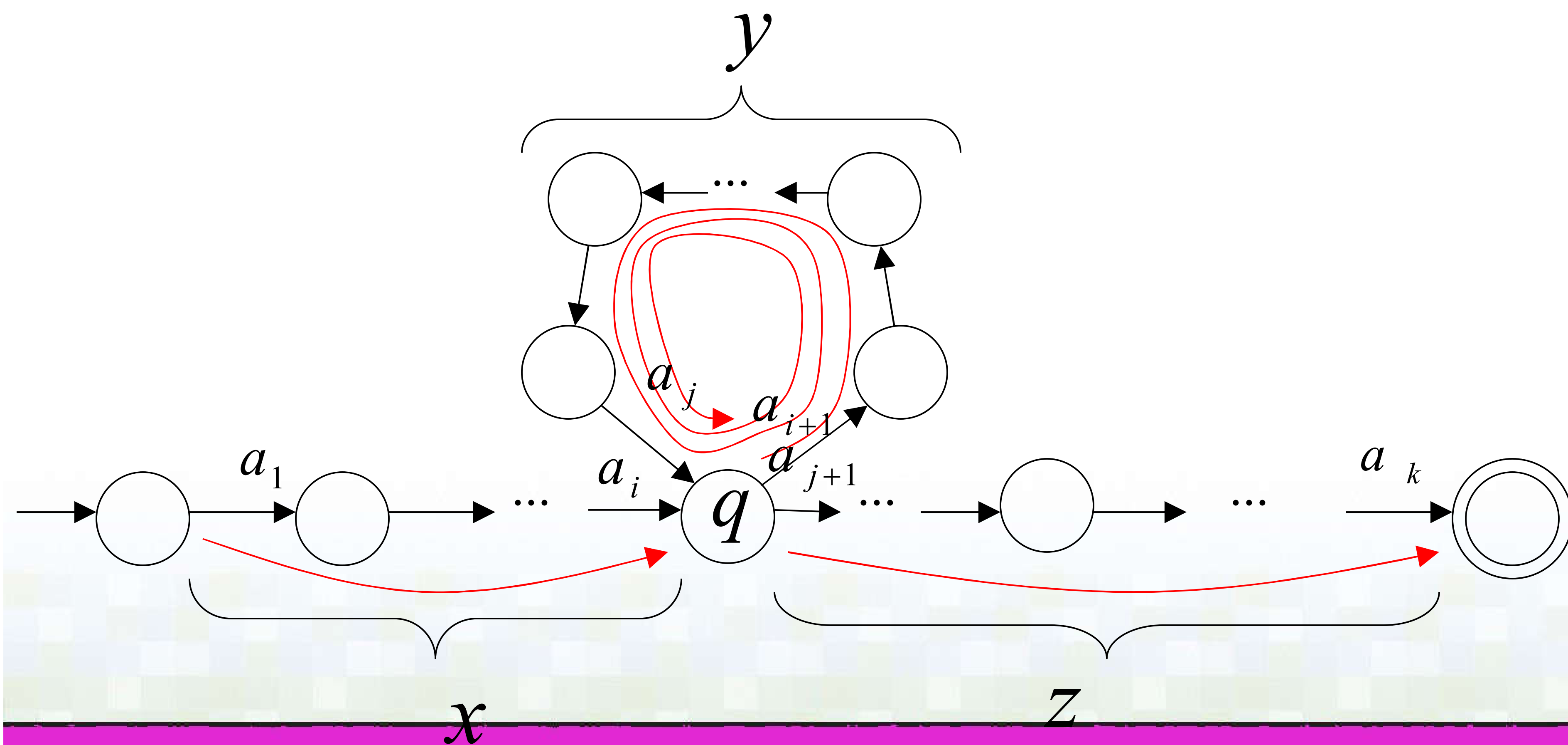
In DFA: w=xyz

Only contains 1st occurrence of q



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

Language recognized by M



Use the pumping lemma to prove that a language is not regular,

- Prove by contradiction
- Assume language L is regular
- Apply the pumping lemma to L
- Using the pumping lemma to obtain contradiction
- Conclude that L is not regular

Method

- Assume language L is regular
- Suppose p is number given by the pumping lemma
- Find string w in L, $|w| \geq p$
- Demonstrate that w cannot be pumped by considering all ways of dividing w into x, y, and z, that is existence of $i \geq 0$:

$$w' = xy^i z \notin L$$

- The existence of w' contradicts the pumping lemma if L were regular. Hence L cannot be regular.

Example

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

is not regular

using the pumping lemma

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

- Assume L is a regular language.
- Because L is infinite, we can apply the pumping lemma

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

Suppose p is the number of the pumping lemma

Looking for a string w : $w \in L$

$$|w| \geq p$$

Found $w = a^p b^p$

By Regular pumping lemma:

It's possible to write: $w = a^p b^p = x\ y\ z$

with $|x\ y| \leq p \quad |y| \geq 1$

$$w = xyz = a^p b^p = \overbrace{a...aa...aa...a}^{p}\overbrace{b...b}^{p}$$

$$\underbrace{a...a}_{x}\underbrace{a...a}_{y}\underbrace{a...ab...b}_{z}$$

Therefore: $y = a^k, \quad 1 \leq k \leq p$

$$x\,y\,z = a^p\,b^p \quad y = a^k, \quad 1 \le k \le p$$

By Regular Pumping Lemma $x\,y^i\,z \in L$

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

That's why: $x\,y^2\,z \in L$

$$x\,y\,z = a^p b^p \qquad y = a^k, \quad 1 \le k \le p$$

By Regular Pumping Lemma $x\,y^2\,z \in L$

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

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

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

However

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

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

CONTRADICTION!!!

Therefore

Assumption “L is regular” is incorrect

Conclusion: L is not regular

Proof that $L = \{a^n b^n \mid n \geq 0\}$ is not regular

- Suppose L is regular. Since L is regular, we can apply the pumping lemma to L.
- Let p be the pumping length for L.
- Choose $w =$ **$a^p b^p$**. Note that $w \in L$ and $|w| \geq p$.
- From the pumping lemma, there exists some x, y, z where $xyz = w$ and $|xy| \leq p$, $|y| > 0$, and $\forall\ i \geq 0$, $xy^i z \in L$.
- Because $|xy| \leq p$, $y = a^k$,(y is all a's).
- We choose **$i = 2$** and **$xy^i z = a^{p+k} b^p$**. Because $|y| > 0$,ther $xy^2 z \notin L$ (the string has more a's than b's). Thus for all possible x, y, z: $xyz = w$, $\exists i$, $xy^i z \notin L$. **Contradiction**
- L is not regular.

Proof that $L = \{a^n b^m \mid n \geq m \geq 0\}$ is not regular

- Suppose L is regular. Since L is regular, we can apply the pumping lemma to L.
- Let p be the pumping length for L.
- Choose $w = \mathbf{a^p b^p}$. Note that $w \in L$ and $|w| \geq p$.
- From the pumping lemma, there exists some x, y, z where $xyz = w$ and $|xy| \leq p$, $|y| > 0$, and $\forall\, i \geq 0$, $xy^iz \in L$.
- Because $|xy| \leq p$, $y = a^k$,(y is all a's).
- We choose $\mathbf{i = 0}$ and $xy^iz = \mathbf{a^{p-k}b^p}$. Because $|y| > 0$,

$xy^0z = xz \notin L$ (the string has **less** a's than b's). Thus for all possible x, y, z: $xyz = w$, $\exists i$, $xy^iz \notin L$. **Contradiction**

- L is not regular.

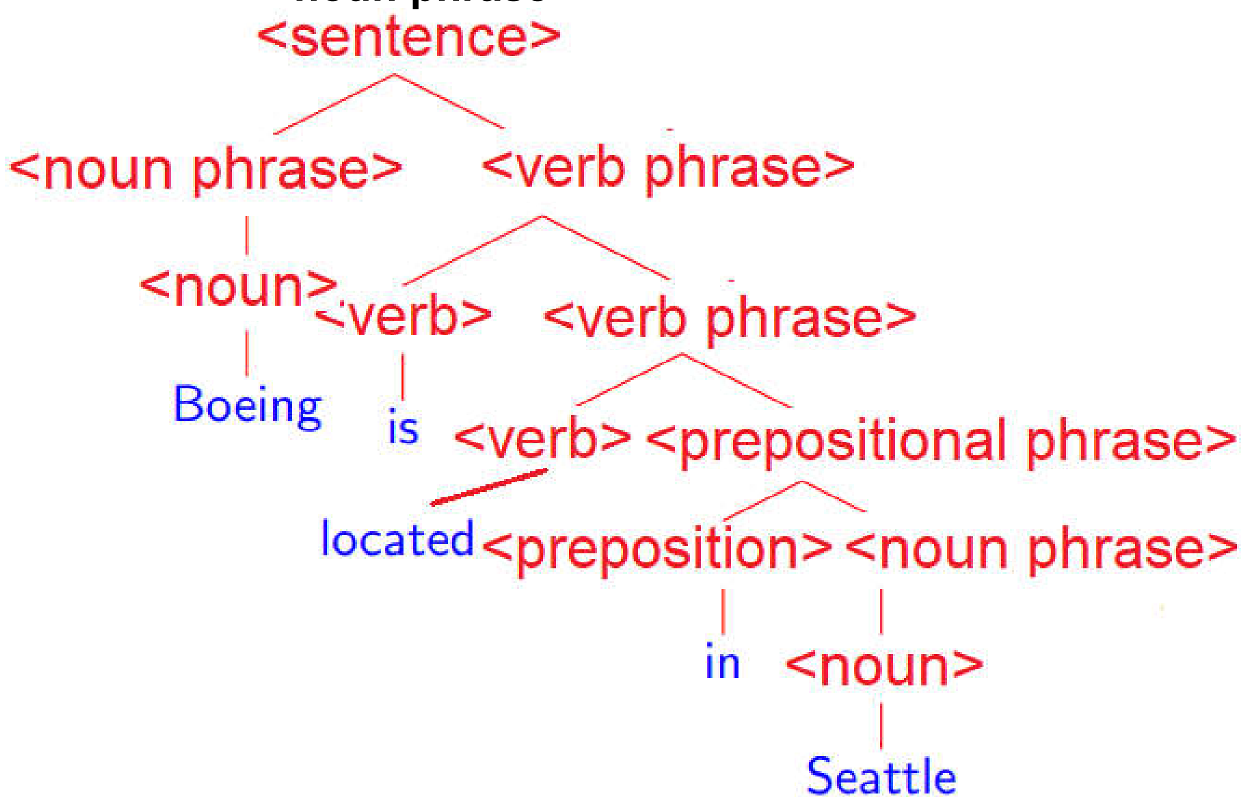
Context-Free Grammars

- Informal description
- Context-Free grammars
- Designing context-free grammars
- Ambiguity
- Chomsky normal form

Imagine a tiny set of syntax rules of English

<sentence>::=<noun phrase> <verb phrase>
<noun phrase>::= <noun>
<verb phrase>::= <verb> <verb phrase>
<verb phrase>::= <verb> <prepositional phrase>
<prepositional phrase>::= <preposition> <noun phrase>

< noun>::= Boeing
<noun> ::= Seattle
<verb>::= is
<verb>::= located
<preposition>::= in



Grammar for arithmetic expressions

<expression> --> number

<expression> --> (<expression>)

<expression> --> <expression> + <expression>

<expression> --> <expression> - <expression>

<expression> --> <expression> * <expression>

<expression> --> <expression> / <expression>

Context Free Grammars (CFG)

A context free grammar G has:

- A set of terminal symbols, Σ
- A set of nonterminal symbols, V
- A start symbol, S, which is a member of Δ
- A set R of production rules of the form A -> w, where A is a nonterminal and w is a string of terminal and nonterminal symbols or ε.

Formal definition of a context free grammar

A *context-free grammar* is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the *variables*,
2. Σ is a finite set, disjoint from V, called the *terminals*,
3. R is a finite set of *rules*, with each rule being a variable and a string of variables and terminals(form of a rule is $A \rightarrow \beta$ where $A \in V$ and $\beta \in (V \cup \Sigma)^*$)
4. $S \in V$ is the start variable.

Context Free Grammar Examples

- Grammar of nested parentheses
  $G_1 = (V, \Sigma, R, S)$ where

  $V = \{S\}$

  $\Sigma = \{ (,) \}$

  $R = \{ S \rightarrow (S), S \rightarrow SS, S \rightarrow \varepsilon \}$

Context Free Grammar Examples

The grammar of decimal numbers

$G_2 = (V, \Sigma, R, S)$, $V = \{S, A, B, C\}$,
$\Sigma = \{+,-, .,0, 1, 2,\ldots, 9\}$

R: $S \rightarrow +A \mid -A \mid A$

$A \rightarrow B.B \mid B$

$B \rightarrow BC \mid C$

$C \rightarrow 0 \mid 1 \mid 2 \mid . . . . \mid 9$

Derivations

- When X consists only of terminal symbols, it is a string of the language denoted by the grammar.
- Each iteration of the loop is a derivation step.
- If an iteration has several nonterminals to choose from at some point, the rules of derivation would allow any of these to be applied.

Definitions

- If u, v, and w are strings of variables and terminals, and $A \rightarrow w$ is a rule of the grammar, we say that uAv *yields* uwv, written $uAv \Rightarrow uwv$.
- Say that u *derives* v, written $u \Rightarrow^* v$,
  - if $u = v$ or
  - if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and
    $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$.

Example

- Consider grammar $G_1 = (\{S\}, \{(,)\}, \{S \rightarrow (S), S \rightarrow SS, S \rightarrow \varepsilon\}, S)$
- A derivation that generate string (()()) is

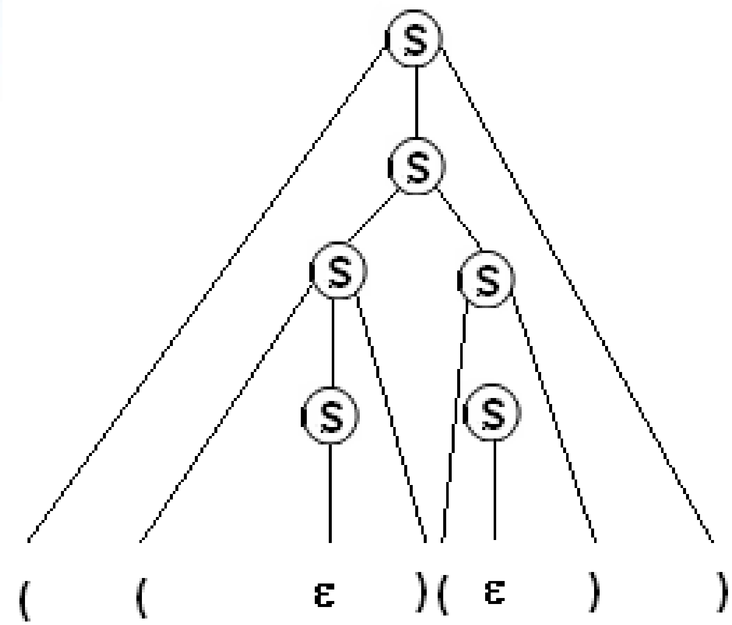
$$
\begin{aligned}
S &\Rightarrow (S) \\
&\Rightarrow (SS) \\
&\Rightarrow ((S)S) \\
&\Rightarrow (()S) \\
&\Rightarrow (()(S)) \\
&\Rightarrow (()())
\end{aligned}
$$

Derivation Tree (Parse Tree)

Derivation tree is constructed with

1) Each tree vertex is a variable (nonterminal) or terminal or epsilon
2) The root vertex is S
3) Interior vertices are from V, leaf vertices are from ∑ or ε
4) An interior vertex A has children, in order, left to right,

   $X_1, X_2, \ldots, X_k$ when there is a rule in R of the form $A \rightarrow X_1 X_2 \ldots X_k$
5) A leaf can be ε only when there is a production $A \rightarrow \varepsilon$ and the leaf's parent can have only this child.



Example: Derivation tree of string (()())

Designing context free grammars

- Many CFLs are the union of simpler CFLs. To construct a CFG for a CFL, you can break into simpler pieces, do so and then construct individual grammars for each piece then merge the grammars to get the necessary CFG.
- If a CFL is regular, construct a DFA for that language and convert DFA into an equivalent CFG
- Certain CFLs contain strings with two substrings that are “linked”, construct a CFG with a rule of the form $R \rightarrow uRv$
- In more complex languages, the strings may contain certain structures that appear recursively as part of other (or the same) structures.

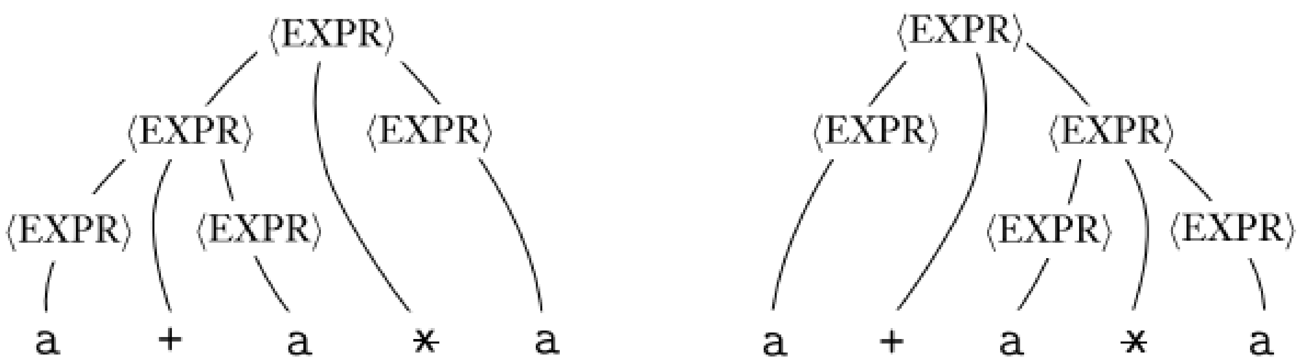
Ambiguity

- A string w is derived *ambiguously* in context-free grammar G if it has two or more different leftmost derivations.
- Grammar G is *ambiguous* if it generates some string ambiguously.

Example of an ambiguous grammar

Consider grammar G_3

<EXPR> → <EXPR> + <EXPR> | <EXPR> * <EXPR> |
(<EXPR>) | a



This grammar generates 2 different parse trees for string a + a * a

Grammar G_3 is ambiguous

Disambiguation

- Find an unambiguous grammar that generates the same language.

  <EXPR> $\rightarrow$ <EXPR> + <TERM> | <TERM>

  <TERM> $\rightarrow$ <TERM> * <FACTOR> | <FACTOR>

  <FACTOR> $\rightarrow$ (<EXPR>) | a

- Some context-free languages are *inherently ambiguous* because they can be generated only by ambiguous grammar.
  Example: $\{a^i b^j c^k \mid i = j \text{ or } j = k\}$.

Chomsky Normal Form – CNF

A grammar where every production is either of the form
$A \rightarrow BC$ or $A \rightarrow c$
where A, B, C are arbitrary variables and c an arbitrary symbol.
If language contains ε, then we allow $S \rightarrow \varepsilon$
where S is start symbol, and forbid S on RHS.

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a$$

Nonterminal Nonterminal

Terminal

Example

$S \to AS$	$S \to AS$
$S \to a$	$S \to AAS$
$A \to SA$	$A \to SA$
$A \to b$	$A \to aa$
CNF	Non CNF

Theorem

- Any context-free language is generated by a context-free grammar in Chomsky normal form
- PROOF IDEA
  - Convert a grammar G into CNF

Method

- The conversion to Chomsky Normal Form has four main steps:
  1. Get rid of all ε productions.
  2. Get rid of all productions where RHS is one variable.
  3. Replace every production that is too long by shorter productions.
  4. Move all terminals to productions where RHS still have one terminal.

Remove ε - rules

ε - rule: $X \rightarrow \varepsilon$

Nullable variables: $Y \Rightarrow \ldots \Rightarrow \varepsilon$

Example:

$$S \rightarrow aMb$$

$$M \rightarrow aMb$$

$$M \rightarrow \varepsilon$$

Nullable variable

ε - rule :

Remove ε - rules

$$S \rightarrow aMb$$

$$M \rightarrow aMb$$

$$\cancel{M \rightarrow \varepsilon}$$

Replace $M \rightarrow \varepsilon$

$$S \rightarrow aMb \mid ab$$

$$M \rightarrow aMb \mid ab$$

- Determine the nullable variables (those that generate ε)
- Go through all productions, and for each, omit every possible subset of nullable variables.
- Delete all productions with empty RHS.
- If the start variable is nullable then create a new start state S' , and add the rules to the grammar S' → S | ε

Remove unit rules

Unit rule is the rule of the form:

$$X \to Y$$

where both X and Y are nonterminals

Example:

$$S \to aA$$

$$A \to a$$

$$\boxed{A \to B}$$

$$\boxed{B \to A}$$

Unit rule

$$B \to bb$$

Remove unit rules

For all rules of form B → u, add the rule A → u unless this was a unit rule previously removed

$$S \to aA$$

$$A \to a$$

$$\cancel{A \to B}$$

$$\cancel{B \to A}$$

$$B \to bb$$

Replace $A \to B$

$$S \to aA \mid aB$$

$$A \to a \mid bb$$

$$B \to A$$

$$B \to bb$$

Remove unit rules

$$S \to aA \mid aB$$

$$A \to a$$

$$\cancel{B \to A}$$

$$B \to bb$$

replace

$$B \to A$$

$$S \to aA \mid aB \mid aA$$

$$A \to a \mid bb$$

$$B \to bb \mid a$$

Remove unit rules

Remove duplicate rules

$$S \rightarrow \cancel{aA} \mid aB \mid aA$$
$$A \rightarrow a \mid bb$$
$$B \rightarrow bb \mid a$$

Result

$$S \rightarrow aA \mid aB$$
$$A \rightarrow a \mid bb$$
$$B \rightarrow bb \mid a$$

Convert a CFG to Chomsky Normal Form

- Consider grammar G:

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

G is not in CNF

We will convert grammar G to CNF

Replace Long Productions by Shorter Ones

- Replace each rule $A \rightarrow u_1u_2 \cdot \cdot \cdot u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with the set of rules
  - $A \rightarrow u_1A_1$
  - $A_1 \rightarrow u_2A_2$
  - $A_2 \rightarrow u_3A_3, \ldots,$
  - $A_{k-2} \rightarrow u_{k-1}u_k$.
    The Ai's are new variables
- Replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \rightarrow u_i$.

Example

- Convert G to CNF

$$S \rightarrow aSb \mid \varepsilon$$

Make new start variable

- The new start variable will not appear on the RHS of any rule

$$S' \rightarrow S \mid \varepsilon$$
$$S \rightarrow aSb \mid \varepsilon$$

Remove ε-rules

- Grammar before removing

  $S' \rightarrow S \mid \varepsilon$

  $S \rightarrow aSb \mid \varepsilon$

- Grammar after removing

  $S' \rightarrow S \mid \varepsilon$

  $S \rightarrow aSb \mid ab$

Remove unit rules

- Grammar before removing S' → S

  S' → S| ε

  S → aSb | ab

  Grammar after removing S' → S

  S' → aSb | ab | ε

  S → aSb | ab

Replace long rules and terminals

- Before replacing long rules

  $S' \rightarrow aSb \mid ab \mid \varepsilon$

  $S \rightarrow aSb \mid ab$

- Replace $S \rightarrow aSb$ using new variable A_1

  $S' \rightarrow aA_1 \mid ab \mid \varepsilon$

  $A_1 \rightarrow Sb$

  $S \rightarrow aA_1 \mid ab$

  $A_1 \rightarrow Sb$

- Replace a with U, b with V and add rules $U \rightarrow a$, $V \rightarrow b$

  $S' \rightarrow UA_1 \mid UV \mid \varepsilon$

  $A_1 \rightarrow SV$

  $S \rightarrow UA_1 \mid UV$

  - $U \rightarrow a$, $V \rightarrow b$