Lecture 12
Simplification of Context-Free Grammars and Normal Forms

COT 4420

Theory of Computation

Chapter 6

Normal Forms for CFGs

1. Chomsky Normal Form CNF

   Productions of form

   $A \to BC$ $A, B, C \in V$

   $A \to a$ $a \in T$

2. Greibach Normal Form GNF

   Productions of form

   $A \to aX$ $A \in V, \; a \in T, X \in V^*$

λ-productions

- Any production of a CFG of the form

$$A \rightarrow \lambda$$

is called a λ-production.

- Any variable A for which the derivation

$$A \Rightarrow^* \lambda$$

is possible is called nullable.

Removing λ-productions

- Theorem: Given a grammar G with λ not in L(G) , the set of nullable variables V_N can be found using an algorithm.
- Proof:

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following until no new variables are added to V_N:

   For all productions $B \to A_1A_2...A_n$ where $A_1 , A_2 , ... , A_n$ are in V_N, put B into V_N.

Removing λ-productions

- Theorem: Let G be any CFG with λ not in L(G), then there exists an equivalent G having no λ-productions.
- Algorithm:

1. Find the set V_N of all nullable variables.
2. For all productions of the form $A \rightarrow x_1x_2\ldots x_m$, $m \geq 1$ where $x_i \in V \cup T$:

   We put this production in the new production set, as well as all those generated by replacing nullable variables with λ in all possible combinations.

   Exception: If all x_i's are nullable, do not include $A \rightarrow \lambda$

Removing λ-productions Example

- Example: Nullable variables V_N: A, B, C

$S \rightarrow ABaC$

$A \rightarrow BC$

$B \rightarrow b \mid \lambda$

$C \rightarrow D \mid \lambda$

$D \rightarrow d$

$S \rightarrow ABaC \mid BaC \mid AaC \mid ABa \mid$
$\quad aC \mid Ba \mid Aa \mid a$

$A \rightarrow BC \mid B \mid C$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

Removing λ-productions Proof

Proof: We need to show that:

1. If $w \neq \lambda$ and $A \Rightarrow^*_{old} w$, then $A \Rightarrow^*_{new} w$.
2. If $A \Rightarrow^*_{new} w$ then $w \neq \lambda$ and $A \Rightarrow^*_{old} w$.

Proof of (1): By induction on the number of steps by which A derives w in the old grammar.

Basis: If in the old grammar, A derives w in one step, then $A \rightarrow w$ must be a production. Since $w \neq \lambda$, this production must appear in the new grammar as well. Therefore, $A \Rightarrow^*_{new} w$.

Removing λ-productions Proof – cont'd

Induction step: We assume the theorem is true for derivation steps of fewer than k. We show it for $A \Rightarrow^*_{old} w$ which has k steps.

Let the first step be $A \Rightarrow_{old} X_1 \ldots X_n$, then w can be broken into $w = w_1 \ldots w_n$, where $X_i \Rightarrow^*_{old} w_i$, for all i, in fewer than k steps. Because of the induction hypothesis we have: $X_i \Rightarrow^*_{new} w_i$

The new grammar has a production $A \rightarrow_{new} X_1 \ldots X_n$, therefore A derives w in the new grammar.

Removing unit productions

- A unit production is one whose right-hand side has only one variable. $A \to B$

Use a **dependency graph**: Whenever the grammar has a unit-production $C \to D$, create an edge (C,D)

- If E =>* F using only unit productions, whenever $F \to \alpha$ is a non-unit production, add $E \to \alpha$.
- Remove unit productions

Removing unit productions Example

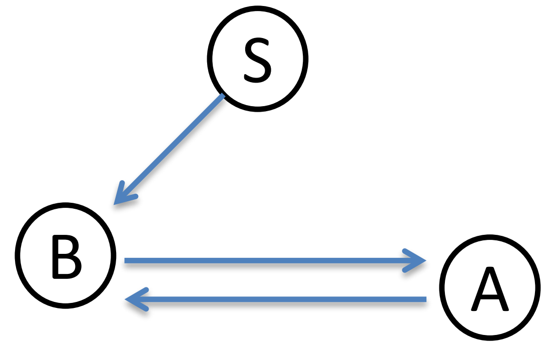
Original Grammar

$S \rightarrow Aa \mid B$

$B \rightarrow A \mid bb$

$A \rightarrow a \mid bc \mid B$

Dependency Graph



New Grammar

$S \rightarrow Aa \mid bb \mid bc \mid a$

$B \rightarrow bb \mid bc \mid a$

$A \rightarrow a \mid bc \mid bb$

S=>*B, S=>*A,

A=>*B, B=>*A

Remove useless productions

- Variable A is useful if there exist some $w \in L(G)$ such that:

$$S \Rightarrow^* xAy \Rightarrow^* w$$

Otherwise it is useless.

Example 1:

$S \rightarrow aSb \mid ab \mid A$

$A \rightarrow bAa$

A does not derive terminal strings

Example 2:

$S \rightarrow aSb \mid ab$

$A \rightarrow bAa \mid ba$

A is not reachable

Useless productions

The order is important!

- To remove useless productions:(follow these steps)
  1. Eliminate variables that derive no terminal

     $S \rightarrow aAb \qquad A \rightarrow bAa$

     A is useless
  2. Eliminate unreachable variables

     $S \rightarrow AC \qquad C \rightarrow aAb \qquad B \rightarrow Ab$

     B is not reachable

Remove useless productions Example

$S \to aS \mid A \mid C$

$A \to a$

$B \to aa$

$C \to aCb$

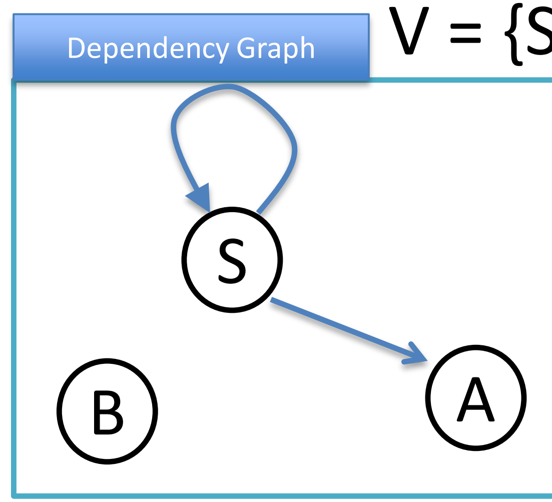
C is useless, so we remove variable C and its productions.

After step (1), every remaining symbol derives some terminal.

Remove useless productions Example

$S \to aS \mid A$
$A \to a$
$B \to aa$

V = {S, A, B}



Construct a **dependency graph** and determine the unreachable variables. (For every rule of the form $C \to xDy$, there is an edge from C to D.)

B is not reachable!

Cleaning up the grammar

1. Eliminate λ-productions
2. Eliminate unit productions
3. Eliminate useless variables

The order is important because removing λ-productions, will introduce new unit productions or useless variables.

Theorem: Let L be a CFL that does not contain λ. Then there exist a context-free grammar that generates L and does not have any useless productions, λ-productions, or unit-productions.

Converting to CNF

$A \to BC$
$A \to a$

$A, B, C \in V$
$a \in T$

- Theorem: Every context-free language L is generated by a Chomsky Normal Form (CNF) grammar.
- Proof: Let G be a CFG for generating L.

  Step1: First clean the grammar G. (remove λ-productions and unit-productions)

  Step2: For every production $A \to x_1x_2...x_n$, if n = 1, x_1 is a terminal (since there is no unit productions).

  If $n \geq 1$, for every terminal $a \in T$, introduce a variable B_a. Replace a with B_a and add $B_a \to a$ to the set of productions.

Step 2 - Example

$A \rightarrow GcDe$

$A \rightarrow G\mathbf{B_c}De$

$B_c \rightarrow c$

Step 2 - Example

$A \rightarrow GcDe$

$A \rightarrow GB_cD\mathbf{B_e}$

$B_c \rightarrow c$

$B_e \rightarrow e$

Converting to CNF – Cont'd

- Every production is of the form:

$A \rightarrow K_1\ K_2\ K_3 \ldots K_n$ $\qquad A, K_i \in V$

or

$A \rightarrow a$ $\qquad a \in T$

Step 3: Break right sides longer than 2 into chain of productions:

$A \rightarrow K_1\ Z_1$

$Z_1 \rightarrow K_2\ Z_2 \quad \ldots$

A -> BCDE is replaced by
A -> BF, F -> CG, and G -> DE.

Greibach Normal Form

GNF

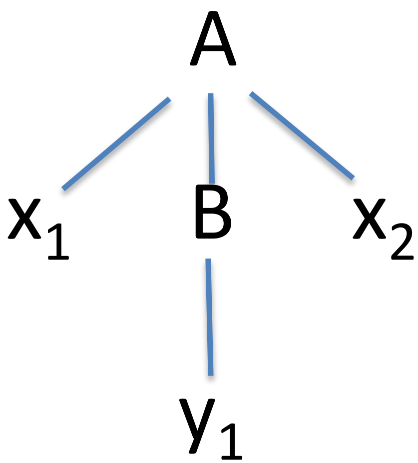
$A \rightarrow aX$

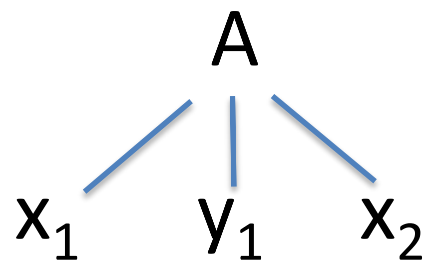
$A \in V$
$a \in T$
$X \in V^*$

Theorem: Every context-free language L is generated by a Greibach Normal Form (GNF) grammar.

Lemma1 (theorem 6.1 in textbook): Let G=(V, T, S, P) be a CFG. Suppose P contains a production of the form $A \rightarrow x_1B\ x_2$. Assume that A and B are different variables and that $B \rightarrow y_1 \mid y_2 \mid \ldots \mid y_n$ is the set of all productions in P which have B as the left side.

We can then remove $A \rightarrow x_1B\ x_2$ from P and add $A \rightarrow x_1y_1x_2 \mid x_1y_2x_2 \mid \ldots \mid x_1y_nx_2$
And have the same language.





These derive the same sentential forms

$A \rightarrow ABa$
$B \rightarrow AA \mid b \mid ZC$

⇒

$A \rightarrow AAAa \mid Aba \mid AZCa$

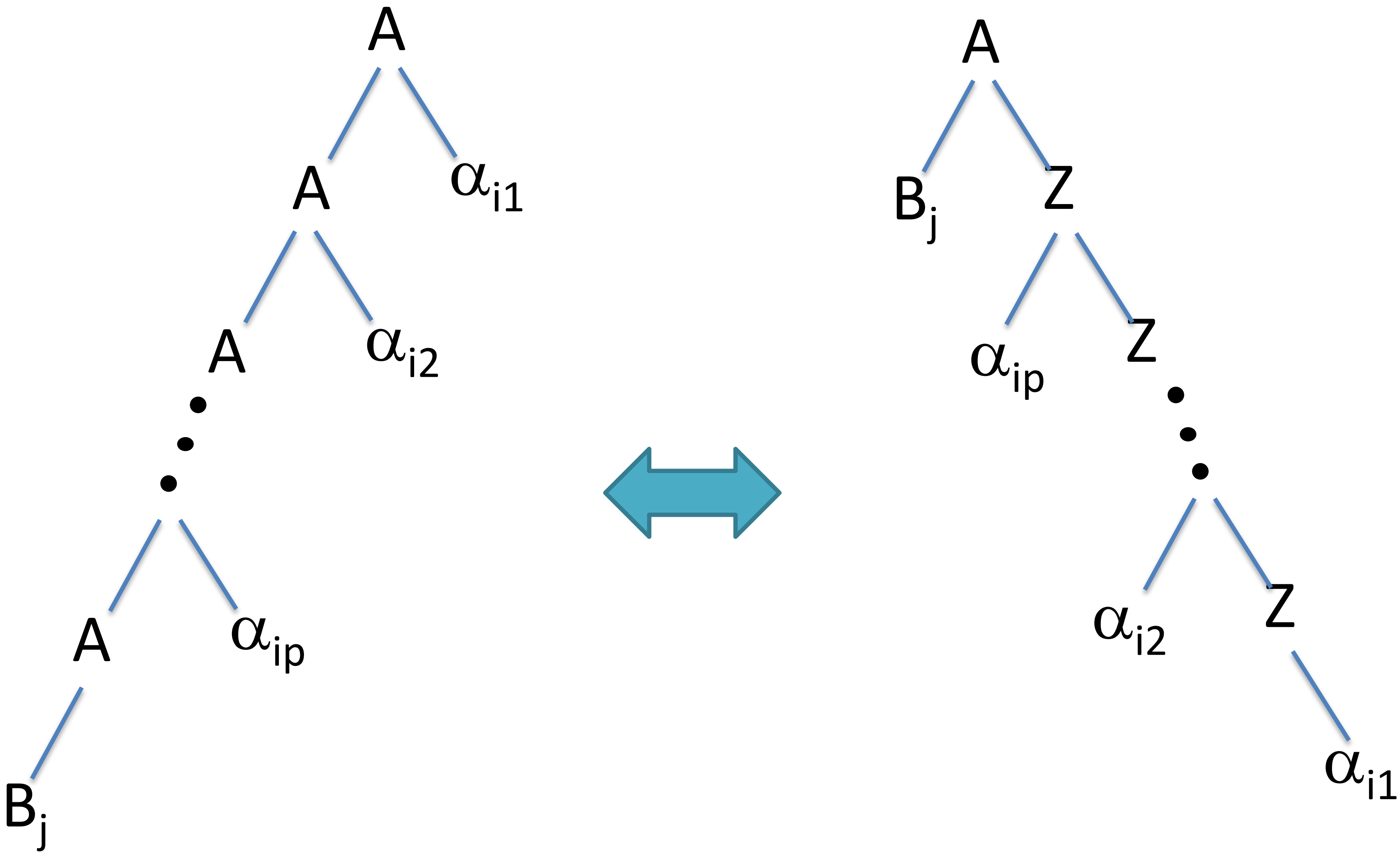
Lemma2, removing left recursion:

Let G=(V, T, S, P) be a CFG. $A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_n$ be the set of A-productions that have A as the first symbol on the R.H.S. And let $A \rightarrow \beta_1 | \beta_2 | \ldots | \beta_m$ be all the other A-productions.

We can remove the left recursive A-productions and add:

$A \rightarrow \beta_i \quad 1 \leq i \leq m$ $\qquad Z \rightarrow \alpha_i \quad 1 \leq i \leq n$

$A \rightarrow \beta_i Z$ $\qquad Z \rightarrow \alpha_i Z$


A
A
α_i1
A
α_i2
A
α_ip
B_j
A
B_j
Z
α_ip
Z
α_i2
Z
α_i1

Converting to GNF

$A \to aX$

$A \in V$
$a \in T$
$X \in V^*$

Theorem: Every context-free language L is generated by a Greibach Normal Form (GNF) grammar.

Step1: Rewrite the grammar into Chomsky Normal Form.

Step2: Relabel all variables as $A_1, A_2, \ldots A_n$

Step3: Transform all productions into form:

a. $A_i \to A_j x_j$ $\quad i < j$ and $x_j \in V^*$ $\quad$ or

b. $A_i \to a\, x_j$ $\quad$ or

c. $Z_i \to A_j x_j$

Converting to GNF - Example

$S \to SS \mid BC$

$B \to CB \mid a$

$C \to SB \mid b$

$S = A_1$, $B = A_2$, $C = A_3$

$A_1 \to A_1A_1 \mid A_2A_3$

$A_2 \to A_3A_2$

$A_3 \to A_1A_2$

$A_2 \to a$

$A_3 \to b$

Converting to GNF - Example

$A_1 \rightarrow A_1A_1$ apply lemma 2 to remove left recursion

$A_1 \rightarrow A_2A_3$

$A_1 \rightarrow A_2A_3Z_1$

$Z_1 \rightarrow A_1$

$Z_1 \rightarrow A_1Z_1$

~~$A_1 \rightarrow A_1A_1$~~

✓ (a) $A_1 \rightarrow A_2A_3$

✓ (a) $A_2 \rightarrow A_3A_2$

$A_3 \rightarrow A_1A_2$

✓ (b) $A_2 \rightarrow a$

✓ (b) $A_3 \rightarrow b$

✓ (a) $A_1 \rightarrow A_2A_3Z_1$

✓ (c) $Z_1 \rightarrow A_1$

✓ (c) $Z_1 \rightarrow A_1Z_1$

Converting to GNF - Example

$A_3 \rightarrow A_1A_2$ apply lemma 1 to replace A_1

$A_3 \rightarrow A_2A_3A_2$

$A_3 \rightarrow A_2A_3Z_1A_2$

Apply lemma 1 again

$A_3 \rightarrow A_3A_2A_3A_2$

$A_3 \rightarrow aA_3A_2$

$A_3 \rightarrow A_3A_2A_3Z_1A_2$

$A_3 \rightarrow aA_3Z_1A_2$

✓ (a) $A_1 \rightarrow A_2A_3$

✓ (a) $A_2 \rightarrow A_3A_2$

~~$A_3 \rightarrow A_1A_2$~~

✓ (b) $A_2 \rightarrow a$

✓ (b) $A_3 \rightarrow b$

✓ (a) $A_1 \rightarrow A_2A_3Z_1$

✓ (c) $Z_1 \rightarrow A_1$

✓ (c) $Z_1 \rightarrow A_1Z_1$

$A_3 \rightarrow A_3A_2A_3A_2$

✓ (b) $A_3 \rightarrow aA_3A_2$

$A_3 \rightarrow A_3A_2A_3Z_1A_2$

✓ (b) $A_3 \rightarrow aA_3Z_1A_2$

Converting to GNF - Example

$A_3 \rightarrow A_3A_2A_3A_2$ apply lemma2 to remove recursion

$A_3 \rightarrow A_3A_2A_3Z_1A_2$

$A_3 \rightarrow bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2$

$Z_2 \rightarrow A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

✓ (a) $A_1 \rightarrow A_2A_3$

✓ (a) $A_2 \rightarrow A_3A_2$

✓ (b) $A_2 \rightarrow a$

✓ (b) $A_3 \rightarrow b$

✓ (a) $A_1 \rightarrow A_2A_3Z_1$

✓ (c) $Z_1 \rightarrow A_1$

✓ (c) $Z_1 \rightarrow A_1Z_1$

~~$A_3 \rightarrow A_3A_2A_3A_2$~~

✓ (b) $A_3 \rightarrow aA_3A_2$

~~$A_3 \rightarrow A_3A_2A_3Z_1A_2$~~

✓ (b) $A_3 \rightarrow aA_3Z_1A_2$

Converting to GNF - Example

✓ (a) $A_1 \rightarrow A_2A_3 \mid A_2A_3Z_1$

✓ (a) $A_2 \rightarrow A_3A_2$

✓ (b) $A_2 \rightarrow a$

✓ (b) $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

✓ (c) $Z_1 \rightarrow A_1 \mid A_1Z_1$

✓ (c) $Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

Now everything is in the form of step 3. Note that A_n is in the form of GNF.

Converting to GNF

Step4: For every production of the form $A_{n-1} \rightarrow A_n x_n$ use lemma 1 to convert to correct GNF form. Continue to A_1.

For all Z-productions, use lemma 1 to convert to correct GNF form.

Example –Cont'd

$A_1 \rightarrow A_2A_3 \mid A_2A_3Z_1$

$A_2 \rightarrow A_3A_2$

✓ $A_2 \rightarrow a$

✓ $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_1 \rightarrow A_1 \mid A_1Z_1$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

For $A_2 \rightarrow A_3A_2$

we write:

$A_2 \rightarrow bA_2 \mid aA_3A_2A_2 \mid aA_3Z_1A_2A_2 \mid bZ_2A_2 \mid$
$aA_3A_2Z_2A_2 \mid aA_3Z_1A_2Z_2A_2$

Example –Cont'd

$A_1 \rightarrow A_2A_3 \mid A_2A_3Z_1$

✓ $A_2 \rightarrow bA_2 \mid aA_3A_2A_2 \mid aA_3Z_1A_2A_2 \mid bZ_2A_2 \mid aA_3A_2Z_2A_2 \mid aA_3Z_1A_2Z_2A_2 \mid a$

✓ $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_1 \rightarrow A_1 \mid A_1Z_1$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

For $A_1 \rightarrow A_2A_3 \mid A_2A_3Z_1$

we write:

$A_1 \rightarrow bA_2A_3 \mid aA_3A_2A_2A_3 \mid aA_3Z_1A_2A_2A_3 \mid bZ_2A_2A_3 \mid aA_3A_2Z_2A_2A_3 \mid aA_3Z_1A_2Z_2A_2A_3 \mid aA_3$

$A_1 \rightarrow bA_2A_3Z_1 \mid aA_3A_2A_2A_3Z_1 \mid aA_3Z_1A_2A_2A_3Z_1 \mid bZ_2A_2A_3Z_1 \mid aA_3A_2Z_2A_2A_3Z_1 \mid aA_3Z_1A_2Z_2A_2A_3Z_1 \mid aA_3Z_1$

Example –Cont'd

- ✓ $A_1 \rightarrow bA_2A_3 \mid aA_3A_2A_2A_3 \mid aA_3Z_1A_2A_2A_3 \mid bZ_2A_2A_3 \mid aA_3A_2Z_2A_2A_3 \mid aA_3Z_1A_2Z_2A_2A_3 \mid aA_3 \mid bA_2A_3Z_1 \mid aA_3A_2A_2A_3Z_1 \mid aA_3Z_1A_2A_2A_3Z_1 \mid bZ_2A_2A_3Z_1 \mid aA_3A_2Z_2A_2A_3Z_1 \mid aA_3Z_1A_2Z_2A_2A_3Z_1 \mid aA_3Z_1$

- ✓ $A_2 \rightarrow bA_2 \mid aA_3A_2A_2 \mid aA_3Z_1A_2A_2 \mid bZ_2A_2 \mid aA_3A_2Z_2A_2 \mid aA_3Z_1A_2Z_2A_2 \mid a$

- ✓ $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_1 \rightarrow A_1 \mid A_1Z_1$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

Example –Cont'd

- ✓ $A_1 \rightarrow bA_2A_3 \mid aA_3A_2A_2A_3 \mid aA_3Z_1A_2A_2A_3 \mid bZ_2A_2A_3 \mid aA_3A_2Z_2A_2A_3 \mid aA_3Z_1A_2Z_2A_2A_3 \mid aA_3 \mid bA_2A_3Z_1 \mid aA_3A_2A_2A_3Z_1 \mid aA_3Z_1A_2A_2A_3Z_1 \mid bZ_2A_2A_3Z_1 \mid aA_3A_2Z_2A_2A_3Z_1 \mid aA_3Z_1A_2Z_2A_2A_3Z_1 \mid aA_3Z_1$

- ✓ $A_2 \rightarrow bA_2 \mid aA_3A_2A_2 \mid aA_3Z_1A_2A_2 \mid bZ_2A_2 \mid aA_3A_2Z_2A_2 \mid aA_3Z_1A_2Z_2A_2 \mid a$

- ✓ $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_1 \rightarrow A_1 \mid A_1Z_1$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

Example –Cont'd

- ✓ $A_1 \rightarrow bA_2A_3 \mid aA_3A_2A_2A_3 \mid aA_3Z_1A_2A_2A_3 \mid bZ_2A_2A_3 \mid aA_3A_2Z_2A_2A_3 \mid aA_3Z_1A_2Z_2A_2A_3 \mid aA_3 \mid bA_2A_3Z_1 \mid aA_3A_2A_2A_3Z_1 \mid aA_3Z_1A_2A_2A_3Z_1 \mid bZ_2A_2A_3Z_1 \mid aA_3A_2Z_2A_2A_3Z_1 \mid aA_3Z_1A_2Z_2A_2A_3Z_1 \mid aA_3Z_1$

- ✓ A_2 … $Z_2A_2 \mid$ aA_3

Use lemma1 for all Z-productions

- ✓ $A_3 \rightarrow b \mid aA_3A_2 \mid aA_3Z_1A_2 \mid bZ_2 \mid aA_3A_2Z_2 \mid aA_3Z_1A_2Z_2$

$Z_1 \rightarrow A_1 \mid A_1Z_1$

$Z_2 \rightarrow A_2A_3A_2 \mid A_2A_3Z_1A_2 \mid A_2A_3A_2Z_2 \mid A_2A_3Z_1A_2Z_2$

The CYK Parser

The CYK membership algorithm

Input:

Grammar G in Chomsky Normal Form

String $w = a_1 a_2 .... a_n$

Output:

find if $w \in L(G)$

The Algorithm

Define:

w_{ij} : is a substring $a_i....a_j$

V_{ij} : { $A \in V : A =>^* w_{ij}$ }

$A \in V_{ii}$ if and only if G contains $A \rightarrow a_i$

$A \in V_{ij}$ if and only if G contains $A \rightarrow BC$, and

$B \in V_{ik}$, and $C \in V_{k+1j}$, ($k \in \{i, i+1, ..., j-1\}$)

The Algorithm

1. Compute V_{11}, V_{22}, … ,V_{nn}
2. Compute V_{12}, V_{23}, … ,$V_{n-1,n}$
3. Compute V_{13}, V_{24},…., $V_{n-2,n}$
4. And so on….

If $S \in V_{1n}$ then $w \in L(G)$, otherwise $w \notin L(G)$.

Example

Grammar G and string w is given:

$S \rightarrow AB$ $\quad\quad\quad\quad$ $w = aabbb$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

Example

1. Compute V_{11}, V_{22}, … ,V_{55}

Note that: $A \in V_{ii}$ if and only if G contains $A \rightarrow a_i$

V_{11} = ? Is there a rule that directly derives a_1 ?

$V_{11} = \{A\}$

V_{22} = ? Is there a rule that directly derives a_2 ?

$V_{22} = \{A\}$

V_{33} = ? Is there a rule that directly derives a_3 ?

$V_{33} = \{B\}$

$V_{44} = \{B\}$, $V_{55} = \{B\}$

Example

2. Compute V_{12}, V_{23}, … ,V_{45}

Note that: $A \in V_{ij}$ if and only if G contains $A \rightarrow BC$, and $B \in V_{ik}$, and $C \in V_{k+1j}$ for all k's

$V_{12} = ?\ \{ A : A \rightarrow BC , B \in V_{11} , C \in V_{22} \}$ Variable $\overset{?}{\rightarrow}$ AA

$V_{12} = \{\}$

$V_{23} = ?\ \{ A : A \rightarrow BC , B \in V_{22} , C \in V_{33} \}$ Variable $\overset{?}{\rightarrow}$ AB

$V_{23} = \{S, B\}$

$V_{34} = ?\ \{ A : A \rightarrow BC , B \in V_{33} , C \in V_{44} \}$ Variable $\overset{?}{\rightarrow}$ BB

$V_{34} = \{A\}$

$V_{45} = ?\ \{ A : A \rightarrow BC , B \in V_{44} , C \in V_{55} \}$ Variable $\overset{?}{\rightarrow}$ BB

$V_{45} = \{A\}$

Example

3. Compute V_{13}, V_{24}, V_{35}

$V_{13} = ?\ \{ A : A \rightarrow BC , B \in V_{11} , C \in V_{23} \}$ Variable $\overset{?}{\rightarrow}$ AS, AB

$\{ A : A \rightarrow BC, B \in V_{12} , C \in V_{33} \}$

$V_{13} = \{S, B\}$

$V_{24} = ?\ \{ A : A \rightarrow BC , B \in V_{22} , C \in V_{34} \}$ Variable $\overset{?}{\rightarrow}$ AA

$\{ A : A \rightarrow BC, B \in V_{23} , C \in V_{44} \}$ Variable $\overset{?}{\rightarrow}$ SB, BB

$V_{24} = \{A\}$

$V_{35} = ?\ \{ A : A \rightarrow BC , B \in V_{33} , C \in V_{45} \}$ Variable $\overset{?}{\rightarrow}$ BA

$\{ A : A \rightarrow BC, B \in V_{34} , C \in V_{55} \}$ Variable $\overset{?}{\rightarrow}$ AB

$V_{35} = \{S, B\}$

Example

4. Compute V_{14}, V_{25}

$V_{14} = ?\ \{ A : A \rightarrow BC\ ,\ B \in V_{11}\ ,\ C \in V_{24} \}$ Variable $\overset{?}{\rightarrow}$ AA

$\{ A : A \rightarrow BC,\ B \in V_{12}\ ,\ C \in V_{34} \}$

$\{ A : A \rightarrow BC,\ B \in V_{13}\ ,\ C \in V_{44} \}$ Variable $\overset{?}{\rightarrow}$ SB, BB

$V_{14} = \{A\}$

$V_{25} = ?\ \{ A : A \rightarrow BC\ ,\ B \in V_{22}\ ,\ C \in V_{35} \}$ Variable $\overset{?}{\rightarrow}$ AS, AB

$\{ A : A \rightarrow BC,\ B \in V_{23}\ ,\ C \in V_{45} \}$ Variable $\overset{?}{\rightarrow}$ SA, BA

$\{ A : A \rightarrow BC,\ B \in V_{24}\ ,\ C \in V_{55} \}$ Variable $\overset{?}{\rightarrow}$ AB

$V_{25} = \{S, B\}$

Example

5. Compute V_{15}

$V_{15} = ?\ \{ A : A \rightarrow BC , B \in V_{11} , C \in V_{25} \}$ Variable $\overset{?}{\rightarrow}$ AS, AB

$\{ A : A \rightarrow BC, B \in V_{12} , C \in V_{35} \}$

$\{ A : A \rightarrow BC, B \in V_{13} , C \in V_{45} \}$ Variable $\overset{?}{\rightarrow}$ SA, BA

$\{ A : A \rightarrow BC, B \in V_{14} , C \in V_{55} \}$ Variable $\overset{?}{\rightarrow}$ AB

$V_{15} = \{S, B\}$

$S \in V_{15}$, therefore $w = aabbb \in L(G)$

	1 **a**	2 **a**	3 **b**	4 **b**	5 **b**
1	{A}				
2		{A}			
3			{B}		
4				{B}	
5					{B}

	1	2	3	4	5
	a	**a**	**b**	**b**	**b**
1	{A}	{}			
2		{A}	{S, B}		
3			{B}	{A}	
4				{B}	{A}
5					{B}

	1 **a**	2 **a**	3 **b**	4 **b**	5 **b**
1	{A}	{}	{S, B}		
2		{A}	{S, B}	{A}	
3			{B}	{A}	{S, B}
4				{B}	{A}
5					{B}

	1 **a**	2 **a**	3 **b**	4 **b**	5 **b**
1	{A}	{}	{S, B}	{A}	{S, B}
2		{A}	{S, B}	{A}	{S, B}
3			{B}	{A}	{S, B}
4				{B}	{A}
5					{B}

Approximate time complexity:

$$O(|w|^2 \cdot |w|) = O(|w|^3)$$

Number of V_{ij}'s to be computed

If |w| = n
n(n-1)/2

Number of evaluations in each V_{ij}

at most n