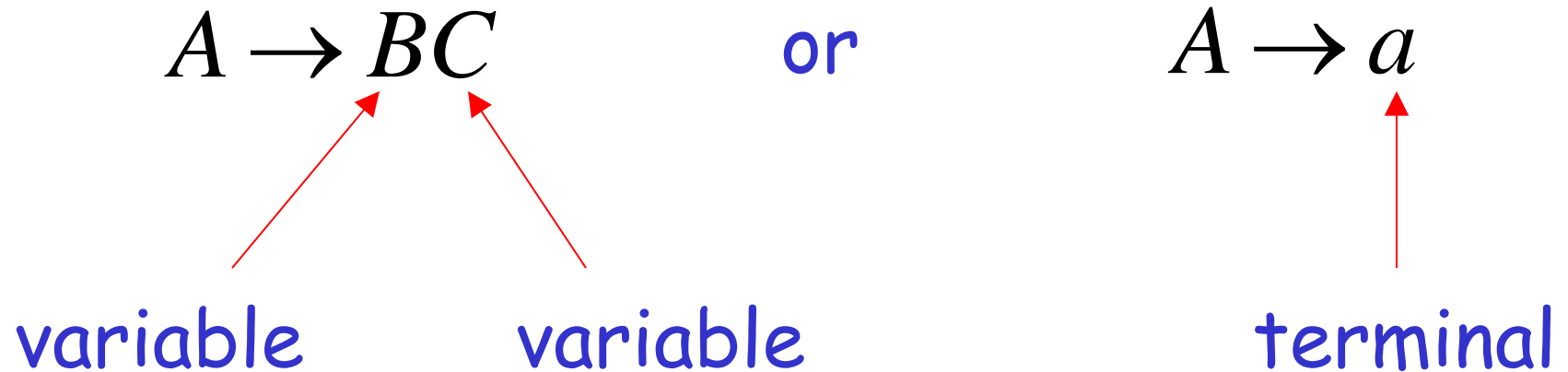


# Normal Forms for Context-free Grammars

# Chomsky Normal Form

Each productions has form:



## Examples:

$$S \rightarrow AS$$

$$S \rightarrow a$$

$$A \rightarrow SA$$

$$A \rightarrow b$$

Chomsky  
Normal Form

$$S \rightarrow AS$$

$$S \rightarrow AAS$$

$$A \rightarrow SA$$

$$A \rightarrow aa$$

Not Chomsky  
Normal Form

# Conversion to Chomsky Normal Form

• Example:  $S \rightarrow ABa$

$A \rightarrow aab$

$B \rightarrow Ac$

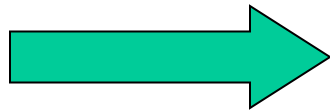
Not Chomsky  
Normal Form

Introduce variables for terminals:  $T_a, T_b, T_c$

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$



$$S \rightarrow ABT_a$$

$$A \rightarrow T_aT_aT_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

Introduce intermediate variable:  $V_1$

$$S \rightarrow ABT_a$$

$$A \rightarrow T_a T_a T_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$



$$S \rightarrow AV_1$$

$$V_1 \rightarrow BT_a$$

$$A \rightarrow T_a T_a T_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

Introduce intermediate variable:  $V_2$

$$S \rightarrow AV_1$$

$$V_1 \rightarrow BT_a$$

$$A \rightarrow T_aT_aT_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$



$$S \rightarrow AV_1$$

$$V_1 \rightarrow BT_a$$

$$A \rightarrow T_aV_2$$

$$V_2 \rightarrow T_aT_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

# Final grammar in Chomsky Normal Form:

$$S \rightarrow AV_1$$

$$V_1 \rightarrow BT_a$$

$$A \rightarrow T_aV_2$$

$$V_2 \rightarrow T_aT_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

## Initial grammar

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$



In general:

From any context-free grammar  
(which doesn't produce  $\lambda$ )  
not in Chomsky Normal Form

we can obtain:

An equivalent grammar  
in Chomsky Normal Form

# The Procedure

First remove:

Nullable variables

Unit productions

Then, for every symbol  $a$ :

Add production  $T_a \rightarrow a$

In productions: replace  $a$  with  $T_a$

New variable:  $T_a$

Replace any production  $A \rightarrow C_1 C_2 \cdots C_n$

with  $A \rightarrow C_1 V_1$

$V_1 \rightarrow C_2 V_2$

$\dots$

$V_{n-2} \rightarrow C_{n-1} C_n$

New intermediate variables:  $V_1, V_2, \dots, V_{n-2}$

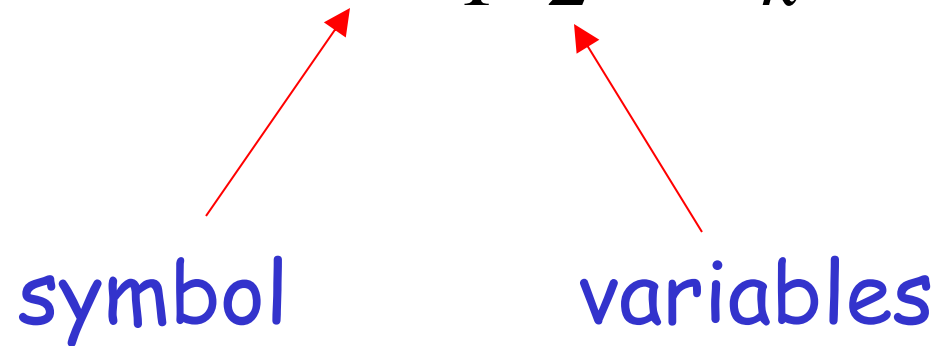
**Theorem:** For any context-free grammar  
(which doesn't produce  $\lambda$  )  
there is an equivalent grammar  
in Chomsky Normal Form

# Observations

- Chomsky normal forms are good for parsing and proving theorems
- It is very easy to find the Chomsky normal form for any context-free grammar

# Greibach Normal Form

All productions have form:

$$A \rightarrow a V_1 V_2 \cdots V_k \quad k \geq 0$$


symbol

variables

## Examples:

$$S \rightarrow cAB$$

$$A \rightarrow aA \mid bB \mid b$$

$$B \rightarrow b$$

Greibach  
Normal Form

$$S \rightarrow abSb$$

$$S \rightarrow aa$$

Not Greibach  
Normal Form



## Conversion to Greinbach Normal Form:

$$S \rightarrow abSb$$

$$S \rightarrow aa$$



$$S \rightarrow aT_bST_b$$

$$S \rightarrow aT_a$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

Greinbach  
Normal Form

**Theorem:** For any context-free grammar  
(which doesn't produce  $\lambda$ )  
there is an equivalent grammar  
in Greinbach Normal Form

# Observations

- Greinbach normal forms are very good for parsing
- It is hard to find the Greinbach normal form of any context-free grammar

# The CYK Parser

# The CYK Membership Algorithm

## Input:

- Grammar  $G$  in Chomsky Normal Form
- String  $w$

## Output:

find if  $w \in L(G)$

this claim. The algorithm we will describe here is called the CYK algorithm, after its originators J. Cocke, D. H. Younger, and T. Kasami. The algorithm works only if the grammar is in Chomsky normal form and succeeds by breaking one problem into a sequence of smaller ones in the following way. Assume that we have a grammar  $G = (V, T, S, P)$  in Chomsky normal form and a string

$$w = a_1 a_2 \cdots a_n.$$

We define substrings

$$w_{ij} = a_i \cdots a_j,$$

and subsets of  $V$

$$V_{ij} = \left\{ A \in V : A \overset{*}{\Rightarrow} w_{ij} \right\}.$$

Clearly,  $w \in L(G)$  if and only if  $S \in V_{1n}$ .

To compute  $V_{ij}$ , observe that  $A \in V_{ii}$  if and only if  $G$  contains a production  $A \rightarrow a_i$ . Therefore,  $V_{ii}$  can be computed for all  $1 \leq i \leq n$  by inspection of  $w$  and the productions of the grammar. To continue, notice that for  $j > i$ ,  $A$  derives  $w_{ij}$  if and only if there is a production  $A \rightarrow BC$ , with  $B \xRightarrow{*} w_{ik}$  and  $C \xRightarrow{*} w_{k+1j}$  for some  $k$  with  $i \leq k, k < j$ . In other words,

$$V_{ij} = \bigcup_{k \in \{i, i+1, \dots, j-1\}} \{A : A \rightarrow BC, \text{ with } B \in V_{ik}, C \in V_{k+1, j}\}. \quad (6.8)$$

An inspection of the indices in (6.8) shows that it can be used to compute all the  $V_{ij}$  if we proceed in the sequence

1. Compute  $V_{11}, V_{22}, \dots, V_{nn}$
2. Compute  $V_{12}, V_{23}, \dots, V_{n-1, n}$
3. Compute  $V_{13}, V_{24}, \dots, V_{n-2, n}$

# The Algorithm

Input example:

- Grammar  $G$ :
  - $S \rightarrow AB$
  - $A \rightarrow BB$
  - $A \rightarrow a$
  - $B \rightarrow AB$
  - $B \rightarrow b$
- String  $w$ :  $aabbbb$



*aabbbb*  
1 2 3 4 5

a

$v_{11}$

a

$v_{22}$

b

$v_{33}$

b

$v_{44}$

b

$v_{55}$

aa

$v_{12}$

ab

$v_{23}$

bb

$v_{34}$

bb

$v_{45}$

aab

$v_{13}$

abb

$v_{24}$

bbb

$v_{35}$

aabb

$v_{14}$

abbb

$v_{25}$

aabbb

$v_{15}$

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a	a	b	b	b
A	A	B	B	B
<hr/>				
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a	a	b	b	b
<i>A</i>	<i>A</i>	<i>B</i>	<i>B</i>	<i>B</i>

---

aa	ab	bb	bb
----	----	----	----

	<i>S, B</i>	<i>A</i>	<i>A</i>
--	-------------	----------	----------

---

aab	abb	bbb
-----	-----	-----

aabb	abbb
------	------

aabbb

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a

a

b

b

b

A

A

B

B

B

aa

ab

bb

bb

S,B

A

A

aab

abb

bbb

S,B

A

S,B

aabb

abbb

A

S,B

aabbb

S,B

Therefore:  $aabbb \in L(G)$

Time Complexity:  $|w|^3$

**Observation:** The CYK algorithm can be easily converted to a parser (bottom up parser)