

# Course 6

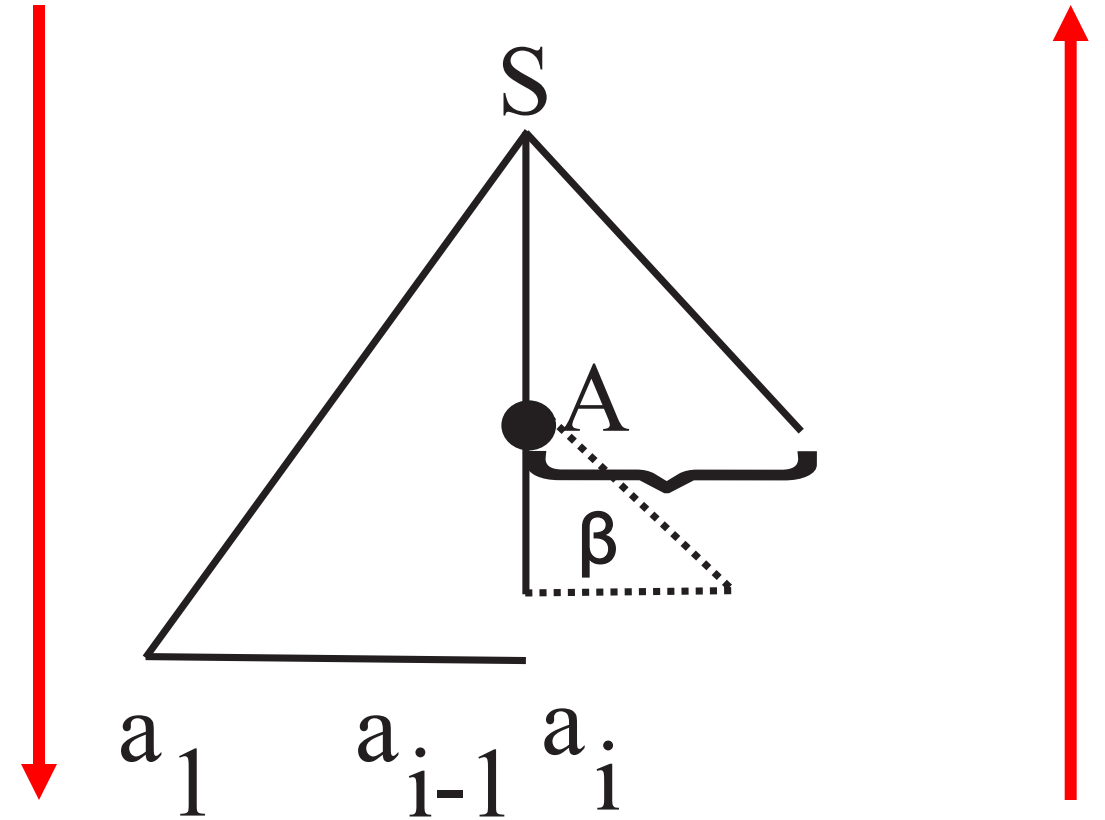
# Problem: Parsing (construct the parse tree)

**if** the *source program is syntactically correct*  
    **then** construct syntax tree  
    **else** "syntax error"

*source program is syntactically correct* =  $w \in L(G) \Leftrightarrow S \overset{*}{\Rightarrow} w$

# Parsing

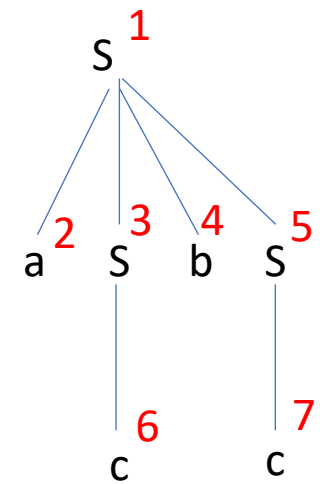
- How:
  1. Top-down vs. Bottom-up
  2. Recursive vs. linear



# Result – parse tree -representation

- Arbitrary tree – child sybling representation
- Sequence of derivations  $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = w$
- String of production – index associated to prod – which prod is used at each derivation step: 1,4,3,...

index	Info	Parent	Right sibling
1	S	0	0
2	a	1	0
3	S	1	2
4	b	1	3
5	S	1	4
6	c	3	0
7	c	5	0



# Descendent recursive parser

- Example

$S \rightarrow aSbS \mid aS \mid c$

# Formal model

- Configuration

$(s, i, \alpha, \beta)$

where:

- $s$  = state of the parsing, can be:
  - $q$  = normal state
  - $b$  = back state
  - $f$  = final state - corresponding to success:  $w \in L(G)$
  - $e$  = error state – corresponding to insuccess:  $w \notin L(G)$
- $i$  – position of current symbol in input sequence  
 $w = a_1a_2...a_n, i \in \{1,...,n+1\}$
- $\alpha$  = working stack, stores the way the parse is built
- $\beta$  = input stack, part of the tree to be built

Define moves between configurations

Initial configuration:  
 $(q, 1, \varepsilon, S)$

Final configuration:  
 $(f, n+1, \alpha, \varepsilon)$

# Expand

WHEN: head of input stack is a nonterminal

$$(q, i, \alpha, A\beta) \vdash (q, i, \alpha A_1, \gamma_1 \beta)$$

where:

$A \rightarrow \gamma_1 \mid \gamma_2 \mid \dots$  represents the productions corresponding to A

1 = first prod of A



# Advance

WHEN: head of input stack is a terminal = current symbol from input

$$(q, i, \alpha, a_i \beta) \vdash (q, i+1, \alpha a_i, \beta)$$

# Momentary insuccess

WHEN: head of input stack is a terminal  $\neq$  current symbol from input

$$(q, i, \alpha, a_i \beta) \vdash (\textcolor{red}{b}, i, \alpha, a_i \beta)$$

# Back

WHEN: head of working stack is a terminal

$$(b, i, \alpha a, \beta) \vdash (b, i-1, \alpha, a\beta)$$

# Another try

WHEN: head of working stack is a nonterminal

$(b, i, \alpha A_j, \gamma_j \beta) \vdash (q, i, \alpha A_{j+1}, \gamma_{j+1} \beta)$  , if  $\exists A \rightarrow \gamma_{j+1}$   
 $(b, i, \alpha, A \beta)$ , otherwise with the exception  
 $(e, i, \alpha, \beta)$ , if  $i=1$ ,  $A=S$ , **ERROR**

# Success

$$(q, n+1, \alpha, \varepsilon) \vdash (\textcolor{red}{f}, n+1, \alpha, \varepsilon)$$

# Algorithm

## Algorithm Descendent Recursive

**INPUT:**  $G, w = a_1a_2...a_n$

**OUTPUT:** string of productions and message

$config = (q, 1, \varepsilon, S);$

//initial configuration ( $s, i, \alpha, \beta$ )

**while** ( $s \neq f$ ) and ( $s \neq e$ ) **do**

**if**  $s = q$

**then if** ( $i = n + 1$ ) and  $IsEmpty(\beta)$

**then**  $Success(config)$

**else**

**if**  $Head(\beta) = A$

**then**  $Expand(config)$

**else**

**if**  $Head(\beta) = a_i$

**then**  $Advance(config)$

**else**  $MomentaryInsucces(config)$

**else**

**if**  $s = b$

**then**

**if**  $Head(\alpha) = a$

**then**  $Back(config)$

**else**  $AnotherTry(config)$

$endWhile$

**if**  $s = e$  **then**  $message "Error"$

**else**  $message "Sequence accepted";$

$BuildStringOfProd(\alpha)$

# $w \in L(G)$ - HOW

- Process  $\alpha$ :
  - From left to right (reverse if stored as stack)
  - Skip terminal symbols
  - Nonterminals – index of prod
- Example:  $\alpha = S_1 a S_2 a S_3 c b S_3 c$



# When the algorithm never stops?

- $S \rightarrow S\alpha$  – expand infinitely (left recursive)