Course 6

Problem: Parsing (construct the syntax tree)

if the *source program is sintactically correct*

then construct syntax tree

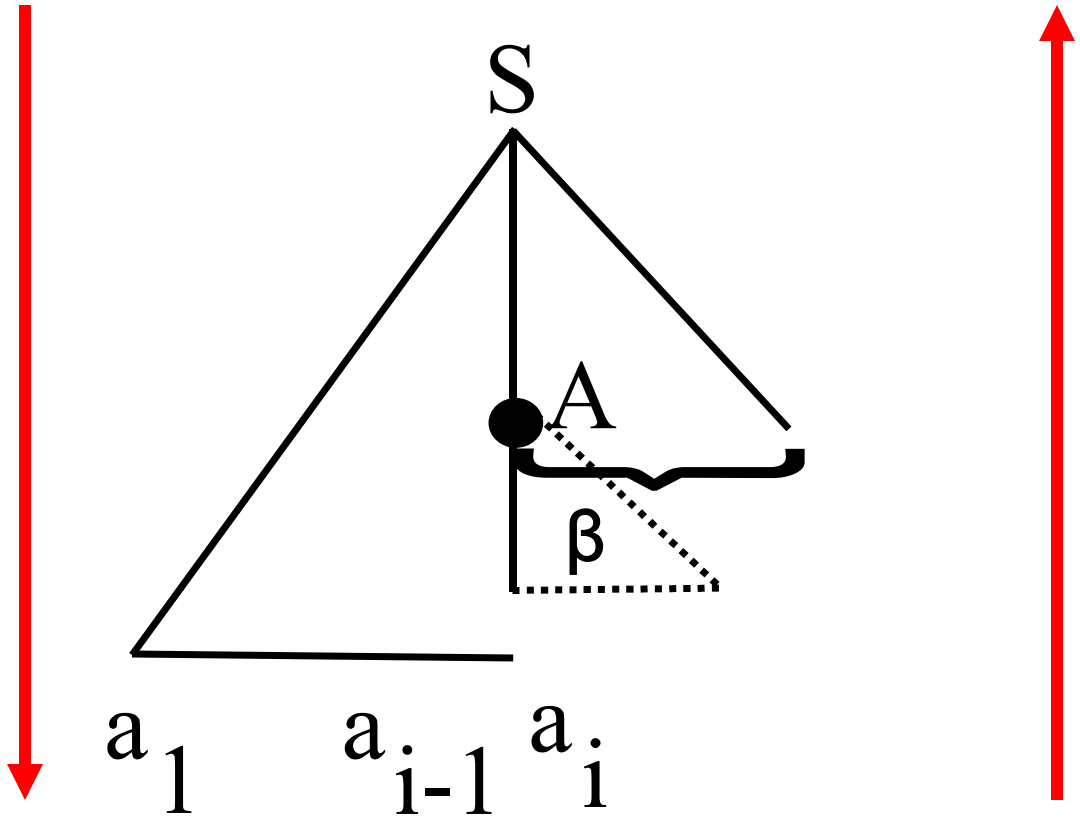
else ”syntax error”

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

Parsing

- Cfg G = (N, **Σ**, P,S) check if w ∈ L(G)
- Construct parse/syntax tree

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

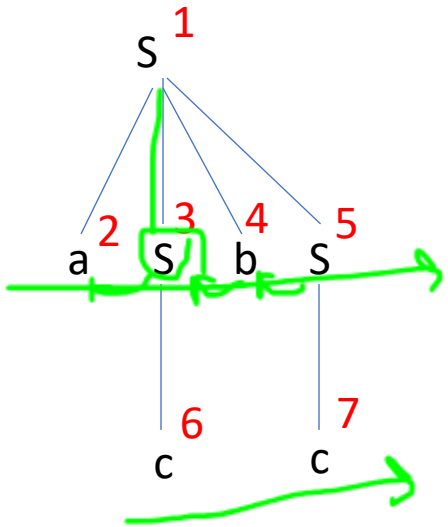


	Descendent	Ascendent
Recursive	Descendent recursive parser	Ascendent recursive parser
Linear	LL(k): LL(1)	LR(k): LR(0), SLR, LR(1), LALR

Result – parse tree -representation

- Arbitrary tree – child sybling representation
- Sequence of derivations S => α_1 => α_2 =>… => α_n = w
- String of production – index associated to prod – which prod is used at each derivation step

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 -> aSbS | aS | c

w = aacbc

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

  $w = a_1 a_2 \ldots a_n$, $i \in \{1,\ldots,n+1\}$
- α = working stack, stores the way the parse is built
- β = input stack, part of the tree to be built

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

Define moves between configurations

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:

$A \rightarrow \gamma_1 \mid \gamma_2 \mid \ldots$ 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 ≠ current symbol from input

$(q,i, \alpha, a_i\beta) \vdash (b,i, \alpha, \beta)$

Back

WHEN: head of working stack is a terminal

(b,i, αa, β) ⊢ (b,i-1, α, aβ)

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, α, β), if i=1, A =S, **ERROR**

Success

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

Algorithm

Algorithm Descendent Recursive

INPUT: G, w =$a_1a_2...a_n$
OUTPUT: string of productions and message

```
config = (q,1, ε,S);                                   //initial configuration (s,i,α,β)
while (s ≠ f) and (s ≠ e) do
   if s = q
     then if (i=n+1) and IsEmpty(β)
               then Success(config)
               else
                    if Head(β) = A
                       then Expand(config)
                       else
                          if  Head(β) = a_i
                              then Advance(config)
                              else MomentaryInsuccess(config)
     else
          if s = b
             then
                 if Head(α) = a
                   then Back(config)
                   else AnotherTry(config)
endWhile
if s = e  then message"Error"
          else message "Sequence accepted";
               BuildStringOfProd(α)
```

w ∈ L(G) - HOW

- Process $\boldsymbol{\alpha}$:
  - From left to right (reverse if stored as stack)
  - Skip terminal symbols
  - Nonterminals – index of prod

- Example: $\boldsymbol{\alpha}$ = S_1 a S_2 a S_3 c b S_3 c

When the algorithm will never finish? (loop infinitely)

- A -> Aα |b //left recursive