Course 6

Problem: Parsing (construct the parsee tree)

if the *source program is sintactically correct*

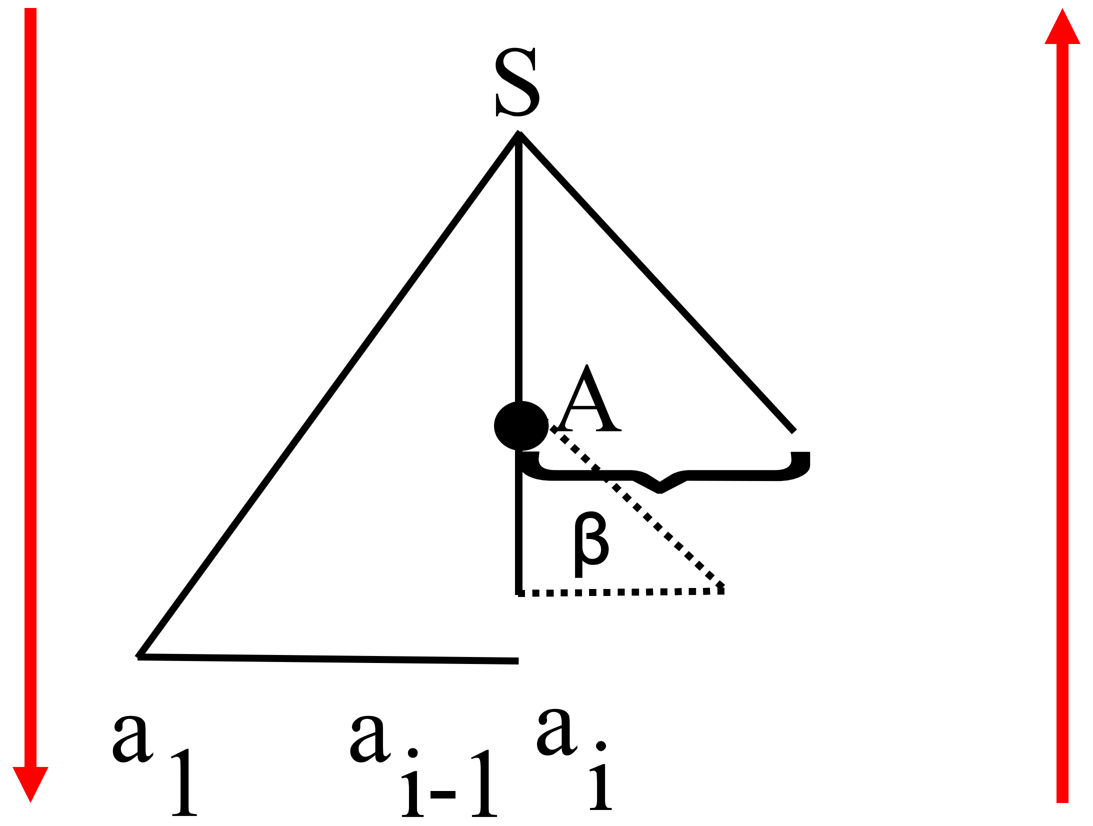
    then construct syntax tree

    else ”syntax error”

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

Parsing

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



Descendent recursive parser

- Example

S -> aSbS | aS | c

Formal model

- Configuration

$$(\mathbf{s}, \mathbf{i}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

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

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

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

Define moves between configurations

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

Expand

WHEN: head of input stack is a nonterminal

$(q,i, \boldsymbol{\alpha}, A\boldsymbol{\beta}) \vdash (q,i, \boldsymbol{\alpha}A_1, \boldsymbol{\gamma}_1\boldsymbol{\beta})$

where:

$A \rightarrow \boldsymbol{\gamma}_1 \mid \boldsymbol{\gamma}_2 \mid \ldots$ represents the productions corresponding to A

1 = first prod of A

Advance

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

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

Momentary insuccess

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

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

Back

WHEN: head of working stack is a terminal

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

Another try

WHEN: head of working stack is a nonterminal

$(b, i, \boldsymbol{\alpha} A_j, \boldsymbol{\gamma}_j \boldsymbol{\beta}) \vdash$ $(q, i, \boldsymbol{\alpha} A_{j+1}, \boldsymbol{\gamma}_{j+1} \boldsymbol{\beta})$, if $\exists\ A \to \boldsymbol{\gamma}_{j+1}$

$(b, i, \boldsymbol{\alpha}, A \boldsymbol{\beta})$, otherwise with the exception

$(e, i, \boldsymbol{\alpha}, \boldsymbol{\beta})$, 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 =$a_1a_2\ldots a_n$
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 MomentaryInsuccess(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 ∈ 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 never stops?

- S->S**α** – expand infinitely (left recursive)