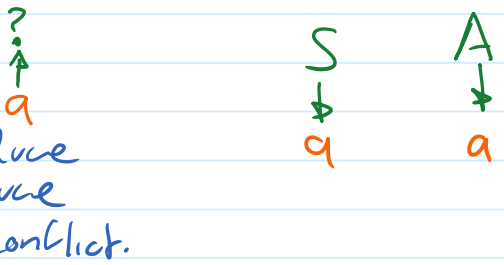
11 Grammar Issues with Top-Down Parsers

Monday, March 11, 2024 11:57 AM

- AMBIGUITY

Not an issue for top-Down parsers.

e.g $S \to a \mid A$, $A \to a$



- LACK OF PAIRWISE DISJOINTNESS

for a non-terminal A
more than one body of a rule of A
begins with the same prefix.

E.g. $A \to aD \mid aC \mid aaB \mid b$

```
FUNCTION parse_A()
  IF token = 'a' THEN
    getToken()
    ???
```

The token does not gives us enough information to decide which body to apply.

Solution 1.- read more tokens.

Solution 2.- hack the parser:- choose first body then backtrack.

Solution 3.- Change the grammar.

REMEDY: LEFT FACTORING

Replace: $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \ldots \mid \Gamma_1 \mid \Gamma_2 \mid \ldots$

α is a common prefix

Replace: $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \dots \mid \gamma_1 \mid \gamma_2 \mid \dots$

α is a common prefix

With: $A \to \alpha A' \mid \Gamma_1 \mid \Gamma_2 \mid \dots$

$A' \to \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots$

E.g.

$A \to aD \mid aC \mid aaB \mid b$

(α = a, β = D; α = a, β = C; α = aa, β = B; Γ = b)

Left factor:

$A \to aA' \mid b$

$A' \to D \mid C \mid aB$

Excersize:

⊗ $S \to zzyAy \mid zzyz \mid zzy \mid zzx \mid yyA \mid yyy$

⊗ $A \to xxAy \mid xzyA \mid yzx$

Left factor:

⊗ $S \to zzS' \mid yyA \mid yyy$

⊗ $S' \to yAy \mid yz \mid y \mid x$

$S \to zzS' \mid yyS''$

$S'' \to A \mid y$

$S' \to yS''' \mid x$

$S''' \to Ay \mid z \mid \lambda$

$A \to xA' \mid yzx$

$A' \to xAy \mid zyA$

- LEFT RECURSION
  - Direct
  - Indirect

$A \to Aa \mid Ab \mid c$

{ c, ca, cb, cabaab }

$A \to Bx$

$B \to Cy$

$A \to Aa \mid Ab \mid c$

$\{c, ca, cb, cabaab\}$

```
FUNCTION parse_A()
  parse_A()
  …
  …
```

$A \to Bx$
$B \to Cy$
$C \to Az \mid w$

parse_A()
↳ parse_B()
↳ parse_C()
↳ parse_A()

$A \to c \mid Aa \mid Ab$

```
FUNCTION parse_A()
  IF token = 'c' THEN
    getToken()
  ELSE
    parse_A()
    …
    …
```

REMEDY: DIRECT LEFT RECURSION ELIMINATION

Replace $A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid .. \mid \beta_2 \mid \beta_2 \mid \beta_3 \mid \ldots$

none Begin with A

by:

$A \to \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid \ldots$

$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \mid \ldots \mid \lambda$

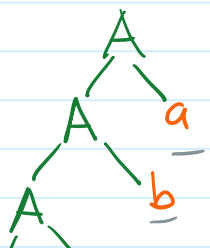
e.g. 1.

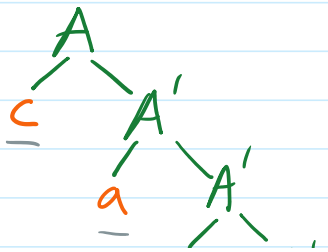
$A \to \underbrace{Aa}_{\alpha_1} \mid \underbrace{Ab}_{\alpha_2} \mid \underbrace{c}_{\beta} \Rightarrow$

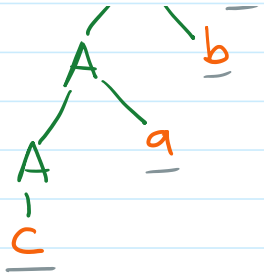
$A \to cA'$

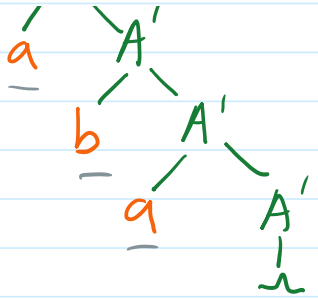
$A' \to aA' \mid bA' \mid \lambda$

caba









e.g #2

$E \to \underbrace{E + T}_{\alpha} \mid \underbrace{T}_{\beta}$

$T \to T * F \mid F$

$F \to int \mid (E)$

Problem: Left Recursion.

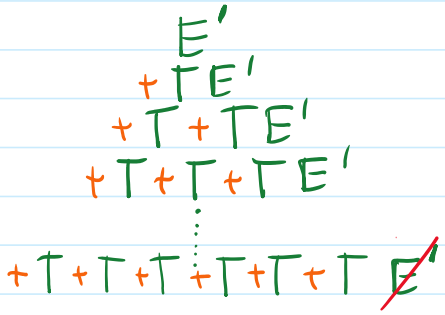
$E \to TE'$

$E' \to +TE' \mid \epsilon$

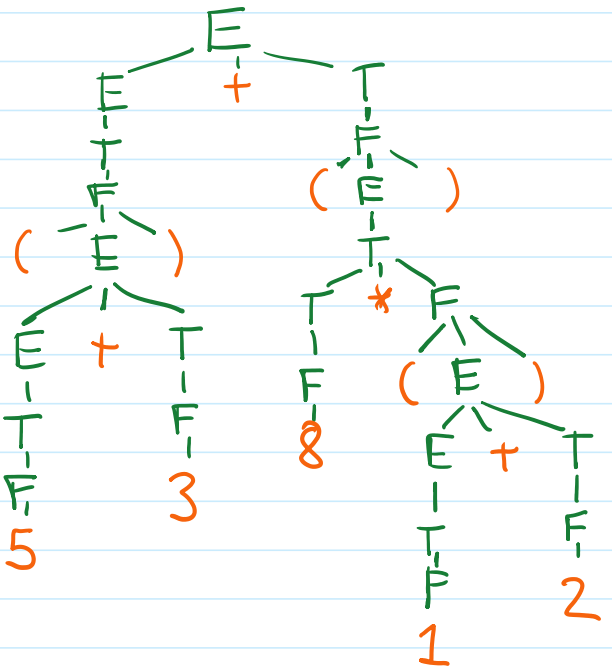
$T \to FT'$

$T' \to *FT' \mid \epsilon$

$F \to int \mid (E)$



(5+3) + (8*(1+2))



Extended BNF

$E \to T\{+T\}$

$T \to F\{*F\}$

$F \to int \mid (E)$

```
FUNCTION parse_E()
  parse_T()
  WHILE token = '+' DO
    getToken()
    parse_T()
```

```
FUNCTION parse_T()
  parse_F()
  WHILE token = '*' DO
    getToken()
    parse_F()
```

```
    getToken()                    getToken()
    parse_T()                     parse_F()
  END                           END
END.                          END.

FUNCTION parse_F()
  IF isInteger(token) THEN
    getToken()
  ELSIF token = '(' THEN
    getToken()
    parse_E()
    IF token = ')' THEN
      getToken()
    ELSE
      error("Unmatched parenthesis")
    END
  ELSE
    error("Factor expected")
  END
END.
```