Eliminating Left Recursion from Production

Left Recursive Grammar

- A left-recursive grammar has a non-terminal A such that

  $A \Rightarrow^{+} A\alpha$

- Top-down parsing methods (LL(1) and RD) cannot handle left-recursive grammars.

- Left-recursion in grammars can be eliminated by transformations.

- A simpler case is that of grammars with immediate left recursion, where there is a production of the form $A \rightarrow A\alpha$.

- Indirect left recursion
  - $A \rightarrow B\alpha$
  - $B \rightarrow A\alpha_1$

Elimination of Immediate Left Recursion

- A simpler case is that of grammars with immediate left recursion, where there is a production of the form $A \rightarrow A\alpha$
- Two productions

  $A \rightarrow A\alpha \mid \beta$

  can be transformed to

  $A \rightarrow \beta A'$,

  $A' \rightarrow \alpha A' \mid \epsilon$

In general

In general, a group of productions:

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m$

$\mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

can be transformed to

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$,

$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \epsilon$

Types of Left Recursive Grammar

Immediate Left Recursion

$$
\begin{aligned}
\underline{E} &\rightarrow \underline{E} + T \\
&\quad |\ \underline{E} - T \\
&\quad |\ T \\
\underline{T} &\rightarrow \underline{T} * F \\
&\quad |\ \underline{T} / F \\
&\quad |\ F \\
F &\rightarrow (E) \ |\ id
\end{aligned}
$$

Indirect Left Recursion

```
A -> Cd
B -> Ce
C -> A | B | f
```

$F \rightarrow (E) \ |\ id$

Eliminating Direct Left Recursion

$S \rightarrow R\,a \mid A\,a \mid a$
$R \rightarrow a\,b$
$A \rightarrow A\,R \mid A\,T \mid b$
$T \rightarrow T\,b \mid a$

After eliminating left recursive grammar equiv. to the above is:

$S \rightarrow R\,a \mid A\,a \mid a$

$R \rightarrow a\,b$

$A \rightarrow b\,A'$
$A' \rightarrow R\,A' \mid T\,A' \mid \epsilon$
$T \rightarrow a\,T'$
$T' \rightarrow b\,T' \mid \epsilon$

Left Recursion Elimination - An Example

Equivalent left-recursive but unambiguous grammar is:

E → E + T

E → T,

T → T F

T → F,

F → F∗

F → P,

P → (E) | a | b

Equivalent non-left-recursive grammar is:

E → T E’

E’ → +T E’ | epsilon

T → F T’

T’ → F T’ | epsilon

F → P F’

F’ → ∗F’ | epsilon

P → (E) | a | b

Eliminating Indirect Left Recursion

```
A -> Cd
B -> Ce
C -> A | B | f
```

- ❑ Arrange all the non-terminals into some arbitrary order

A < B < C

For each terminal, replace the production rule, eliminate any immediate left recursion among.

- Take the production of A alone.

  $A \rightarrow Cd$

  There is no left-recursion. Add this to the result.

- Take the production of B, now consider the production of A(as it is in the result)

  $B \rightarrow C\, e$

Since the production for C is not in the result, we won’t consider its production.

There is no left recursion, add the production of B to result.

Eliminating Indirect Left Recursion

- Consider the production of C,

  C→ A | B | f

  A → Cd
  B→ C e

- By looking into the productions in the result, the derivation of A and B results in left recursion.

  C -> C d | C e | f

  results in left-recursive grammar.

- Eliminate the immediate left-recursion from the productions of C. C -> C d | C e | f

  C -> f C’

  C’ -> d C’ | e C’ | epsilon

  Add the productions of C to the result

- Grammar after eliminating left recursion

  A → Cd
  B→ C e
  C -> f C’
  C’ -> d C’ | e C’ | epsilon