UPES

UNIVERSITY WITH A PURPOSE

Content of this lecture

- What is left recursion and left factoring
- How to remove these

Left Recursion

- A grammar is ***left recursive*** if it has a non-terminal A such that there is a derivation

  $A \overset{+}{\Rightarrow} A\alpha$ for some string α

- Top-down parsing techniques **cannot** handle left-recursive grammars
  - Conversion of left-recursive grammar into an equivalent non-recursive grammar is essential
- Possible ways of left-recursion
  - may appear in a single step of the derivation (*immediate left-recursion*) or
  - may appear in more than one step of the derivation

UPES
UNIVERSITY WITH A PURPOSE

Immediate Left-Recursion

$A \rightarrow A\,\alpha \mid \beta$ where β does not start with A

$\Downarrow$ eliminate immediate left recursion

$A \rightarrow \beta\, A'$

$A' \rightarrow \alpha\, A' \mid \varepsilon$ an equivalent grammar

In general,

$A \rightarrow A\,\alpha_1 \mid \ldots \mid A\,\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n$ where $\beta_1 \ldots \beta_n$ do not start with A

$\Downarrow$ eliminate immediate left recursion

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

$A' \rightarrow \alpha_1\, A' \mid \ldots \mid \alpha_m\, A' \mid \varepsilon$ an equivalent grammar

UPES
UNIVERSITY WITH A PURPOSE

Immediate Left Recursion example

- $E \rightarrow E+T \mid T$
- $T \rightarrow T*F \mid F$
- $F \rightarrow (E) \mid id$

Immediate Left Recursion example

- E→E+T|T
- T→T*F|F
- F→(E)|id

- E→TE`
- E`→+TE`|λ
- T→FT`
- T`→*FT`| λ
- F→(E)|id

UPES
UNIVERSITY WITH A PURPOSE

Immediate Left Recursion example

- S→a|^|(T)
- T→T,S|S

Immediate Left Recursion example

- S→a|^|(T)
- T→T,S|S

- S→a|^|(T)
- T→ST`
- T`→,ST`| λ

UPES
UNIVERSITY WITH A PURPOSE

Left-Recursion -- Problem

- A grammar cannot be *immediately left-recursive*, but it still can be *left-recursive*
- Just elimination of the immediate left-recursion does not guarantee a grammar which is not left-recursive

$S \rightarrow Aa \mid b$

$A \rightarrow Sc \mid d$ This grammar is not immediately left-recursive, but it is still left-recursive

$\underline{S} \Rightarrow Aa \Rightarrow \underline{S}ca$ or

$\underline{A} \Rightarrow Sc \Rightarrow \underline{A}ac$ causes to a left-recursion

- Solution: *eliminate all left-recursions from the grammar*

UPES
UNIVERSITY WITH A PURPOSE

Eliminate Left-Recursion -- Algorithm

- Arrange non-terminals in some order: $A_1 \dots A_n$

- for i **from** 1 **to** n **do** {

  - **for** j **from** 1 **to** i-1 **do** {

    replace each production

    $$A_i \rightarrow A_j \gamma$$

    by

    $$A_i \rightarrow \alpha_1 \gamma \mid \dots \mid \alpha_k \gamma$$

    where $A_j \rightarrow \alpha_1 \mid \dots \mid \alpha_k$

  }

  - eliminate immediate left-recursions among A_i productions

}

Eliminate Left-Recursion -- Example

$S \rightarrow Aa \mid b$
$A \rightarrow Ac \mid Sd \mid f$

- Order of non-terminals: S, A

for S:

- we do not enter the inner loop.
- there is no immediate left recursion in S.

for A:

- Replace $A \rightarrow Sd$ with $A \rightarrow Aad \mid bd$
  So, we will have $A \rightarrow Ac \mid Aad \mid bd \mid f$
- Eliminate the immediate left-recursion in A

$$A \rightarrow bdA' \mid fA'$$
$$A' \rightarrow cA' \mid adA' \mid \varepsilon$$

So, the resulting equivalent grammar which is not left-recursive is:

$S \rightarrow Aa \mid b$
$A \rightarrow bdA' \mid fA'$
$A' \rightarrow cA' \mid adA' \mid \varepsilon$

Indirect Left Recursion example

- $S \rightarrow Ab$
- $A \rightarrow Sc|d$

Indirect Left Recursion example

- S→Ab
- A→Sc|d

- Step 1: Order the non terminals S, A as A1 and A2 respectively.

  A1→A2b (1)

  A2→A1c|d (2)
- Step 2: Put value of 1 in 2

  A1→A2b (3)

  A2→A2bc|d (4)
- Step 3: Again substitute A1=S and A2=A in 3 and 4

  S→Ab

  A→Abc|d
- Step 4: Remove immediate left recursion from 6

  S→Ab

  A→dA`

  A`→bcA`|λ

Left-Factoring

- Top-down *parser without backtracking* (predictive parser) insists that the grammar must be *left-factored*

grammar ➔ a new equivalent grammar suitable for predictive parsing

stmt → `if` expr `then` stmt `else` stmt |
`if` expr `then` stmt

After seeing `if`, we cannot decide which production rule to choose to re-write *stmt* in the derivation

UPES
UNIVERSITY WITH A PURPOSE

Left-Factoring (cont.)

- In general,

  $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$ where α is non-empty and the first symbols of β_1 and β_2 (if they have one) are different

- Choice involved when processing α

  A to $\alpha\beta_1$ or

  A to $\alpha\beta_2$

- Re-write the grammar as follows:

  $A \rightarrow \alpha A'$

  $A' \rightarrow \beta_1 \mid \beta_2$ so, we can immediately expand A to $\alpha A'$

UPES
UNIVERSITY WITH A PURPOSE

Left-Factoring -- Algorithm

- For each non-terminal A with two or more alternatives (production rules) with a common non-empty prefix, let say

$$A \rightarrow \alpha\beta_1 \mid \ldots \mid \alpha\beta_n \mid \gamma_1 \mid \ldots \mid \gamma_m$$

convert it into

$$A \rightarrow \alpha A' \mid \gamma_1 \mid \ldots \mid \gamma_m$$
$$A' \rightarrow \beta_1 \mid \ldots \mid \beta_n$$

UPES
UNIVERSITY WITH A PURPOSE

Left-Factoring – Example1

$$A \rightarrow \underline{a}bB \mid \underline{a}B \mid cdg \mid cdeB \mid cdfB$$

$$\Downarrow$$

$$A \rightarrow aA' \mid \underline{cd}g \mid \underline{cd}eB \mid \underline{cd}fB$$

$$A' \rightarrow bB \mid B$$

$$\Downarrow$$

$$A \rightarrow aA' \mid cdA''$$

$$A' \rightarrow bB \mid B$$

$$A'' \rightarrow g \mid eB \mid fB$$

Left-Factoring – Example2

$$A \rightarrow ad \mid a \mid ab \mid abc \mid b$$

$$\Downarrow$$

$$A \rightarrow aA' \mid b$$

$$A' \rightarrow d \mid \varepsilon \mid b \mid bc$$

$$\Downarrow$$

$$A \rightarrow aA' \mid b$$

$$A' \rightarrow d \mid \varepsilon \mid bA''$$

$$A'' \rightarrow \varepsilon \mid c$$

UPES
UNIVERSITY WITH A PURPOSE

Left Factoring example

- $S \rightarrow iCtS \mid iCtSeS \mid b$
- $C \rightarrow d$

UPES
UNIVERSITY WITH A PURPOSE

Left Factoring example

- S→iCtS|iCtSeS|b
- C→d

- S→iCtSS`|b
- S`→λ|eS
- C→d

THANK YOU

UPES
UNIVERSITY WITH A PURPOSE

Dr Anurag Jain
Assistant Professor (SG), Virtualization Department, School of Computer Science,
University of Petroleum & Energy Studies, Dehradun - 248 007 (Uttarakhand)
Email: anurag.jain@ddn.upes.ac.in , dr.anuragjain14@gmail.com
Mobile: (+91) -9729371188