<u>Compiler Construction (CS-636)</u>

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

(Week 7) Lecture 13 & 14

Objectives: Learning objectives of these lectures are

- ***Students will able to understand:***
  - ***What is Left Recursion?***
  - ***Which is Direct Left Recursion?***
  - ***Which is Indirect Left Recursion?***
  - ***How to eliminate Direct Left Recursion?***
  - ***How to eliminate Indirect Left Recursion?***
- ***Students will able to understand:***
  - ***What is Left Factoring?***
  - ***What is Left Factored Grammar?***

Text Book & Resources:

1. Compilers Principles Techniques and Tools (2nd Edition) by Alfread V. Aho, Ravi Sethi.
2. Introduction to Computer Theroy By Daniel I.A. Cohen.

Videos Links:

https://youtu.be/PFAEPT8rjdY (Part 1)

https://www.youtube.com/watch?v=xu-keaYj088 (Part 2)

https://www.youtube.com/watch?v=OfH5UvUghlY (Part 3)

https://youtu.be/zD4xJ8HkXmk (Part 4)

https://www.youtube.com/watch?v=COdq7B96KCg (Part 5)

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

Left Recursion:

A grammar is left recursive if it has a non-terminal A such that there is a derivation A →Ab for some string a. Top-down parsing methods cannot handle left-recursive grammars, so a transformation is needed to eliminate left recursion.

In this lecture, we will discussed two types of left recursions

- ***Direct left recursion***
- ***Indirect left recursion***

Elimination of left recursion:

We first discussed direct or immediate left recursion, where there is a production of the form A→Ab. Here, we study the general case. We showed how the left-recursive pair of productions A→Aα | β could be replaced by the non-left-recursive productions:

A→Aα | β

After Elimination

A→ βM

M→αM | $\in$

without changing the strings derivable from A. This rule by itself suffices for

many grammars.

Description of Above Example

- First of identify the left recursion
- For every term which contain left recursion mention as " α " beside it. E.g A→Aα | β Here A produces A→Aα and A→ β . So first production contain the left recursion. These productions can be written as A→ βM . For every β we can write these type of production with another non-terminal M(any Symbol can be used instead of M).
- For every α, new M symbol will be used for the production. E.g. M→αM | $\in$

Example # 1

- ⇨ A grammar is left recursive if it has a non-terminal A such that there is a derivation A→ AX for some string X.
- ⇨ If production is of the form A→AX, it is called immediate left recursion.

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

$A \rightarrow AX \mid B$

Replace it with

$A \rightarrow BA`$

$A` \rightarrow XA` \mid \in$

Generic Example # 2

Remove immediate left recursion by following technique.

$A \rightarrow AX_1 \mid AX_2 \mid \ldots\ldots\ldots\mid AX_m \mid B_1 \mid \ldots\ldots\ldots\ldots \mid B_n.$

Then replace by

$A \rightarrow B_1A` \mid B_2A_1 \mid \ldots\ldots\ldots\mid B_nA`$

$A` \rightarrow X_1A` \mid X_2A_1 \mid \ldots\ldots\ldots\mid X_mA` \mid \in$

Example # 3

Given CFG	**After Elimination of Left Recursion**
$E \rightarrow E + T$	$E \rightarrow TE`$
$E \rightarrow E - T$	$E \rightarrow +TE` \mid \in$
$E \rightarrow T$	$E` \rightarrow -TE` \mid \in$
$T \rightarrow T * F$	$T \rightarrow FT`$
$T \rightarrow T/ F$	$T` \rightarrow *FT` \mid \in$
$T \rightarrow F$	$T` \rightarrow /FT` \mid \in$
$F \rightarrow (E)$	$F \rightarrow (E)$
$F \rightarrow id$	$F \rightarrow id$

Example # 4

Given CFG	**Answer**
$A \rightarrow ABd / Aa / a$ $B \rightarrow Be / b$	$A \rightarrow aA'$ $A' \rightarrow BdA' / aA' / \in$ $B \rightarrow bB'$ $B' \rightarrow eB' / \in$

Compiler Construction (CS-636)

Dr. Naseer Ahmed Sajid **email id: naseer@biit.edu.pk Whatsapp# 0346-5100010**

Example # 5

Given CFG	**Answer**
E → E + E / E x E / a	E → aA A → +EA / xEA / ∈

Example # 6

Given CFG	**Answer**
E → E + T / T T → T x F / F F → id	E → TE’ E’ → +TE’ / ∈ T → FT’ T’ → xFT’ / ∈ F → id

Example # 7

Given CFG	**Answer**
S → (L) / a L → L , S / S	S → (L) / a L → SL’ L’ → ,SL’ / ∈

Example # 8

Given CFG	**Answer**
S → S0S1S / 01	S → 01A A → 0S1SA / ∈

Example # 9

Given CFG	**Answer**
S → A A → Ad / Ae / aB / ac B → bBc / f	S → A A → aBA’ / acA’ A’ → dA’ / eA’ / ∈ B → bBc / f

Example # 10

Given CFG	**Answer**
A → AAα / β	A → βA’

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

	A’ → AαA’ / ∈

Indirect Left Recursion

Removing indirect left recursion:

S -> Aa | b | ∈

A -> Ac | Sd | ∈ S=> Aa=>Sda

Answer

S -> Aa | b | ∈

A -> Ac | Aad | bd |d | ∈ (Replaced S values three times in this production and then apply direct method)

S -> Aa | b |∈

A ->dA` | bdA` | A` (d,bd, ∈ are takes as β)

A` ->cA` | adA` | ∈ (c and ad are takes as α)

Example # 1

Given CFG A → Ba / Aa / c B → Bb / Ab / d	**Answer** **<u>Step-01:</u>** First let us eliminate left recursion from A → Ba / Aa / c Eliminating left recursion from here, we get- A → BaA’ / cA’ A’ → aA’ / ∈ Now, given grammar becomes- A → BaA’ / cA’ A’ → aA’ / ∈ B → Bb / Ab / d **<u>Step-02:</u>** Substituting the productions of A in B → Ab,

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

	we get the following grammar- A → BaA' / cA' A' → aA' / ∈ B → Bb / BaA'b / cA'b / d **Step-03:** Now, eliminating left recursion from the productions of B, we get the following grammar- A → BaA' / cA' A' → aA' / ∈ B → cA'bB' / dB' B' → bB' / aA'bB' / ∈

Example # 2

Given CFG	**Answer**
X → XSb / Sa / b S → Sb / Xa / a	**Step-01:** First let us eliminate left recursion from X → XSb / Sa / b Eliminating left recursion from here, we get- X → SaX' / bX' X' → SbX' / ∈ Now, given grammar becomes- X → SaX' / bX' X' → SbX' / ∈ S → Sb / Xa / a

Dr. Naseer Ahmed Sajid **email id: naseer@biit.edu.pk Whatsapp# 0346-5100010**

	Step-02: Substituting the productions of X in S → Xa, we get the following grammar- X → SaX' / bX' X' → SbX' / ∈ S → Sb / SaX'a / bX'a / a **Step-03:** Now, eliminating left recursion from the productions of S, we get the following grammar- X → SaX' / bX' X' → SbX' / ∈ S → bX'aS' / aS' S' → bS' / aX'aS' / ∈

Example # 3

Given CFG	**Answer**
S → Aa / b A → Ac / Sd / ∈	**Step-01:** First let us eliminate left recursion from S → Aa / b This is already free from left recursion. **Step-02:** Substituting the productions of S in A → Sd, we get the following grammar- S → Aa / b

	$A \rightarrow Ac$ / Aad / bd / $\in$ **Step-03:** Now, eliminating left recursion from the productions of A, we get the following grammar- $S \rightarrow Aa$ / b $A \rightarrow bdA'$ / A' $A' \rightarrow cA'$ / adA' / $\in$

Left factoring:

Left factoring is a process by which the grammar with common prefixes is transformed to make it useful for Top down parsers.

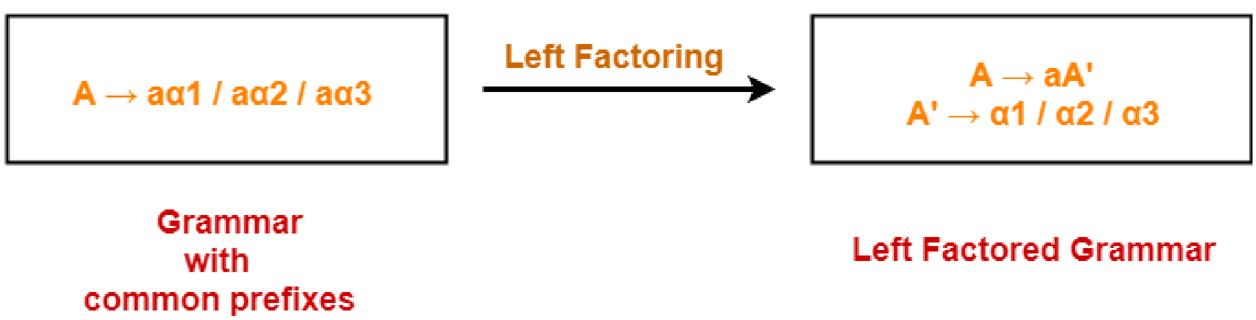
How?

In left factoring,

- We make one production for each common prefixes.
- The common prefix may be a terminal or a non-terminal or a combination of both.
- Rest of the derivation is added by new productions.

The grammar obtained after the process of left factoring is called as **Left Factored Grammar**.

Example # 1



Given CFG

$A \rightarrow XB_1 | XB_2 | \ldots\ldots\ldots\ldots\ldots XB_n | R$ when choice b/w two

Then alternative is not clear

Dr. Naseer Ahmed Sajid **email id: naseer@biit.edu.pk Whatsapp# 0346-5100010**

A -> XA` | R then left- factor it

A` -> B_1 | B_2 | B_3 |…………..|B_n

Description

In the given CFG, we can see that in the productions X is repeated in every production except R. Then we can take X as common and for all B's used A-dash and write the production as
A -> XA` | R

Then write the production for A-dash which shows all the values.

A` -> B_1 | B_2 | B_3 |…………..|B_n

Example #2

Given CFG	**Answer**
S → iEtS / iEtSeS / a E → b	S → iEtSS' / a S' → eS / ∈ E → b

Example #3

Given CFG	**Answer**
A → aAB / aBc / aAc	**Step-01:** A → aA' A' → AB / Bc / Ac Again, this is a grammar with common prefixes. **Step-02:** A → aA' A' → AD / Bc D → B / c

Example #4

Given CFG	**Answer**
S → bSSaaS / bSSaSb / bSb / a	**Step-01:**

Compiler Construction (CS-636)

Dr. Naseer Ahmed Sajid **email id: naseer@biit.edu.pk Whatsapp# 0346-5100010**

<table>
<tr><td></td><td>S → bSS' / a

S' → SaaS / SaSb / b

Again, this is a grammar with common prefixes.

<u>Step-02:</u>

S → bSS' / a

S' → SaA / b

A → aS / Sb</td></tr>
</table>

Example #5

<table>
<tr><td>Given CFG
S → aSSbS / aSaSb / abb / b</td><td>Answer
<u>Step-01:</u>

S → aS' / b

S' → SSbS / SaSb / bb

Again, this is a grammar with common prefixes.

<u>Step-02:</u>

S → aS' / b

S' → SA / bb

A → SbS / aSb</td></tr>
</table>

Example #6

<table>
<tr><td>Given CFG
S → a / ab / abc / abcd</td><td>Answer
<u>Step-01:</u>

S → aS'

S' → b / bc / bcd / ∈</td></tr>
</table>

Dr. Naseer Ahmed Sajid email id: naseer@biit.edu.pk Whatsapp# 0346-5100010

	Again, this is a grammar with common prefixes. **Step-02:** S → aS' S' → bA / ∈ A → c / cd / ∈ Again, this is a grammar with common prefixes. **Step-03:** S → aS' S' → bA / ∈ A → cB / ∈ B → d / ∈

Example #7

Given CFG	Answer
S → aAd / aB A → a / ab B → ccd / ddc	S → aS' S' → Ad / B A → aA' A' → b / ∈ B → ccd / ddc