

Theory of Automata & Formal Languages

MALAY BHATT

Outline

- Top-Down Parsing
 - Left Factoring of Grammar
 - Elimination of Left Recursion
- Pumping Lemma for Context-Free Languages
- Context Sensitive Grammar (Recursive Language)
- Unrestricted Grammar (Recursively Enumerable Language)

Top-Down Parsing

- We can eliminate non-Determinism **upto some extent** by re-writing the grammar rules
- There are two main techniques :
 1. Left-Factoring of Grammar
 2. Elimination of Left Recursion

Left Recursion

- Productions of the form
$$A \rightarrow A \alpha \mid \beta$$
are left recursive

Ex. $E \rightarrow E + T \mid T$

Here, A matches with E

α matches with $+ T$

β matches with T

Left Recursion

- When one of the productions in a grammar is left recursive then a Top Down parser (Left-Most Derivation) loops forever

- $E \rightarrow E + T$

- $\rightarrow E + T + T$

- $\rightarrow E + T + T + T + T \dots$

and if we replace forcefully (To stop infinite loop)
'E' with 'T' then

$\rightarrow T + T + T + T + T \dots$

Immediate Left-Recursion Elimination (Example – 1)

Rewrite every left-recursive production

$$A \rightarrow A \alpha \mid \beta \mid \gamma \mid A \delta$$

into a right-recursive production:

$$\begin{aligned} A &\rightarrow \beta A_R \mid \gamma A_R \\ A_R &\rightarrow \alpha A_R \mid \delta A_R \mid \varepsilon \end{aligned}$$

Ex. $E \rightarrow E + T \mid E - T \mid T$

Here,

A matches with E

α matches with + T

δ matches with - T

β matches with T

$$\begin{aligned} E &\rightarrow T E_R \\ E_R &\rightarrow + T E_R \mid - T E_R \mid \varepsilon \end{aligned}$$

Immediate Left-Recursion Elimination (Example – 2)

Rewrite every left-recursive production

$$A \rightarrow A \alpha \mid \beta \mid \gamma \mid A \delta$$

into a right-recursive production:

$$\begin{aligned} A &\rightarrow \beta A_R \mid \gamma A_R \\ A_R &\rightarrow \alpha A_R \mid \delta A_R \mid \varepsilon \end{aligned}$$

$$S \rightarrow S0S1S \mid 01$$

$$S \rightarrow 01 S_R$$

$$S_R \rightarrow 0S1S S_R \mid \varepsilon$$

Immediate Left-Recursion Elimination (Example – 3)

Rewrite every left-recursive production

$$A \rightarrow A \alpha \mid \beta \mid \gamma \mid A \delta$$

into a right-recursive production:

$$\begin{aligned} A &\rightarrow \beta A_R \mid \gamma A_R \\ A_R &\rightarrow \alpha A_R \mid \delta A_R \mid \varepsilon \end{aligned}$$

$$\begin{array}{l} S \rightarrow (L) \mid a \\ L \rightarrow L, S \mid S \end{array} \quad \longrightarrow \quad \begin{array}{l} S \rightarrow (L) \mid a \\ L \rightarrow SL_R \\ L_R \rightarrow ,SL_R \mid \varepsilon \end{array}$$

Immediate Left-Recursion Elimination (Example – 4)

Rewrite every left-recursive production

$$A \rightarrow A\alpha \mid \beta \mid \gamma \mid A\delta$$

into a right-recursive production:

$$A \rightarrow \beta A_R \mid \gamma A_R$$
$$A_R \rightarrow \alpha A_R \mid \delta A_R \mid \varepsilon$$

$$S \rightarrow A$$

$$A \rightarrow Ad \mid Ae \mid aB \mid ac$$

$$B \rightarrow bBc \mid f$$



$$S \rightarrow A$$

$$A \rightarrow aBA_R \mid acA_R$$

$$A_R \rightarrow dA_R \mid eA_R \mid \varepsilon$$

$$B \rightarrow bBc \mid f$$

Immediate Left-Recursion Elimination (Example – 5)

$$A \rightarrow ABd \mid Aa \mid a$$
$$B \rightarrow Be \mid b$$

Here,

A matches with A
 α matches with Bd
 δ matches with a
 β matches with a

$$A \rightarrow a A_R$$
$$A_R \rightarrow Bd A_R \mid a A_R \mid \varepsilon$$

Here,

A matches with B
 α matches with e
 β matches with b

$$B \rightarrow b B_R$$
$$B_R \rightarrow e B_R \mid \varepsilon$$

Indirect Left-Recursion Elimination

$$\begin{aligned} A &\rightarrow Ba \mid \mathbf{Aa} \mid c \\ B &\rightarrow Bb \mid Ab \mid d \end{aligned}$$

Step-01:

First let us eliminate left recursion from $A \rightarrow Ba \mid Aa \mid c$

Now, given grammar becomes-

$$A \rightarrow Ba\mathbf{A_R} \mid c\mathbf{A_R}$$

$$\mathbf{A_R} \rightarrow a\mathbf{A_R} \mid \epsilon$$

$$B \rightarrow Bb \mid Ab \mid d$$

Indirect Left-Recursion Elimination

$$A \rightarrow BaA_R \mid cA_R$$

$$A_R \rightarrow aA_R \mid \varepsilon$$

$$B \rightarrow Bb \mid Ab \mid d$$

Step-02:

Substituting the productions of A in $B \rightarrow Ab$, we get the following grammar-

$$A \rightarrow BaA_R \mid cA_R$$

$$A_R \rightarrow aA_R \mid \varepsilon$$

$$B \rightarrow Bb \mid BaA_R b \mid cA_R b \mid d$$

Indirect Left-Recursion Elimination

$$A \rightarrow BaA_R \mid cA_R$$

$$A_R \rightarrow aA_R \mid \varepsilon$$

$$B \rightarrow Bb \mid BaA_R b \mid cA_R b \mid d$$

Step-03:

Now, eliminating left recursion from the productions of B, we get the following grammar-

$$A \rightarrow BaA_R \mid cA_R$$

$$A_R \rightarrow aA_R \mid \varepsilon$$

$$B \rightarrow cA_R bB_R \mid dB_R$$

$$B_R \rightarrow bB_R \mid aA_R bB_R \mid \varepsilon$$

Example Left Recursion Elimination

$$A \rightarrow BC \mid a$$

$$B \rightarrow CA \mid Ab$$

$$C \rightarrow AB \mid CC \mid a$$

$i = 1$: nothing to do

$i = 2, j = 1$: $B \rightarrow CA \mid \underline{A}b$

$$\Rightarrow B \rightarrow CA \mid \underline{BC}b \mid \underline{a}b$$

$$\Rightarrow_{(imm)} B \rightarrow CAB_R \mid abB_R$$

$$B_R \rightarrow CbB_R \mid \varepsilon$$

$i = 3, j = 1$: $C \rightarrow \underline{A}B \mid CC \mid a$

$$\Rightarrow C \rightarrow \underline{BC}B \mid \underline{a}B \mid CC \mid a$$

$i = 3, j = 2$: $C \rightarrow \underline{B}CB \mid aB \mid CC \mid a$

$$\Rightarrow C \rightarrow \underline{CAB_R}CB \mid \underline{abB_R}CB \mid aB \mid CC \mid a$$

$$\Rightarrow_{(imm)} C \rightarrow abB_RCB C_R \mid aB C_R \mid a C_R$$

$$C_R \rightarrow AB_RCB C_R \mid CC_R \mid \varepsilon$$

Practice Problem-1

$$X \rightarrow XSb \mid Sa \mid b$$
$$S \rightarrow Sb \mid Xa \mid a$$

Practice Problem - 2

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

A General Systematic Left Recursion Elimination Method

Input: Grammar G with no cycles or ε -productions

Arrange the non-terminals in some order A_1, A_2, \dots, A_n

for $i = 1, \dots, n$ **do**

for $j = 1, \dots, i-1$ **do**

 replace each

$$A_i \rightarrow A_j \gamma$$

 with

$$A_i \rightarrow \delta_1 \gamma \mid \delta_2 \gamma \mid \dots \mid \delta_k \gamma$$

 where

$$A_j \rightarrow \delta_1 \mid \delta_2 \mid \dots \mid \delta_k$$

enddo

 eliminate the *immediate left recursion* in A_i

enddo

Pumping Lemma for Context-Free Languages

Suppose L is a context-free language. Then there is an integer n so that for every $u \in L$ with $|u| \geq n$, u can be written as $u = vwxyz$ for some strings v , w , x , y , and z satisfying

1. $|wy| > 0$
2. $|wxy| \leq n$
3. for every $m \geq 0$, $vw^mxy^mz \in L$

Pumping Lemma for Context-Free Languages

$\{WW \mid W \text{ belongs to } \{a,b\}^*\}$

$$u = a^n b^n a^n b^n \quad (w = a^n b^n)$$

Case :1 $wxy = a^n$ or $wxy = b^n$ or $wxy = a^n$ or $wxy = b^n$

For $m=2$, $(v = \epsilon, z = b^n a^n b^n)$

$$vw^2xy^2z = a^{2n} b^n a^n b^n$$

does not belong to L

Case :2 $wxy = a^{n-2} b^2$ or $wxy = b^{n-2} a^2$ or $wxy = a^{n-2} b^2$

For $m=2$, $(v = a^2, z = b^{n-2} a^n b^n)$

$$vw^2xy^2z = a^2 a^{2n-4} b^4 b^{n-2} a^n b^n = a^{2n+2} b^{n+2} a^n b^n$$

does not belong to L

Context Sensitive Language (Recursive Language)

A context-sensitive grammar (CSG) is an unrestricted grammar in which no production is length-decreasing. In other words, every production is of the form $\alpha \rightarrow \beta$, where $|\beta| \geq |\alpha|$.

A language is a context-sensitive language (CSL) if it can be generated by a context-sensitive grammar.

Context Sensitive Language (Method-1)

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Let's derive string: **aabbcc**

S
aSBc
aabcBc
aabBcc
aabbcc

$S \rightarrow aSBc \mid abc$
 $cB \rightarrow Bc$
 $bB \rightarrow bb$

Let's derive string: **aaabbbccc**

S
aSBc
a aSBcBc
aaabcBcBc
aaabBccBc
aaabbccBc
aaabbcbcc
aaabbBccc
aaabbbccc

Context Sensitive Language (Method-2)

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Let's derive string: **aabbcc**

S
aB S c
A Ba bcc
A aB bcc
Aa bbcc

$S \rightarrow aBSc \mid abc$

$Ba \rightarrow aB$

$Bb \rightarrow bb$

Let's derive string: **aaabbbccc**

S
aBSc
aBaBScc
aaBBabccc
aaBaBbccc
aaaBBbccc
aaaBbbccc
aaabbbccc

Unrestricted Grammar

(Recursively Enumerable Language)

An *unrestricted* grammar is a 4-tuple $G = (V, \Sigma, S, P)$, where V and Σ are disjoint sets of variables and terminals, respectively. S is an element of V called the start symbol, and P is a set of productions of the form


$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (V \cup \Sigma)^*$ and α contains at least one variable.

A Turing machine T with input alphabet Σ accepts a language $L \subseteq \Sigma^*$ if $L(T) = L$. T decides L if T computes the characteristic function $\chi_L : \Sigma^* \rightarrow \{0, 1\}$. A language L is *recursively enumerable* if there is a TM that accepts L , and L is *recursive* if there is a TM that decides L .

Remember that there are *three* possible outcomes of executing a Turing machine over a given input. The Turing machine may :

- Halt and accept the input;
- Halt and reject the input; or
- Never halt.

 **Recursive Language**
Recursive Enumerable Language

A language is **recursive** if there exists a Turing machine that **accepts** every string of the language and **rejects** every string (over the same alphabet) that is not in the language.

A language is **recursively enumerable** if there exists a Turing machine that accepts every string of the language, and does not accept strings that are not in the language. (Strings that are not in the language may be rejected or may cause the Turing machine to go into an infinite loop.)

Recursively Enumerable Language (Unrestricted Grammar)

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Let's derive string: **aabbcc**

S
SABC
FABCABC
FABACBC
FABABCC
FAABBCC
aABBBCC
aaBBCC
aabBCC
aabbCC
aabbccC
aabbcc

$S \rightarrow SABC \mid FABC$

$BA \rightarrow AB$

$CA \rightarrow AC$

$CB \rightarrow BC$

$FA \rightarrow a$ ($|\alpha| > |\beta|$)

$aA \rightarrow aa$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Chomsky Hierarchy

