Theory of Computation

Dr Samayveer Singh

Grammar

Grammar

- For the simplicity, let’s consider two types of description of sentences in English
  - S → <noun> <verb> <adverb>
  - S → <noun> <verb>
    - <noun> → Sam
    - <noun> → Ram
    - <verb> → walked
    - <verb> → ate
    - <adverb> → slowly
    - <adverb> → quickly

Grammar

- A Grammar consists of 4-tuple (V_N, Σ, P, S) where
  - V_N = {<noun> <verb> <adverb>}
  - Σ = {Ram, Sam, ate, walked, slowly, quickly}
  - P is the collection of rules.
  - S is the special symbol denoting a sentence.

- The sentences are obtained by (i) starting with S. (ii) replacing words using the productions. and (iii) terminating when a string of terminals is obtained.

Definition of a Grammar

- A grammar or phase structure grammar is given by (V_N, Σ, P, S) where
  - V_N is a finite nonempty set whose elements are called variables or non-terminals.
  - Σ is a finite nonempty set whose elements are called terminals.
  - P is a set of productions or substitution rules of the form

$$\alpha \rightarrow \beta$$

    α is a variable and β is a string of variables and terminals
  - S is a special variable called the start symbol or variable.

If G=({S}, {0, 1}, {S→0S1, S→^}, S) Find L(G)

$S \rightarrow \Lambda$

$S \rightarrow 0S1$

$S \rightarrow 0\Lambda 1 \Rightarrow 01$

$S \rightarrow 0S1$

$\rightarrow 00S11$

$\rightarrow 0011$

$S \rightarrow 000111$

$L(G) = \{0^n 1^n \mid n \geq 0\}$

If G = ({ S}, {a}, {S→SS}, S), find the language generated by G.

Sol^n

$S \rightarrow SS$

$S \rightarrow SSS$

$S \rightarrow SSSS$

$L(G) = \{\phi\}$

Let G =({S, C}, {a, b}, P, S), where P consists of S→aCa, C→aCa | b. Find L(G).

Sol[n]

$S \rightarrow aCa$
$\rightarrow aba$

$S \rightarrow aCa$
$\rightarrow aaCaa$
$\rightarrow aabaa$

$S \rightarrow aCa$
$\rightarrow aaCaa$
$\rightarrow aaaCaaa$
$\rightarrow aaabaaa$

$$L(G) = \{ a^n b a^n \mid n \geq 1 \}$$

$C \rightarrow aCa \mid b$

$C \rightarrow aCa$
$C \rightarrow b$

If G is S → aS | bS | a | b, find L(G).

Sol^n.

$S \to a$

$S \to b$

$S \to aa$

$S \to ab$

$S \to ba$

$S \to bb$

$S \to aS$

$\to aaS$

$\to aaa$ — aab

— bba — bbb

$L(G) = \{a, b\}^+$ OR

$L(G) = \{a, b\}^* - \{\Lambda\} = \{a, b\}^+$

Let L be the set of all palindromes over {a, b}. Construct a grammar G generating L.

(i) Λ

(ii) a, b

(iii) x — axa & bxb

$$P: \begin{cases} S \to \Lambda \\ S \to a, \ S \to b \\ S \to aSa, \ S \to bSb \end{cases}$$

$$G = (\{S\}, \{a, b\}, P, \{S\})$$

Construct a grammar generating $L = \{ wcw^T \mid w \in \{a, b\}^* \}$.

$S \to c$

$S \to aSa$

$S \to bSb$

P: $\boxed{S \to aSa \mid bSb \mid c}$

$G = (\{S\}, \{a, b, c\}, \{P\}, \{S\})$

Find a grammar generating

$L = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$

$L = L_1 \cup L_2$

$L_1 = \{a^n b^n \mid n \geq 1\}$ ✓

$L_2 = \{a^n b^n c^i \mid n \geq 1, i \geq 1\}$

P

$$\boxed{\begin{array}{ll} S \to A & A \to aAb \mid ab \\ & S \to Sc \end{array}}$$

$G = (\{S, A\},$
$\{a, b, c\}, P,$
$S)$

$S \to Sc$
$\to Scc$
$\to Acc$
$\to aAbcc$
$\to aaAbbcc \to aaabbbcc$

aaabbbcc

$S \to A$
$\to aAb$
$\to aaAbb$
$\to aaabbb$

Find a grammar generating $L = \{a^j b^n c^n | n \geq 1, j \geq 0\}$

P:

$S \to aS$

$S \to A$

$A \to bAc \mid bc$

$G = (\{S, A\}, \{a, b, c\}, P, S)$

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

$S \to aS\alpha \mid a\alpha$

P: $S \to aSBC \mid aBC$

$aB \to ab$

$CB \to BC$

$bB \to bb$

$bC \to bc$

$cC \to cc$

$G = (\{S, B, C\}, \{a, b, c\}, P, \{S\})$

$a^3\alpha^3$

aaabbbccc

$a^n b^n c^n$

$a^n \alpha^n$

$\alpha = BC$

aaaBCBCBC

aaabCBCBC

aaabBCBCC

aaabbBCCC

aaabbbCCC

aaabbbcCC

aaabbbccC

aaabbbccc

Chomsky Classification of Languages

- According to Chomosky, there are four types of grammars − Type 0, Type 1, Type 2, and Type 3.

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Type - 0 Grammar

- **Type-0 grammars** generate recursively enumerable languages.
- The productions have no restrictions. They are any phase structure grammar including all formal grammars.
- They generate the languages that are recognized by a Turing machine.
- The productions can be in the form of $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ where $\boldsymbol{\alpha}$ is a string of terminals and non-terminals with at least one non-terminal and $\boldsymbol{\alpha}$ cannot be null. $\boldsymbol{\beta}$ is a string of terminals and non-terminals.
- Example
  - $S \rightarrow ACaB$
  - $Bc \rightarrow acB$
  - $CB \rightarrow DB$
  - $aD \rightarrow Db$

Type - 1 Grammar

- **Type-1 grammars** generate context-sensitive languages. The productions must be in the form

$$\boldsymbol{\alpha\ A\ \beta \rightarrow \alpha\ \gamma\ \beta}$$

where $\mathbf{A \in V_N}$ (Non-terminal) and $\boldsymbol{\alpha, \beta, \gamma \in (\sum \cup V_N)^*}$ (Strings of terminals and non-terminals)

- If $\mathbf{A \rightarrow \gamma}$**, then** $\mathbf{|A| <= |\gamma|}$
- The strings $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ may be empty, but $\boldsymbol{\gamma}$ must be non-empty.
- The rule $\mathbf{S \rightarrow \varepsilon}$ is allowed if S does not appear on the right side of any rule. The languages generated by these grammars are recognized by a linear bounded automaton.
- Example

$AB \longrightarrow AbBc$

$A \longrightarrow bcA$

$B \longrightarrow b$

Type - 2 Grammar

- **Type-2 grammars** generate context-free languages.
- The productions must be in the form $\mathbf{A} \rightarrow \boldsymbol{\gamma}$

  where $\mathbf{A} \in \mathbf{V_N}$ (Non terminal) and $\boldsymbol{\gamma} \in (\sum \cup \mathbf{V_N})^*$ (String of terminals and non-terminals).
- These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.
- Example

  $S \rightarrow X\ a$

  $X \rightarrow a$

  $X \rightarrow aX$

  $X \rightarrow abc$

  $X \rightarrow \varepsilon$

Type - 3 Grammar

- **Type-3 grammars** generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.
- The productions must be in the form **$X \rightarrow a$ or $X \rightarrow aY$**

  where **$X, Y \in V_N$** (Non terminal) and **$a \in \sum$** (Terminal)
- The rule **$S \rightarrow \varepsilon$** is allowed if **S** does not appear on the right side of any rule.
- Example

$$X \rightarrow \varepsilon$$

$$X \rightarrow a \mid aY$$

$$Y \rightarrow b$$

Examples

- Find the highest type number which can be applied to the following productions:

  (a) S → *Aa*, A → cB | A, B→abc. → Type 2

  (b) S→*ASB* | *d*, *A* → *aA*

  (c) S → *aS* | *ab*

  Type 3

Chomsky Hierarchy

Grammar Types(Phrase-structure Grammars):