BCSE304L
Theory of Computation

Lecture 4

Dr. Saritha Murali
(SCOPE, VIT Vellore)

16-12-2022

Grammars

Grammar

- A grammar for the English language tells us whether a particular sentence is well-formed or not.
- If the sentence is correct grammatically then that sentence will be the part of grammar, otherwise not.
- “I am going to school.” - valid
- “I going am to school.” - invalid

- Grammar is a set of rules used to define a language.
- It is the structure of the strings in the language.

Grammar

Definition: A grammar is a 4-tuple G = (N, T, P, S), where

- N is a finite set of symbols called ***non-terminals*** (uppercase letters)
- T is a finite set of symbols called ***terminals*** (lowercase letters)
- S $\in$N is the ***start symbol***
- P is the set of ***productions*** of the form $\alpha \rightarrow \beta$

Note: $N \cap T = \phi$

Derivation

- Non terminals - N
- Terminals – T
- Total alphabet - $V = \mathrm{N} \cup T$

$\alpha, \beta \in V^*$ are strings in V^*

$\alpha \Rightarrow \beta$ β is obtained from α in one step

$\alpha \Rightarrow^* \beta$ β is obtained from α in zero or more steps

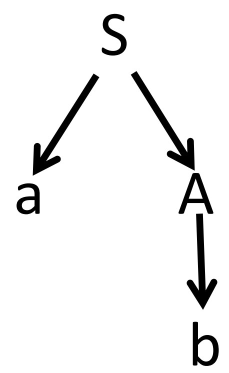
$\Rightarrow^*$ reflexive transitive closure of $\Rightarrow$

Grammar (Examples)

G = ({S, A}, {a, b}, P, S), where

P: $S \rightarrow aA$

$A \rightarrow b$



L(G) = { ab }

Grammar

Let G = (N, T, P, S) be a grammar.

Then L(G) is called the language generated by G.

L(G) = $\{ w \in T^* / S \overset{*}{\Rightarrow} w \}$

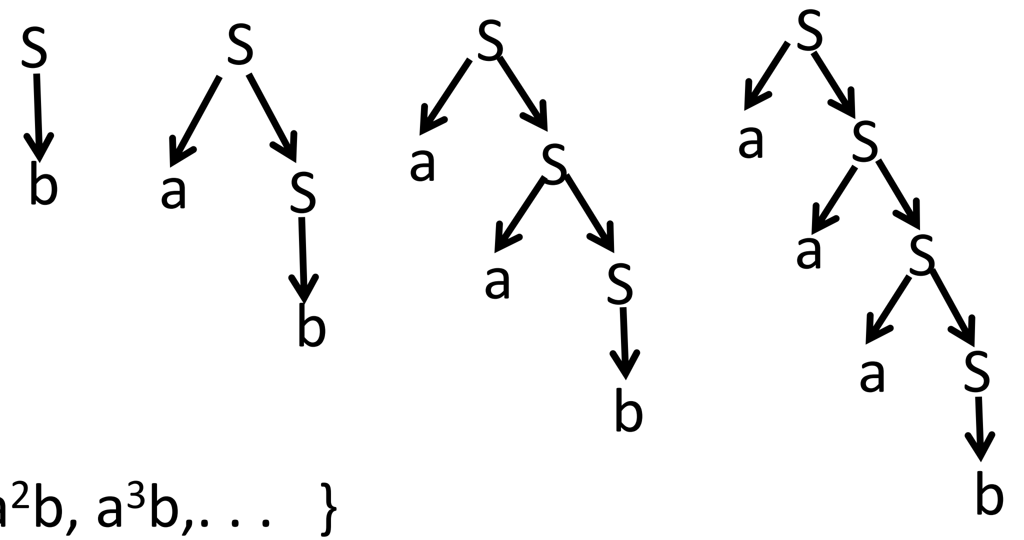
- Grammars are language generating device
- Automata are language accepting device

Grammar

G = ({S}, {a, b}, P, S), where

P: $S \rightarrow aS$

$S \rightarrow b$



$L(G) = \{ b, ab, a^2b, a^3b, \ldots \}$

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

Grammars and Languages

1) G= ({S}, {a}, P, S), where

P: $S \rightarrow aS$ (rule 1)

$S \rightarrow a$ (rule 2)

$S \Rightarrow \mathbf{a}$

$S \Rightarrow aS \Rightarrow \mathbf{aa}$

$S \Rightarrow aS \Rightarrow aaS \Rightarrow \mathbf{aaa}$

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

2) G = ({S}, {a, b}, P, S), where

P: $S \rightarrow aS$ (rule 1)

$S \rightarrow b$ (rule 2)

$S \Rightarrow \mathbf{b}$

$S \Rightarrow aS \Rightarrow \mathbf{ab}$

$S \Rightarrow aS \Rightarrow aaS \Rightarrow \mathbf{aab}$

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

Grammars and Languages

3) G= ({S}, {a, b }, P, S), where

P: $S \rightarrow aSb$ (rule 1)

$S \rightarrow ab$ (rule 2)

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

4) G= ({S}, {a, b, c }, P, S), where

P: $S \rightarrow aSa$ (rule 1)

$S \rightarrow bSb$ (rule 2)

$S \rightarrow c$ (rule 3)

L(G) = { $wcw^R \mid w \in \{a, b\}^*$ }

Grammars and Languages

3) G= ({S}, {a, b }, P, S), where

P: $S \to aSb$ (rule 1)

$S \to ab$ (rule 2)

$S \Rightarrow$ **ab**

$S \Rightarrow aSb \Rightarrow$ **aabb**

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow$ **aaabbb**

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

4) G= ({S}, {a, b, c }, P, S), where

P: $S \to aSa$ (rule 1)

$S \to bSb$ (rule 2)

$S \to c$ (rule 3)

$S \Rightarrow$ **c**

$S \Rightarrow aSa \Rightarrow$ **aca**

$S \Rightarrow bSb \Rightarrow$ **bcb**

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow$ **abcba**

$S \Rightarrow bSb \Rightarrow baSab \Rightarrow$ **bacab**

L(G) = { wcw^R | $w \in \{a, b\}^*$ }

Grammars and Languages

5) G= ({S}, {a, b, c }, P, S), where

P: $S \rightarrow aSBc$ (rule 1) $\quad S \Rightarrow$ **abc**

$S \rightarrow abc$ (rule 2) $\quad S \Rightarrow aSBc \Rightarrow aabcBc \Rightarrow aabBcc \Rightarrow$ **aabbcc**

$cB \rightarrow Bc$ (rule 3)

$bB \rightarrow bb$ (rule 4) $\quad S \Rightarrow aSBc \Rightarrow aaSBcBc \Rightarrow aaabcBcBc$

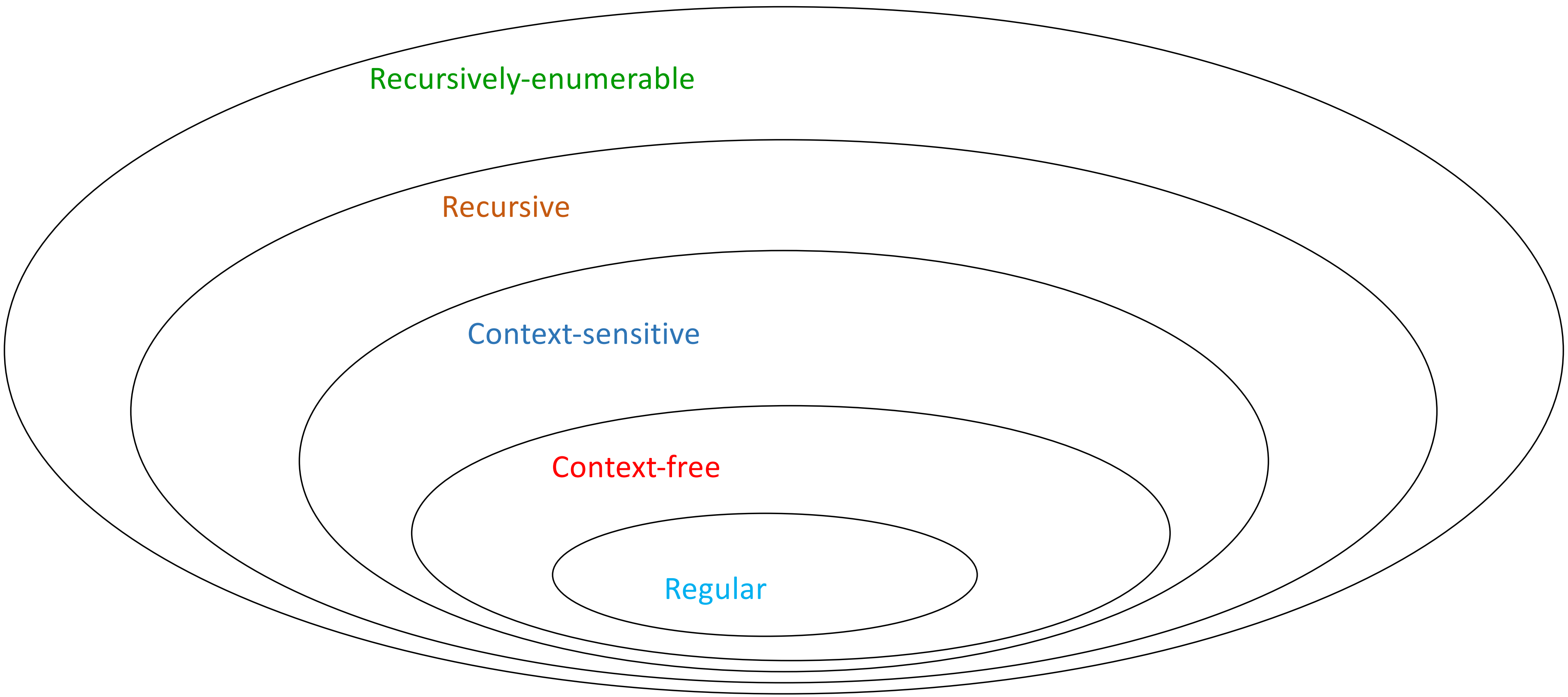
$\Rightarrow aaabBccBc \Rightarrow aaabBcBcc \Rightarrow aaabBBccc$

$\Rightarrow aaabbBccc \Rightarrow$ **aaabbbccc**

L(G) = { $a^n b^n C^n$ | n ≥ 1 }

Chomsky Hierarchy

Non-recursively enumerable



Chomsky Hierarchy

	Language	Grammar	Machine	Example
Type 3	**Regular languages**	**Regular grammars** • **Right-linear grammars** • **Left-linear grammars**	**Finite-state automata**	**a***
Type 2	**Context-free languages**	**Context-free grammars**	**Push-down automata**	$a^n b^n$
Type 1	**Context-sensitive languages**	**Context-sensitive grammars**	**Linear-bound automata**	$a^n b^n c^n$
Type 0	**Recursive languages** **Recursively enumerable languages**	**Unrestricted grammars**	**Turing machines**	**any computable function**

Read as: unrestricted grammars can generate the languages that can be accepted by a Turing machine.