- UNIT -3

- FORMAL LANGUAGES and GRAMMARS

- CSE322
- Formal Languages and Automata Theory

Intro to Languages

- English grammar tells us if a given combination of words is a valid sentence.

- The syntax of a sentence concerns its form while the semantics concerns
- its meaning.
- e.g. the mouse wrote a poem

- From a syntax point of view this is a valid sentence.

- From a semantics point of view not so fast…perhaps in Disney land

- Natural languages (English, French, Portuguese, etc) have very complex rules of syntax and not necessarily well-defined.

Formal Language

- Formal language – is specified by well-defined set of rules of syntax
- We describe the sentences of a formal language using a grammar.
- Two key questions:
- 1 - Is a combination of words a valid sentence in a formal language?
- 2 – How can we generate the valid sentences of a formal language?
  - Formal languages provide models for both natural languages and programming languages.

Grammars

- A formal *grammar G* is any compact, precise mathematical definition of a language *L*.
  - As opposed to just a raw listing of all of the language's legal sentences, or just examples of them.
- A grammar implies an algorithm that would generate all legal sentences of the language.
  - Often, it takes the form of a set of recursive definitions.
- A popular way to specify a grammar recursively is to specify it as a *phrase-structure grammar.*

Grammars (Semi-formal)

Example: A grammar that generates a subset of the English language

$$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle$$

$$\langle noun\_phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

$$\langle article \rangle \to a$$

$$\langle article \rangle \to the$$

-

$$\langle noun \rangle \to boy$$

$$\langle noun \rangle \to dog$$

$$\langle verb \rangle \to runs$$

$$\langle verb \rangle \to sleeps$$

- A derivation of "the boy sleeps":

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \ boy \ \ \langle verb \rangle \\
&\Rightarrow the \ \ boy \ \ sleeps
\end{aligned}
$$

- A derivation of "a dog runs":

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ dog \ \langle verb \rangle \\
&\Rightarrow a \ dog \ runs
\end{aligned}
$$

- Language of the grammar:

L = { "a boy runs",
"a boy sleeps",
"the boy runs",
"the boy sleeps",
"a dog runs",
"a dog sleeps",
"the dog runs",
"the dog sleeps" }

Notation

$$\langle noun \rangle \rightarrow boy$$

$$\langle noun \rangle \rightarrow dog$$

Variable
or
Non-terminal

Production
rule

Terminal
Symbols of
the vocabulary

Symbols of
the vocabulary

Basic Terminology

- A ***vocabulary/alphabet***, V is a finite nonempty set of elements called symbols.
  - Example: V = {a, b, c, A, B, C, S}
- A ***word/sentence*** over V is a string of finite length of elements of V.
  - Example: Aba
- The ***empty/null string***, λ is the string with no symbols.
- V^* is the set of all words over V.
  - Example: V^* = {Aba, BBa, bAA, cab …}
- A ***language*** over V is a subset of V^*.
  - We can give some criteria for a word to be in a language.

Phrase-Structure Grammars

- *A phrase-structure grammar* (abbr. PSG) $G = (V,T,S,P)$ is a 4-tuple, in which:
  - V is a set of *nonterminals (that is represented by Upper Case).*
  - T is a set of symbols called *terminals*
  - S, the *start symbol.*
    - in our example the start symbol was "sentence".
  - P is a set of *productions* (to be defined).
    - Rules for substituting one sentence fragment for another
    - Every production rule must contain at least one nonterminal on its left side.

Phrase-structure Grammar

- **EXAMPLE:**

- Let $G = (V, T, S, P)$,

- Where $V=\{ A, B, S\}$
- $T = \{a, b\}$,
- S is a start symbol
- $P = \{S \rightarrow ABa, A \rightarrow BB, B \rightarrow ab, A \rightarrow Bb\}$.

G is a Phrase-Structure Grammar.

What sentences can be generated with this grammar?

Derivation

- # Definition

- Let $G=(V,T,S,P)$ be a phrase-structure grammar.
- Let $w_0=lz_0r$ (the concatenation of l, z_0, and r) $w_1=lz_1r$ be strings over V.
- If $z_0 \rightarrow z_1$ is a production of G we say that w1 is directly derivable from w0 and we write $w_o \Rightarrow w_1$.
- If $w_0, w_1, \ldots, w_n$ are strings over V such that $w_0 \Rightarrow w_1, w_1 \Rightarrow w_2, \ldots, w_{n-1} \Rightarrow w_n$, then we say that w_n is derivable from w_0, and write $w_0 \Rightarrow^* w_n$.
- The sequence of steps used to obtain w_n from w_o is called a derivation.

Language

- Let G(V,T,S,P) be a phrase-structure grammar.

- The language generated by G (or the language of G)
- denoted by L(G) , is the set of all strings of terminals
- that are derivable from the starting state S.

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

Language L(G)

- **EXAMPLE:**

- Let $G = (V, T, S, P)$, where $V = \{, A, S\}$, $T = \{a, b\}$, S is a start symbol and $P = \{S \to aA, S \to b, A \to aa\}$.

- The language of this grammar is given by $L(G) = \{b, aaa\}$;

1. we can derive aA from using $S \to aA$, and then derive aaa using $A \to aa$.
2. We can also derive b using $S \to b$.

Another example

- Grammar:

G=(V,T,S,P) V={S} T={a,b} P =
$$S \to aSb$$
$$S \to \lambda$$

- Derivation of sentence :

$$ab$$

$$S \Rightarrow aSb \Rightarrow ab$$

$$S \to aSb \qquad S \to \lambda$$

- Grammar: $S \to aSb$

  $S \to \lambda$

- Derivation of sentence $aabb$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$S \to aSb$ $\qquad\qquad$ $S \to \lambda$

- Other derivations:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$$

$$\Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

So, what's the language of the grammar with the productions?

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

- Language of the grammar with the productions: $S \rightarrow aSb$

$$S \rightarrow \lambda$$

$$L = \{a^n b^n : n \geq 0\}$$

Another Example

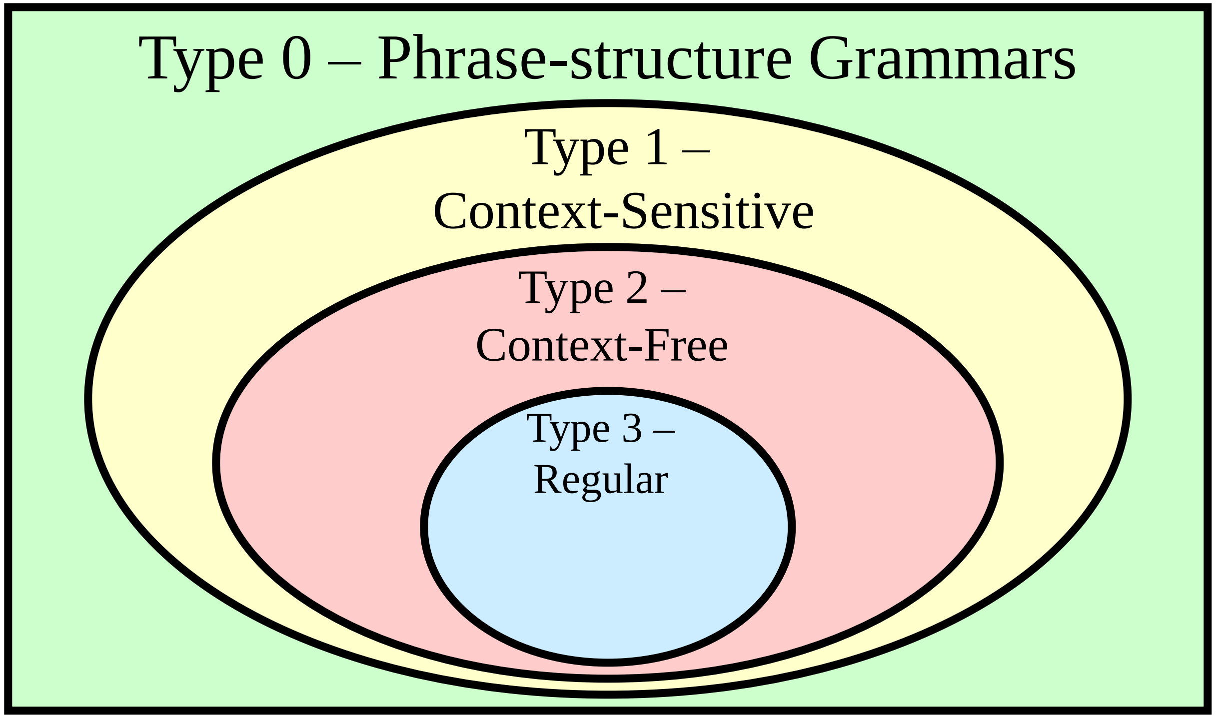
- Let $G = (\overbrace{\{A, B, S\}}^{V}, \overbrace{\{a, b\}}^{T}, S,$
- $\{\overbrace{S \rightarrow ABa,\ A \rightarrow BB,\ B \rightarrow ab,\ AB}^{P} \rightarrow b\}).$
- One possible derivation in this grammar is:

  $S \Rightarrow ABa \Rightarrow Aaba \Rightarrow BBaba \Rightarrow Bababa$

  $\Rightarrow abababa.$

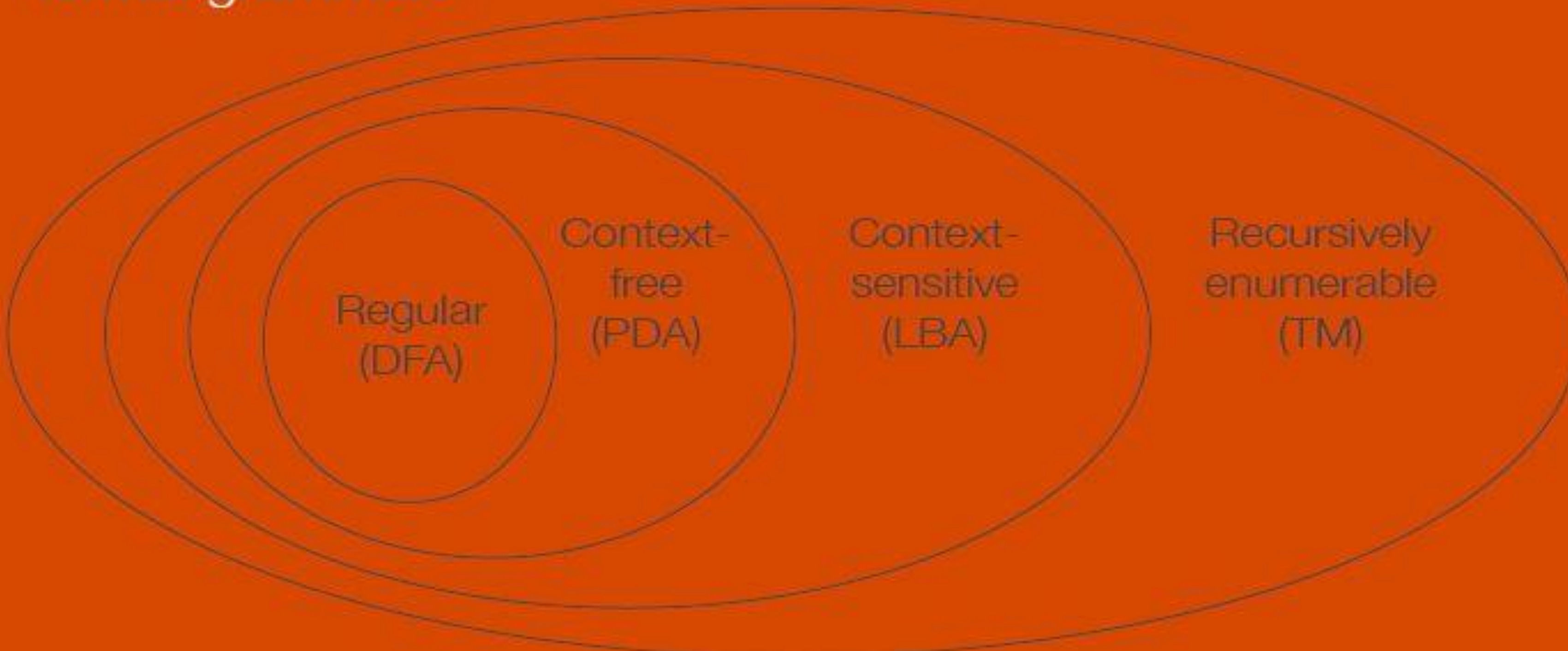
Types of Grammars - Chomsky hierarchy of languages

- Venn Diagram of Grammar Types:



The hierarchy

- A containment hierarchy (strictly nested sets) of classes of formal grammars



The hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown
Type-3	Regular	Regular	Finite

Defining the PSG(Phrase Structure Grammar) Types and Rules for type of productions in these Grammars

- Type 0: Phase-structure grammars – no restrictions on the production rules

- Type 1: Context-Sensitive PSG:
  - All after fragments are either longer than the corresponding before fragments, or empty:
  - if b → a, then $|b| <= |a| \quad \vee \quad a = \varepsilon$

- Type 2: Context-Free PSG:
  - All before fragments have length 1 and are non-terminals:
  - if b → a, then $|b| = 1$ ($b \in N$).

- Type 3: Regular PSGs:
  - All before fragments have length 1 and non-terminals
  - All after fragments are either single terminals, or a pair of a terminal followed by a nonterminal.
  - if b → a, then $a \in T \quad \vee \quad a \in TN$.

Classifying grammars

- Given a grammar, we need to be able to find the smallest class in which it belongs. This can be determined by answering three questions:
- Are the left hand sides of all of the productions single non-terminals?
- If yes, does each of the productions create at most one non-terminal and is it on the right?
- **Yes – regular** **No – context-free**
- If not, can any of the rules reduce the length of a string of terminals and non-terminals?
- **Yes – unrestricted** **No – context-sensitive**

Definition: Context-Free Grammars

Grammar $G = (V, T, S, P)$

Vocabulary

Terminal symbols

Start variable

Productions of the form:

$$A \to x$$

Non-Terminal

String of variables and terminals

Derivation Tree of A Context-free Grammar

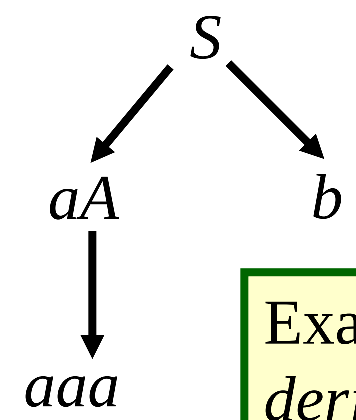
- Represents the language using an ordered rooted tree.

- Root represents the starting symbol.
- Internal vertices represent the nonterminal symbol that arise in the production.
- Leaves represent the terminal symbols.

- If the production $A \to w$ arise in the derivation, where w is a word, the vertex that represents A has as children vertices that represent each symbol in w, in order from left to right.

Language Generated by a Grammar

- Example: Let $G = (\{S,A,\},\{a,b\}, S, \{S \rightarrow aA, S \rightarrow b, A \rightarrow aa\})$. What is $L(G)$?
- Easy: We can just draw a tree of all possible derivations.
  - We have: $S \Rightarrow aA \Rightarrow aaa$.
  - and $S \Rightarrow b$.
- Answer: $L = \{aaa, b\}$.



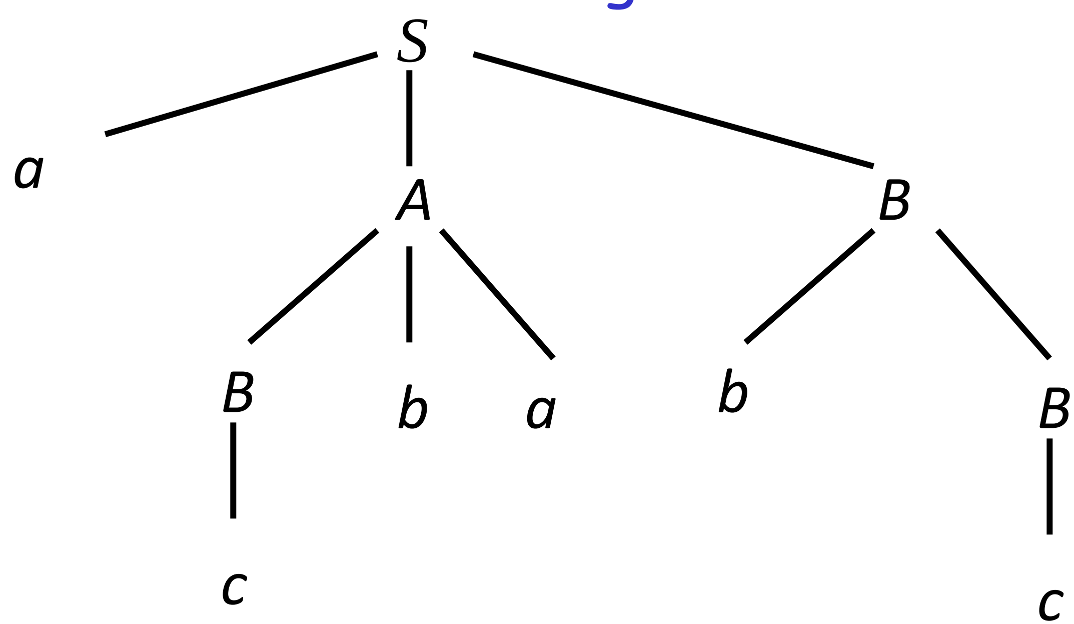
Example of a *derivation tree* or *parse tree* or *sentence diagram.*

Example: Derivation Tree

- Let G be a context-free grammar with the productions $P = \{S \rightarrow aAB,\ A \rightarrow Bba,\ B \rightarrow bB,\ B \rightarrow c\}$. The word w = acbabc can be derived from S as follows:

  $$S \Rightarrow aAB \rightarrow a(Bba)B \Rightarrow acbaB \Rightarrow acba(bB) \Rightarrow acbabc$$

  Thus, the derivation tree is given as follows:



Generating Infinite Languages

- A simple PSG can easily generate an infinite language.
- Example: $S \rightarrow 11S$, $S \rightarrow 0$ ($T = \{0,1\}$).
- The derivations are:
  - $S \Rightarrow 0$
  - $S \Rightarrow 11S \Rightarrow 110$
  - $S \Rightarrow 11S \Rightarrow 1111S \Rightarrow 11110$
  - and so on…

$L = \{(11)^*0\}$ – the set of all strings consisting of some number of concatenations of 11 with itself, followed by 0.

Another example

- Construct a PSG that generates the language $L = \{\mathbf{0}^n\mathbf{1}^n \mid n\in\mathbf{N}\}$.
  - **0** and **1** here represent symbols being concatenated *n* times, not integers being raised to the *n*th power.
- **Solution strategy:** Each step of the derivation should preserve the invariant that the number of **0**'s = the number of **1**'s in the template so far, and all **0**'s come before all **1**'s.
- **Solution:** $S \rightarrow \mathbf{0}S\mathbf{1}$, $S \rightarrow \lambda$.

Context-Sensitive Languages

- The language $\{ a^n b^n c^n \mid n \geq 1\}$ is context-sensitive but not context free.
- A grammar for this language is given by:
- $S \rightarrow aSBC \mid aBC$
- $CB \rightarrow BC$
- $aB \rightarrow ab$
- $bB \rightarrow bb$
- $bC \rightarrow bc$
- $cC \rightarrow cc$

Terminal and non-terminal

- A derivation from this grammar is:-
- $S \Rightarrow aSBC$
- $\Rightarrow aaBCBC$ (using $S \rightarrow aBC$)
- $\Rightarrow aabCBC$ (using $aB \rightarrow ab$)
- $\Rightarrow aabBCC$ (using $CB \rightarrow BC$)
- $\Rightarrow aabbCC$ (using $bB \rightarrow bb$)
- $\Rightarrow aabbcC$ (using $bC \rightarrow bc$)
- $\Rightarrow aabbcc$ (using $cC \rightarrow cc$)
- which derives $a^2b^2c^2$.

- **Language Of Grammar-**
-
- Language of Grammar is the set of all strings that can be generated from that grammar.
- If the language consists of finite number of strings, then it is called as a **Finite language**.
- If the language consists of infinite number of strings, then it is called as an **Infinite language**.

Example-01:

Consider a grammar G = (V , T , P , S) where-

V = { S }

T = { a , b }

P = { $S \rightarrow aSbS$, $S \rightarrow bSaS$, $S \rightarrow \in$ }

S = { S }

This grammar generates the strings having equal number of a's and b's.

So, Language of this grammar is-

L(G) = { ∈ , ab , ba , aabb , bbaa , abab , baba , …… }

This language consists of infinite number of strings.

Therefore, language of the grammar is infinite.

Consider a grammar $G = (V, T, P, S)$ where-

$V = \{ S, A, B, C \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow ABC, A \rightarrow a, B \rightarrow b, C \rightarrow c \}$

$S = \{ S \}$

This grammar generates only one string "abc".

So, Language of this grammar is-

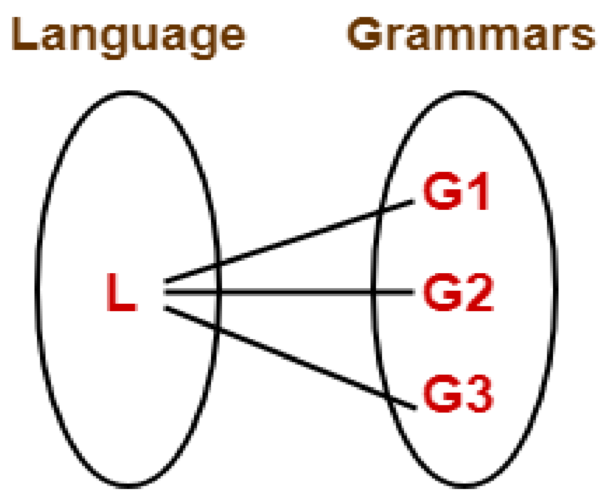
$L(G) = \{ abc \}$

This language consists of finite number of strings.

Therefore, language of the grammar is finite.

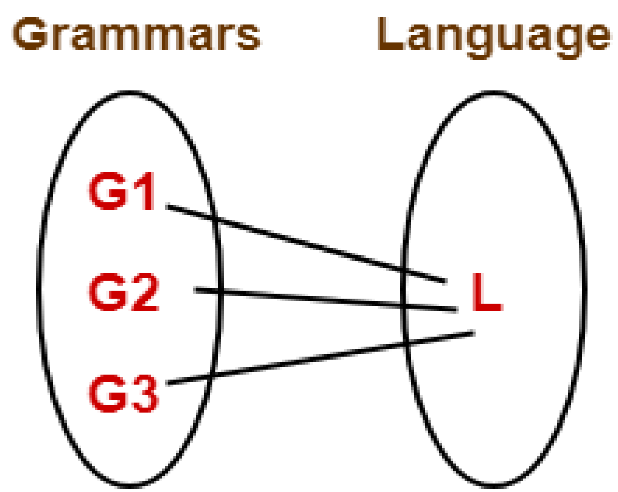
Important Concept-

- For any given grammar, the language generated by it is always unique.
- For any given language, we may have more than one grammar generating that language.



1 : m

(One to many relation)



m : 1

(Many to one relation)

Grammar G1-

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

The language generated by this grammar is-

L(G1) = { ab }

Grammar G2-

$$S \rightarrow AB$$

$$A \rightarrow \in$$

$$B \rightarrow ab$$

The language generated by this grammar is-

L(G2) = { ab }

Here,

Both the grammars generate a unique language.

But given a language L(G) = { ab }, we have two different grammars generating that language.

This justifies the above concept.

-

Steps to convert regular expressions directly to regular grammars and vice versa

Type	Regular Expression	RLG	LLG
Single terminal	e	$S \rightarrow e$	$S \rightarrow e$
Union operation	$(e+f)$	$S \rightarrow e \mid f$	$S \rightarrow e \mid f$
Concatenation	ef	$S \rightarrow eA,\ \ A \rightarrow f$	$S \rightarrow Af,\ \ A \rightarrow e$
Star closure	e^*	$S \rightarrow eS \mid \epsilon$	$S \rightarrow Se \mid \epsilon$
Plus closure	e^+	$S \rightarrow eS \mid e$	$S \rightarrow Se \mid e$
Star closure on union	$(e+f)^*$	$S \rightarrow eS \mid fS \mid \epsilon$	$S \rightarrow Se \mid Sf \mid \epsilon$
Plus closure on union	$(e+f)^+$	$S \rightarrow eS \mid fS \mid e \mid f$	$S \rightarrow Se \mid Sf \mid e \mid f$
Star closure on concatenation	$(ef)^*$	$S \rightarrow eA \mid \epsilon;$ $A \rightarrow fS$	$S \rightarrow Af \mid \epsilon;$ $A \rightarrow Se$
Plus closure on concatenation	$(ef)^+$	$S \rightarrow eA\,;$ $A \rightarrow fS \mid f$	$S \rightarrow Af;$ $A \rightarrow Se \mid e$

step by step find RLG and LLG for regex $0*(1(0+1))*0*(1(0+1))*$. At each step, same color is used to match part of regex getting translated into corresponding part in grammar.

Preparing RLG

1.	0*(1(0+1))*	$S \to 0S$
2.	0*(1(0+1))*	$S \to 0S \mid A \mid \epsilon$
3.	0*(1(0+1))*	$S \to 0S \mid A \mid \epsilon$ $A \to 1B$
4.	0*(1(0+1))*	$S \to 0S \mid A \mid \epsilon$ $A \to 1B$ $B \to 0A \mid 1A \mid 0 \mid 1$

Preparing LLG

1.	$0^*(1(0+1))^*$	$S \to A \mid \epsilon$
2.	$0^*(1(0+1))^*$	$S \to A \mid \epsilon$ $A \to A10 \mid A11 \mid B$
3.	$0^*(1(0+1))^*$	$S \to A \mid \epsilon$ $A \to A10 \mid A11 \mid B$ $B \to B0 \mid 0$

- **Example 1**
- Consider the regular expression $(a + b)^*a$. We will now construct a regular grammar for this regular expression. For every terminal symbol a, we create a regular grammar with the rule *S* \arrow a, start symbol *S*. We then apply the transformations to these regular grammars, progressively constructing the regular grammar.

First consider the expression $a + b$. We create two regular grammars:

$$S1 \rightarrow a \text{ and}$$
$$S2 \rightarrow b$$

where $S1$ and $S2$ are the start symbols. Clearly, these grammars recognise the regular expressions a and b respectively.

Now, we apply the union transformation for regular grammars to get:

$$S3 \rightarrow a \mid b$$
$$S1 \rightarrow a$$
$$S2 \rightarrow b$$

where $S3$ is the start symbol. This grammar obviously recognises $a + b$.

Next, we consider the expression $(a + b)^*$. We already have a regular grammar for $(a + b)$, so now we apply the Kleene star transformation on the regular grammar:

$$S4 \rightarrow a \mid b \mid \varepsilon$$
$$S3 \rightarrow a \mid b$$
$$S1 \rightarrow aS3$$
$$S2 \rightarrow bS3$$

where $S4$ is the start symbol.

Recall that we need a regular grammar that recognises $(a + b)^*a$. We thus consider again the regular expression a. Again, we create a regular grammar that describes the language:

Recall that we need a regular grammar that recognises $(a + b)^*a$. We thus consider again the regular expression a. Again, we create a regular grammar that describes the language:

$$S5 \rightarrow a$$

where $S5$ is the start symbol.

We now construct the catenation of the regular grammar describing $(a + b)^*$ together with this one. We simply apply the transformation that catenates two regular grammars, to get:

$$S4 \rightarrow a \mid b \mid \varepsilon$$
$$S3 \rightarrow aS5 \mid bS5$$
$$S1 \rightarrow aS3$$
$$S2 \rightarrow bS3$$
$$S5 \rightarrow a$$

where $S4$ is the start symbol.

This regular grammar is equivalent to the regular expression $(a + b)^*a$.