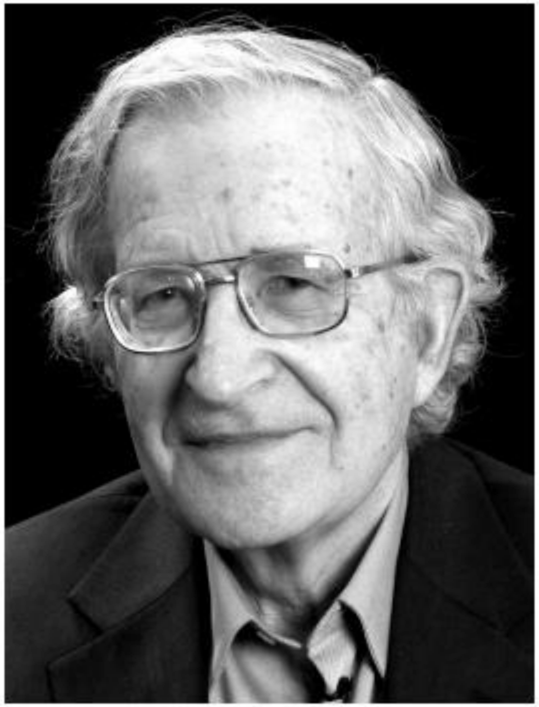
Context Free Languages

Context Free Grammars

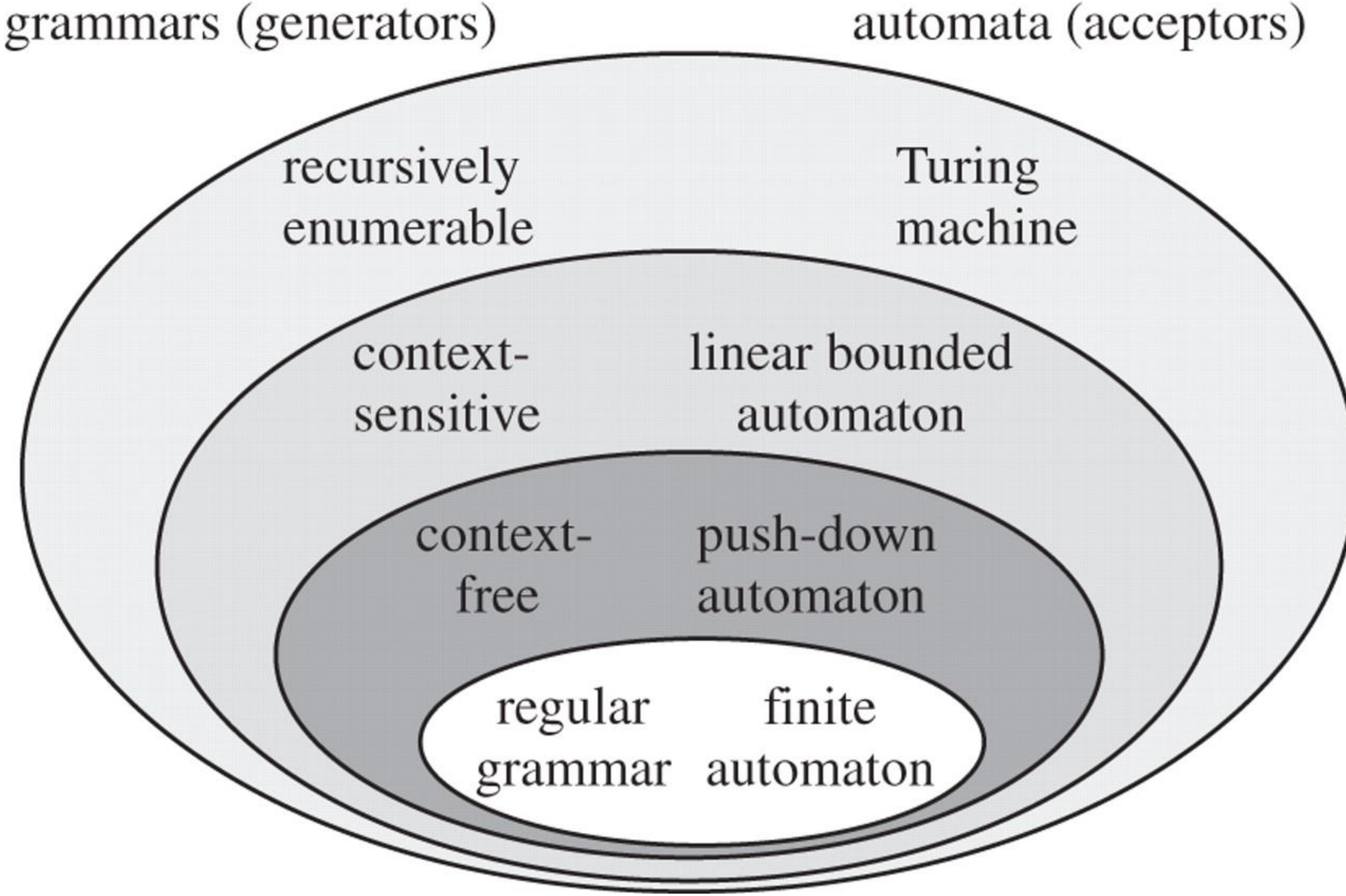
Context-Free Grammars



Noam Chomsky
(linguist, philosopher, logician, and activist)

In the formal languages of computer science and linguistics, the **Chomsky hierarchy** is a **hierarchy** of classes of formal grammars. This **hierarchy** of grammars was described by Noam **Chomsky** in 1956.

Chomsky Hierarchy



The Hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursive Enumerable	Turing Machine
Type-1	Context Sensitive	Context Sensitive	Linear-Bound
Type-2	Context Free	Context Free	Pushdown
Type-3	Regular	Regular	Finite

How production rules look like

Type	Grammar	Production rules
Type 0	unrestricted	$\alpha \rightarrow \beta$
Type 1	context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type 2	**context-free**	$A \rightarrow \gamma$
Type 3	regular	$A \rightarrow aB$ or $A \rightarrow Ba$

A grammar generates sentences (strings) in a language

Examples

Consider the grammar

$$
\begin{aligned}
S &\rightarrow AB && (1)\\
A &\rightarrow C && (2)\\
CB &\rightarrow Cb && (3)\\
C &\rightarrow a && (4)
\end{aligned}
$$

where $\{a, b\}$ are terminals, and $\{S, A, B, C\}$ are non-terminals.

Examples

Consider the grammar

$$
\begin{align}
S &\rightarrow AB \tag{1}\\
A &\rightarrow C \tag{2}\\
CB &\rightarrow Cb \tag{3}\\
C &\rightarrow a \tag{4}
\end{align}
$$

where $\{a, b\}$ are terminals, and $\{S, A, B, C\}$ are non-terminals.
We can derive the phrase "ab" from this grammar in the following way:

$$
\begin{aligned}
S &\rightarrow AB, \text{ from (1)}\\
&\rightarrow CB, \text{ from (2)}\\
&\rightarrow Cb, \text{ from (3)}\\
&\rightarrow ab, \text{ from (4)}
\end{aligned}
$$

Examples

Consider the grammar

$$
\begin{align}
S &\rightarrow NounPhrase \quad VerbPhrase \tag{5}\\
NounPhrase &\rightarrow SingularNoun \tag{6}\\
SingularNoun \quad VerbPhrase &\rightarrow SingularNoun \quad comes \tag{7}\\
SingularNoun &\rightarrow John \tag{8}
\end{align}
$$

We can derive the phrase "John comes" from this grammar in the following way:

$$
\begin{aligned}
S &\rightarrow NounPhrase \quad VerbPhrase, \text{ from (1)}\\
&\rightarrow SingularNoun \quad VerbPhrase, \text{ from (2)}\\
&\rightarrow SingularNoun \quad comes, \text{ from (3)}\\
&\rightarrow John \quad comes, \text{ from (4)}
\end{aligned}
$$

Type	Grammar	Production rules
Type 0	unrestricted	$\alpha \to \beta$
Type 1	context-sensitive	$\alpha A \beta \to \alpha \gamma \beta$
Type 2	**context-free**	$A \to \gamma$
Type 3	regular	$A \to aB$ or $A \to Ba$

Definition (Context-Free Grammar)

A context-free grammar is a tuple $G = (V, T, P, S)$ where

- V is a finite set of variables (nonterminals, nonterminals vocabulary);
- T is a finite set of terminals (letters);
- $P \subseteq V \times (V \cup T)^*$ is a finite set of rewriting rules called productions,
  - We write $A \rightarrow \beta$ if $(A, \beta) \in P$;
- $S \in V$ is a distinguished start or "sentence" symbol.

Definition (Context-Free Grammar)

A context-free grammar is a tuple $G = (V, T, P, S)$ where

- V is a finite set of variables (nonterminals, nonterminals vocabulary);
- T is a finite set of terminals (letters);
- $P \subseteq V \times (V \cup T)^*$ is a finite set of rewriting rules called productions,
  - We write $A \to \beta$ if $(A, \beta) \in P$;
- $S \in V$ is a distinguished start or "sentence" symbol.

Example: $G_{0^n1^n} = (V, T, P, S)$ where

- $V = \{S\}$;
- $T = \{0, 1\}$;
- P is defined as

$$\begin{array}{lcl} S & \to & \varepsilon \\ S & \to & 0S1 \end{array}$$

- $S = S$.

Palindromes

$G_{pal} = (\{P\}, \{0,1\}, A, P)$

$$
\begin{array}{llll}
1. & P & \rightarrow & \epsilon \\
2. & P & \rightarrow & 0 \\
3. & P & \rightarrow & 1 \\
4. & P & \rightarrow & 0P0 \\
5. & P & \rightarrow & 1P1
\end{array}
$$

A context-free grammar for palindromes

Derivation:

- Let $G = (V, T, P, S)$ be a context-free grammar.
- Let $\alpha A \beta$ be a string in $(V \cup T)^* V (V \cup T)^*$
- We say that $\alpha A \beta$ yields the string $\alpha \gamma \beta$, and we write $\alpha A \beta \Rightarrow \alpha \gamma \beta$ if

$$A \rightarrow \gamma \text{ is a production rule in } G.$$

- For strings $\alpha, \beta \in (V \cup T)^*$, we say that α derives β and we write $\alpha \stackrel{*}{\Rightarrow} \beta$ if there is a sequence $\alpha_1, \alpha_2, \ldots, \alpha_n \in (V \cup T)^*$ s.t.

$$\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \cdots \alpha_n \Rightarrow \beta.$$

$\Rightarrow$ is also called direct derivation.

$\stackrel{i}{\Rightarrow}$ is to mean that the i th production is used in the direct derivation.

$\stackrel{*}{\Rightarrow}$ is reflexive and transitive closure of $\Rightarrow$

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

T is the set of symbols $\{+, *, (,), a, b, 0, 1\}$ and P is the set of productions

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

T is the set of symbols $\{+, *, (,), a, b, 0, 1\}$ and P is the set of productions

- Can you find how the following is true.

$$E \overset{*}{\Rightarrow} (a1 + b0 * a1)$$

Compact Notation for Productions

It is convenient to think of a production as "belonging" to the variable of its head. We shall often use remarks like "the productions for A" or "A-productions" to refer to the productions whose head is variable A. We may write the productions for a grammar by listing each variable once, and then listing all the bodies of the productions for that variable, separated by vertical bars. That is, the productions $A \to \alpha_1,\ A \to \alpha_2, \ldots, A \to \alpha_n$ can be replaced by the notation $A \to \alpha_1|\alpha_2|\cdots|\alpha_n$. For instance, the grammar for palindromes from Fig. 5.1 can be written as $P \to \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$.

CFL Definition

The language $L(G)$ accepted by a context-free grammar $G = (V, T, P, S)$ is the set

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

Leftmost and Rightmost Derivations

- Derivations are not unique.

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow} I * E \underset{lm}{\Rightarrow} a * E \underset{lm}{\Rightarrow}$$

$$a * (E) \underset{lm}{\Rightarrow} a * (E + E) \underset{lm}{\Rightarrow} a * (I + E) \underset{lm}{\Rightarrow} a * (a + E) \underset{lm}{\Rightarrow}$$

$$a * (a + I) \underset{lm}{\Rightarrow} a * (a + I0) \underset{lm}{\Rightarrow} a * (a + I00) \underset{lm}{\Rightarrow} a * (a + b00)$$

We can also summarize the leftmost derivation by saying $E \overset{*}{\underset{lm}{\Rightarrow}} a * (a + b00)$, or express several steps of the derivation by expressions such as $E * E \overset{*}{\underset{lm}{\Rightarrow}} a * (E)$.

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

$$E \underset{rm}{\Rightarrow} E * E \underset{rm}{\Rightarrow} E * (E) \underset{rm}{\Rightarrow} E * (E + E) \underset{rm}{\Rightarrow}$$

$$E * (E + I) \underset{rm}{\Rightarrow} E * (E + I0) \underset{rm}{\Rightarrow} E * (E + I00) \underset{rm}{\Rightarrow} E * (E + b00) \underset{rm}{\Rightarrow}$$

$$E * (I + b00) \underset{rm}{\Rightarrow} E * (a + b00) \underset{rm}{\Rightarrow} I * (a + b00) \underset{rm}{\Rightarrow} a * (a + b00)$$

This derivation allows us to conclude $E \overset{*}{\underset{rm}{\Rightarrow}} a * (a + b00)$. $\square$

Exercise

Consider the following grammar:

$$\begin{array}{rcl} S & \to & AS \mid \varepsilon. \\ S & \to & aa \mid ab \mid ba \mid bb \end{array}$$

Give leftmost and rightmost derivations of the string *aabbba*.

$$G_{pal} = (\{P\}, \{0,1\}, A, P)$$

$$\begin{array}{llll} 1. & P & \rightarrow & \epsilon \\ 2. & P & \rightarrow & 0 \\ 3. & P & \rightarrow & 1 \\ 4. & P & \rightarrow & 0P0 \\ 5. & P & \rightarrow & 1P1 \end{array}$$

A context-free grammar for palindromes

Prove that $L(G_{pal})$ is the set of palindromes over the given alphabet.

$$G_{pal} = (\{P\}, \{0,1\}, A, P)$$

$$\begin{array}{llll} 1. & P & \to & \epsilon \\ 2. & P & \to & 0 \\ 3. & P & \to & 1 \\ 4. & P & \to & 0P0 \\ 5. & P & \to & 1P1 \end{array}$$

A context-free grammar for palindromes

Prove that $L(G_{pal})$ is the set of palindromes over the given alphabet.

- This proof has two parts ($\Rightarrow$ and $\Leftarrow$)

$$G_{pal} = (\{P\}, \{0,1\}, A, P)$$

1.	P	$\rightarrow$	ϵ
2.	P	$\rightarrow$	0
3.	P	$\rightarrow$	1
4.	P	$\rightarrow$	$0P0$
5.	P	$\rightarrow$	$1P1$

A context-free grammar for palindromes

Prove that $L(G_{pal})$ is the set of palindromes over the given alphabet.

- This proof has two parts (⇒ and ⇐)
  1) $(w = w^R) \Rightarrow w \in L(G_{pal})$
  2) $w \in L(G_{pal}) \Rightarrow (w = w^R)$

$$(w = w^R) \Rightarrow w \in L(G_{pal})$$

1.	P	$\rightarrow$	ϵ
2.	P	$\rightarrow$	0
3.	P	$\rightarrow$	1
4.	P	$\rightarrow$	$0P0$
5.	P	$\rightarrow$	$1P1$

A context-free grammar for palindromes

- Proof [by induction on $|w|$:

BASIS: We use lengths 0 and 1 as the basis.

If $|w| = 0$ or $|w| = 1$, then w is ϵ, 0, or 1.

Since there are productions $P \rightarrow \epsilon$, $P \rightarrow 0$, and $P \rightarrow 1$, we conclude that $P \stackrel{*}{\Rightarrow} w$ in any of these basis cases.

INDUCTION: Suppose $|w| \geq 2$. Since $w = w^R$, w must begin and end with the same symbol

- Note, $w \in L(G_{pal})$ is same $P \stackrel{*}{\Rightarrow}$ w

BASIS: We use lengths 0 and 1 as the basis.

If $|w| = 0$ or $|w| = 1$, then w is ϵ, 0, or 1.

Since there are productions $P \to \epsilon$, $P \to 0$, and $P \to 1$, we conclude that $P \overset{*}{\Rightarrow} w$ in any of these basis cases.

INDUCTION: Suppose $|w| \geq 2$. Since $w = w^R$, w must begin and end with the same symbol

BASIS: We use lengths 0 and 1 as the basis.

If $|w| = 0$ or $|w| = 1$, then w is ϵ, 0, or 1.

Since there are productions $P \to \epsilon$, $P \to 0$, and $P \to 1$, we conclude that $P \overset{*}{\Rightarrow} w$ in any of these basis cases.

INDUCTION: Suppose $|w| \geq 2$. Since $w = w^R$, w must begin and end with the same symbol

Inductive Hypothesis: Let for $|w| \leq k \; where \; (w = w^R)$, $P \overset{*}{\Rightarrow}$ w is true.

Inductive Step: We need to show for $|w| = k + 1, P \overset{*}{\Rightarrow}$ w is true.

Note, $w = 0x0$ or w $= 1x1$, where $|x| = k - 1$.

Then, $P \Rightarrow 0P0 \overset{*}{\Rightarrow} 0x0$ (Since $|x| \leq k$, so $P \overset{*}{\Rightarrow} x$ is true).

So, $P \overset{*}{\Rightarrow}$ w is true. With a similar argument, $P \Rightarrow 1P1 \overset{*}{\Rightarrow} 1x1$

$$w \in L(G_{pal}) \Rightarrow (w = w^R)$$

- Proof [by induction on number of steps in the derivation]:

BASIS: If the derivation is one step, then it must use one of the three productions that do not have P in the body. That is, the derivation is $P \Rightarrow \epsilon$, $P \Rightarrow 0$, or $P \Rightarrow 1$. Since ϵ, 0, and 1 are all palindromes, the basis is proven.

INDUCTION:

- Assume for n steps it is true.
- Then, show for (n+1) steps it must be true.

Left as an exercise.

Sentential Forms

$G = (V, T, P, S)$ is a CFG, then any string α in $(V \cup T)^*$ such that $S \stackrel{*}{\Rightarrow} \alpha$ is a *sentential form*.

Sentential Forms

$G = (V, T, P, S)$ is a CFG, then any string α in $(V \cup T)^*$ such that $S \stackrel{*}{\Rightarrow} \alpha$ is a *sentential form.*

1.	E	$\rightarrow$	I
2.	E	$\rightarrow$	$E + E$
3.	E	$\rightarrow$	$E * E$
4.	E	$\rightarrow$	(E)
5.	I	$\rightarrow$	a
6.	I	$\rightarrow$	b
7.	I	$\rightarrow$	Ia
8.	I	$\rightarrow$	Ib
9.	I	$\rightarrow$	$I0$
10.	I	$\rightarrow$	$I1$

A context-free grammar for simple expressions

$$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (I + E)$$

Sentential Forms

$G = (V, T, P, S)$ is a CFG, then any string α in $(V \cup T)^*$ such that $S \overset{*}{\Rightarrow} \alpha$ is a *sentential form*.

If $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$, then α is a *left-sentential form*,

and if $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$, then α is a *right-sentential form*.

Note that the language $L(G)$ is those sentential forms that are in T^*; i.e., they consist solely of terminals.

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E+E \\
3. & E & \rightarrow & E*E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

A context-free grammar for simple expressions

$$
E \Rightarrow E*E \Rightarrow E*(E) \Rightarrow E*(E+E) \Rightarrow E*(I+E)
$$

- Is this sentential form left-sentential? Or right-sentential?

Exercise 5.1.2: The following grammar generates the language of regular expression $\mathbf{0}^*\mathbf{1}(\mathbf{0}+\mathbf{1})^*$:

$$\begin{array}{lcl} S & \rightarrow & A1B \\ A & \rightarrow & 0A \mid \epsilon \\ B & \rightarrow & 0B \mid 1B \mid \epsilon \end{array}$$

Give leftmost and rightmost derivations of the following strings:

* a) 00101.

b) 1001.

c) 00011.

Note, the given grammar is not a regular grammar (even-though it generates a regular language).

Can you find $L(G)$?

- $S \rightarrow aS|bS|a|b|\epsilon$

Can you find $L(G)$?

- $S \to aS|bS|a|b|\epsilon$

- Answer: All strings. Σ^*

Can you find $L(G)$?

1. $S \to S_1S|\epsilon,$
2. $S_1 \to aS_1b|ab$

Can you find $L(G)$?

1. $S \to S_1S|\epsilon$,
2. $S_1 \to aS_1b|ab$

Recall that the $S \to aSb|\epsilon$ generates $\{a^nb^n|n \geq 0\}$.

Can you find $L(G)$?

1. $S \rightarrow S_1 S | \epsilon,$
2. $S_1 \rightarrow aS_1 b | ab$

Recall that the $S \rightarrow aSb|\epsilon$ generates $\{a^n b^n | n \geq 0\}$.

Starting from S_1 we get $\{a^n b^n | n \geq 1\}$

Can you find $L(G)$?

1. $S \rightarrow S_1 S | \epsilon,$
2. $S_1 \rightarrow a S_1 b | ab$

Recall that the $S \rightarrow aSb|\epsilon$ generates $\{a^n b^n | n \geq 0\}$.

Starting from S_1 we get $\{a^n b^n | n \geq 1\}$

The answer:

$a^{n_1} \mathrm{b}^{\mathrm{n}_1} a^{n_2} \mathrm{b}^{\mathrm{n}_2} \ldots a^{n_k} \mathrm{b}^{\mathrm{n}_\mathrm{k}} \in L(G)$

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

Can you find $L(G)$?

$$S \to SS \mid S \mid [S] \mid (\,) \mid [\,]$$

Can you find $L(G)$?

$$S \to SS \mid S \mid [S] \mid (\,) \mid [\,]$$

Set of all balanced parentheses with alphabet $\{\ (,\ ),\ [,\ ]\ \}$

Can you find $L(G)$?

1. $S \rightarrow aB|bA$
2. $B \rightarrow b|bS|aBB$
3. $A \rightarrow a|aS|bAA$

Can you find $L(G)$?

1. $S \rightarrow aB|bA$
2. $B \rightarrow b|bS|aBB$
3. $A \rightarrow a|aS|bAA$

Produces strings with equal number of a's and b's.

Can you find $L(G)$?

1. $S \to SaSbS|SbSaS|\epsilon$

Can you find $L(G)$?

1. $S \rightarrow SaSbS|SbSaS|\epsilon$

Produces strings with equal number of a's and b's.

With one difference than the previous CFG. What is it?