Context-Free Languages

- The class of context-free languages generalizes over the class of regular languages, i.e., every regular language is a context-free language.
- The reverse of this is not true, i.e., every context-free language is not necessarily regular. For example, as we will see $\{0^k1^k \mid k \geq 0\}$ is context-free but not regular.
- Many issues and questions we asked for regular languages will be the same for context-free languages:

  Machine model – PDA (Push-Down Automata)

  Descriptor – CFG (Context-Free Grammar)

  Pumping lemma for context-free languages (and find CFL's limit)

  Closure of context-free languages with respect to various operations

  Algorithms and conditions for finiteness or emptiness

- Some analogies don't hold, e.g., non-determinism in a PDA makes a difference and, in particular, deterministic PDAs define a subset of the context-free languages.
- We will only talk on non-deterministic PDA here.

- Informally, a Context-Free Language (CFL) is a language generated by a Context-Free Grammar (CFG).

- What is a CFG?

- Informally, a CFG is a set of rules for deriving (or *generating*) strings (or sentences) in a language.

- Note: A grammar generates a string, whereas a machine accepts a string

- **Example CFG:**

<sentence> –> <noun-phrase> <verb-phrase>	(1)
<noun-phrase> –> <proper-noun>	(2)
<noun-phrase> –> <determiner> <common-noun>	(3)
<proper-noun> –> John	(4)
<proper-noun> –> Jill	(5)
<common-noun> –> car	(6)
<common-noun> –> hamburger	(7)
<determiner> –> a	(8)
<determiner> –> the	(9)
<verb-phrase> –> <verb> <adverb>	(10)
<verb-phrase> –> <verb>	(11)
<verb> –> drives	(12)
<verb> –> eats	(13)
<adverb> –> slowly	(14)
<adverb> –> frequently	(15)

- **Example Derivation:**

<sentence>	=> <noun-phrase> <verb-phrase>	by (1)
	=> <proper-noun> <verb-phrase>	by (2)
	=> Jill <verb-phrase>	by (5)
	=> Jill <verb> <adverb>	by (10)
	=> Jill drives <adverb>	by (12)
	=> Jill drives frequently	by (15)

- Informally, a CFG consists of:

  - A set of replacement *rules*, each having a Left-Hand Side (LHS) and a Right-Hand Side (RHS).
  - Two types of symbols; *variables* and *terminals*.
  - LHS of each rule is a *single* variable (no terminals).
  - RHS of each rule is a string of *zero or more* variables and terminals.
  - A *string* consists of only terminals.

- Formally, a Context-Free Grammar (CFG) is a 4-tuple:

G = (V, T, P, S)

V - A finite set of variables or *non-terminals*

T - A finite set of *terminals* (*V* and *T* do not intersect: *do not use same symbols*)
This is our ∑

P - A finite set of *productions*, each of the form A –> α, where *A* is in *V* and
α is in $(V \cup T)^*$

Note that α may be ε

S - A starting non-terminal (*S* is in *V*)

- **Example CFG for** $\{0^k1^k \mid k \geq 0\}$**:**

G = ({S}, {0, 1}, P, S)   // *Remember: G = (V, T, P, S)*

P:

(1) S –> 0S1   or just simply S –> 0S1 | ε

(2) S –> ε

- **Example Derivations:**

S	=> 0S1	(1)	S	=> ε	(2)
	=> 01	(2)			

S	=> 0S1	(1)
	=> 00S11	(1)
	=> 000S111	(1)
	=> 000111	(2)

- Note that G "generates" the language $\{0^k1^k \mid k \geq 0\}$

- **Example CFG for** ?**:**

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

  P:

(1)	S –> ABC	
(2)	A –> aA	A –> aA | ε
(3)	A –> ε	
(4)	B –> bB	B –> bB | ε
(5)	B –> ε	
(6)	C –> cC	C –> cC | ε
(7)	C –> ε	

- **Example Derivations:**

S	=> ABC	(1)	S	=> ABC	(1)
	=> BC	(3)		=> aABC	(2)
	=> C	(5)		=> aaABC	(2)
	=> ε	(7)		=> aaBC	(3)
				=> aabBC	(4)
				=> aabC	(5)
				=> aabcC	(6)
				=> aabc	(7)

- Note that G generates the language a*b*c*

Formal Definitions for CFLs

- Let G = (V, T, P, S) be a CFG.

- **Observation:** **"–>"** forms a relation on V and $(V \cup T)^*$

- **Definition:** Let *A* be in *V*, and *B* be in $(V \cup T)^*$, A –> B be in *P*, and let α and β be in $(V \cup T)^*$. Then:

$$\alpha A\beta \Rightarrow \alpha B\beta$$

  In words, $\alpha A\beta$ *directly derives* $\alpha B\beta$, or in other words $\alpha B\beta$ follows from $\alpha A\beta$ by the application of exactly one production from *P*.

- **Observation:** **"=>"** forms a relation on $(V \cup T)^*$ and $(V \cup T)^*$.

- **Definition:** Suppose that $\alpha_1, \alpha_2, \ldots, \alpha_m$ are in $(V \cup T)^*$, $m \geq 1$, and

$$\begin{array}{l} \alpha_1 \Rightarrow \alpha_2 \\ \alpha_2 \Rightarrow \alpha_3 \\ \quad \vdots \\ \alpha_{m-1} \Rightarrow \alpha_m \end{array}$$

  Then $\alpha_1 \Rightarrow^* \alpha_m$

  In words, α_m follows from α_1 by the application of *zero or more* productions. Note that: $\alpha \Rightarrow^* \alpha$.

- **Observation:** "**=>***" forms a relation on $(V \cup T)^*$ and $(V \cup T)^*$.

- **Definition:** Let α be in $(V \cup T)^*$. Then α is a *sentential form* if and only if $S \Rightarrow^* \alpha$.

- **Definition:** Let $G = (V, T, P, S)$ be a context-free grammar. Then the *language generated* by G, denoted L(G), is the set:

$$\{w \mid w \text{ is in } T^* \text{ and } S \Rightarrow^* w\}$$

- **Definition:** Let *L* be a language. Then *L* is a *context-free language* if and only if there exists a context-free grammar *G* such that L = L(G).

- **Definition:** Let G_1 and G_2 be context-free grammars. Then *G1* and *G2* are *equivalent* if and only if $L(G_1) = L(G_2)$.

- **Theorem:** Let *L* be a regular language. Then *L* is a context-free language. (or, RL $\subseteq$CFL)

- **Proof:** (by induction)

  We will prove that if *r* is a regular expression then there exists a CFG *G* such that L(r) = L(G). The proof will be by induction on the number of operators in *r*.

  Basis: Op(r) = 0

  Then *r* is either Ø, ε, or ***a***, for some symbol ***a*** in Σ.

  For Ø:

  Let G = ({S}, {}, P, S) where P = {}

  For ε:

  Let G = ({S}, {}, P, S) where P = {S –> ε}

  For **a**:

  Let G = ({S}, {a}, P, S) where P = {S –> **a**}

Inductive Hypothesis:

Suppose that for any regular expression r, where $0 \leq op(r) \leq k$, that there exists a CFG G such that L(r) = L(G), for some k>=0.

Inductive Step:

Let r be a regular expression with op(r)=k+1. Then $r = r_1 + r_2$, $r = r_1r_2$ or $r = r_1*$.

Case 1) $r = r_1 + r_2$

Since r has k+1 operators, one of which is +, it follows that r_1 and r_2 have at most k operators. From the inductive hypothesis it follows that there exist CFGs $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$ such that $L(r_1) = L(G_1)$ and $L(r_2) = L(G_2)$. Assume without loss of generality that V_1 and V_2 have no non-terminals in common, and construct a grammar G = (V, T, P, S) where:

$V = V_1 \cup V_2 \cup \{S\}$

$T = T_1 \cup T_2$

$P = P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$

Clearly, L(r) = L(G).

Case 2) $r = r_1 r_2$

Let $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$ be as in Case 1, and construct a grammar $G = (V, T, P, S)$ where:

$V = V_1 \cup V_2 \cup \{S\}$
$T = T_1 \cup T_2$
$P = P_1 \cup P_2 \; U \; \{S \rightarrow S_1S_2\}$

Clearly, $L(r) = L(G)$.

Case 3) $r = (r_1)^*$

Let $G_1 = (V_1, T_1, P_1, S_1)$ be a CFG such that $L(r_1) = L(G_1)$ and construct a grammar $G = (V, T, P, S)$ where:

$V = V_1 \cup \{S\}$
$T = T_1$
$P = P_1 \cup \{S \rightarrow S_1S, S \rightarrow \varepsilon\}$

Clearly, $L(r) = L(G)$. •

- The preceding theorem is constructive, in the sense that it shows how to construct a CFG from a given regular expression.

- **Example #1:**

  $r = a^*b^*$

  $r = r_1r_2$

  $r_1 = r_3^*$

  $r_3 = a$

  $r_2 = r_4^*$

  $r_4 = b$

- **Example #1:** a*b*

$r_4 = b$	$S_1 \to b$
$r_3 = a$	$S_2 \to a$
$r_2 = r_4^*$	$S_3 \to S_1 S_3$ $S_3 \to \varepsilon$
$r_1 = r_3^*$	$S_4 \to S_2 S_4$ $S_4 \to \varepsilon$
$r = r_1 r_2$	$S_5 \to S_4 S_3$

- **Example #2:**

  $r = (0+1)^*01$

  $r = r_1 r_2$

  $r_1 = r_3^*$

  $r_3 = (r_4 + r_5)$

  $r_4 = 0$

  $r_5 = 1$

  $r_2 = r_6 r_7$

  $r_6 = 0$

  $r_7 = 1$

- **Example #2:** (0+1)*01

$r_7 = 1$	$S_1 \rightarrow 1$
$r_6 = 0$	$S_2 \rightarrow 0$
$r_2 = r_6 r_7$	$S_3 \rightarrow S_2 S_1$
$r_5 = 1$	$S_4 \rightarrow 1$
$r_4 = 0$	$S_5 \rightarrow 0$
$r_3 = (r_4 + r_5)$	$S_6 \rightarrow S_4,\ S_6 \rightarrow S_5$
$r_1 = r_3^*$	$S_7 \rightarrow S_6 S_7$ $S_7 \rightarrow \varepsilon$
$r = r_1 r_2$	$S_8 \rightarrow S_7 S_3$

- **Definition:** A CFG is a regular grammar if each rule is of the following form:

  – A –> a
  – A –> aB
  – A –> ε

  where *A* and *B* are in *V*, and *a* is in *T*

- **Theorem:** A language *L* is a regular language iff there exists a regular grammar *G* such that L = L(G).

- **Proof:** Exercise. •Develop translation fromRegular form -> DFA; and
  DFA -> regular grammar]

- **Observation:** The grammar S –> 0S1 | ε is not a regular grammar.

- **Observation:** A language may have several CFGs, some regular, some not (The fact that the preceding grammar is not regular does not in and of itself prove that 0^n1^n is not a regular language).

- **Definition:** Let G = (V, T, P, S) be a CFG. A tree is a <u>derivation (or parse) tree</u> if:

  – Every vertex has a label from $V \cup T \cup \{\varepsilon\}$
  – The label of the root is S
  – If a vertex with label A has children with labels $X_1, X_2, \ldots, X_n$, from left to right, then

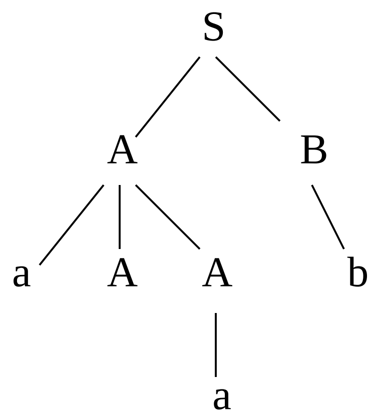
$$A \rightarrow X_1, X_2, \ldots, X_n$$

  must be a production in P
  – If a vertex has label ε, then that vertex is a leaf and the only child of its' parent

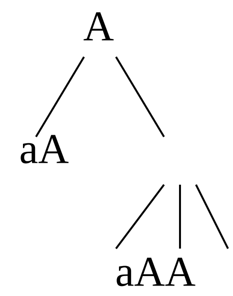
- More Generally, a derivation tree can be defined with any non-terminal as the root.

- **Example:**

S –> AB
A –> aAA
A –> aA
A –> a
B –> bB
B –> b



yield = aAab

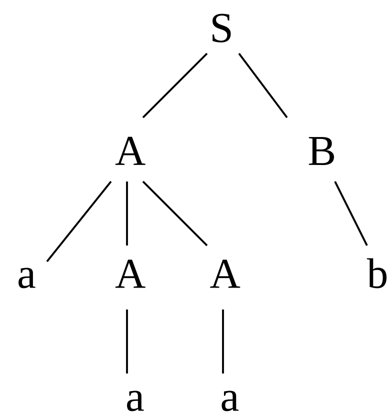


yield = aaAA

- **Notes:**
  - Root can be any non-terminal
  - Leaf nodes can be terminals or non-terminals
  - A derivation tree with root S shows the productions used to obtain a sentential form

- **Observation:** Every derivation corresponds to one derivation tree.

S => AB
=> aAAB
=> aaAB
=> aaaB
=> aaab



Rules:
S –> AB
A –> aAA
A –> aA
A –> a
B –> bB
B –> b

- **Observation:** Every derivation tree corresponds to one or more derivations.

leftmost:
S => AB
=> aAAB
=> aaAB
=> aaaB
=> aaab

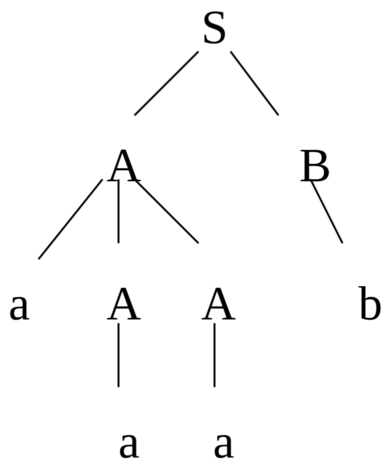
rightmost:
S => AB
=> Ab
=> aAAb
=>aAab
=> aaab

mixed:
S => AB
=> Ab
=> aAAb
=> aaAb
=> aaab

- **Definition:** A derivation is *leftmost (rightmost)* if at each step in the derivation a production is applied to the leftmost (rightmost) non-terminal in the sentential form.
  - The first derivation above is leftmost, second is rightmost, the third is neither.

- **Observation:** Every derivation tree corresponds to exactly one leftmost (and rightmost) derivation.

S => AB
=> aAAB
=> aaAB
=> aaaB
=> aaab

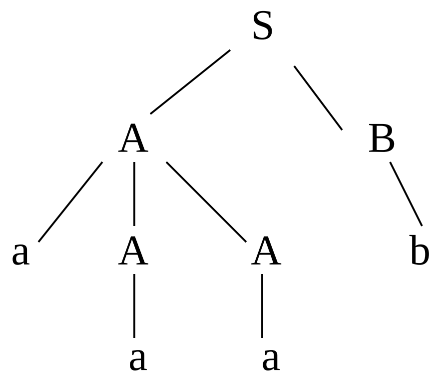


- **Observation:** Let G be a CFG. Then there may exist a string *x* in L(G) that has more than 1 leftmost (or rightmost) derivation. Such a string will also have more than 1 derivation tree.

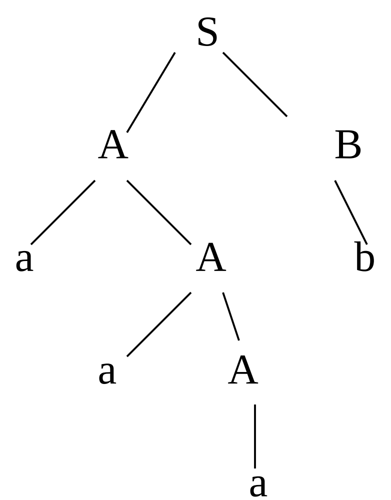
- **Example:** Consider the string *aaab* and the preceding grammar.

S –> AB
A –> aAA
A –> aA
A –> a
B –> bB
B –> b

S => AB
=> aAAB
=> aaAB
=> aaaB
=> aaab



S => AB
=> aAB
=> aaAB
=> aaaB
=> aaab



- The string has two left-most derivations, and therefore has two distinct parse trees.

- **Definition:** Let G be a CFG. Then G is said to be ambiguous if there exists an x in L(G) with >1 leftmost derivations. Equivalently, G is said to be ambiguous if there exists an x in L(G) with >1 parse trees, or >1 rightmost derivations.

- **Note:** Given a CFL L, there may be more than one CFG G with L = L(G). Some ambiguous and some not.

- **Definition:** Let L be a CFL. If every CFG G with L = L(G) is ambiguous, then L is inherently ambiguous.

- An ambiguous Grammar:

  E -> I $\sum$ ={0,…,9, +, *, (,)}

  E -> E + E

  E -> E * E

  E -> (E)

  I -> ε | 0 | 1 | … | 9

A leftmost derivation
E=>E*E
=>I*E
=>3*E+E
=>3*I+E
=>3*2+E
=>3*2+I
=>3*2+5

- A string: 3*2+5
- Two parse trees:

  * on top, & + on top

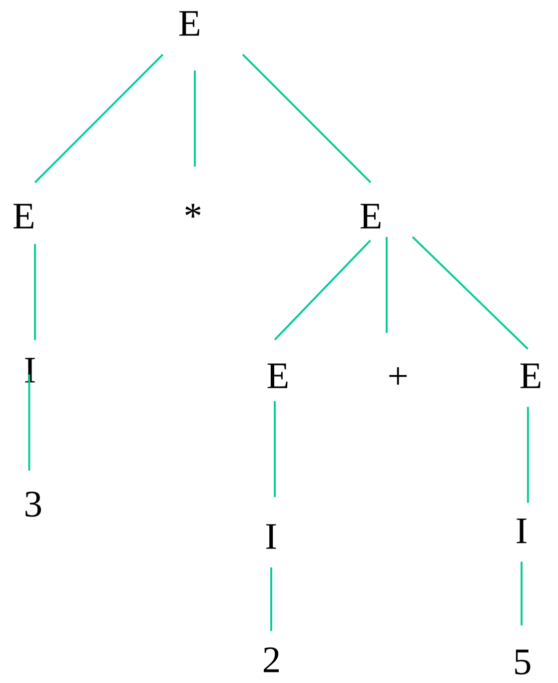
  & two left-most derivation:

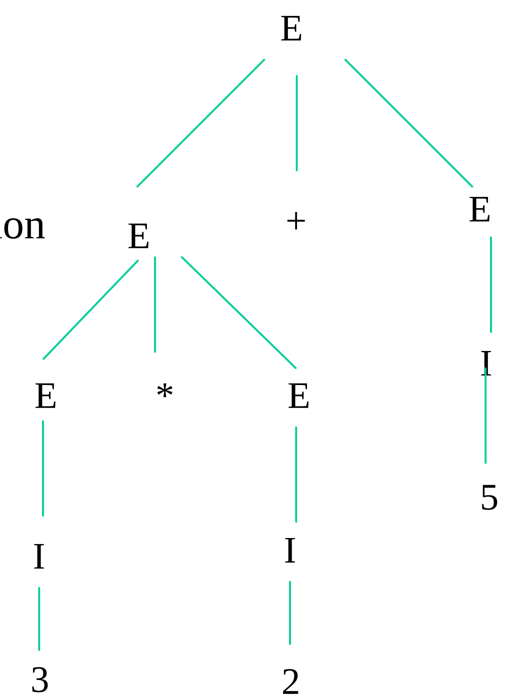
Another leftmost derivation
E=>E+E
=>E*E+E
=>I*E+E
=>3*E+E
=>3*I+E
=>3*2+I
=>3*2+5

E -> I
E -> E + E
E -> E * E
E -> (E)
I -> ε | 0 | 1 | … | 9

∑ ={0,…,9, +, *, (,)}

E=>E*E
=>I*E
=>3*E+E
=>3*I+E
=>3*2+E
=>3*2+I
=>3*2+5





Another leftmost derivation
E=>E+E
=>E*E+E
=>I*E+E
=>3*E+E
=>3*I+E
=>3*2+I
=>3*2+5

Ambiguous grammar:

E -> I

E -> E + E

E -> E * E

E -> (E)

I -> ε | 0 | 1 | … | 9

- Disambiguation of the Grammar:

$\sum$ ={0,,,9, +, *, (,)}

E -> T | E + T   *// This T is a non-terminal, do not confuse with* $\sum$

T -> F | T * F

F -> I | (E)

I -> ε | 0 | 1 | … | 9

- A string: 3*2+5
- Only one parse tree & one left-most derivation now:

    + on top: *TRY PARSING THE EXPRESSION NOW*

- A language may be *Inherently ambiguous*:

$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

- An ambiguous grammar:

S -> AB | C

A -> aAb | ab

B -> cBd | cd

C -> aCd | aDd

D -> bDc | bc

- Try the string: *aabbccdd*, two different derivation trees
- Grammar CANNOT be disambiguated for this (not showing the proof)

Rules:
S -> AB | C
A -> aAb | ab
B -> cBd | cd

C -> aCd | aDd
D -> bDc | bc

String *aabbccdd* belongs to two different parts of the language:
$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

Derivation 1 of *aabbccdd*:

S => AB
=> aAbB
=> aabbB
=> aabb cBd
=> aabbccdd

Derivation 2 of *aabbccdd*:

S => C
=> aCd
=> aaDdd
=> aa bDc dd
=> aabbccdd

- Potential algorithmic problems for context-free grammars:

  - Is L(G) empty?
  - Is L(G) finite?
  - Is L(G) infinite?
  - Is $L(G_1) = L(G_2)$?
  - Is G ambiguous?
  - Is L(G) inherently ambiguous?
  - Given ambiguous G, construct unambiguous G' such that L(G) = L(G')
  - Given G, is G "minimal?"