Context-free Languages

Foundations of Computing Science

Pallab Dasgupta
Professor,
Dept. of Computer Sc & Engg

Context-free Grammar (CFG)

A *context-free grammar (CFG)* is a 4-tuple (V, Σ, R, S), where

- V is a finite set called the *variables*
- Σ is a finite set, disjoint from V, called the *terminals*
- R is a finite set of *rules*, with each rule being a variable and a string of variables and terminals
- $S \in V$ is the *start variable*

Few Terminologies / Notions:

- If u, v and w are strings of variables and terminals, and $A \rightarrow w$ is a rule of the grammar, we say that uAv *yields* uwv, written as $uAv \Rightarrow uwv$
- u *derives* v, written as $u \overset{*}{\Rightarrow} v$, if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and

  $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
- The language generated by some *context-free grammar (CFG)*, G, is called the *context-free language (CFL)*, $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

Example of Context-free Grammars

Example-1:

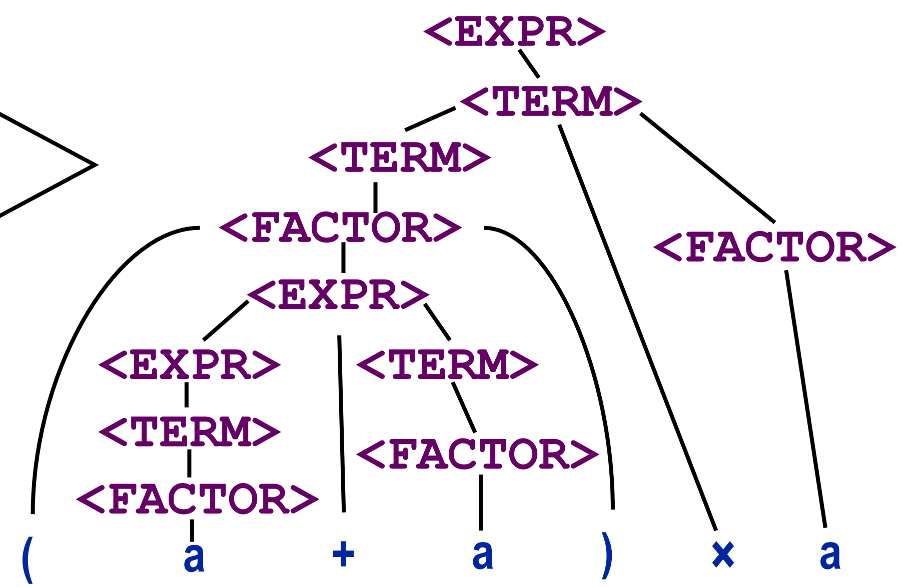
$G_1 = (\{S\}, \{(\,,\,)\}, R, S)$, where set of rules ($R$), is `S → (S) | SS | ε`

Here, $L(G_1)$ = all strings of properly nested parenthesis

Example-2:

$G_2 = (V, \Sigma, R,$ <EXPR> $)$, where V is {<EXPR>, <TERM>, <FACTOR>} and Σ is $\{a, +, \times, (,)\}$. The rules (R) are

```
<EXPR> → <EXPR> + <TERM> | <TERM>
<TERM> → <TERM> × < FACTOR > | < FACTOR >
<FACTOR> → ( <EXPR> ) | a
```



Parse Tree for the string (a + a) x a

Designing Context-free Grammars

Example:

Let $L(G_3)$ = equal number of 1s and 0s follow each other

= $\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$

The grammar for the language $\{0^n1^n \mid n \geq 0\}$ is $S_1 \rightarrow 0S_11 \mid \varepsilon$ and

the grammar for the language $\{1^n0^n \mid n \geq 0\}$ is $S_2 \rightarrow 1S_20 \mid \varepsilon$

Therefore, The complete grammar for the grammar $L(G_3)$ is

$$S \rightarrow S_1 \mid S_2 \;;\; S_1 \rightarrow 0S_11 \mid \varepsilon \;;\; S_2 \rightarrow 1S_20 \mid \varepsilon$$

Designing CFG for Regular Languages:

- Regular Languages → DFA
- If $\delta(q_i, a) = q_j$ is a transition in DFA; add the rule $R_i \rightarrow aR_j$ to CFG
- Add the rule $R_i \rightarrow \varepsilon$ to CFG if q_i is an accept state
- Make R_0 the start variable of CFG, where q_0 is the start state of DFA

Ambiguity

A string *w* is derived *ambiguously* in context-free grammar *G* iff it has two or more different leftmost derivations

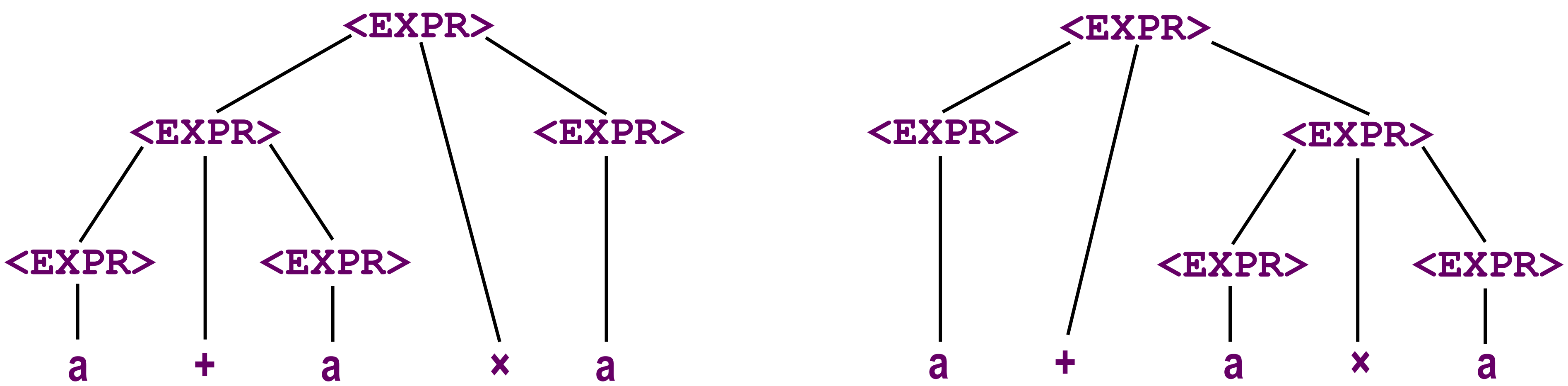
Grammar *G* is *ambiguous* iff it generates some string ambiguously

Example:

- Consider the grammar G_4:

<EXPR> → <EXPR> + <EXPR> | <EXPR> × <EXPR> | (<EXPR>) | a

- G_4 generates the string $a + a \times a$ ambiguously



Two Parse Trees for the Same String $a + a \times a$

Chomsky Normal Form

A context-free grammar is in *Chomsky Normal Form* iff every rule is of the form

$A \rightarrow BC$ and $A \rightarrow a$, where

- *a* is any *terminal*
- *A*, *B* and *C* are any *variables* – except that *B* and *C* may not be start variable
- In addition, we permit the rule $S \rightarrow \varepsilon$, where S is the *start variable*

Theorem:

- Any context-free language is generated by a context-free grammar in Chomsky normal form

Any CFG → Chomsky Normal Form

A Transformation Example:

$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \varepsilon$

↓

$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \varepsilon$

↓

$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB \mid a$
$A \rightarrow B \mid S \mid \varepsilon$
$B \rightarrow b \mid \varepsilon$

→

$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \rightarrow B \mid S \mid \varepsilon$
$B \rightarrow b$

↑

$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$
$A \rightarrow B \mid S$
$B \rightarrow b$

↑

$S_0 \rightarrow S \mid ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow B \mid S$
$B \rightarrow b$

→

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow B \mid S \mid b$
$B \rightarrow b$

↓

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow S \mid b \mid ASA \mid aB \mid a \mid SA \mid AS$
$B \rightarrow b$

↓

$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$
$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$
$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$
$A_1 \rightarrow SA$
$U \rightarrow a$
$B \rightarrow b$

Pushdown Automaton (PDA)

A *pushdown automata (PDA)* is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- Q, Σ, Γ and F are all finite sets
- Q is the *set of states*
- Σ is the *input alphabet*
- Γ is the *stack alphabet*
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow P(Q \times \Gamma_\varepsilon)$ is the transition relation
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of *accept states*

Examples of PDA

Let the PDA M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, \$\}$

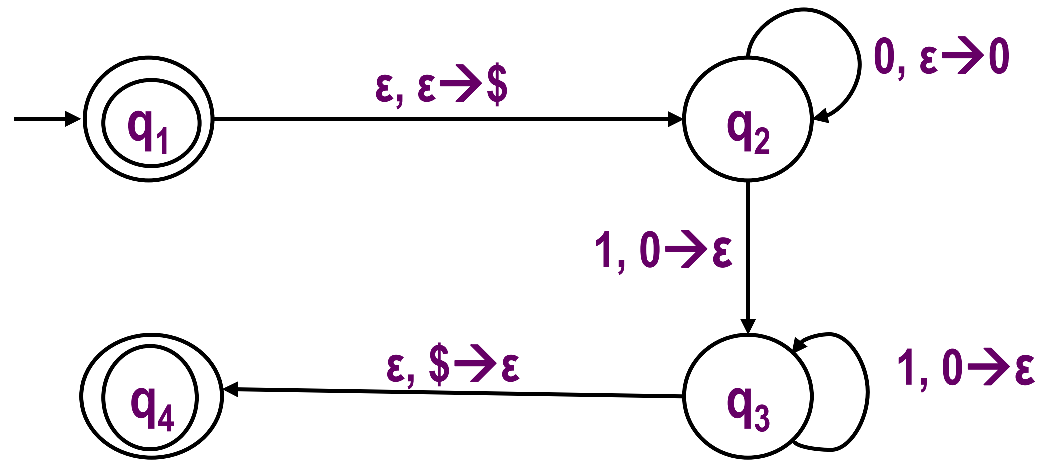
$F = \{q_1, q_4\}$

δ is given by

Input	0			1			ε		
Stack	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

PDA M_1 recognizes the language,

$L(M_1) = \{0^n 1^n \mid n \geq 0\}$



Acceptance/Recognition by PDA

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *accepts* input w if

- w can be written as $w = w_1 w_2 \ldots w_m$, where each $w_i \in \Sigma_\varepsilon$
- Sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exists (the strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation)
- The following *three* conditions are satisfied:
  - $r_0 = q_0$ and $s_0 = \varepsilon$
    [M starts out properly, in the start state and with an empty stack]
  - For $i = 0, 1, \ldots, m-1$; we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
    [M moves properly according to the state, stack & next input symbol]
  - $r_m \in F$ [an accept state occurs at the input end]

Theorem:

- A language is context-free if and only if some pushdown automaton recognizes it
  - Every regular language is context-free

Pumping Lemma for CFL

If *A* is a context-free language, then there is a number *p* (the pumping length), where, if *s* is any string in *A* of length at least *p*, then *s* may be divided into five pieces *s* = *uvxyz* satisfying the following conditions:

- For each $i \geq 0$, $uv^ixy^iz \in A$
- $|vy| > 0$, and
- $|vxy| \leq p$

Examples:

- The following languages (denoted by B, C, D) are not context-free:
  - $B = \{a^nb^nc^n \mid n \geq 0\}$
  - $C = \{a^ib^jc^k \mid 0 \leq i \leq j \leq k\}$
  - $D = \{ww \mid w \in \{0,1\}^*\}$