Context-Free Languages

03 February 2021 23:45

Content-Free languages form the next step from regular languages.

Regular :	DFA	Reg. expressions
Content-Free :	PDA	Content-Free Grammar.

CFL's are useful in parsing programming languages.

Content-Free Grammars

Consists of Substitution Rules or Productions.

$A \to 0A1$

$A \to B$

$B \to b$

Variables : Capital letters A, B.

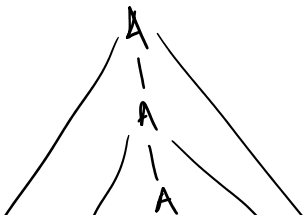
Terminals : Small letters, numbers, other symbols : 0, 1, b.

$A \to 0A1 \to 00A11 \to 000A111$

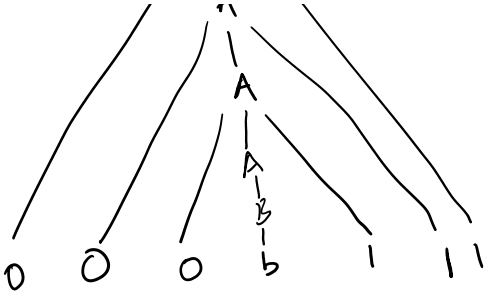
$\to 000B111 \to 000b111$

Start Variable : On the LHS of the first rule.

* Generate strings starting from start variable, and repeatedly replacing the variables in the resulting string using rules. Repeat till no variables remain.



CFG generating 000b111.



(( )) , (( ) (( ))) , (( ))) (( )

A programming language will need to check for properly nested parenthesis. $L = \{ w \in \{(,)\}^* \mid w \text{ is properly nested} \}$

$A = (^*)^*$. A is regular.

$A \cap L = \{(^n)^n \mid n \geq 0\}$ is not regular.

Hence L is not regular. We will see that L is context-free.

$B = 0^n 1^n$

$S \to 0S1$ OR $S \to A$

$S \to \varepsilon$ $A \to 0A1 \mid \varepsilon$

Even English sentences can follow such a grammar. We have words and we generate sentences in various ways from NOUN, VERB, PREPOSITION, ADJECTIVE etc. (read example in book - page 101).

Examples: (1) $S \to (A)$

$A \to \varepsilon \mid aA \mid ASA$

(2) $S \to \varepsilon \mid aSb$.

Def 2.2: A context-free grammar (CFG) is a 4-tuple (V, Σ, R, S), where.

Def 2.2.

4-tuple (V, Σ, R, S), where.

(1) V is a finite set, called variables.

(2) Σ is a finite set, disjoint from V, called terminals.

(3) R is a set of rules, each of the form

Variable $\longrightarrow$ String of variables and terminals.

(4) $S \in V$ is the start variable.

Yields: One step $uAv \Rightarrow uwv$ (using $A \to w$)

Derives: u derives v, $u \overset{*}{\Rightarrow} v$ if $u = v$ or

if $\exists$ a sequence $u_1, u_2 \ldots u_k$ such that

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v.$$

Language of G, $L(G)$ is given by

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

Why context-free?

Each variable is expanded not based on context.

Rules are not of the type $a V b \longrightarrow a \times b$

$b V a \longrightarrow b y a$

Rules do not depend on context. There are context-sensitive languages, though we won't

context-sensitive languages, though we won't see them in this course.

Example 2.3 : $G = \{ \{S\}, \{a, b\}, R, S \}$.

R : $S \to aSb \mid SS \mid \varepsilon$

If $a = ($ and $b =)$, this yields the set of all properly nested parentheses.

Designing CFG's

(1) If A and B are CFG's, we can construct their union by having a rule

$$S \to S_A \mid S_B.$$

Then rules of A and B.

(2) All regular languages can be expressed using a CFG.

One variable for each state of a DFA. R_0 corresponds to q_0 (start state).

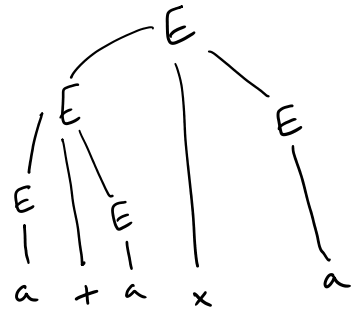
$\delta(q_i, a) = q_j$ corresponds to $R_i \to aR_j$

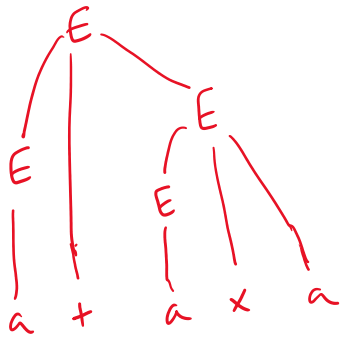
For $q_j \in F$, we add $R_j \to \varepsilon$.

Exercise: Verify that this grammar generates the same language as the DFA.

Ambiguity

G: $E \to E+E \mid E \times E \mid (E) \mid a$





When the parser parses 'a+a×a', it is not sure how this expression was derived. Does + or × have precedence?

In English: She called the man with an Iphone.

$E \to E+T \mid T$

$T \to T \times F \mid F$

$F \to (E) \mid a$

} This grammar generates the same language as above but is not ambiguous.

Def 2.7: A string w is derived ambiguously in a CFG G if it has two or more distinct leftmost derivations. Grammar G is ambiguous if it generates some strings ambiguously.

Chomsky Normal Form

simplified form for G.

Chomsky Normal

* Helpful to have a simplified form for G.
* Need to be efficient to check if $w \in L(G)$.
* Should not have loops (in derivation).
* Should not have empty derivations, or useless rules.
* Should have simple rules that are easy to check.

Def 2.8: A context-free grammar is in **Chomsky Normal Form** if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is a terminal, A, B, C are variables and B and C should not be the start variable.

Also, we allow $S \to \varepsilon$.