Theory of Computation: CS-202

Context Free Language

Outline

- Context Free Languages
- Context Free Grammar (CFG)
- Derivation Tree
- Simplification of CFG

Context Free Language

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

If we take $a = (\ \&\ b =)$

Then this language describes a simple kind of nested structure found in programming.

i.e. it will accept all the strings of type (()), ((()))

but not (()

⇒ This shows that some properties of programming languages requires something beyond regular languages.

⇒ Context free language is perhaps the most important aspect of formal language theory as it applies to programming language.

Context Free Grammar

A grammar $G = (V, T, S, P)$ is said to be context free if all production in P have the form

$$A \rightarrow x$$

where $A \rightarrow V$ & $x \rightarrow (V \cup T)^*$

A language L is said to be context free iff there is a context free grammar G such that $L = L(G)$

Note: RG is restricted in two ways:

- left side must be a single variable
- right side has a special form

To make grammar powerful, we must relax some of these restrictions so by retaining restriction of L.H.S. but permitting anything on the right, we can get context free grammar.

Example

- $G = (\{S\}, \{a, b\}, S, P)$

$$\text{P}: \quad S \to aSa$$
$$S \to bSb$$
$$S \to \lambda$$

is context free

$S \to aSa \to aaSaa \to aabSbaa \to aabbaa$

this makes it clear that

$$L(G) = \{ww^R : w \in (a, b)^*\}$$

Note:

Above language is CF

Note: Regular and linear grammar are Context Free.

but Context Free Grammar is not necessarily linear.

Derivation

- If there is a production $A \rightarrow \alpha$ then we say that A derives α and is denoted by $A \Rightarrow \alpha$

  $\alpha A \beta \Rightarrow \alpha \gamma \beta$ if $A \rightarrow \gamma$ is a production.

- If $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ then $\alpha_1 \Rightarrow \alpha_n$

- Given a grammar G and a string w of terminals in L(G), we can write $S \Rightarrow w$.

- If $S \Rightarrow \alpha$ where α is a string of terminals and non terminals of G then we say that α is a sentential form of G.

Derivation

Left-most and right-most derivations are two important concepts in formal language theory and the study of context-free grammars, often used in the context of parsing and generating strings according to a given grammar.

Left-Most Derivation:

- In a left-most derivation, one starts with the start symbol of a grammar and repeatedly replace the leftmost non-terminal symbol in the current string with a production rule's right-hand side until you derive the desired string.

1. S → ABC
2. A → a
3. B → b
4. C → c

S → ABC (Replace S with ABC)
→ aBC (Replace A with a)
→ abC (Replace B with b)
→ abc (Replace C with c)

Right-Most Derivation:

- In a right-most derivation, you start with the start symbol of a grammar and repeatedly replace the rightmost non-terminal symbol in the current string with a production rule's right-hand side until you derive the desired string.

S → ABC (Replace S with ABC)
→ ABc (Replace C with c)
→ Abc (Replace B with b)
→ abc (Replace A with a)

Note: An ambiguous grammar is one that produces more than one leftmost (rightmost) derivation of a sentence.

Leftmost and Rightmost Derivations

If CFG that is not linear, a derivation may involve sentential forms with more than one variables, so in such cases we have a choice in order by which variables are replaced.

$$G = (\{A, B, S\}, \{a, b\}, S, P)$$

P:

1. $S \to AB$
2. $A \to aaA$
3. $A \to \lambda$
4. $B \to Bb$
5. $B \to \lambda$

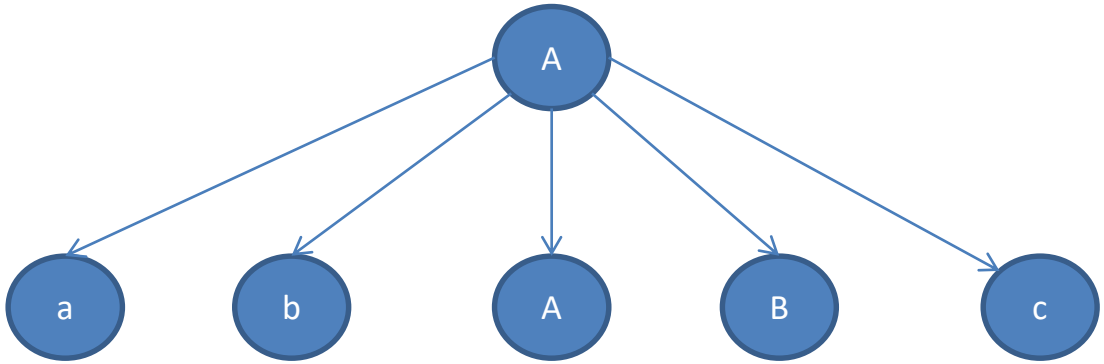
$$S \underset{1}{\to} AB \underset{2}{\to} aaAB \underset{3}{\to} aaB \underset{4}{\to} aaBb \underset{5}{\to} aab$$

$$S \underset{1}{\to} AB \underset{4}{\to} ABb \underset{5}{\to} Ab \underset{2}{\to} aaAb \underset{3}{\to} aab$$

Parse/syntax or Derivation tree

It is an ordered tree in which nodes are labeled with the left sides of productions and in which the children of a node represent its corresponding right side.

$$A \rightarrow abABc$$



Definition of Derivation Tree

Let $G = (V, T, S, P)$ be a CFG. An ordered tree is a derivation tree for G iff it has the following properties:

1. The root is labeled *S*.
2. Every leaf has a label from $T \cup \{\lambda\}$.
3. Every interior vertex (a vertex i.e. not leaf) has a leaf.
4. If a vertex has a label $A \in V$ & it's children are labeled (L to R) $a_1, a_2, \ldots, a_n$ then P must contain a production of form
$$A \rightarrow a_1 a_2 \ldots a_n$$
5. A leaf labeled λ has no sibling (no children)

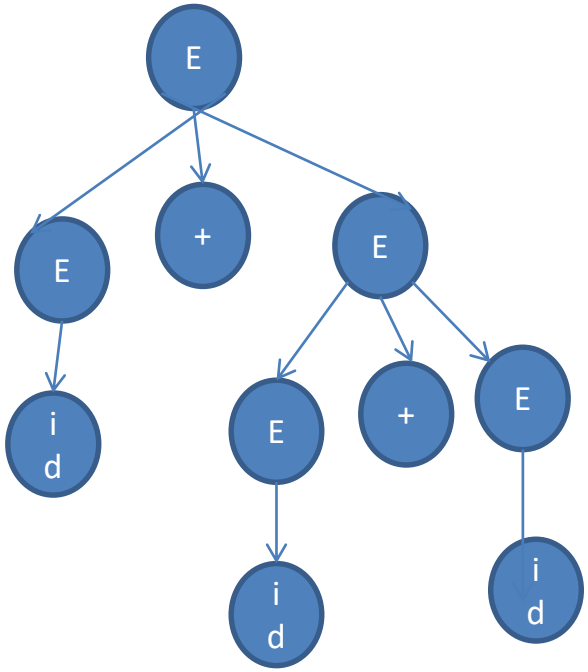
Note: A tree has properties 3, 4 & 5, but in which property 2 does not necessarily hold then property 2 can be replaced by 2(a).

2 (a) Every leaf has a label from $V \cup T \cup \{\lambda\}$ is said to be a partial derivation tree.

Example:

$$E \rightarrow E + E \mid E * E \mid id$$

string id + id + id



Ambiguity

- A Grammar can have more than one parse tree for a string

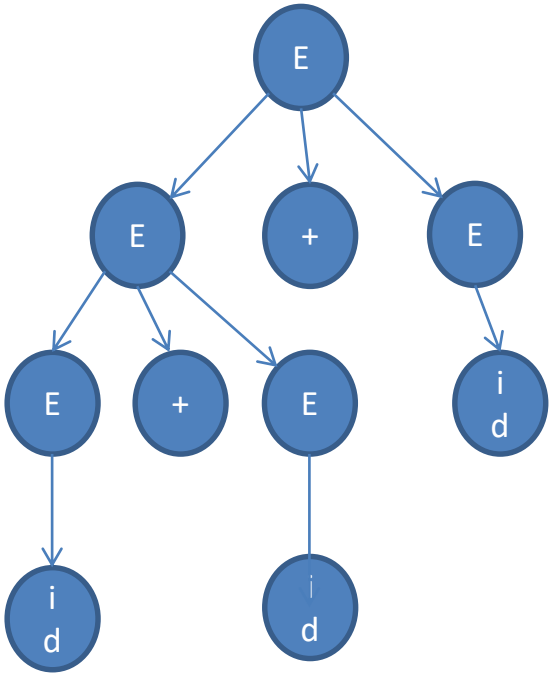
Consider the grammar G with productions

$E \rightarrow E + E \mid E * E \mid id$

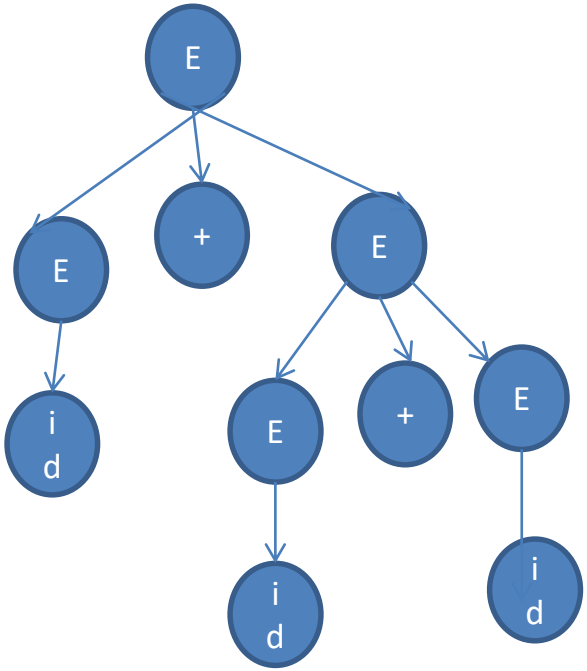
Consider grammar E → E + E| E * E |id

Derive “id+id+id”.

String “id+id+id” or 9+5+2 has two parse trees



(a)

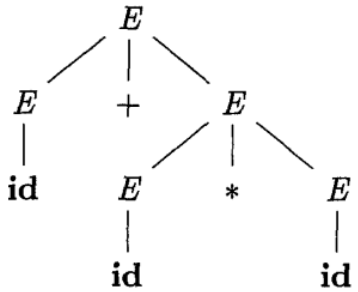


(b)

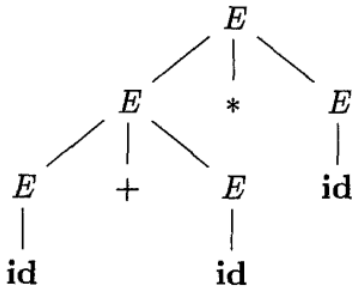
id+id+id has two parse trees

Consider grammar E → E + E| E * E |id

Derive “id+id*id”.



(a)



(b)

String “id+id*id” or 9+5*2 has two parse trees

Parsing

- Parsing: all production of form

$$S \rightarrow x$$

Find all x that can be derived from S in one step. If none of these mach useful the given string w, we go to the next round, in which we apply all applicable production to the left most variable of every x.

Example: Exhaustive search parsing-

$$S \rightarrow SS \mid aSb \mid bSa \mid \lambda$$

and the string w = aabb

1. $S \rightarrow SS$
2. $S \rightarrow aSb$
3. $S \rightarrow bSa$
4. $S \rightarrow \lambda$

1. $S \rightarrow SS \rightarrow SSS$
2. $S \rightarrow SS \rightarrow aSbS$
3. $S \rightarrow SS \rightarrow bSaS$
4. $S \rightarrow SS \rightarrow S$

Now for sentential form

1. $S \rightarrow aSb \rightarrow aSSb$
2. $S \rightarrow aSb \rightarrow aaSbb$
3. $S \rightarrow aSb \rightarrow abSab$
4. $S \rightarrow aSb \rightarrow ab$

For target string, $aabb$

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$$

Simplification of CFG

Simplification of CFG

A useful substitution rule-

Theorem: Let $G = (V, T, S, P)$ be a context free grammar suppose that P contains a production of the form

$$A \to x_1 B x_2$$

$$B \to y_1 \mid y_2 \mid \ldots \mid y_n$$

is the best of productions in P which have B as left side.

Let $\hat{G} = \left(V, T, S, \hat{P}\right)$ be the grammar in which $\hat{P}$ is constructed by deleting $A \to x_1 B x_2$ from P and adding to it

$$A \to x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid \ldots \mid x_1 y_n x_2$$

Then $L\left(\hat{G}\right) = L(G)$

Example:

Consider $G = (\{A, B\}, \{a, b, c\}, A, P)$

with production:

$A \rightarrow a \mid aaA \mid abBc$

$B \rightarrow abbA \mid b$

By using sub. rule

$\hat{G} = A \rightarrow a \mid aaA \mid ababbAc \mid abbc$

$B \rightarrow abbA \mid b$

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

Removing useless production

Let $G = \{V, T, S, P\}$ be a CFG. A variable $A \in V$ is said to be useful iff there is at least one $w \in L(G)$ s.t.

$$S \overset{*}{\rightarrow} xAy \overset{*}{\rightarrow} w$$

$$x, y \in (V \cup T)^*$$

A variable that is not useful is called useless.

Example:

$S \rightarrow A$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bA$

here variable B is useless & so the production $B \rightarrow bA$

Removing λ- production

Any production of a CFG of the form $A \to \lambda$ is called λ- production. Any variable A for which the derivation

$$A \overset{*}{\to} \lambda$$

is possible, is called null able.

Example:

1. $S \rightarrow ABaC$
2. $A \rightarrow BC$
3. $B \rightarrow b \mid \lambda$
4. $C \rightarrow D \mid \lambda$
5. $D \rightarrow \mathrm{d}$

Solution:

$S \rightarrow ABaC \mid AaC \mid Aa \mid BaC \mid aC \mid Ba \mid a \mid$**ABa**

$A \rightarrow B \mid C \mid BC$

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

Removing unit production

Any production of a CFG of the form

$$A \to B$$

where $A, B \in V$ is called a unit production. To remove unit production we use substitution rule.

Example:

$S \rightarrow Aa \mid B$

$B \rightarrow A \mid bb$

$A \rightarrow a \mid bc \mid B$

$\Rightarrow S \rightarrow Aa$

$\mathrm{B} \rightarrow bb$

$\mathrm{A} \rightarrow a \mid bc$

Then new rule,

$S \rightarrow Aa \mid bb \mid a \mid bc$

$B \rightarrow a \mid bc \mid bb$

$A \rightarrow a \mid bc \mid bb$

Here removal of unit product has made B and the associated production useless.

Suggested Books

P. Linz, An Introduction to Formal Language and Automata, Narosa Publisher.
K. L. P. Mishra, N. Chandrasekaran, Theory of Computer Science: Automata, Languages andComputation, PHI.