Context Free Languages

- Context Free Grammars
- Parsing Arithmetic Expression
- Removing λ-productions
- Normal forms

Context Free Grammar (CFG)

- Definition: A CFG, G, is a PSG G=(N, T, P, S) with productions of the form $A \rightarrow \beta$, $A \in N$, $\beta \in (N \cup T)^*$.
- CFGs are used in defining the syntax of programming languages and in parsing arithmetic expressions.
- A language generated from CFG is called Context Free Language (CFL).
- Ex.

a) $S \rightarrow aB$
$\quad B \rightarrow bA|b$
$\quad A \rightarrow a$

b) $S \rightarrow aB|A$
$\quad A \rightarrow aA|a|CBA$
$\quad B \rightarrow \lambda$
$\quad C \rightarrow c$

CFG: cont'd

- Let G be a CFG, then x Є L(G) iff S ➔ x in zero or more steps over G.
- x Є L(G) can as well be obtained from a **derivation tree** or **parse tree**. The root of the tree is S and x is the collection of leaves from left to right.
- **Left most derivation**: employs the reduction of the left most non-terminal
- **Right most derivation**: employs the reduction of the right most non-terminal

CFG: cont'd

- If a derivation of a string x has two different left most derivations, then the grammar is said to be **ambiguous**. Otherwise **unambiguous**.

  (i.e. a grammar is ambiguous if it can produce more than one parse tree for a particular sentence.
- Ex.

  1. G1 = (N, T, P, S) with productions:

     $S \rightarrow AB$

     $A \rightarrow aA|a$

     $B \rightarrow bB|b$

     let x = aaabbb

     a) find a left most and right most derivations for x

     b) draw the parse tree for x

CFG: cont'd

2. G2 = (N, T, P, S) with productions:

   $S \rightarrow SbS|ScS|a$

   let x = abaca Є L(G2)

   a) find a left most and right most derivations for x

   b) draw the parse tree for x

3. Is G1 ambiguous? Is G2?

Parsing Arithmetic Expression

- Consider the following grammar:

  $E \rightarrow T \mid E + T \mid E - T$

  $T \rightarrow F \mid T * F \mid T/F$

  $F \rightarrow a \mid b \mid c \mid (E)$

  Draw parse trees for

  a) a*b+c b) a+b*c c) (a+b)*c   d) a-b-c

Removing λ-productions

- Let G be a CFG and A → α, A Є N
  1. If α = λ, then A → α is a λ-production
  2. If α ≠ λ for all productions, then G is λ-free
  3. If A => λ in zero or more steps, then A is called **nullable** or λ-generating non-terminal
- Ex. Let G=(N, T, P, S) be a CFG with productions:

  S → ABaC

  A → B

  B → b| λ

  C → c| λ

  Find the non-terminals which are nullable.

Removing λ-productions: cont'd

- Let G=(N, T, P, S) be a CFG with λ-productions. Construct G'=(N, T, P', S), a λ-free CFG as follows:
  1. Put all non λ-productions of P in P'
  2. For all nullable non-terminals, put productions in P' by removing one or more nullable non-terminals on the right side productions.

     Thus, L(G) \ {λ} = L(G')

  Ex1: Construct G' for the previous example.

  Ex2: Construct G' for a grammar G with productions:

  $S \rightarrow aS \mid AB$

  $A \rightarrow \lambda$

  $B \rightarrow \lambda$

  $D \rightarrow b$

Removing λ-productions: cont'd

- <u>Theorem</u>: For any CFG G, there exists a CFG G' with λ-free productions such that L(G) \{λ} = L(G').
- A production A → α is called relevant/useful iff there exists a derivation of some x∈L(G) that uses the production. Otherwise, it's called irrelevant/useless.

  Ex. S → aSb | A | λ

  A → aA | λ

Normal Forms

- When the productions in a CFG G satisfy certain restrictions, G is said to be in a normal form.
- We'll see two normal forms: CNF and GNF

1. Chomsky Normal Form (CNF)

- Let G=(N, T, P, S) be a CFG and A → α be a production of G.
    1. If α = B, B in N, then A → α is called a **Unit** production
    2. If |α|>1 and there exists a terminal substring of α, then A → α is called a **Secondary** production
    3. If α contains more than two non-terminals, then A → α is called a **Tertiary** production

Chomsky Normal Form (CNF)

- Definition: Let G = (N, T, P, S) be a λ-free CFG, then G is said to be in CNF if all its productions are of the form:

  A → BC where A, B, C ∈ N

  OR

  A → a, a ∈ T, A ∈ N

Ex. G with productions

S → AB

A → a

B → b

CNF: cont'd

- Theorem: Any λ-free CFL can be generated by a CFG in CNF.

  proof:

  Let G = (N, T, P, S) be a λ-free grammar such that L = L(G). Construct a grammar G which is in CNF.

  Steps:

1. Replace all Unit productions as follows:
   - For any A Є N on the LHS of the Unit production, denote U(A) and non-Unit productions of A by N(A)
   - For each A Є N and U(A) ≠ Ø replace U(A) by

     {A → α | A=>B in one or more steps, and B→α Є N(B)}

CNF: cont'd

2. Replace all Secondary productions as follows:

   For any a Є T, a substring of a Secondary production, replace a by A_a where A_a is a new non-terminal and $A_a \rightarrow a$.

3. Replace all Tertiary productions as follows:

   If $A \rightarrow B_1B_2 \ldots B_m$, m>2, then replace the production by:

   $A \rightarrow B_1B_1'$

   $B_1' \rightarrow B_2B_2'$

   $B_2' \rightarrow B_3B_3'$

   …

   $B_{m-2}' \rightarrow B_{m-1}B_m$, where $B_1', \ldots, B_{m-2}'$ are all unique new non-terminals that do not appear in any other production.

CNF: cont'd

- Ex. Convert the following grammars to CNF

  1. Let G be a CFG with productions:

     $S \rightarrow A \mid ABA$

     $A \rightarrow aA \mid a \mid B$

     $B \rightarrow bB \mid b$

  2. Let G be a CFG with productions:

     $S \rightarrow aAD$

     $A \rightarrow aB \mid bAB$

     $B \rightarrow b$

     $D \rightarrow d$

Identification of non-CFLs

- Pumping lemma for CFLs
  (Reading assignment)

Greibach Normal Form (GNF)

- Let G=(N, T, P, S) be a λ-free CFG, then if all the productions of G are of the form

  $A \rightarrow a\alpha,\ A \in N,\ a \in T,\ \alpha \in N^*$

  then G is said to be in GNF

- <u>Theorem G1</u>: If $A \rightarrow \alpha_1 B \alpha_2$ is a production in a CFG G and $B \rightarrow \beta_1|\beta_2|\beta_3|\ldots|\beta_k$ are all productions with B on the LHS, then

  $A \rightarrow \alpha_1 B\ \alpha_2$ can be replaced by

  $A \rightarrow \alpha_1\beta_1\alpha_2|\ \alpha_1\beta_2\alpha_2|\ \ldots\ |\ \alpha_1\beta_k\alpha_2$

  without affecting L(G).

GNF: cont'd

- Theorem G2: If in a CFG there is a production $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1|\beta_2|\ldots|\beta_m$, such a production is called **left recursive**, and $A \rightarrow \beta_1|\beta_2|\beta_3|\ldots|\beta_m$ are the remaining productions with A on the LHS.

  Then an equivalent grammar can be constructed by introducing a new non-terminal, A', and replacing all these productions by:

  $A \rightarrow \beta_1|\beta_2|\beta_3|\ldots|\beta_m| \beta_1A'| \beta_2A'|\ldots| \beta_mA'$

  $A' \rightarrow \alpha_1| \alpha_2|\ldots| \alpha_n| \alpha_1A'| \alpha_2A'|\ldots| \alpha_nA'$

GNF: cont'd

- Theorem GNF: Any λ-free CFG G can be converted into a grammar in GNF.

Proof:

Let G be a λ-free CFG. To convert G into GNF use the steps below:

1. Convert G into CNF, G'
2. Rename the non-terminals in G' as

   $A_1, A_2, \ldots, A_m$ $(m>=1)$
3. Convert all the productions into

   $A_i \rightarrow a\alpha$ or $A_i \rightarrow A_j\alpha$ with $j > i$

To convert to the form $A_i \rightarrow A_j\alpha$ with $j > i$, do the following:

Substitute for A_j according to Theorem G1.

If there exist left recursive productions with A_i on the LHS, then introduce a new non-terminal A_i' and apply Theorem G2.

GNF: cont'd

4. After the 3[rd] step, the productions will be of the form
   i. $A_i \rightarrow A_j\alpha$, $j > i$, $\alpha \in (N \cup N')^*$ where N' stands for the new non-terminals A_i introduced.
   ii. $A_i \rightarrow a\alpha$, $a \in T$, $\alpha \in (N \cup N')^*$ or
   iii. $A_i' \rightarrow x\alpha$, $x \in (N \cup T)$, $\alpha \in (N \cup N')^*$

Replace (i) by using Theorem G1

(iii) by using Theorem G2

GNF: cont'd

Ex. Convert to GNF

1. Let G be with productions

   $S \rightarrow AB$

   $A \rightarrow BS \mid b$

   $B \rightarrow SA \mid a$

2. Let G be with productions

   $A_1 \rightarrow A_2A_2 \mid a$

   $A_2 \rightarrow A_1A_2 \mid b$

Closure Properties of CFGs

- <u>Theorem</u>: CFGs are closed under:
  a) Union
  b) Concatenation
  c) Kleen star(*)