Chapter 1

Properties of Context Free Languages

1.1 Chomsky Normal Form

- Every CFL can be generated by a CFG in which all productions are of the form $A \rightarrow BC$ or $A \rightarrow a$, where $A$, $B$ and $C$ are variables and $a$ is terminal. This form is called Chomsky Normal Form.

- Eliminating Useless Symbols
  - A symbol $X$ in a CFG $G = \{V, T, P, S\}$ is called *useful* if there exist a derivation of a terminal string from $S$ where $X$ appears somewhere, else it is called *useless*.
  - A symbol $X$ is called *generating* if some terminal string can be derived from $X$.
  - A symbol $X$ is called *reachable* if it can be reached from $S$.
  - To remove non-generating symbols we find them first and then remove all the productions in which they occur, same thing we do with non-reachable.
  - To remove both we should first remove non-generating then non-reachable. Clearly if we follow this order there will be no non-reachable symbol but we can also prove that there will be no non-generating symbol also. On the other hand if we first remove non-reachable symbols and then non-generating symbols then it possible to create non-reachable symbols after second step.
  - If a symbol is useful then it is both generating and reachable.
  - Converse of above statement is not true. For e.g. in CFG $S \rightarrow ABC, B \rightarrow b$
    
    $B$ is both reachable and generating but still not useful.

- Claim: If in a CFG $G$ we remove non-generating and then non-reachable symbols then there will be no useless symbol. Moreover, let $G'$ be the CFG we get this way then $L(G) = L(G')$.

- Finding non-generating symbols: To find non-generating symbols we will actually find the generating symbols using induction. Basically, first find straightforward generating symbols then go to each production and see if body of that production is only made up of generating symbols. Keep iterating, until we don’t get a new generating symbol in some iteration.

- Finding non-reachable symbols: Again instead of finding non-reachable symbols we’ll actually find the reachable ones. And again we will use induction for that, in a CFG $G$ clearly $S$ is reachable, and if $A$ is a symbol which is reachable then all symbols present in production with $A$ as head are also reachable.

- Eliminating $\epsilon$-Productions
A production of type $A \rightarrow \epsilon$ is called an $\epsilon$-production.

Claim: If a language $L$ has a CFG, then $L - \{\epsilon\}$ has a CFG without $\epsilon$-productions.

To remove $\epsilon$-productions, we can again use the inductive technique.

• Eliminating Unit Productions

A unit production is the production of the form $A \rightarrow B$, where $A$ and $B$ are both variables.

We can inductively find all the pairs $(A, B)$ such that $A \Rightarrow B$ using only unit productions. Let us call such pairs unit pair.

Algorithm for finding unit pairs:

* Basis: Create the initial set of unit pairs by first finding straightforward unit pairs and pairs of type $(A, A)$

* Induction: If $(A, B)$ and $(B, C)$ is a unit pair then $(A, C)$ is also a unit pair.

1.2 Closure properties

1.3 Decision properties

• Testing the emptiness of a CFG $G$ is decidable.

• Testing the membership of a string $w$ in a CFG $G$ is also decidable.

• The following problems are undecidable?
  1. Is a given $G$ ambiguous?
  2. Is a given CFL inherently ambiguous?
  3. Is the intersection of two CFL’s empty?
  4. Are two CFL’s same?

• Exercises:
  1. Is $L(G)$ finite, for a given CFG $G$
  2. Does $L(G)$ contain at least 100 strings?
  3. Given a CFG $G$ and one of its variables $A$, is there any sentential form in which $A$ is the first symbol.
  4. Modify the CYK algorithm so that it can report the distinct parse trees for the given input.