Context Free Grammars- II

Recall definitions of CFG, CFL, derivations, derivation trees.

Proof of Chomsky Normal Form: Put in a new variable \( C_a \) for each terminal \( a \). If rhs (right hand side) of a production has only one letter, a terminal, leave it alone. If it has more than one letter, then replace each terminal \( a \) on the rhs by \( C_a \). Add all productions of the form \( C_a \rightarrow a \).

Now each production \( A \rightarrow X_1X_2\ldots X_l \), (with \( l \geq 3 \)), we introduce new variables \( D_1, D_2, \ldots D_{l-2} \) and replace this production by the set of productions:

\[
A \rightarrow X_1D_1; \quad D_1 \rightarrow X_2D_2; \quad D_2 \rightarrow X_3D_3; \ldots; \quad D_{l-3} \rightarrow X_{l-2}D_{l-2}; \quad D_{l-2} \rightarrow X_{l-1}X_l.
\]

Closure Properties: The set of CFL’s is closed under union, concatenation and Kleene star.

Proof (For Union alone): If \( G_1, G_2 \) are two CFG’s, a CFG for \( L(G_1) \cup L(G_2) \) is defined by first renaming all variables in \( G_2 \) to be different from variables of \( L_1 \), then taking all the productions of both grammars plus the two productions \( S \rightarrow S_1S_2 \), where \( S_1, S_2 \) are the start symbols of \( G_1, G_2 \) resp.

Claim The set of CFL’s is not closed under intersection.

Example It is known that \( \{a^ib^ic| \} \) is not be a CFL.

Parsing Algorithm for CFL’s

The input is string \( x \) of length \( n \). We first put the CFG in Chomsky Normal Form.

We use Dynamic Programming. We denote by \( x(i, j) \) the substring of \( x \) starting from its \( i \) th letter to its \( j \) th letter. We will compute the quantities

\[
S_{ij} = \{ A \in V : A \Rightarrow^* x(i, j) \}.
\]

To do this, we branch on the first production used in a derivation of \( x(i, j) \). This leads to a Dynamic Prog recursion with the observation that \( A \) is in \( S_{ij} \) if there is a production of the form \( A \rightarrow BC \), where for some \( k, i \leq k \leq j - 1, B \in S_k \) and \( C \in S_{k+1,j} \), both strictly smaller problems. It was very important for this that we first got rid of \( \epsilon \) and unit productions. (Think about this point again.)

Emptiness and finiteness problems for CFL’s are decidable. But cofiniteness (whether a CFL contains all but a finite number of strings) and equivalence of two CFL’s are undecidable.
Proof for Emptiness: Assume the CFG is in CNF with set of productions $P$. For variable $A$, define $f(A)$ to be the minimum depth of a derivation tree for $A \Rightarrow^* \alpha$ for some terminal string $\alpha$; $f(A)$ is thought of as $\infty$ if there is no such derivation. Let

$$F_i = \{ A : f(A) = i \}.$$ 

We will show how to find all the $F_i$ first. $F_1 = \{ A : A \rightarrow a \text{ is a production} \}.$

We claim that $F_{i+1}$ is precisely the set of $A$ for which, (i) $A \notin F_1 \cup F_2 \cup \ldots F_{i-1}$ and (ii) there is a production $A \rightarrow BC$ such that $B \in F_i$ and $C \in F_1 \cup F_2 \cup \ldots F_i$ or $C \in F_i$ and $B \in F_1 \cup F_2 \cup \ldots F_i$. (Why?)

Thus $F_{i+1}$ may be found from $F_i$.

This implies that if for some $i$, $F_i$ is empty, then $F_{i+1}, F_{i+2}, \ldots$ are all empty. Now it is easy to see that we can construct $F_1 \cup F_2 \cup F_3 \ldots$ and therefore test emptiness.