CS 360
Naomi Nishimura

Context-free languages

1 Closure properties

In Theorem 7.24 of the textbook, several closure properties for the class of context-free languages are proved using substitution and Theorem 7.23. Here we give the constructions explicitly, assuming that for two context-free languages L_1 and L_2 we have associated grammars $G_1 = (V_1, T_1, P_1, S_1)$ and $G_2 = (V_2, T_2, P_2, S_2)$. In what follows we need to ensure that $V_1 \cap V_2 = \emptyset$, which can be accomplished by renaming of variables.

To show that the class of context-free languages is closed under union, we show how we can form from G_1 and G_2 a grammar $G = (V, T, P, S)$ such that $L(G) = L_1 \cup L_2$. We set $V = V_1 \cup V_2 \cup \{S\}$, $T = T_1 \cup T_2$, and $P = P_1 \cup P_2 \cup \{S \to S_1\} \cup \{S \to S_2\}$.

The proof for closure under concatenation is similar, where $L(G) = L_1L_2$ and G is defined by $V = V_1 \cup V_2 \cup \{S\}$, $T = T_1 \cup T_2$, and $P = P_1 \cup P_2 \cup \{S \to S_1S_2\}$.

For Kleene star, we define G so that $L(G) = L_1^*$ by setting $V = V_1 \cup \{S\}$, $T = T_1$, and $P = P_1 \cup \{S \to S_1S\} \cup \{S \to \epsilon\}$.

2 Regular implies CFL

We can show that if L is regular, then L is CFL either by constructing a PDA that mimics an automaton D such that $L(D) = L$ (essentially ignoring the stack) or by constructing a context-free grammar G such that $L(G) = L$ using a regular expression α such that $L(\alpha) = L$.

To construct the grammar, we can use induction on the number of operations in α. If $\alpha = \emptyset$, we create a grammar without any rules. For $\alpha = a \in \Sigma$, we construct a grammar with the rule $S \to a$, and for $\alpha = \epsilon$ with the rule $S \to \epsilon$.

We use as our induction hypothesis the claim that if β has fewer than k operations, there exists grammar G_β such that $L(G_\beta) = L(\beta)$. We now consider α with k operations, and in particular the last operation used to construct α. We can then decompose α as $\alpha = \alpha_1 + \alpha_2$, or $\alpha = \alpha_1\alpha_2$, or $\alpha = \alpha_1^*$. Since each of α_1 and α_2 (if it exists) has fewer than k operations, we can use the induction hypothesis to form grammars G_1 and G_2, $L(G_1) = L(\alpha_1)$ and $L(G_2) = L(\alpha_2)$. We can then use the closure property constructions from the previous section to complete G.

3 Decision problems for CFL's

Since we are not concerned with the details of the time complexity of the algorithms for these problems, we can get by with a simpler presentation than that given in the textbook.

To determine if a string w is in a CFL L, we first observe that we can find a grammar G in Chomsky Normal Form such that $L(G) = L - \{\epsilon\}$. If $w = \epsilon$, we can check membership in L during the conversion algorithm. Otherwise, we know that if there is a derivation of w in G, the length of the derivation will be $2|w| - 1$. Since the number of rules is finite, the number of derivations of length $2|w| - 1$ is finite, and hence membership can be tested by trying them all (if w is found the answer is "yes" and if w is not found the answer is "no").

To determine if a CFL L is empty, we can either test for reachability (see the text) or use the pumping lemma for context-free languages, in a manner similar to the algorithm for the emptiness problem for regular languages. We can check for membership of ϵ in the conversion to a grammar G in CNF, returning "no" if $\epsilon \in L$. Otherwise, for $n = 2^{p+1}$, where $p + 1$ is the number of variables in G, we check to see if there is any string of length at most n in the language using the algorithm found in the previous paragraph. Since the total number of strings to check is finite, the algorithm executes in a finite amount of time.

To determine if a CFL L is finite, again we can use the pumping lemma in the manner used to check finiteness of regular languages. In this case we determine if there is any string w where $n \leq |w| < 2n$ is in the language.