Removal of Direct Left Recursion

…on of an arbitrary context-free grammar, rule applications can generate terminal … any position and in any order in a derivation. For example, derivations in … G_1 generate terminals to the right of the variable, while derivations in G_2 generate … both sides.

$$G_1: S \to Aa \qquad\qquad G_2: S \to aAb$$
$$A \to Aa \mid b \qquad\qquad A \to aAb \mid \lambda$$

…ch normal form adds structure to the generation of the terminals in a derivation. … built in a left-to-right manner with one terminal added on each rule application. …on $S \stackrel{+}{\Rightarrow} uAv$, where A is the leftmost variable, the string u is called the *terminal* … sentential form. Our objective is to construct a grammar in which the terminal …eases with each rule application.

… grammar G_1 provides an example of rules that do the exact opposite of what is … The variable A remains as the leftmost symbol until the derivation terminates with … of the rule $A \to b$. Consider the derivation of the string *baaa*

$$\begin{aligned} S &\Rightarrow Aa \\ &\Rightarrow Aaa \\ &\Rightarrow Aaaa \\ &\Rightarrow baaa. \end{aligned}$$

…s of the left-recursive rule $A \to Aa$ generate a string of a's but do not increase … of the terminal prefix. A derivation of this form is called ***directly left-recursive***. … grows only when the non–left-recursive rule is applied.

… important component in the transformation to Greibach normal form is the ability … left-recursive rules from a grammar. The technique for replacing left-recursive …strated by the following examples.

a) $A \to Aa \mid b$ b) $A \to Aa \mid Ab \mid b \mid c$ c) $A \to AB \mid BA \mid a$

$B \to b \mid c$

… generated by these rules are ba^*, $(b \cup c)(a \cup b)^*$, and $(b \cup c)^*a(b \cup c)^*$, respec… The left recursion builds a string to the right of the recursive variable. The recursive … is terminated by an A rule that is not left-recursive. To build the string in a left-… manner, the nonrecursive rule is applied first and the remainder of the string is … by right recursion. The following rules generate the same strings as the previous … without using direct left recursion.

a) $A \to bZ \mid b$ b) $A \to bZ \mid cZ \mid b \mid$ c) $A \to BAZ \mid aZ \mid BA \mid a$

$Z \to aZ \mid a$ $\quad Z \to aZ \mid bZ \mid a \mid b$ $\quad Z \to BZ \mid B$

$B \to b \mid c$

The rules in (a) generate ba^* with left recursion replaced by right recursion. With these rule the derivation of $baaa$ increases the length of the terminal prefix with each rule applicatio

$$\begin{aligned} A &\Rightarrow bZ \\ &\Rightarrow baZ \\ &\Rightarrow baaZ \\ &\Rightarrow baaa \end{aligned}$$

The removal of the direct left recursion requires the addition of a new variable to t grammar. This variable introduces a set of right-recursive rules. Direct right recursion caus the recursive variable to occur as the rightmost symbol in the derived string.

To remove direct left recursion, the A rules are divided into two categories: the le recursive rules

$$A \rightarrow Au_1 \mid Au_2 \mid \dots \mid Au_j$$

and the rules

$$A \rightarrow v_1 \mid v_2 \mid \dots \mid v_k,$$

in which the first symbol of each v_i is not A. A leftmost derivation from these rules cons of applications of left-recursive rules followed by the application of a rule $A \rightarrow v_i$, wh ends the direct left recursion. Using the technique illustrated in the previous examples, construct new rules that initially generate v_i and then produce the remainder of the str using right recursion.

The A rules

$$A \rightarrow v_1 \mid \dots \mid v_k \mid v_1Z \mid \dots \mid v_kZ$$

initially place one of the v_i's on the left-hand side of the derived string. If the string cont a sequence of u_i's, they are generated by the Z rules

$$Z \rightarrow u_1Z \mid \dots \mid u_jZ \mid u_1 \mid \dots \mid u_j$$

using right recursion.

Example 4.7.1

A set of rules is constructed to generate the same strings as

$$A \rightarrow Aa \mid Aab \mid bb \mid b$$

without using direct left recursion. These rules generate $(b \cup bb)(a \cup ab)^*$. The direc recursion in derivations using the original rules is terminated by applying $A \rightarrow b$ or $A \rightarrow$ To build these strings in a left-to-right manner, we use the A rules

$$A \rightarrow bb \mid b \mid bbZ \mid bZ$$

to generate the leftmost symbols of the string. The Z rules generate $(a \cup ab)^+$ usin right-recursive rules

$$Z \rightarrow aZ \mid abZ \mid a \mid ab.$$