PART III

Context-Free Languages—Solutions

Part III
Context-Free Languages—Solutions

14 Phrase-Structure Grammars

73. Given this formal definition of derivation, what would be the effect of extending P to $(V \cup \Sigma)^* \times (V \cup \Sigma)^*$, of allowing ε on the left-hand side of productions?

    (Solution)

    It P contains any production $\varepsilon \longrightarrow \beta$ then β can be inserted anywhere—any string $\alpha = \alpha_l\alpha_r$ directly derives $\alpha_l\beta\alpha_r$ for any partitioning of α into α_l and α_r. Potentially more significantly, there will be no string that cannot be rewritten—productions for ε can always apply. Thus we would lose the notion of a *terminating* derivation, one in which the last sentential form cannot be rewritten.

15 Context-Free Grammars

Lemma 2 *Suppose $G = \langle V, \Sigma, S, P\rangle$ is a CFG. Then for all $X \in V$ and $w \in \Sigma^*$, if t is a derivation tree for w from X in G there is both a left-most derivation of w from X in G and a right-most derivation of w from X in G corresponding to t and these derivations are unique.*

74. Prove the lemma.

    (Solution)

    From the previous lemma we have that if there is such a derivation tree t then there is a derivation of w from X in G. Intuitively, it seems clear that there should be a unique left-most derivation corresponding to t, for suppose, for contradiction, that there were two distinct left-most derivations:

$$\alpha_1 \Longrightarrow \cdots \Longrightarrow \alpha_{k-1} \Longrightarrow \alpha_k \Longrightarrow \cdots$$
$$\alpha_1 \Longrightarrow \cdots \Longrightarrow \alpha_{k-1} \Longrightarrow \alpha'_k \Longrightarrow \cdots$$

where α_k and α'_k are the first sentential forms on which they disagree. Since both are left-most derivations, they must, at this step, both rewrite the left-most non-terminal of α_{k-1}, which is unique. Since they both must correspond to t, they both must rewrite that non-terminal to the same string, that is, they must both apply the same production. It follows that $\alpha_k = \alpha'_k$ contradicting our choice of these as the first forms on which the derivations differ.

We can also prove this by induction on the structure of t.

(BASIS:)

Suppose $d(t) = 1$. Then $X \stackrel{1}{\Longrightarrow} w$ is a derivation for w from X in G corresponding to t, it is the only such derivation, and it is simultaneously a left-most and a right-most derivation.

(IND:)

Suppose $d(t) = k + 1$ and that the lemma holds for all derivation trees in G with depth no greater than k. By the definition of derivation trees,

$$w = w_1 v_1 w_2 \cdots w_{n-1} v_n w_n$$

and t consists of a depth one tree for $w_1 Y_1 w_2 \cdots w_{n-1} Y_n w_n$ from X in G with trees (of depth no greater than k) for v_1 from Y_i in G attached at the 'Y_i's. By the induction hypothesis there are unique left-most derivations corresponding to each of these smaller trees. Now, any left-most derivation corresponding to t must apply each of the productions in the derivation from Y_i before any of the productions in the derivation from Y_{i+1}. Moreover, since the left-most derivation from Y_i is unique, it must apply these in exactly the same order as they are applied in that derivation. Consequently, the derivation that first applies

$$X \longrightarrow w_1 Y_1 w_2 \cdots w_{n-1} Y_n w_n$$

and proceeds to apply each of the productions of the left-most derivations from the Y_i in order is the unique left-most derivation of $w_1 v_1 w_2 \cdots w_{n-1} v_n w_n$ from X in G corresponding to t.

The proofs for right-most derivations are similar.

15.1 Determining the Language Generated by CFG

Non-inductive Proof that $L \subseteq L(G)$

75. Prove that $L(G_{ab}) = \{a^i b^i \mid i \geq 0\}$ by proving the remaining direction—that $L(G_{ab}) \subseteq \{a^i b^i \mid i \geq 0\}$.

    (Solution)

    To show that $S \underset{G_{ab}}{\overset{*}{\Longrightarrow}} w \in \Sigma^*$ implies that $w \in \{a^i b^i \mid i \geq 0\}$, by induction on the length of the derivation:

    (BASIS)

    $S \underset{G_{ab}}{\Longrightarrow} w$ implies $w = \varepsilon$ which is just $a^0 b^0$.

    (IND:)

    Suppose $S \overset{n+1}{\Longrightarrow} G_{ab} w \in \Sigma^*$ and if $S \underset{G_{ab}}{\overset{k}{\Longrightarrow}} v \in \Sigma^*$ where $k \leq n$ then $v \in \{a^i b^i \mid i \geq 0\}$. Then, from the grammar

    $$S \underset{G_{ab}}{\Longrightarrow} aSb \underset{G_{ab}}{\overset{*}{\Longrightarrow}} avb = w$$

    where $S \underset{G_{ab}}{\overset{n}{\Longrightarrow}} v$. Thus, by IH, $v \in \{a^i b^i \mid i \geq 0\}$ and, consequently

    $$w \in \{a^i b^i \mid i \geq 1\} \subseteq \{a^i b^i \mid i \geq 0\}.$$

16 Some Closure Properties of the class CFL

76. Why do we need to assume that the sets of non-terminals are disjoint?

    (Solution)

    Otherwise there could be derivations in which productions of G_1 and productions of G_2 were both used, potentially deriving strings not derivable in either.

77. Prove the claim.

($\Rightarrow$)

Suppose $w \in L(G)$. Then $S \overset{*}{\underset{G}{\Longrightarrow}} w$ which, from the construction of G, implies that either $S \underset{G}{\Longrightarrow} S_1 \overset{*}{\underset{G}{\Longrightarrow}} w$ or $S \underset{G}{\Longrightarrow} S_2 \overset{*}{\underset{G}{\Longrightarrow}} w$. Since the variables of G_1 and G_2 are disjoint, the derivation of w from S_i can involve only productions from P_i. Consequently, either $S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} w$ and, hence, $w \in L(G_1) \cup L(G_2)$.

($\Leftarrow$)

If $w \in L(G_1) \cup L(G_2)$ then either $S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} w$. In either case $S \underset{G}{\Longrightarrow} S_i \overset{*}{\underset{G}{\Longrightarrow}} w$ witnesses $w \in L(G)$.

78. Give a construction that, from $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$ and $G_2 = \langle V_2, \Sigma, S_2, P_2 \rangle$, builds a CFG G that generates $L(G_1) \cdot L(G_2)$. Argue that your construction is correct, that $w \in L(G)$ iff $w \in L(G_1) \cdot L(G_2)$.

(Solution)

Let

$$G = \langle V_1 \cup V_2 \cup \{S\}, \Sigma, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1 S_2\} \rangle.$$

Then

$$\begin{aligned} S \overset{*}{\underset{G}{\Longrightarrow}} w \in \Sigma^* &\Leftrightarrow S \underset{G}{\Longrightarrow} S_1 S_2 \overset{*}{\underset{G}{\Longrightarrow}} w \\ &\Leftrightarrow w = w_1 w_2 \text{ and } S_i \overset{*}{\underset{G_i}{\Longrightarrow}} w_i \\ &\Leftrightarrow w = w_1 \cdot w_2, \quad w_1 \in L(G_1) \text{ and } w_2 \in L(G_2). \end{aligned}$$

16.1 Closure of CFL under substitution

79. Show that closure of CFL under substitution into CFL implies closure of CFL under union, concatenation and Kleene closure.

(Solution)

Let

$$\begin{array}{rlllcl} G_{a+b} & \text{be} & S \longrightarrow a \mid b & \text{then} & L(G_{a+b}) & = & \{a, b\} \\ G_{ab} & \text{be} & S \longrightarrow ab & \text{then} & L(G_{ab}) & = & \{ab\} \\ G_{a^*} & \text{be} & S \longrightarrow \varepsilon \mid aS & \text{then} & L(G_{a^*}) & = & \{a^*\} \end{array}$$

Thus these are each in CFL. Let L_a and L_b be any CFLs. Let $f = \{a \mapsto L_a, b \mapsto L_b\}$. Then

$$f(L(G_{a+b})) = L_a \cup L_b \qquad f(L(G_{ab})) = L_a \cdot L_b \qquad \text{and} \qquad f(L(G_{a^*})) = (L_a)^*.$$

16.2 Constructing Grammars for CFLs

80. Give a CFG for $L_{80} = \{a^i b^j c^k \mid i \neq j \text{ or } j \neq k\}$ by decomposing the language into trivial languages.

(Solution:)

$L_{80} = L_A \cup L_B$ where $L_A = \{a^i b^j c^k \mid i \neq j,\ 0 \leq i,j,k\}$ and $L_B = \{a^i b^j c^k \mid j \neq k,\ 0 \leq i,j,k\}$.
$L_A = L_C \cup L_D$ where $L_C = \{a^i b^j c^k \mid i < j,\ 0 \leq i,j,k\}$ and $L_D = \{a^i b^j c^k \mid i > j,\ 0 \leq i,j,k\}$.
$L_B = L_E \cup L_F$ where $L_E = \{a^i b^j c^k \mid j < k,\ 0 \leq i,j,k\}$ and $L_F = \{a^i b^j c^k \mid j > k,\ 0 \leq i,j,k\}$.
$L_C = L_G \cdot L_H \cdot L_I$ where $L_G = \{a^i b^i \mid 0 \leq i\}$, $L_H = \{b^i \mid 0 < i\}$ and $L_I \{c^i \mid 0 \leq i\}$.
$L_D = L_J \cdot L_G \cdot L_I$ where $L_J = \{a^i \mid 0 < i\}$.
$L_E = L_K \cdot L_L \cdot L_M$ where $L_K = \{a^i \mid 0 \leq i\}$, $L_L = \{b^i c^i \mid 0 \leq i\}$ and $L_M = \{c^i \mid 0 < i\}$.
$L_F = L_K \cdot L_H \cdot L_L$.

Then

$$\begin{array}{rcll}
S & \longrightarrow & A & \mid B \\
A & \longrightarrow & C & \mid D \\
B & \longrightarrow & E & \mid F \\
C & \longrightarrow & GHI & \\
D & \longrightarrow & HJI & \\
E & \longrightarrow & KLM & \\
F & \longrightarrow & KJL & \\
G & \longrightarrow & aGb & \mid \varepsilon \\
H & \longrightarrow & bH & \mid b \\
I & \longrightarrow & cI & \mid \varepsilon \\
J & \longrightarrow & aK & \\
K & \longrightarrow & aK & \mid \varepsilon \\
L & \longrightarrow & bLc & \mid \varepsilon \\
M & \longrightarrow & cI &
\end{array}$$

Note that the decomposition already gives the set of strings derived by each of the non-terminals.

81. Show that $L(\mathcal{G}_2) \subseteq L_2$.

(Solution:)

The trivial derivation is just $S \Longrightarrow \varepsilon$ which certainly yields a string with equal numbers of 'a's and 'b's. Every other derivation starts with $S \Longrightarrow SS$, $S \Longrightarrow aSb$, or $\Longrightarrow bSa$, all three of which preserve the property of having equal numbers of 'a's and 'b's.

82. Show that $L_2 \subseteq L(\mathcal{G}_2)$.

(Solution:)

Suppose $w \in L_2$. Then either $w = \varepsilon$, in which case S derives w in one step: $S \Longrightarrow \varepsilon$, or w contains at least one 'a' and one 'b'. Assume, for induction that every string in L_2 that is strictly shorter than w is derivable from S in $\mathcal{G}_2$. Let w_1 be the shortest non-empty prefix of w that contains equal numbers of 'a's and 'b's. Then $w = w_1w_2$ for some w_2 that also contains equal number of 'a's and 'b's. If w_2 is non-empty, then both w_1 and w_2 are strictly shorter than w and, by the induction hypothesis, derivable from S in $\mathcal{G}_2$. Thus,

$$S \Longrightarrow SS \overset{*}{\Longrightarrow} w_1w_2 = w.$$

Suppose, on the other hand, that the shortest non-empty prefix of w that contains equal numbers of 'a's and 'b's is w itself. Then either w begins with 'a' and ends with 'b' or *vice versa*: $w = aw_1b$ or $w = bw_1a$. Moreover, w_1 has equal numbers of 'a's and 'b's and is strictly shorter than w and, consequently, by the induction hypothesis, is derivable from S in $\mathcal{G}_2$. Thus,

$$S \Longrightarrow aSb \overset{*}{\Longrightarrow} aw_1b \qquad \text{or} \qquad S \Longrightarrow aSb \overset{*}{\Longrightarrow} bw_1a.$$

83. Show that the CFG of Section 15.1 can be converted to $\mathcal{G}_2$.

(Solution:)

We can simply substitute the rhs of the productions for A and B for their occurrences in the productions for S:

$$S \longrightarrow a(Sb) \mid b(Sa) \mid SS \mid \varepsilon.$$

84. Let L_D be the set of all strings of balanced parenthesis. Show that this is a CFL by first defining it inductively.

(Solution:)

First of all, the empty string is a string of balanced parentheses. (If you choose to interpret the language as the *positive* strings of balanced parentheses there is no problem—your base case will just be '()' instead.) From this basis, if one has a string of balanced parentheses and encloses it in a pair of parentheses the result is still a string of balanced parentheses. This only gets parentheses nested in a single stack. Multiple stacks of parentheses can be constructed by starting with two strings of balanced parentheses and concatenating them. Thus:

- $\varepsilon \in L_D$.
- If $w \in L_D$ then '(' $\cdot\, w \,\cdot$ ')' $\in L_D$.
- If $w_1, w_2 \in L_D$ then $w_1 \cdot w_2 \in L_D$.
- Nothing else.

As a CFG:

$$S \longrightarrow \varepsilon \mid (S) \mid SS$$

Then we claim that $L_S = L_D$. That $w \in L_s \Rightarrow w \in L_D$ can be verified from the analysis on which the inductive definition is based. For the other direction (informally), suppose that $w \in L_D$. Consider the parenthesis matching the initial parenthesis of the string. If this matching parenthesis is the final parenthesis of the string then $w =$ '(' $\cdot\, w' \cdot$ ')', where w' is a strictly smaller string of balanced parentheses. Thus it is derived by

$$S \Longrightarrow (S) \overset{*}{\Longrightarrow} (w') = w$$

where the existence of the derivation of w' from S is justified by the induction hypothesis. If, on the other hand, the parenthesis matching the initial parenthesis is *not* the last parenthesis then $w =$ '('$\cdot w' \cdot$')'$\cdot w_2$, where '(' $\cdot\, w' \cdot$ ')' (call it w_1) and w_2 are both strings of balanced parentheses strictly smaller than w. Then, again by the IH,

$$S \Longrightarrow SS \overset{*}{\Longrightarrow} w_1 S \overset{*}{\Longrightarrow} w_1 w_2 = w.$$

17 Normal Forms for CFGs

17.1 Useless Symbols

85. Let G be the grammar in which P is:

$$\begin{aligned} S &\longrightarrow aSb \mid aXY \mid \varepsilon \\ X &\longrightarrow aX \mid aS \\ Y &\longrightarrow Yb \\ Z &\longrightarrow Xb \end{aligned}$$

(a) What is Productive(G)?

(Solution)
$\{a, b, S, X, Z\}$.

(b) Let G_1 be the grammar in which $P_1 = P \cap (\text{Productive}(G) \times (\text{Productive}(G))^*)$. What is P_1.?

(Solution)

$$S \longrightarrow aSb \mid \varepsilon \qquad X \longrightarrow aX \mid aS \qquad Z \longrightarrow Xb$$

(c) What is Reachable(G_1)?

(Solution)
$\{S, a, b\}$.

(d) Let G_2 be the grammar in which $P_2 = P_1 \cap (\text{Reachable}(G_1) \times (\text{Reachable}(G_1))^*)$. What is P_2?

(Solution)
$S \longrightarrow aSb \mid \varepsilon$.

(e) What is Reachable(G)?

(Solution)
$\{S, X, Y, a, b\}$.

(f) Let G_3 be the grammar with $P_3 = P \cap (\text{Reachable}(G) \times (\text{Reachable}(G))^*)$. What is P_3?

(Solution)

$$S \longrightarrow aSb \mid aXY \mid \varepsilon \qquad X \longrightarrow aX \mid aS \qquad Y \longrightarrow Yb$$

(g) What is Productive(G_3)?

(Solution)

$\{a, b, S, X\}$.

(h) Let G_4 be the grammar with $P_4 = P_3 \cap (\text{Productive}(G_3) \times (\text{Productive}(G_3))^*)$. What is P_4?

(Solution)

$$S \longrightarrow aSb \mid \varepsilon \qquad X \longrightarrow aX \mid aS$$

17.2 ε-Productions

86. Prove that if G is a positive CFG then there is no derivation of ε from any non-terminal in G.

(Solution)

If G has no ε-productions then the right-hand side of each production has at least one symbol. Hence, it is at least as long as the left-hand side and, consequently, every sentential form in a production is at least as long as its predecessor. Since the initial sentential from (S) has length one, every sentential form has length one. Therefore, ε, having length zero, cannot be derived.

87. Give an example of a CFG that includes an ε-production but cannot derive the empty string.

(Solution)

How about:

$$S \longrightarrow aA \qquad A \longrightarrow \varepsilon.$$

88. What happens if $L(G)$ is *not* positive?

(Solution)

If $L(G)$ is not positive then S is nullable and there must be some production $S \longrightarrow \gamma$ for which $S \longrightarrow \varepsilon \in \text{Nulled}_G(S \longrightarrow \gamma)$. Note that every other ε-production in the nulled grammar can be safely removed, but if we remove $S \longrightarrow \varepsilon$ also the grammar will no longer derive ε.

17.3 Unit Productions

89. Why can't we remove useless symbols first?

    (Solution:)

    The process of eliminating ε-productions may will make every symbol that derives only the empty string unproductive. Similarly, any symbol that is reachable only in the middle of some sequence of unit productions will become unreachable when unit productions are eliminated.

17.4 Chomsky Normal Form (CNF)

90. Convert the grammar

$$\begin{array}{rrcl}G: & S & \longrightarrow & T \mid S+T \\ & T & \longrightarrow & x \mid (S)\end{array}$$

to CNF.

(Solution:)

First we remove the unit productions:

$$\begin{array}{rrclll}G_1: & S & \longrightarrow & S+T & \mid (S) & \mid x \\ & T & \longrightarrow & (S) & \mid x &\end{array}$$

Then we add the pre-terminals:

$$\begin{array}{rrclll}G_2: & S & \longrightarrow & SAT & \mid LSR & \mid x \\ & T & \longrightarrow & LSR & \mid x & \\ & A & \longrightarrow & + && \\ & L & \longrightarrow & (&& \\ & R & \longrightarrow &) &&\end{array}$$

Note that, as 'x' is generated only by non-branching rules (i.e., of the form '$X \longrightarrow \sigma$'), there is no need to add a preterminal for 'x'—since no occurrences of 'x' get replaced, the corresponding non-terminal will be unreachable.

Finally, we convert the three-branching rules to binary form:

$$
\begin{array}{rrcl}
G_2: & S & \longrightarrow & SS_1 \mid LS_2 \mid x \\
& S_1 & \longrightarrow & AT \\
& S_2 & \longrightarrow & SR \\
& T & \longrightarrow & LT_1 \mid x \\
& T_1 & \longrightarrow & SR \\
& A & \longrightarrow & + \\
& L & \longrightarrow & (\\
& R & \longrightarrow &)
\end{array}
$$

91. Suppose G is a CFG in CNF and $S \overset{*}{\underset{G}{\Longrightarrow}} w$. Give an upper bound on the length of derivations of w from S in G.

(Solution)

Note that there will be exactly one production of the form $A \longrightarrow \sigma$ for each symbol in w, thus exactly $|w|$ of these. Furthermore, each production of the form $A \longrightarrow BC$ adds exactly one to the length of the sentential form to which it is applied. Since the length of the initial sentential form is 1, there can be no more than $|w| - 1$ of these. Hence, the derivation can have no more than $n + n - 1 = 2n - 1$ steps, which is to say it can include no more than $2n$ sentential forms.

92. Give a lower bound on the length of derivations of w from S in G under the same assumptions.

(Solution)

Here we note that the only productions that increase the length of a sentential form in a derivation in G are the ones of form $A \longrightarrow BC$ and these increase its length by exactly one. Consequently, the derivation must employ at least $|w| - 1$ of these productions. As we just argued, it must also employ $|w|$ productions of the form $A \longrightarrow \sigma$. Thus, there must be at least $2n - 1$ steps to the derivation and it must include at least $2n$ sentential forms.

It follows that every derivation in a CNF grammar is of length exactly $2|w|$ (has $2|w| - 1$ steps).

We can get both the upper and lower bound easily by appealing to the form of the derivation trees. Note that, for any grammar in CNF, these consist of a binary branching portion with a “fringe” of unary branching productions at its leaves. The corresponding derivations will include a step for each of the internal nodes of the tree, which is to say, for each of the nodes in the binary part. Note, also, that this binary portion has exactly $|w|$ leaves. It remains only to establish that the number of nodes in a binary tree with n leaves is $2n-1$. This is an easy induction on the structure of the tree: the trivial binary tree has one leaf and $2-1=1$ nodes; a binary tree constructed from two binary trees with n_1 and n_2 leaves (and, by IH, $2n_1-1$ and $2n_2-1$ nodes) has n_1+n_2 leaves and $2n_1-1+2n_2-1+1=2(n_1+n_2)-1$ nodes.

93. Prove that the class of CFLs is closed under reversal:

$$L^{\mathrm{R}} = \{w^{\mathrm{R}} \mid w \in L\}.$$

(Solution)

For $G = \langle V, \Sigma, S, P\rangle$ let $G' = \langle V, \Sigma, S, P'\rangle$ where $A \longrightarrow \alpha^{\mathrm{R}} \in P' \Leftrightarrow A \longrightarrow \alpha \in P$ (i.e., P' is P with the right-hand side if its productions reversed; of course, this effects only the binary productions of G). To show that $L(G') = L(G)^{\mathrm{R}}$ we will prove that $X \overset{*}{\underset{G}{\Longrightarrow}} \alpha$ iff $X \overset{*}{\underset{G'}{\Longrightarrow}} \alpha^{\mathrm{R}}$ for all $X \in V$, $\alpha \in (V \cup \Sigma)^*$, by induction on the number of steps in the derivation. We will use derivations with 1 step as the base case.

(Basis:)

$$\begin{aligned} X \underset{G}{\Longrightarrow} \alpha \quad &\Leftrightarrow \quad X \longrightarrow \alpha \in P \\ &\Leftrightarrow \quad X \longrightarrow \alpha^{\mathrm{R}} \in P' \\ &\Leftrightarrow \quad X \underset{G'}{\Longrightarrow} \alpha^{\mathrm{R}}. \end{aligned}$$

(Ind:)

$$\begin{aligned} X \overset{n>1}{\underset{G}{\Longrightarrow}} \alpha \quad &\Leftrightarrow \quad X \underset{G}{\Longrightarrow} BC \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1\alpha_2 = \alpha \text{ and } B \overset{<n}{\underset{G}{\Longrightarrow}} \alpha_1, C \overset{<n}{\underset{G}{\Longrightarrow}} \alpha_2 \text{ for some } B, C \in V \\ &\Leftrightarrow \quad X \underset{G'}{\Longrightarrow} CB \overset{*}{\underset{G'}{\Longrightarrow}} {\alpha_2}^{\mathrm{R}}{\alpha_1}^{\mathrm{R}} \text{ for some } B, C \in V \qquad \text{(by IH)} \\ &\Leftrightarrow \quad X \overset{n>1}{\underset{G'}{\Longrightarrow}} \alpha^{\mathrm{R}}. \end{aligned}$$

It follows, then, that

$$w \in L(G) \Leftrightarrow S \overset{*}{\underset{G}{\Longrightarrow}} w \Leftrightarrow S \overset{*}{\underset{G'}{\Longrightarrow}} w^{\mathrm{R}} \Leftrightarrow w^{\mathrm{R}} \in L(G')$$

which is to say, $L(G') = L(G)^{\mathrm{R}}$.

17.5 Greibach Normal Form (GNF)

94. Suppose G is a CFG in GNF and $S \overset{*}{\underset{G}{\Longrightarrow}} w$. Give both an upper and a lower bound on the length of derivations of w from S in G.

    (Solution)

    This is nearly immediate. Every production in a GNF grammar adds exactly one terminal to the sentential form to which it is applied. Hence there are exactly $|w|$ steps in the derivation of w from S and the derivation includes $n + 1$ sentential forms.

18 Deciding Membership

95. Give an algorithm to test if $\varepsilon \in L(\mathcal{G})$, where $\mathcal{G}$ is any CFG.

    (Solution)

    Compute Nullable($\mathcal{G}$). $S \in$ Nullable($\mathcal{G}$) iff $\varepsilon \in L(\mathcal{G})$.

96. Put this together with the ideas of this section to give a completely general algorithm that decides membership for any CFG.

    (Solution)

    Given $\mathcal{G}$ and w. If $w = \varepsilon$ then return TRUE iff $S \in$ Nullable($\mathcal{G}$). Otherwise convert $\mathcal{G}$ to GNF. Systematically generate all derivations in $\mathcal{G}$ of length $|w|$. If the last sentential form of one of these is w then return TRUE, otherwise return FALSE.

18.1 Recursive Descent Parsing

97. Carry out the procedure for Parse(S, ab) for the grammar:

$$S \longrightarrow AB \qquad A \longrightarrow Sa \qquad A \longrightarrow \varepsilon \qquad B \longrightarrow bS \qquad B \longrightarrow \varepsilon$$

with the productions ordered left to right as given.

(Solution)

The stack of recursive calls looks something like:

```
Parse(S,ab)
  Parse(AB,ab)
    Parse(SaB,ab)
      Parse(ABaB,ab)
        etc.
```