Derivations beginning with \((AdditiveExpression)\) produce correctly formed expressions with additive operators, multiplicative operators, and parentheses. For example,

\[
\begin{align*}
(AdditiveExpression) & \Rightarrow (AdditiveExpression) + (MultiplicativeExpression) \\
& \Rightarrow (MultiplicativeExpression) + (MultiplicativeExpression) \\
& \Rightarrow (UnaryExpression) + (MultiplicativeExpression) \\
& \Rightarrow (Identifier) + (MultiplicativeExpression) \\
& \Rightarrow (Identifier) + (MultiplicativeExpression) \ast (MultiplicativeExpression)
\end{align*}
\]

begins such a derivation. Derivations from \((UnaryExpression)\) can produce literals, variables, or \((Expression)\) to obtain nested parentheses.

The rules that define identifiers, literals, and expressions show how the design of a large language is decomposed into creating rules for frequently recurring subsets of the language. The resulting variables \((Identifier)\), \((Literal)\), and \((Expression)\) become the building blocks for higher-level rules.

The start symbol of the grammar is \((CompilationUnit)\) and the derivation of a Java program begins with the rule

\[
(CompilationUnit) \Rightarrow (PackageDeclaration)_{opt} (ImportDeclarations)_{opt} (TypeDeclarations)_{opt}.
\]

A string of terminal symbols derivable from this rule is a syntactically correct Java program.

---

**Exercises**

1. Let \(G\) be the grammar

\[
S \rightarrow abSc \mid A \\
A \rightarrow cAd \mid cd.
\]

a) Give a derivation of \(ababcdc\).
b) Build the derivation tree for the derivation in part (a).
c) Use set notation to define \(L(G)\).

2. Let \(G\) be the grammar

\[
S \rightarrow ASB \mid \lambda \\
A \rightarrow aAb \mid \lambda \\
B \rightarrow bBa \mid ba.
\]

a) Give a leftmost derivation of \(aabbba\).
b) Give a rightmost derivation of \(abaabbbabbaa\).
c) Build the derivation tree for the derivations in parts (a) and (b).
d) Use set notation to define L(G).

3. Let G be the grammar

\[
S \rightarrow SAB \mid \lambda \\
A \rightarrow aA \mid a \\
B \rightarrow bB \mid \lambda.
\]

a) Give a leftmost derivation of \textit{abbaab}.
b) Give two leftmost derivations of \textit{aa}.
c) Build the derivation tree for the derivations in part (b).
d) Give a regular expression for L(G).

4. Let DT be the derivation tree

```
S
\ /
/  \nA --\- B
   /   \
/    \n a    a
  / \  / \
A  A  B
 /   / \
/   /  \\
 a  a  b
```

a) Give a leftmost derivation that generates the tree DT.
b) Give a rightmost derivation that generates the tree DT.
c) How many different derivations are there that generate DT?

5. Give the leftmost and rightmost derivations corresponding to each of the derivation trees given in Figure 3.3.

6. For each of the following context-free grammars, use set notation to define the language generated by the grammar.

\[
\begin{align*}
&\text{a) } S \rightarrow aSb \mid \lambda \\
&\quad B \rightarrow bB \mid b \\
&\text{b) } S \rightarrow aSbb \mid A \\
&\quad A \rightarrow cA \mid c \\
&\text{c) } S \rightarrow abSc \mid A \\
&\quad A \rightarrow cdAb \mid A \\
&\text{d) } S \rightarrow aSb \mid A \\
&\quad A \rightarrow cAd \mid cBd \\
&\quad B \rightarrow aBb \mid ab \\
&\text{e) } S \rightarrow aSb \mid aB \\
&\quad B \rightarrow bb \mid b
\end{align*}
\]

7. Construct a grammar over \{a, b, c\} whose language is \{a^n b^{2n} c^m \mid n, m \geq 0\}.
8. Construct a grammar over \{a, b, c\} whose language is \{a^n b^m c^{2n+m} \mid n, m \geq 0\}.
9. Construct a grammar over \{a, b, c\} whose language is \{a^n b^m c^i \mid 0 \leq n + m \leq i\}.
10. Construct a grammar over \( \{a, b\} \) whose language is \( \{a^m b^n \mid 0 \leq n \leq m \leq 3n\} \).

11. Construct a grammar over \( \{a, b\} \) whose language is \( \{a^m b^i a^n \mid i = m + n\} \).

12. Construct a grammar over \( \{a, b\} \) whose language contains precisely the strings with the same number of \( a \)'s and \( b \)'s.

* 13. Construct a grammar over \( \{a, b\} \) whose language contains precisely the strings of odd length that have the same symbol in the first and middle positions.

14. For each of the following regular grammars, give a regular expression for the language generated by the grammar.

\[
\begin{align*}
\text{a) } & S \rightarrow aA \\
& A \rightarrow aA \mid bA \mid b \\
\text{b) } & S \rightarrow aA \\
& A \rightarrow aA \mid bB \\
& B \rightarrow bB \mid \lambda
\end{align*}
\]

\[
\begin{align*}
\text{c) } & S \rightarrow aS \mid bA \\
& A \rightarrow bB \\
& B \rightarrow aB \mid \lambda \\
\text{d) } & S \rightarrow aS \mid bA \mid \lambda \\
& A \rightarrow aA \mid bS
\end{align*}
\]

For Exercises 15 through 25, give a regular grammar that generates the described language.

15. The set of strings over \( \{a, b, c\} \) in which all the \( a \)'s precede the \( b \)'s, which in turn precede the \( c \)'s. It is possible that there are no \( a \)'s, \( b \)'s, or \( c \)'s.

16. The set of strings over \( \{a, b\} \) that contain the substring \( aa \) and the substring \( bb \).

17. The set of strings over \( \{a, b\} \) in which the substring \( aa \) occurs at least twice. (Hint: Beware of the substring \( aaa \).)

18. The set of strings over \( \{a, b\} \) that contain the substring \( ab \) and the substring \( ba \).

19. The set of strings over \( \{a, b\} \) in which the number of \( a \)'s is divisible by three.

20. The set of strings over \( \{a, b\} \) in which every \( a \) is either immediately preceded or immediately followed by \( b \), for example, \( baab, aba, \) and \( b \).

21. The set of strings over \( \{a, b\} \) that do not contain the substring \( aba \).

22. The set of strings over \( \{a, b\} \) in which the substring \( aa \) occurs exactly once.

23. The set of strings of odd length over \( \{a, b\} \) that contain exactly two \( b \)'s.

* 24. The set of strings over \( \{a, b, c\} \) with an odd number of occurrences of the substring \( ab \).

25. The set of strings over \( \{a, b\} \) with an even number of \( a \)'s or an odd number of \( b \)'s.

26. The grammar in Figure 3.1 generates \( (b^i a^j b^k)^+ \), the set of all strings with a positive, even number of \( a \)'s. Prove this.

27. Prove that the grammar given in Example 3.2.2 generates the prescribed language.

28. Let \( G \) be the grammar

\[
\begin{align*}
S & \rightarrow aSb \mid B \\
B & \rightarrow bB \mid b
\end{align*}
\]

Prove that \( L(G) = \{a^n b^m \mid 0 \leq n < m\} \).
29. Let $G$ be the grammar

$$S \rightarrow aSa | B$$
$$B \rightarrow bbBdd | C$$
$$C \rightarrow bd.$$  

a) What is $L(G)$?  

b) Prove that $L(G)$ is the set given in part (a).

* 30. Let $G$ be the grammar

$$S \rightarrow aSbS | aS | \lambda.$$  

Prove that every prefix of a string in $L(G)$ has at least as many $a$'s as $b$'s.

31. Let $G$ be a context-free grammar and $w \in L(G)$. Prove that there is a rightmost derivation of $w$ in $G$.

32. Let $G$ be the grammar

$$S \rightarrow aS | Sb | ab.$$  

a) Give a regular expression for $L(G)$.  

b) Construct two leftmost derivations of the string $aabb$.  

c) Build the derivation trees for the derivations from part (b).  

d) Construct an unambiguous grammar equivalent to $G$.

33. For each of the following grammars, give a regular expression or set-theoretic definition for the language of the grammar. Show that the grammar is ambiguous and construct an equivalent unambiguous grammar.

✓ a) $S \rightarrow aaS | aaaaaS | \lambda$  

b) $S \rightarrow aSA | \lambda$  

A $\rightarrow bA | \lambda$  

c) $S \rightarrow aSb | aAb$  

A $\rightarrow cAd | B$  

B $\rightarrow aBb | \lambda$  

d) $S \rightarrow AaSbB | \lambda$  

A $\rightarrow aA | a$  

B $\rightarrow bB | \lambda$  

e) $S \rightarrow A | B$  

A $\rightarrow abA | \lambda$  

B $\rightarrow aBb | \lambda$
34. Let $G$ be the grammar

$$
S \rightarrow aA | \lambda \\
A \rightarrow aA | bB \\
B \rightarrow bB | b.
$$

a) Give a regular expression for $L(G)$.
b) Prove that $G$ is unambiguous.

35. Let $G$ be the grammar

$$
S \rightarrow aS | aA | a \\
A \rightarrow aAb | ab.
$$

a) Give a set-theoretic definition of $L(G)$.
b) Prove that $G$ is unambiguous.

36. Let $G$ be the grammar

$$
S \rightarrow aS | bA | \lambda \\
A \rightarrow bA | aS | \lambda.
$$

Give a regular expression for $L(G)$. Is $G$ ambiguous? If so, give an unambiguous grammar that generates $L(G)$. If not, prove it.

37. Construct unambiguous grammars for the languages $L_1 = \{a^n b^n c^m \mid n, m > 0\}$ and $L_2 = \{a^n b^m c^m \mid n, m > 0\}$. Construct a grammar $G$ that generates $L_1 \cup L_2$. Prove that $G$ is ambiguous. This is an example of an inherently ambiguous language. Explain, intuitively, why every grammar generating $L_1 \cup L_2$ must be ambiguous.

38. Use the definition of Java in Appendix IV to construct a derivation of the string $1.3e2$ from the variable $<Literal>$.

*39. Let $G_1$ and $G_2$ be the following grammars:

$$
G_1: \quad S \rightarrow aABb \\
A \rightarrow aA | a \\
B \rightarrow bB | b
$$

$$
G_2: \quad S \rightarrow AABB \\
A \rightarrow AA | a \\
B \rightarrow BB | b
$$

a) For each variable $X$, show that the right-hand side of every $X$ rule of $G_1$ is derivable from the corresponding variable $X$ using the rules of $G_2$. Use this to conclude that $L(G_1) \subseteq L(G_2)$.
b) Prove that $L(G_1) = L(G_2)$. 
40. A right-linear grammar is a context-free grammar, each of whose rules has one of the following forms:
   i) \( A \to w \), or
   ii) \( A \to wB \),

   where \( w \in \Sigma^* \). Prove that a language \( L \) is generated by a right-linear grammar if, and only if, \( L \) is generated by a regular grammar.

41. Try to construct a regular grammar that generates the language \( \{a^n b^n \mid n \geq 0\} \). Explain why none of your attempts succeed.

42. Try to construct a context-free grammar that generates the language \( \{a^n b^n c^n \mid n \geq 0\} \).
   Explain why none of your attempts succeed.

---

**Bibliographic Notes**

Context-free grammars were introduced by Chomsky [1956], [1959]. Backus-Naur form was developed by Backus [1959]. This formalism was used to define the programming language ALGOL; see Naur [1963]. The BNF definition of Java is given in Appendix IV. The equivalence of context-free languages and the languages generated by BNF definitions was noted by Ginsburg and Rice [1962].

Properties of ambiguity are examined in Floyd [1962], Cantor [1962], and Chomsky and Schutzenberger [1963]. Inherent ambiguity was first noted in Parikh [1966]. A proof that the language in Exercise 37 is inherently ambiguous can be found in Harrison [1978]. Closure properties for ambiguous and inherently ambiguous languages were established by Ginsburg and Ullian [1966a, 1966b].