Research in axiomatic semantics was begun by Floyd (1967) and further developed by Hoare (1969). The semantics of a large part of Pascal was described by Hoare and Wirth (1973) using this method. The parts they did not complete involved functional side effects and goto statements. These were found to be the most difficult to describe.

The technique of using preconditions and postconditions during the development of programs is described (and advocated) by Dijkstra (1976) and also discussed in detail in Gries (1981).

Good introductions to denotational semantics can be found in Gordon (1979) and Sney (1977). Introductions to all three semantics description methods discussed in this chapter can be found in Marcorse et al. (1976). Another good reference for much of the material of this chapter is Pagan (1981). The form of the denotational semantic functions in this chapter is similar to that in Meyer (1990).

**Review Questions**

1. Define *syntax* and *semantics*.
2. Who are language descriptions for?
3. Describe the operation of a general language generator.
4. Describe the operation of a general language recognizer.
5. What is the difference between a sentence and a sentential form?
6. Define a left recursive grammar rule.
7. What three extensions are common to most EBNFs?
8. Distinguish between static and dynamic semantics.
9. What purpose do predicates serve in an attribute grammar?
10. What is the difference between a synthesized and an inherited attribute?
11. How is the order of evaluation of attributes determined for the trees of a given attribute grammar?
12. What is the primary use of attribute grammars?
13. What is the problem with using a software pure interpreter for operational semantics?
14. Explain what the preconditions and postconditions of a given statement mean in axiomatic semantics.
15. Describe the approach of using axiomatic semantics to prove the correctness of a given program.
16. Describe the basic concept of denotational semantics.
17. In what fundamental way do operational semantics and denotational semantics differ?

**Problem Set**

1. The two mathematical models of language description are generation and recognition. Describe how each can define the syntax of a programming language.
2. Write EBNF descriptions for the following:
   a. A Java class definition header statement
   b. A Java method call statement
   c. A `switch` statement
   d. A `union` definition
   e. A `float` literal
3. Rewrite the BNF of Example 3.4 to give `+` precedence over `*` and force + to be right associative.
4. Rewrite the BNF of Example 3.4 to add the `++` and `--` unary operators of Java.
5. Write a BNF description of the Boolean expressions of Java, including the three operators `&`, `|`, and `!`, and the relational expressions.
6. Using the grammar in Example 3.2, show a parse tree and a leftmost derivation for each of the following statements:
   a. `A = A * (B + (C * A))`
   b. `B = C * (A + C + B)`
   c. `A = A * (B + C)`
   d. `A = A * (B + C)`
   e. `A = (A + B) * C`
   f. `A = B + C * A`
   g. `A = (A * (B + C))`
   h. `A = (C * (A + B))`
7. Using the grammar in Example 3.4, show a parse tree and a leftmost derivation for each of the following statements:
   a. `S = A * (B + C)`
   b. `S = (A + B) * C`
   c. `S = (A * (B + C))`
   d. `S = (C * (A + B))`
8. Prove that the following grammar is ambiguous:
   `<S> -> <A>`
   `<A> -> <A> + <A> 1 <id>`
   `<id> -> 1 <id>`
9. Modify the grammar of Example 3.4 to add a unary minus operator that has higher precedence than either `+` or `-`.
10. Describe, in English, the language defined by the following grammar:
    `<S> -> <A> <B> <C>`
    `<A> -> a <A> 1 a`
    `<B> -> b <B> 1 b`
    `<C> -> c <C> 1 c`
11. Consider the following grammar:
\[ S \rightarrow A b \]
\[ A \rightarrow A a \]
\[ B \rightarrow a \]
Which of the following sentences are in the language generated by this grammar?
- a. baaab
- b. bbbab
- c. bbbaaaa
- d. bbaab

12. Consider the following grammar:
\[ S \rightarrow a S c \]
\[ A \rightarrow c A \]
\[ B \rightarrow d \]
Which of the following sentences are in the language generated by this grammar?
- a. abcd
- b. acbdc
- c. accebc
- d. acd
- e. acce

13. Write a grammar for the language consisting of strings that have \( k \) copies of the letter \( a \) followed by the same number of copies of the letter \( b \), where \( k > 0 \). For example, the strings \( ab \), \( aaabb \), and \( aaaaaaabb \) are in the language but \( a, ab, ba, \) and \( aaab \) are not.

14. Draw parse trees for the sentences \( aab \) and \( aaababbb \), as derived from the grammar of Problem 13.

15. Convert the BNF of Example 3.1 to EBNF.
16. Convert the BNF of Example 3.3 to EBNF.

17. Convert the following EBNF to BNF:
\[ S \rightarrow A ( b A ) \]
\[ A \rightarrow a ] \]

18. Using the virtual machine instructions given in Section 3.5.1.1, give an operational semantic definition of the following:
- a. Pascal \( \text{repeat} \)
- b. Ada \( \text{for} \)
- c. Fortran \( \text{do} \) of the form: \( \text{do } N \text{ } K = \text{start, end, step} \)

19. Compute the weakest precondition for each of the following assignment statements and postconditions:
- a. \( a = 2 \times (b - 1) - 1, (a > 0) \)
- b. \( b = (c + 10) / 3, (b > 6) \)
- c. \( a = a + 2 \times b - 1, (a > 1) \)
- d. \( x = 2 \times y + x - 1, (x > 11) \)

20. Compute the weakest precondition for each of the following sequences of assignment statements and their postconditions:
- a. \( a = 2 \times b + 1, b = a - 3 \)
- b. \( a = 3 \times (2 \times b + a), b = 2 \times a - 1 \)

21. Write a denotational semantics mapping function for the following statements:
- a. Ada \( \text{for} \)
- b. Pascal \( \text{repeat} \)
- c. Java \( \text{Boolean expressions} \)
- d. Java \( \text{for} \)
- e. C \( \text{switch} \)

22. What is the difference between an intrinsic attribute and a nonintrinsic-synthesized attribute?

23. Write an attribute grammar whose BNF basis is that of Example 3.6 in Section 3.4.5, but whose language rules are as follows: Data types cannot be mixed in expressions, but assignment statements need not have the same types on both sides of the assignment operator.

24. Write an attribute grammar whose base BNF is that of Example 3.2 and whose type rules are the same as for the assignment statement example of Section 3.4.5.

25. Prove the following program is correct:
\( (x = V, \text{ and } y = V) \)
\( \text{temp} = x; \)
\( x = y; \)
\( y = \text{temp}; \)
\( (x = V, \text{ and } y = V) \)
26. Prove the following program is correct:

\[
\begin{align*}
(n > 0) \\
\text{count} &= n; \\
\text{sum} &= 0; \\
\text{while } \text{count} \neq 0 \text{ do} & \\
& \quad \text{sum} = \text{sum} + \text{count}; \\
& \quad \text{count} = \text{count} - 1; \\
\text{end} \\
& (\text{sum} = 1 + 2 + \ldots + n)
\end{align*}
\]