Symbolic logic provides the basis for logic programming and logic programming languages. The approach of logic programming is to use as a database a collection of facts and rules that state relationships between facts, and to use an automatic inferencing process to check the validity of new propositions, assuming the facts and rules of the database are true. This approach is the one developed for automatic theorem proving.

Prolog is the most widely used logic programming language. The origins of logic programming lie in Robinson's development of the resolution rule for logical inference. Prolog was developed primarily by Colmerauer and Roussel at Marseille, with some help from Kowalski at Edinburgh.

Logic programs should be nonprocedural, which means that the characteristics of the solution are given but the complete process of getting the solution is not.

Prolog statements are facts, rules, or goals. Most are made up of structures, which are atomic propositions, and logic operators, although arithmetic expressions are also allowed.

Resolution is the primary activity of a Prolog interpreter. This process, which uses backtracking extensively, involves mainly pattern matching among propositions. When variables are involved, they can be instantiated to values to provide matches. This instantiation process is called unification.

There are a number of problems with the current state of logic programming. For reasons of efficiency, and even to avoid infinite loops, programmers must sometimes state control flow information in their programs. Also, there are problems with the closed-world assumption and negation.

Logic programming has been used in a number of different areas, primarily in relational database systems, expert systems, and natural language processing.

**BIBLIOGRAPHIC NOTES**

The Prolog language is described in several books. Edinburgh's form of the language is covered in Clocksin and Mellish (1997). The microcomputer implementation is described in Clark and McCabe (1984).

Hogster (1984) is an excellent book on the general area of logic programming. It is the source of the material in this chapter's section on logic programming applications.

**REVIEW QUESTIONS**

1. What are the three primary uses of symbolic logic in formal logic?
2. What are the two parts of a compound term?
3. What is the general form of a proposition in clausal form?
4. Give general (not rigorous) definitions of resolution and unification.
5. What are the forms of Horn clauses?
6. What is the basic concept of declarative semantics?
7. What are the three forms of a Prolog term?
8. What are the syntactic forms and usage of fact and rule statements in Prolog?
9. Explain the two approaches to matching goals to facts in a database.
10. Explain the difference between a depth-first and a breadth-first search when discussing how multiple goals are satisfied.
11. Explain how backtracking works in Prolog.
12. Explain what is wrong with the Prolog statement X is X + 1.
13. What are the two ways a Prolog programmer can control the order of pattern matching during resolution?
14. Explain the generate and test programming strategy in Prolog.
15. Explain the closed-world assumption used by Prolog. Why is this a limitation?
16. Explain the negation problem with Prolog. Why is this a limitation?
17. Explain the connection between automatic theorem proving and Prolog's inferencing process.
18. Explain the difference between procedural and nonprocedural languages.
19. Explain why Prolog systems must do backtracking.
20. What is the relationship between resolution and unification in Prolog?

**PROBLEM SET**

1. Compare the concept of data typing in Ada with that of Prolog
2. Describe how a multiple-processor machine could be used to implement resolution. Could Prolog, as currently defined, use this method?
3. Write a Prolog description of your family tree (based only on facts), going back to your grandparents and including all descendants. Be sure to include all relationships.
4. Write a set of rules for family relationships, including all relationships from grandparents through two generations. Now add these to the facts of Problem 1, and eliminate as many of the facts as you can.
5. Write the following statements in Prolog:
   a. If Fido is a yellow Lab, then Fido is a dog.
   b. If it is Tuesday and it is February, then there is school.
   c. If Fred is a male and Fred is your parent, then Fred is your father.
   d. If X is your parent, then X is your father or X is your mother.
6. Explain two ways in which the list-processing capabilities of Scheme and Prolog are similar.
7. In what way are the list-processing capabilities of Scheme and Prolog different?
8. Write a comparison of Prolog with ML, including two similarities and two differences.
9. From a book on Prolog, learn and write a description of an occur-check problem. Why does Prolog allow this problem to exist in its implementation?

PROGRAMMING EXERCISES

1. Write a Prolog program that finds the maximum of a list of numbers.
2. Write a Prolog program that succeeds if the intersection of two given list parameters is empty.
3. Write a Prolog program that returns a list containing the union of the elements of two given lists.
4. Write a Prolog program that returns the last element of a given list.

BIBLIOGRAPHY