Transactions on Information and Communications Technologies vol 7, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

Prolog as a first programming language

M.P. Lee,[a] J.D. Pryce,[b] A. Harrison[b]

[a] *Computing Information Systems Management Group*

[b] *Software Engineering Group, Royal Military College of Science, Cranfield University, Shrivenham, Swindon, SN6 8LA, UK*

Abstract

This paper documents an experiment in teaching Prolog as a first programming language to first year computing undergraduates. It begins by introducing the main programming paradigms before considering the factors involved in first language choice and the advantages and disadvantages of Prolog. It concludes by illustrating how Prolog can be utilised across much of the computing curriculum.

1 Introduction

Most computing undergraduates are first taught a third generation programming language (3GL) such as Pascal. Unfortunately this first exposure can constrain the students to a permanent procedural view. There is an analogy here to learning natural language, a person's native tongue greatly influences their approach to a second natural language. Therefore the choice of both first programming language and appropriate methods to teach first year undergraduates is crucial (Peacock et alia 1987 & 1988).

A few places have tried teaching a fourth generation language (4GL, Lee et alia 1989) and increasingly fifth generation (5GL) or declarative languages are being taught. The latter are usually functional languages such as (S)ML, Miranda, Hope or Lisp. This paper argues that Prolog ('PROgramming in LOGic') is a suitable first programming language.

Furber (1992) has conducted a survey of 60 UK universities and polytechnics

which shows that the most popular first programming language is Pascal (33 respondents) followed by Modula2 (11), Ada (6) and then by various functional languages (5). Interestingly Prolog is far and way the most popular second programming language (38) followed by C (24) and COBOL (21), but nowhere admits to teaching it first.

Allen (1985) gives the following terminology comparison between procedural, functional and logic programming languages:

	Procedural	Functional	Logic
Module	procedure	function	clause
Component	statement	expression	relation
Naming	assignment	lambda-binding	unification
Result	side-effect	value	constraint

This indicates that the clause is the unit of modularity in Prolog consisting of a series of relations. Importantly naming in Prolog takes place by a process of pattern matching termed unification, which binds a value to a variable. The variable cannot be re-assigned but only undone and re-satisfied. Clauses and relations in Prolog are either true or false so that the result of any Prolog query or goal is ultimately the answer yes or no.

This paper documents an experiment in teaching Prolog as a first programming language to first year computing undergraduates. It begins by considering the factors involved in first language choice, continues with the advantages and disadvantages of Prolog and concludes by illustrating how Prolog can be utilised across much of the computing curriculum.

2 Language choice

Woodhouse (1983) has proposed that a first programming language should satisfy 5 well-defined criteria. The first is that of availability. Prolog is widely available on mainframe, mini- and micro-computers. The public domain version of Prolog-2 for MS-DOS is particularly attractive to educators.

The second criterion is that of teacher knowledge. In the case of Prolog this is aided by a variety of textbooks and articles extolling the virtues of Prolog (e.g. Goble 1989, Leadbetter et alia 1991 and Malpas 1987).

The third requirement, that of ease of teaching and learning, stems from the design of Prolog as an interactive interpreter. In order to run a program in Prolog it is merely necessary to pose a query or goal which may or may not be satisfied. Students do not need to know about input/output (I/O) in order to

Transactions on Information and Communications Technologies vol 7, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

get results since all query variables are automatically displayed.

Prolog requires little initial design since it merely requires some pertinent facts and rules to get a student started. Since it is a weakly typed language beginning students do not need to declare data types. In addition Prolog dispenses with the tedious compile/link/run cycle of 3GLs.

The fourth criterion is that of language utility. Prolog was originally designed for natural language processing, but logic programming turned out to be more general purpose. Some of the educational applications are discussed below.

The last requirement, that of being well-structured for problem solving, springs from its logical simplicity. Prolog has only one main control structure, recursion and one main data structure, the list.

3 Advantages of Prolog

Li (1984) gives six attractive features of Prolog. Firstly specification is programming since we can write executable specifications directly in Prolog. Leadbetter et alia (1991) give an example:

```
maximum (Input1, Input2, Result) :-
  pre_maximum (Input1, Input2),
  post_maximum (Input1, Input2, Result).

pre_maximum (Input1, Input2) :-
  integer(Input1),
  integer(Input2).

post_maximum (Input1, Input2, Result) :-
  (Result = Input1; Result = Input2),
   Result >= Input1,
   Result >= Input2.
```

Notice that post_maximum never compares the two inputs directly (the procedural approach) but merely states what conditions have to hold in any solution. Of course for a beginning programmer the simpler more procedural code is:

```
maximum (Input1, Input2, Input1) :-
  Input1 >= Input2.
maximum (Input1, Input2, Input2) :-
  Input1 <  Input2.
```

Secondly Prolog has a powerful deductive capability which views computation as inference. Thus the program becomes a logical proof of the solution of a given problem. This pure approach begins to break down when procedural

elements like I/O enter, but beginning programmers need not concern themselves with this.

Thirdly Prolog is non-deterministic since programs may generate multiple solutions through backtracking. Students do not need to worry about control structures but merely ask for alternative answers.

Fourthly Prolog is a bi-directional language. Arguments can change from input to output according to demand, procedures can be driven both forwards as well as backwards and the list operator can both split and combine lists. An example is the classical list append or concatenate procedure:

```
append ([], List, List).
append ([Head|Tail1], List2, [Head|Tail3]) :-
  append (Tail1, List2, Tail3).
```

The first line says that it is a fact that one can append an empty list on to a List to give the same List. This is a base fact or terminating case. The second line says that it is a rule that one can append the list whose first element is Head and whose rest is Tail1, on to List2 to give the list whose first element is also Head but whose rest is Tail 3, if one can append Tail1 on to List 2 to give Tail3. This is a recursive rule. This code works both forwards:

```
?- Append( [ann, bert], [clare, dave], Class).
Class = [ann, bert, clare, dave]
```

and backwards to decompose a list into component lists:

```
?- append(List1, List2, [ann, bert]).
List1 = []           List2 = [ann, bert] ?;
List1 = [ann]        List2 = [bert] ?;
List1 = [ann, bert] List2 = [] ?
```

This pure approach begins to break down when procedural elements like assignment using the 'is' operator are used, but beginning programmers need not concern themselves with this.

Fifthly Prolog programs have a double interpretation since they can be read both declaratively, stating what conditions must hold in a solution, and procedurally, stating how to solve a problem.

Sixthly the fact that the main control structure is recursion and the main data structure is the list makes for very elegant solutions to list processing type problems, which bring out Prolog's logical simplicity. However, it has no explicit 'for' or 'while' loops and no arrays. The latter means the language is

Transactions on Information and Communications Technologies vol 7, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

unsuitable for problems involving random access to data structures, such as in numerical analysis, for which other languages, such as Fortran, are more appropriate.

4 Using Prolog as a first programming language

As an experiment the Information Technology undergraduates at the Royal Military College of Science, Cranfield University, have studied Prolog in their first term's computer programming module. This has laid the foundation for the following modules:

First year
- CP1 Computer programming
- DM1 Discrete mathematics

Second year
- FM2 Introduction to formal methods
- DB2 Databases and information retrieval
- PM2 Advanced programming methods

Third year
- LT3 Language theory and compiler design
- KB3 Knowledge based systems
- SA3 Advanced systems analysis

5 Disadvantages of Prolog

Unfortunately Prolog has some important disadvantages for teaching purposes. Firstly it is a difficult language to debug. The standard tracing mechanism, the Box model, is verbose and difficult to follow whilst graphical tracers have had only limited success. This problem is compounded by the tendency of Prolog programs to have long lists of numbered arguments, e.g. Tail1, List2, Tail3 above. These are hard to choose in a systematic but readable way.

Secondly it has no modularity, so although all variables are local to each predicate, all predicates are in a global name space. Some versions of Prolog have module extensions of variable utility.

Thirdly it is an unstandardised language with a variety of dialects. The National Physical Laboratory is currently developing an official (de iure) standard alongside the unofficial (de facto) Edinburgh syntax dialect.

Fourthly it is a weakly typed language which, although good for beginners,

increases the debugging problem and violates database domain integrity.

6 Applications of Prolog

Prolog has traditionally been seen as a language for Artificial Intelligence (AI) programming (as used in our 3rd year course in KBS). It is well suited to this application since it has a built-in backward chaining inference engine which takes an initial goal, fires appropriate rules until some base facts are reached. This involves depth-first searching (Lee et alia 1991). In addition it has a built-in pattern matcher which helps with symbolic processing. However Prolog is not just restricted to AI type applications.

Prolog can usefully be applied to various other parts of the curriculum, particularly formal methods, databases, systems analysis and language theory. In the area of formal methods it can be used to illustrate discrete mathematics (our course DM1), to animate formal specifications (our course FM2) and to write executable specifications (Leadbetter et alia 1991).

For databases (our course DB2) Prolog can be used to demonstrate the operations of the relational algebra (Malpas 1987). In the case of systems analysis (our course SA3) Prolog can be used as a data flow language to turn data flow diagrams directly into code without going through structure charts (Goble 1989). Finally Prolog's Definite Clause Grammar (DCG) subsystem can be used to teach language theory and compiler design (Fiddian et alia 1990 and our course LT3).

In addition we teach Prolog++ in our second year course (PM2) which builds on the first year logic programming with the object-oriented paradigm.

7 Conclusions

The authors concur with Bottino (1992) who, in comparing the use of Pascal and Prolog in secondary education, notes that after an initial simple approach Prolog exhibits a number of drawbacks, both conceptual and technical, when it becomes necessary to obtain a deeper knowledge.

8 References

1. Allen, J. R. Speaking Lisp *Computer Language*, 1985, **July**, 27-33.

2. Bottino, R. M. Comparing different approaches to programming from an educational viewpoint *Computers Educ.,* 1992, **18**, 4, 273-281.

Transactions on Information and Communications Technologies vol 7, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

3. Fiddian, N. J., Gray, W. A. & Howells, D. I. Query language inter-translation in heterogeneous distributed database systems *Database Technology,* 1990, **2**, 3/4, 3-8.

4. Furber, D. A survey of the teaching of programming to computing undergraduates in UK universities and polytechnics *Computer Journal,* 1992, **35**, 5, 530-533.

5. Goble, T. *Structured systems analysis through Prolog,* Prentice-Hall, 1989.

6. Leadbetter, P., Thomas, P. & Weedon, R. *Prolog: from specification to prototype,* Blackwell, 1991.

7. Lee, M. P., Jeffreys, S. & Peacock, D. "dBASE as a first programming language" *Collegiate Microcomputer,* 1989, **VII**, 2, 111-116.

8. Lee, M. P., Harrison, A. & Kent, A. E. Group projects for the software engineering of knowledge based systems, in SEHE '91 (ed G. A. King), pp. 95-107, *Proceedings of the first national conference on Software Engineering in Higher Education*, Southampton Institute, 1991.

9. Li, D. *A Prolog database system,* Research Studies Press, 1984.

10. Malpas, J. *Prolog: a relational language and its applications*, Prentice-Hall, 1987.

11. Peacock, D., Jeffreys, S. & Lee, M. P. The use of a structured project to teach program development, *ACM SIGCSE Bulletin* 1987, **19**, 4, 10-18.

12. Peacock, D., Ralhan, V. K., Lee, M. P. & Jeffreys, S. A first year course in software design and use, *ACM SIGCSE Bulletin*, 1988, **20**, 4, 2-8.

13. Woodhouse, D. Introductory courses in computing: aims and languages *Computer Education,* 1983, **7**, 2, 79-89.