APPROACHES FOR TEACHING PROLOG TO BEGINNERS

Shanshan Yang
Department of Computer Science
University of Warwick
Coventry, CV4 7AL
UK
Shanshan.Yang@warwick.ac.uk

Mike Joy
Department of Computer Science
University of Warwick
Coventry, CV4 7AL
UK
M.S. Joy@warwick.ac.uk

ABSTRACT

Prolog is a very different language compared with procedural or object-oriented languages, and developing a Prolog programming mindset is a challenge for many novices. In this paper, we consider a number of teaching approaches which instructors use to deliver basic ideas about Prolog to novices. We classify these approaches into three categories – "logic based", "declarative features based" and "programs based". Using this classification as a framework, we describe the prerequisite knowledge required for students to learn Prolog effectively. Finally, we describe how the choice of approach has changed over the past 25 years.

Keywords
Prolog, teaching methods, Logic, declarative programming

1. INTRODUCTION

Prolog is a declarative language, and as such is unlike procedural and object-oriented languages. It superficially appears easy to learn because it has simple program constructs and syntax [1]. However, developing a suitable programming mindset is a challenge for many novices, especially for those already schooled in the procedural or object-oriented paradigms [2,3,4]. It is important for a teacher to be aware of the factors [5,6,7,8] affecting students who are learning Prolog, and to be sensitive to the difficulties [9] students may encounter.

There is an old adage 'Well begun is half done', which reminds us the importance of the starting point in a learning process. In the context of learning Prolog, there are a number of approaches which are used for teaching novices during the initial learning phase. We are interested in establishing what sorts of Prolog teaching approaches are currently in use, what prerequisites are required for those approaches, and how they have been used in the past. In order to pursue this goal, we analysed a comprehensive selection of the Prolog textbooks currently available.

2. METHOD

The teaching approaches we have identified are based on the contents of 14 introductory Prolog textbooks. These textbooks are all available within the UK, either through bookshops, or through our University library. Based on publicity material (such as cover sleeves) and module web sites most of them have been highly recommended by Prolog instructors in UK universities, and we have confidence that the approaches taken by the authors are therefore valid. The books span 25 years, since the first publication of a Prolog textbook in the early 1980's by Clocksin and Mellish [10] to the latest written by Bramer in 2005 [11].

For each textbook, we scrutinized the first few chapters in order to identify the teaching approach used, and also (where appropriate) material from the editors' prefaces and from published reviews.

A qualitative analysis of these data was then applied to classify the teaching into a number of distinct approaches. We present the categories, and discuss how the classification is related to the teaching content, and how it may help to build the Prolog mindset. We then discuss how the choice of approach has changed over time.

3. PROLOG TEACHING OVERVIEW

Before we start to discuss the teaching approaches, we should consider at what learning stage the approaches are applied, and what content is covered.

We claim that learning Prolog can be divided into a number of distinct phases. The first phase links novices from different backgrounds into the Prolog community. The following phase extends students' knowledge to cover most or all of the language. A third phase involves developing programming techniques appropriate to the paradigms used by the language. There may also be an additional phase, containing material related to the application of Prolog programming skills to distinct application domains, such as AI or databases.

The first phase naturally happens at the start of Prolog learning process, the second and third phases may take place either consecutively or simultaneously as the leaning process moves forward, and the fourth phase usually follows the second.

There are three entities involved in Prolog programming – the human programmer or user, the Prolog program, and the Prolog system – and there are roles for each of them. The programmer develops a program, which consists of a series of facts and rules which is stored in the system's internal database. The programmer can query the database and wait for system to respond to the query. The system tries to satisfy each query by trying to deduce a response from the known rules and facts and return one or more answers.

There are some features in these processes which are specific to Prolog. The programs only specify the data items and the relationships between rules and facts – the program contains no algorithm to solve the problem. A Prolog program is *declarative*. The Prolog system determines the algorithm at run time to solve the problem by making deduction from the existing facts and rules [12].

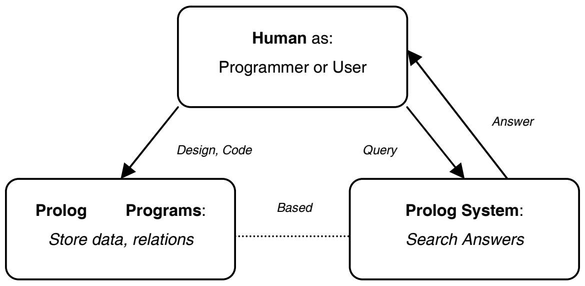

Figure 1: Relations between entities

Figure 1 illustrates the three entities and the relationships between them. The rectangles in the figure represent the three entities, text in italics describes the roles performed by the entities, and the arrows and lines show the relationships between them.

4. APPROACHES CLASSIFICATION

Prolog is a declarative logic language, and a number of different approaches to the delivery of basic Prolog ideas can be used, which can help learners build a Prolog mindset at the beginning of learning process. We have identified three categories of approach: *logic based*, *declarative features* and *programs based*. They are summarised as follows.

4.1 Logic based approaches

Logic based approaches are characterised by initially introducing concepts of mathematic logic and/or logic programming. Prolog is the best known logic programming language, which has first-order predicate logic as its theoretical foundation [13,14], and such an approach ensures consistency with its theoretical roots.

Methods which focus on mathematical logic cover first-order predicate logic, and map it to the Prolog language [13]. Authors who use a logic programming based method describe Prolog as a type of logic programming, and initially demonstrate the features that might be common to all logic programming (such as program constructs) followed by Prolog-specific features (both syntactic and semantic) [15,16]. Maier and Warren [17] use a method targeted at experienced learners which introduces the fundamentals of logic programming through viewing two interpreters – Proplog and Dataplog. These are used to focus on the formal semantics of logic programming, based on the ideas of zero-order logic and first-order logic, as the later one forms the foundation of Prolog. Lucas' method [12] is to combine a logic based approach with a program-based approach, and covers not only logical deduction and logical statements but also describes how to build up hands-on experience of interacting with the Prolog interpreter and handling programs.

4.2 Declarative features approaches

The second approach involves introducing the basic ideas of Prolog by viewing a number of its declarative features. A declarative language is a high level language [18] because instead of supplying instructions to the computer, the programmer supplies a formal specification of the problem to be solved and leaves the computer to decide how to solve it. Prolog as a declarative language includes a combination of declarative features [13], including knowledge specification and descriptive style programming, and contains both declarative and procedural semantics. These features can be covered in different ways while teaching. Some authors using models to describe Prolog such as specification tools [18, 19], database containers [13,15,20,21] and problem solving machines [22]. Some authors describe the relationships between Prolog programs and the Prolog system [12], whereas others present Prolog's declarative nature by describing problems using objects and relationships between them [23,24].

4.2.1 Specification tool based

This sub-approach involves viewing Prolog in its role as a specification tool, used to write specifications of problems and solve them. The methods normally first cover how to write a problem specification using Prolog, followed by materials about how to execute such a specification [18,19].

4.2.2 Database based

This sub-approach uses Prolog to set up and query a database. Initially using a database of facts, the approach then describes how the database can be augmented by more complex relation rules. The Prolog execution model is introduced while viewing the database queries [13, 15, 20, 21].

4.2.3 Problem solving based

Prolog is a problem solving machine, which can represent and solve problems. This sub-approach covers how a problem is described as a Prolog program, and how the Prolog system applies the knowledge related to this problem, to find the solution to the problem automatically [22].

4.2.4 System based

In this sub-approach, an overview of the Prolog system is used to introduce the key Prolog features. This first provides the relationships between Prolog programs and the Prolog system, and then shows how to interact with the Prolog system using different programs. Some logic concepts are also covered [12].

4.2.5 Known facts and relations based

This approach normally describes how to identify objects and relations from real world problems, and identifies basic elements of Prolog that can be used to describe and manipulate those objects and relations [23, 24].

4.3 Program based approaches

These approaches introduce the basic ideas of Prolog by showing the student sample programs. This serves two purposes: firstly, the student sees the program constructs contained in the programs, and secondly, is able to view the interactions between the programmer, the program and the Prolog interpreter. By viewing programs, learners can quickly get a feel of what Prolog programs look like, the components and structure of programs, and the syntax and technical vocabulary used to represent these components. By viewing interactions between that program and the Prolog system, learners can quickly get a feel of what role a Prolog program plays within the programming environment and how it is different from other languages' programs they have met before.

Some authors combine the two purposes [11, 25]. For example, Merritt [25] first provides an example of an adventure game with a natural language interface. Without describing its code in detail, he presents a sample run of this game with the interactions of the user and the Prolog interpreter. Then he presents and explains simple examples (commencing with a short two-line program), which are later linked back into the original motivational adventure game. Bramer [11] uses a short simple program to illustrate the terminologies that are commonly used in Prolog, both for the program framework and for data structures. Lucas [12] combines this approach with the logic based approach which has been mentioned before. He covers logical deduction and logical statements before describing how to build up hands-on experience of handling programs.

4.4 Approaches Evaluation

Entities* / *Categories	***Prolog Program***	***Prolog System***	***Human***
Logic Based	Logic statements	Performing logic deduction	None
Declarative Feature Based	Database, specification	Problem solving machine	Describe problem, query
Program Based	Source files with data and relations	Prolog interpreter	Develop, handle program

Figure 2: Relations between classification and entities

Figure 2 above illustrates how each of the three teaching approaches describes or views the three entities which are involved in developing and using Prolog software. The relationships between the approaches and the entities also suggest which pre-requisites are required for each of the approaches, such as abstract theory and related academic knowledge of computer science. For understanding the roles relative to logic based approaches in the second row of the table, knowledge of first order predicate logic would be required. For understanding the roles in the declarative feature based approach, a student would require knowledge of databases or program specification, which are topics which would normally be delivered elsewhere in a Computer Science course. Understanding the roles in the program based approach just requires basic familiarity with using computer systems.

Approaches	Years	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
Logic Based					🟩			🟩	🟩						🟩		🟩									
Declarative features	Specification Tool Based						🟥				🟥															
	Database Based				🟥		🟥	🟥				🟥														
	Problem Solving Based										🟥															
	System Based																🟥									
	Known Facts and Relations Based																					🟥		🟥		
Programs Based											🟦						🟦									🟦

Figure 3: Textbooks, teaching approaches and years of publication

5. Development of Approaches

After discussing the approaches which are currently available in textbooks, in this section we discuss the development of the choice of teaching approach over time.

We studied 14 textbooks, which contained 18 instances of the approaches we summarised here (since some textbooks have used more than one approach).

Using logic based approaches requires related abstract theory, and the Prolog mindset is developed abstractly. Using program based approaches requires practical experience, and the mindset is developed concretely. Approaches which are based on declarative features require both theoretical and practical knowledge – for instance, understanding databases cannot be done without related database theory and practical experience.

Table 3 illustrates the approaches taken by the textbooks and the years of publication. This shows that the logic based approaches were widely used during the 1980s and early 1990s, whereas declarative based approaches have been popular throughout the 25 years under consideration. The early use of specification tools is perhaps a relatively abstract type of declarative approach, in contrast to the concreteness of the more recent application of known facts and relations. The program based approach was not used before 1990, but continues to be an approach favoured by recent authors.

So we see a trend in which the approach used to introduce students to Prolog has become more practical and concrete, and hence less theoretical and abstract.

6. Conclusion

We have presented a classification of the approaches which textbook authors have adopted for introducing Prolog to novice students – logic based, declarative features based and program based. We have identified the prerequisites required for those approaches, and we have discussed the how the approach taken has become more concrete during the 25 years through which Prolog textbooks have been published.

7. References

[1] Hong J. Guided programming and automated error analysis in an intelligent Prolog tutor, *Int. J. Human-Computer Studies.* **61**, 505-534 (2004).

[2] Kumar A. N., Prolog for imperative programmers, *The Journal of Computing in Small Colleges.* **17**, 167-181 (2002).

[3] Antonio M. L. Jr., Supporting declarative programming through analogy, *The Journal of Computing in Small Colleges.* 53-65 (2001).

[4] Bieliková M. and Návrat P., Learning programming in Prolog using schemata, *ACM SIGPLAN Notices.* **33**, 41-47 (1998).

[5] Bergin S. and Reilly R., Programming: factors that influence success, *ACM SIGCSE Bulletin.* **37**(1), 411-415 (2005).

[6] Rountree N., Rountree J., Robins A. and Hannah R., Interacting factors that predict success and failure in a CS1 course, *ACM SIGCSE Bulletin* **36**(4), 101-104 (2004).

[7] Lewis T.L., Chase J.D, Pérez-Quiñones M.A., and Rosson M.B. The effects of individual differences on CS2 course performance across universities, *ACM SIGCSE Bulletin* **37**(1), 426-430 (2005).

[8] Lahtinen E., Ala-Mutka K. and Järvinen H.-M., A study of the difficulties of novice programmers*,* *ACM SIGCSE Bulletin* **37**(1), 14-18 (2005).

[9] Newmarch J., A plan-based approach to Prolog recursion*, ACM SIGCSE Bulletin* **25**(2), 12-18 (1993).

[10] Clocksin W.F. and Mellish C.S., *Programming in Prolog: Using the ISO Standard.* Springer-Verlag (1981).

[11] Bramer M., *Logic programming with Prolog.* Springer-Verlag (2005).

[12] Lucas B. *Mastering Prolog.* UCL Press (1996).

[13] Malpas J, *Prolog: a relational language and its applications*. Prentice-Hall International (1987).

[14] Horváth T., Sloan, R.H. and Turán G., Learning logic programs by using the product homomorphism method, *Proc. 10th Annual Conference on Computational Learning Theory,* 10-20 (1997).

[15] Clark K.L. and McCabe F.G., *Micro-PROLOG: programming in logic*. Prentice-Hall (1984).

[16] Sterling L. (2nd*), The art of Prolog*: *advanced programming techniques*. Cambridge University Press (1994).

[17] Maier D. and Warren D.S., *Computing with logic: logic programming with Prolog.* The Benjamin and Cummings (1988).

[18] Dodd T., *Prolog: a logical approach*. Oxford University Press (1990).

[19] Rogers J.B., *A Prolog primer.* Addison Wesley (1986).

[20] Marcus C., *Prolog programming: applications for database systems, expert systems, and natural language systems.* Addison Wesley (1986).

[21] Bowen K. *Prolog and expert systems.* Mc Graw-Hall International (1991).

[22] Konigsberger, H.K. and Frank W. G. M., *Prolog from the beginning*. McGraw-Hill (1990).

[23] Bratko I. (3rd), *Prolog programming for artificial intelligence.* Addison Wesley (2001).

[24] Clocksin W.F. and Mellish C.S., *Programming in Prolog: Using the ISO Standard (5th edition).* Springer-Verlag (2003).

[25] Merritt D., *Adventure in Prolog.* Springer-Verlag (1990).