Approaches for Learning Prolog Programming

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

Mike Joy
Department of Computer Science
University of Warwick
Coventry, UK
m.s.joy@warwick.ac.uk

Abstract: Understanding Prolog programming is a challenge for many beginners, and many instructors and researchers have proposed and adopted varied teaching approaches to help learners to understand Prolog easily. However, it is not known whether any of the teaching approaches currently in use is appropriate from a learner's perspective. This paper reports the results of an investigation into which approaches are suitable for Prolog novices and why they are appropriate.

We first categorise the teaching approaches which are currently in use in mainstream Prolog introductory textbooks, and highlight how they have been used over the past 25 years. We then discuss the appropriateness of each approach, by addressing their advantages and disadvantages based on interviews with learners. Using this information, a larger picture of the suitability of these teaching approaches is drawn, and finally we suggest what factors may influence it and discuss possible improvements.

The findings of this study suggest that an approach based on emphasising the declarative features of the language is the most appropriate, as a component of a blended learning strategy.

Keywords: Learning and Teaching Programming, Approaches, Logic Programming,

Prolog

1. Introduction

Learning logic programming is a standard component of the curriculum offered by many universities in the UK (Van Roy et al, 2003). Even though it superficially appears easy to learn because it has simple program constructs and syntax, learning Prolog is still a challenge for many novices (Hong, 2004; Pentti, 1993), and the language itself is often considered difficult to understand. A reason might be that Prolog uses a different paradigm compared to conventional languages, and even experts in other languages can still struggle with Prolog (Kumar, 2002; Lopez, 2001; Schreiweis, 1993).

Indeed, a number of research studies have tried to address this difficulty. Many instructors and researchers have proposed and adopted varied teaching approaches to assist learners to understand Prolog easily. Kumar (2002) identifies a number of misconceptions which are made by imperative programmers when they start to learn programming in Prolog. He also attempts to dispel them by "recasting the semantics of commonly misunderstood data and control constructs in Prolog in terms of the more familiar imperative constructs." Lopez (2001) presents a methodology to teach logic programming by the use of analogy, which is used to "understand something new in terms of something familiar". He suggests giving students declarative programming examples illustrating different concepts, and then asking them to write their own programs which contain the same concepts but in a given analogous domain, so the students can learn from the process of the analogy (Lopez, 2001; Iding, 1997).

However, it is not known whether any such teaching approach is appropriate or not from the learner's perspective. We therefore conducted a study to investigate which approaches are suitable, and why, based on students' learning experiences.

In order to determine which teaching approach is suitable for learners, our research

addresses the following issues. We first identified the teaching approaches which have been adopted in Prolog introductory textbooks, and how they have been used over the past 25 years. We then highlighted the appropriateness of each approach, by addressing their advantages and disadvantages based on learners' opinions obtained through interviews. Using this information, we present a picture of these teaching approaches, and suggest reasons for the suitability (or otherwise) of each. Finally, we suggest how this information might be used to improve the delivery of initial Prolog teaching.

2. Methodology

Two investigations were performed in this research.

First, a literature survey was adopted to identify the approaches used in currently available Prolog introductory textbooks. 14 textbooks were considered, each of which is either currently in use in a UK university, or comes highly recommended by Prolog instructors in an educational institution. The books span 25 years, since the first publication in the early 1980s by Clocksin and Mellish (1981) to the most recent written by Bramer (2005). For each textbook, the first few chapters, and also (where appropriate) material from the editors' prefaces and from published reviews, were scrutinised in order to identify the teaching approach used. A qualitative analysis of these data was then applied to classify the teaching into a number of distinct approaches, which were then analysed to illustrate how the choice of approach has changed over time.

The second investigation involved individual interviews with 11 participants, each of whom was either an undergraduate or a postgraduate student in the authors' Computer Science Department who had previously learned Prolog. An open-ended semi-structured interview format was chosen in order to explore students' experiences of different Prolog learning approaches. A list of the different Prolog teaching approaches were presented and explained to each student, and then the discussion was focused upon the three key topics

– their preferences for Prolog learning approaches, the reasons for their preferences, and suggestions for improving the approaches. Students were asked what they thought about each approach and why, and asked to suggest how the approaches might be improved in future years, and students were also given the opportunity to describe their experience in learning Prolog as whole. The discussions were subsequently analysed by coding transcripts and identifying key themes in the data.

3. Teaching approaches

In order to study which teaching approaches are suitable for Prolog learners, we first identified what approaches are currently available based on Prolog introductory textbooks. In this section, we present an overview of teaching approaches which instructors use to deliver basic ideas about Prolog to novices, and summarise the detailed analysis published elsewhere (Yang and Joy, 2006).

Prolog has first order predicate logic as its theoretical foundation (Chezzi, 1998; Brna, 1993; Cohen, 1988), and for the purpose of delivering basic ideas about Prolog to novices we are interested in how much theoretical background knowledge learners might need in order to help them to understand Prolog easily. We analysed a comprehensive selection of the Prolog textbooks currently available to identify the teaching approaches used. It appears that three categories of distinct approaches have been taken: “logic based”, “declarative features based” and “programs based”. Logic based approaches cover the abstract theories of mathematical logic and/or logic programming. Program based approaches require practical experience, and the Prolog mindset is developed concretely. Approaches which focus on Prolog’s declarative features require both theoretical and practical knowledge.

In the rest of this section, we first cover Prolog teaching background which these approaches relate to, we present the categories, and finally we describe how the choice of

approach has changed over the past 25 years.

3.1. Prolog teaching stage and content

Before we start to discuss the teaching approaches, we first consider the learning stage and learning content the approaches are applied to. The teaching approaches we investigated are focussed on the first phase of the learning process – linking learners from different backgrounds into the Prolog community.

These teaching approaches cover a brief introduction about what Prolog is, rather then the subsequent details of how to make use of it. There are three components involved in the programming environment – human users, the Prolog program, and the Prolog system (Bramer, 2005; Clocksin and Mellish, 2003; Bratko, 2001; Lucas, 1996; Sterling, 1994; Bowen, 1991; Dodd, 1990; Konigsberger and de Bruyn, 1990; Merritt, 1990). 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 the 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. 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.

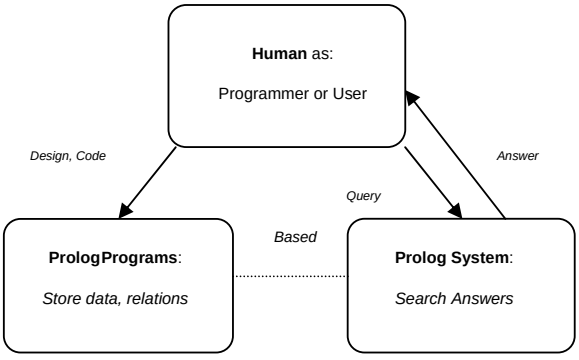

Figure 1: Prolog Learning Content

ITALICS Volume 6 Issue 4, October 2007
ISSN: 1473-7507

There are some features in these processes which are specific to Prolog. Programs only specify the data items and the relationships between rules and facts – a 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. The details of different ways of introducing Prolog to novices are presented in the next section (Lucas, 1996).

3.2. Logic based approaches

Logic based approaches are characterised by initially introducing the concepts of mathematical logic and/or logic programming. Prolog is the best known logic programming language, which has first-order predicate logic as its theoretical foundation, 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 (Malpas, 1987). 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) (Sterling, 1994; Clark and McCabe, 1984). Maier and Warren (1988) 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 and first-order logic, as the latter forms the foundation of Prolog. Lucas' method (1996) 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.

3.3. 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 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 (Dodd, 1990). Prolog as a declarative language includes a combination of declarative features (Malpas, 1987), 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 use models to describe Prolog such as specification tools (Dodd, 1990; Rogers, 1986), database containers (Bowen, 1991; Malpas, 1987; Marcus, 1986; Clark and McCabe, 1984) and problem solving machines (Konigsberger and de Bruyn, 1990). Some authors describe the relationships between Prolog programs and the Prolog system (Lucas, 1996), whereas others present Prolog's declarative nature by describing problems using objects and relationships between them (Clocksin and Mellish, 2003; Bratko, 2001).

3.3.1. 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 (Dodd, 1990; Rogers, 1986).

3.3.2. 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

(Bowen, 1991; Malpas, 1987; Marcus, 1986; Clark and McCabe, 1984).

3.3.3. 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 (Konigsberger and de Bruyn, 1990).

3.3.4. 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 (Lucas, 1996).

3.3.5. Known facts and relations based

This approach 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. For example, Clocksin and Mellish (2003) first use a number of examples in real life to introduce the concepts of objects and their relationships. Then they discuss how to represent objects and their relations in Prolog, and how to make use of them to solve problems.

3.4. Program based approaches

These approaches introduce the basic ideas of Prolog by showing sample programs to the students. 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 while adopting this approach (Bramer, 2005; Merritt, 1990); others combine this approach with other approaches. Lucas (1996) combines this approach with the logic based approach which has been mentioned above.

3.5. Prolog teaching trends

In this section we highlight how the approaches, which are currently available in textbooks, have been used over the past 25 years. 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.

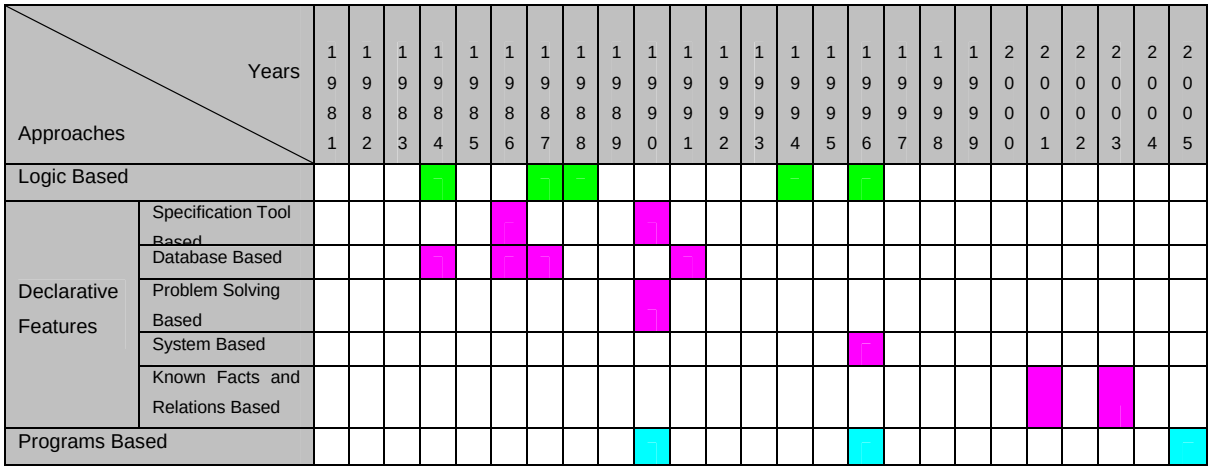

Table 1: Prolog Teaching Trends

Table 1 above 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.

4. The advantages and disadvantages of each approach

It is beyond the scope of this paper to identify with certainty any preferred approach, however the interviews have highlighted many of the advantages and disadvantages associated with each. These suggest that, based on students' past learning experiences, a blend of different approaches would be appropriate.

4.1. Database based approach

The evidence suggests that the database approach is one of the most appropriate approaches for learning Prolog. Although it has not been adopted in our current Prolog module, 8 out of 9 learners mentioned that this approach works for them. As one student commented:

> *"I think after the database approach, it is easy to learn the rest of Prolog"*

In general, learners were very positive about this approach because their prior knowledge of databases appeared to be an abstract learning model which was particularly helpful in

learning Prolog:

> *"You declare something and then you enter some query, and then search for the truth…it is more like a database"*

Prolog can be viewed systematically, and there are similarities between Prolog and database languages such as SQL, in particular the sub topics 'Facts' and 'Queries'. However, although some students were able to transfer their database knowledge to their understanding of Prolog, there were also negative transfer effects. A clear and consistent message emerged from students' comments: there are differences between Prolog and database languages, particularly because Prolog is more complicated and there is "recursion" in Prolog. These differences should be pointed out, otherwise learners might not manage to transfer this knowledge, as illustrated by the following quotes.

> *"Prolog is much more complicated, you specify rules which can be quite complex"*

> *"It is not exactly the same…there will be differences from databases … it should be mentioned at the beginning, otherwise people will get mixed up"*

4.2. Known facts and relations based approach

Only 5 students commented about this approach, but all of them were positive, because learners think it is concrete, and allows them to apply Prolog to real problems.

> *"Because you can take something you know about … something in the world … you create examples in Prolog based on that"*

> *"It is a systematic approach … to transform the problem and requirements into programs"*

4.3. Program based approach

This approach is partially adopted in the authors' department, and uses program examples

to introduce what Prolog is about. It is practical, concrete and close to the real world situation, as the following two comments on the practical nature of the approach highlight.

> *"It is practical … you get to see what is happening … understanding is better when you have practical experience on it than just theory"*

> *"If you show people sample programs … you are able to explain them and show people how to related to things related to the real world … people will be able to pick things up"*

There are mixed comments about this approach – 5 out of 11 claimed that this approach is suitable for them, but 7 of them suggested otherwise. Although some students were able to pick up Prolog quickly by viewing sample programs and their interactions, other learners still struggled:

> *"Some people have very strong feelings about programming languages – they can learn a lot, once they look at the program, but some people can't"*

In order to manage to use this approach, learners also need to have mastered the skill of learning from programs, they need to be able to practice running sample programs, and they also need to have other relevant knowledge, such as data structures, program constructs, and declarative constructs:

> *"Without any previous knowledge in Prolog, you only show me the code and related tips, without any data structures knowledge, that must be very confusing … even you want to test it, you still don't know how to do it"*

There is also the disadvantage that coverage by example requires sufficient code samples to be viewed.

> *"I think it is easy to see things in a quite narrow view … [for example,] [program] construct … you don't have the understanding of this for general purposes … many*

different programs, many different variations, you can't give all the aspects of Prolog"

4.4. Problem solving based approach

The evidence collected from the interviews does not suggest this approach is appropriate for learning Prolog, since the comments for this approach were divided equally between positive and negative. Learners are interested in this approach because it is a new way of viewing Prolog – programming in Prolog is about describing and solving problems:

"You have a problem at the start, you write it on paper how you progress from your thought, like writing down in words, and then query. It is better than doing Prolog from the start"

However, some students believe this approach is too general:

"I think all the languages are used to solve problems"

Furthermore, problem solving as an initial approach was viewed as too complex:

"We attempted to solve problem using Prolog at the start of learning Prolog … but I don't know how to transform problem to statements … how to use Prolog to solve my problem … it is not a syntax and technical problem, it is a problem of how you transform your knowledge into a program"

4.5. System based approach

This approach has not been applied in our department's Prolog module, and we have no evidence to suggest that it is appropriate as an introduction to Prolog. However, learners appeared positive about this approach, since they felt it allows them a deeper understanding about Prolog, especially for learners who already have previous knowledge of other languages:

"I think knowing what is going on behind, knows how it unifies … is useful … especially, if you want to optimise the program"

Not all of the students interviewed were positive about this approach, because they felt it is not necessary to know in depth how the Prolog system works when they start to learn the language, as this might be confusing:

> *"I thing people must know other systems in other programming languages well … such as Java systems … otherwise they might get mixed up … which we don't want it happen"*

4.6. Logic based approach

The evidence collected from student interviews suggests this approach is inappropriate, although it has been adopted in our department's Prolog module for many years – 7 out of 10 participants disagree with this approach, and only 3 out of 11 considered the approach to be appropriate for them. The major advantage is that mathematical logic is the foundation for learning Prolog, and knowledge of discrete mathematics and predicate logic are helpful for understanding Prolog concepts later on. One student remarked:

> *"Prolog was born from the logic …Apart from the operating side of Prolog …having the knowledge of logic works particularly on me"*

Students considered this approach to be abstract, and noted that they needed to learn mathematical logic first:

> *"Logic based approach… you don't get any example in real life"*

Since this is a topic which most students are not familiar with before, this would lengthen the time for them to learn Prolog. In addition, mathematical logic itself in fact is perceived as difficult to understand. Furthermore, even when learners can understand mathematical

logic, there is still a conceptual gap between mathematical logic and Prolog:

> *"There is some gap between logic and Prolog … it is not straight forward … concept shifting from the maths to Prolog"*

4.7. Specification tool based approach

This approach is one of the less popular approaches commented on by students, and although it has not been adopted in our current Prolog module, 6 out of 6 students had negative comments about it. As teachers, we may consider that using an abstract metaphor to introduce Prolog makes it easy and quick to build a whole picture of the language, but this perception is not shared:

> *"Specification only talks about the inputs and outputs; you still have to learn Prolog at the end of the day"*

Furthermore, most students seem not to understand specification when initially introduced to it, and this has led to confusion:

> *"When I think about specification, I think about very long documents, I think about Z notation, which scares me"*

5. Discussions and improvements

In this section we first summarise the appropriateness for each approach, and use the information we have collected to identify possible opportunities for improving these approaches in the future.

Table 2 illustrates which approaches are appropriate for most Prolog beginners, based on the evidence we have collected. The database and the known facts and relations approaches appear to be appropriate, whereas the logic based and the specification tool based approaches might not be suitable for most learners.

Results / Approaches	Appropriate	Inappropriate	Cannot Decide
Logic based		✓	
Specification tool based		✓	
Database based	✓		
Problem solving based			✓
System based			✓
Known facts and relations	✓		
Program based			✓

Table 2: Appropriateness of Teaching Approaches

Table 3 summarises the preferences of the 11 individuals interviewed. The meanings of each symbol are presented as below,

✗☼: means student thinks this approach is inappropriate for him/her, and gave reasons

✓☼: means student thinks this approach is appropriate for him/her, and gave reasons

✓: means student thinks this approach is appropriate for him/her, but did not give any clear reason

✗: means student thinks this approach is inappropriate for him/her, but did not give any clear reason

Student ID / Approaches	1	2	3	4	5	6	7	8	9	10	11
Logic based	×☼		✓	×☼	×☼	×☼	×	✓	✓	×☼	×☼
Specification tool based				×☼	×☼	×☼	×☼	×☼			×
Database based	✓☼			✓☼	✓☼	×☼	✓☼	×☼	✓	✓☼	✓
Problem solving based	×☼			✓☼		×☼	✓☼	✓☼		×☼	×
System based	✓☼			×☼			✓☼	✓☼	×☼		
Known facts and relations based				✓	✓☼	✓☼			✓☼		✓
Program based	✓	×☼	✓☼	✓☼	✓☼	×☼	✓	×☼	×☼	×☼	×☼

Table 3: Learners' Preferences for Each Approach

5.1. Message 1: The most appropriate approach contains both concrete and abstract components

The two most popular approaches – database based and known facts and relations based – both belong to the category of declarative feature based approaches, and contain both concrete and abstract components, combining both theoretical and practical knowledge.

5.2. Message 2: Several approaches are appropriate

Our results suggest that learner preferences are important, since three of the approaches – the program based, the problem solving and the system based approaches – are also suitable for some individual learners. As the former one is practical, the second one provides new ways of viewing Prolog, and the latter one allows learners to be able to understand about Prolog more deeply.

Although the logic based approach is not suitable for most learners, some students still

believe this approach is useful for them.

5.3. Blended learning strategy

Table 3 illustrates that different learners prefer different learning approaches. We can see that students who prefer a concrete program based approach tend not to enjoy the logic style of approach, and conversely this suggests to us that there are two principle styles of learners. In order to address the different styles of learners, we propose to adopt several approaches to teach Prolog; these approaches are blended together and contain both concrete and abstract components (Clark, 2003).

6. Conclusions and further research

We have presented a study of teaching approaches for learning Prolog programming. Those which are currently in use are described, and we have highlighted how the choice of approach has changed over the past 25 years. Using data obtained from interviews with students who have already learned the language, we have compared the approaches used for introducing Prolog to beginners. We note that approaches based on initially emphasizing its declarative features appear to be particularly appropriate, and we also suggest that a blended teaching strategy should be adopted to meet different styles of learners.

In order to develop this research, a larger sample of students, and the possibility of carrying out a controlled experiment on a set of students currently learning Prolog, would help us to understand in more detail the appropriateness of individual approaches, and how successful different blends of those approaches might be.

References

Bowen, K. (1991), *Prolog and Expert Systems,* McGraw-Hill, New York

Bramer, M. (2005), *Logic Programming with Prolog,* Springer-Verlag, New York

Bratko, I. (2001), *Prolog Programming for Artificial Intelligence,* Addison Wesley, Wokingham

Brna, P. (1993) Teaching Prolog Techniques, *The Sixth International Annual Conference of the PEG Group on Knowledge Based Environments for Teaching and Learning*, pp 111-117

Chezzi, C. (1998), *Programming Language Concepts,* Wiley, Chichester

Clark, D. (2003), *Blended learning*, Epic White paper, Brighton

Clark, K.L. and McCabe, F.G. (1984), *Micro-PROLOG: Programming in Logic*, Prentice-Hall, London

Clocksin, W.F. and Mellish, C.S. (1981), *Programming in Prolog,* Springer-Verlag, New York

Clocksin, W.F. and Mellish, C.S. (2003), *Programming in Prolog (5th ed.)*, Springer-Verlag, New York

Cohen, J. (1988), The View of the Origins and Development of Prolog*, Communications of the ACM* 31(1), pp 26-36

Dodd, T. (1990), *Prolog: a Logical Approach*, Oxford University Press, Oxford

Hong, J. (2004), Guided Programming and Automated Error Analysis in an Intelligent Prolog Tutor. *International Journal of Human-Computer Studies,* 61(4), pp 505-534

Iding, M. (1997), How Analogies Foster Learning from Science Texts, *Instructional Science,* 25(4), pp 233-253

Konigsberger, H.K. and de Bruyn, F.W.G.M. (1990), *Prolog from the Beginning*, McGraw-Hill, London

Kumar, A. (2002), Prolog for Imperative Programmers, *Journal of Computing Sciences in Colleges,* 17(6), pp 167-181

Lopez, A.M. (2001), Supporting Declarative Programming through Analogy, *Journal of Computing Sciences in Colleges,* 16(4), pp 53-65

Lucas, B. (1996), *Mastering Prolog,* UCL Press, London

Maier, D. and Warren, D.S. (1988), *Computing with Logic: Logic Programming with Prolog,* Benjamin / Cummings, Wokingham

Malpas, J. (1987), *Prolog: a Relational Language and its Applications*, Prentice-Hall International, London

Marcus, C. (1986), *Prolog Programming: Applications for Database Systems, Expert Systems, and Natural Language Systems,* Addison Wesley, Reading

Merritt, D. (1990), *Adventure in Prolog,* Springer-Verlag, New York

Pentti, R. (1993), Teaching AI through Prolog Programming Techniques, *Computer Education* 20(1), pp 133-139

Rogers, J.B. (1986), *A Prolog Primer,* Addison Wesley, London

Schreiweis, U. (1993), An Integrated Prolog Programming Environment, *ACM SIGPLAN Notices,* 28(2), pp 53-60

Sterling, L. *(*1994), *The Art of Prolog*: *Advanced Programming Techniques*, Cambridge University Press, Cambridge

Van Roy, P., Armstrong, J., Flatt, M. and Magnusson, B. (2003), The Role of Language Paradigms in Teaching Programming, *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pp 269-270

Yang, S. and Joy, M. (2006), Approaches for Teaching Prolog to Beginners*, Proceedings of the 7th Annual Conference of the HEA Network for Information and Computer Sciences*