papers on pragmatics and reasoning try to exploit the natural application of logic programming for knowledge representation and logical reasoning, in order to interface this with the output of the parser and to produce useful and co-operative dialogue from the computer.

This collection of papers will be an important reference and a source of up to date material for the NL researcher, and the panel discussions at the end are stimulating pointers to future developments. The book should make the reader look forward to the next workshop, due to take place in Copenhagen in August 1990.

Expert systems development in Prolog and Turbo Prolog by Peter Smith, Sigma Press (distributed by Wiley & Sons), 1988, 214 pp, £12.95.

Physical Description

2 appendices (an ASCII table and a list of operators)
1 bibliography (not annotated)
13 short exercises
Solutions to 11 selected exercises
13 Full program listings (7 long and 6 short ones)

Introduction

The use of logic programming in commercial applications is getting more popular nowadays. Prolog can be used to build up expert system shells for various applications in financial forecasting, planning, decision-making . . . etc.

The idea of logic programming stems from Alain Colmerauer at Marseilles, Robert Kowalski and Maarten van Emden at Edinbugh and is implemented in *Prolog* (Programming in Logic) by various people at Edinburgh. Very often, Artificial Intelligence (AI) people like to refer it to Edinburgh Prolog. Now, we have various implementations which are slightly different from Edinburgh Prolog, such as Borland's Turbo Prolog. Smith responds to such a trend and writes a book on his remarkable book on expert systems development based on Edinburgh Prolog and Turbo Prolog.

Prolog works by LUSH resolution.[1] It is a declarative programming language. A typical Prolog program consists of *facts* (predicates) and *rules*. It is like a series of first-order predicate logic statements rewritten in Horn clauses.[2] In a sense, it is like a collection of IF . . . THEN statements, e.g.

```
predecessor (X, Y):-    % X is the predecessor of Y IF
    parent (X, Y).      %   X is the parent of Y
```

Many Information Technology (IT) workers choose Prolog for their application because Prolog is easier to learn when compared with other procedural languages like FORTRAN and PASCAL. It can be effectively and efficiently used to build up Natural Language Front-ends, powerful relational database systems and management information systems.

Structure of the book

This book can be divided into two distinct parts: a gentle introduction to expert systems development and some basics of Prolog programming.

The first part deals with the history and application of expert systems. Smith gives a brief introduction to expert system application and a history of the Alvey programme in response to Japan's Fifth Generation project.

[1] LUSH resolution is a special kind of theorem proving technique.
[2] A specially written form of logical clause with only ONE non-negated clause on the left hand side.

According to Smith, a typical expert system consists of a knowledge base, a user interface and an inference engine plus a knowledge acquisition system. He concentrates on how to build up the knowledge base by Prolog predicates (facts), the inference engine by Prolog rules and the user interface.

Six chapters of eleven are devoted to how to write Prolog programs. In each of these chapters, Smith introduces different predicates with small everyday life examples, plus a section on how to rewrite all these in Turbo Prolog. These include debugging facilities (**spy**, **trace**), backtracking, **cuts** and **fail**, recursion and some 20 built-in predicates.

The second half of the book consists of two major parts: user-interface design and the development of an expert system. Smith introduces **Definite Clause Grammar** (DCG) to illustrate the power of Prolog in interpreting context-free grammar rules. He also gives a naive example to illustrate how a Natural Language Front-end can be built: he suggests that the computer can use a predicate to transform the user's input like *I am* into *you*, *my* into *your* and respond to the user.

In his proposed expert system framework, Smith asserts that we can build the knowledge base by Prolog **facts**. The user-interface can be built up by Input/Output facilities (**read**, **write**), DCG rules and his *transform* facilities in his example. The inference engine can be thought of as a collection of Prolog rules, analogous to IF/THEN statements. Next, Smith proposes that we can use **asserta** and **assertz** predicates to store new facts in the knowledge base (i.e. his knowledge acquisition tools!). In addition, he suggests that the use of **clause/2** can be another powerful tool in explaining how the expert system gets the conclusion. According to his example in pp 151–156, he shows us a program which identifies the type of a dog by a collection of facts and rules about different attributes of dogs. After the user has entered the attributes *solid*, *alert* and *long coat*, the computer concludes that the dog the user describes is a Pekingese and displays how it gets the conclusion:

```
identify (pekingese):-    % the rule that computer gets
   group (toy),           %  the conclusion
   appearance (solid),
   appearance (alert),
   coat (long).
```

The computer shows how it gets such a conclusion by displaying:

```
group (toy), appearance (solid), appearance (alert), coat (long).
```

A simple call of **clause/2** can do that job for us! Smith suggests that having the computer to explain how it gets the deduction is not as difficult as we think.

Criticism and Comment

Audience: Novice Programmers

This book is suitable for readers who have no experience in programming or in Computer Science/Artificial Intelligence.

Smith is very helpful in introducing how to use the Prolog interpreter at the beginning of his book:

1 [**user**]. To enter facts and rules interactively without quiting the interpreter.
2 "." every Prolog predicate or fact ends with a full stop. Otherwise a "|:" prompt will pop up. Many beginners are frightened by it and get stuck.
3 ^**Z** (**or** ^**D**) to exit the interpreter.
4 ";" for more answer after Prolog returns the first solution.
5 "," means the conjunction "and".
6 ":-" corresponds to "if". In Turbo Prolog, both "," and ":-" can be spelled out as *and* and *if* respectively.

These little things often annoy and even frighten novice Prolog users when they evoke any Prolog interpreters.

For Turbo Prolog users, Smith has given enough instructions in using the system. However, he does not give any guidance on using the WordStar-like editor. Nor does he explain what symbols are, or why there are no full stop. in *predicates* in Turbo Prolog.

A good glossary can be found in chapter 4, which is very easy to read, when compared with the so-called Prolog classic textbooks.

In my opinion, this book is *not* suitable for practitioners, students or researchers in AI. This book oversimplifies what an expert system is, by using naive, toy exercises (e.g. pub guide, tictactoe and the travelling salesman). The so-called *case studies* in chapter 11 are far from the expert systems in the real world. There are no technical details of how to build up a robust Natural Language Front-end.

Assessment

The part on Prolog programming style is weak. Smith does not explicate how to write Prolog declaratively. There is no particular didactic strategy in this book.

In addition, Smith does not distinguish red and green cuts in his book. Nor does he emphasize that using (red) cuts is not a good Prolog programming style.

The treatment of user interface design is inadequate. Smith mentions DCG briefly on p 137 but he does not illustrate how an expert system can make use of it. Instead, he introduces a *transform/2* mechanism which is only a naive translation device, not a practical user-interface.

Smith does not give sufficient exercises (13 exercises in total) for the readers. Not all of them are given solutions.

Style and Layout

Smith has a talent in writing in simple, easy-to-read prose. His definition of some Prolog jargons like instantiation, object, is impressive. There are a few diagrams in this book which are helpful in understanding the exercises. The layout of the book is clear.

Errors

Though the book claims to provide tested and working programs, I found the following printing errors:

	Errors	**Should be:**
p 119	write('My name is),	write('My name is '),
p 167	option2:meno):-	option2(ameno):-
p 168	member(:[_|Y]):-	member(X,[_|Y]):-
	member(X,Y).	member(X,Y).
	areas([' town_centre ',	areas([' town_centre',
	warwick_road,paddock_lane).	warwick_road,paddock_lane]).
p 172	route(Town,Trown,_:Town],	route(Town,Town,_,[Town],
	0):-!.	0):-!.
	route(Town1,Town2,Covered,	route(Town1,Town2,Covered,
	[Town1|:y],Dist):-	[Town1|Way],Dist):-
	go(smalltown,bigtown15).	go(smalltwon,bigtown,15).
p 182	input_comp(N:,[Comp_name|L1]):-	input_comp(N,[Comp_name|L1]):-
p 183	input_cond(:m,[Cond_name|L1]):-	input_cond(Num,[Cond_name|L1]):-

Selected Bibliography

Brakto, I, 1986. *Prolog Programming for Artificial Intelligence*, Wokingham: Addison Wesley
Clocksin, W and Mellish, C, 1984. *Programming in Prolog*, Second Edition, New York: Springer-Verlag
Gallier, JH, 1987. *Logic for Computer Science: Foundations of Automatic Theorem Proving*, Singapore: John Wiley

Database applications using Prolog by Robert Lucas, Ellis Horwood Ltd, 1988, pp 159, £19.95.

The Alvey programme over the last few years has, to some extent, been an attempt to spread the new "fifth generation" methods out of the ivory towers and into industry and commerce. In many ways this has been successful, and numerous collaborations between academic research groups and commercial and public sector organizations have been established. Nevertheless there is still a widespread view that as yet the new technology has not been fully transferred. In particular, in the field of knowledge-based systems, and specifically Prolog-based ones, there seems to be a lack of significant "real" applications. This situation makes recent government cutbacks in funding for such collaborations difficult to counter, reinforces a tendency for the research community to be inward looking and encourages industrial partners to perceive that community as impractical symbol pushers. One aspect of *Database applications using Prolog* by Robert Lucas, the product of collaboration between the Department of Computer Science at Coventry Polytechnic and local industry, is that it makes a welcome contribution to the process of technology transfer. It has to be said, however, that the book is concerned more with the first stage of transfer, producing prototype systems, than the final stage of fully operational industrial systems.

We are told (p 12) that "the whole thrust of the work described was based on the observation that we can use a database as a repository for facts for the Prolog system". An important aim of the work was, therefore, to provide an efficient interface between a Prolog front-end and a conventional database. As well as a thorough discussion of the construction of this interface and three prototype applications developed using the system, a good part of this short book (about 40 pp) is devoted to reviewing the elements of Prolog, relational databases and expert systems. While this is moderately successful it suffers from attempting to cover in a very short space material which is accessible elsewhere. On balance, however, the completeness gained is probably worthwhile, particularly because it encourages the use of Prolog from a logical point of view.

The major focus of the book is the design and implementation of a tightly coupled interface between Prolog and a commercially available relational database (the Mimer database). The detailed account of the problems which arise in the construction of the interface provide a useful source of information for others involved in similar projects, even though the implementation, in Pascal, is specific to the Mimer database. The final implementation is tested for performance. This is clearly of crucial importance because an underlying assumption of the composite Prolog plus database design is that it can achieve efficiencies better than a pure Prolog system, and also that these efficiencies are acceptable in comparison to other database access methods. The tests carried out, although rather limited, show tentative support for the design. The real value of this part of the account, however, is methodological, some tests are much better than no tests at all.

The strength of this book is in the example applications. The most entertaining of these, devised in collaboration with the police, is concerned with the detection of burglaries. Apparently this activity is bogged down with paper work; scene of the crime reports are made on paper, background data is supplied on paper and so on. One aspect of the project was to provide a flexible database to replace this paper work. The other was to model a particular aspect of the detection task carried out by the burglary squad: the value of information on the modus operandi of burglars, for example whether they prefer upstairs windows or are deterred by dogs. We are suitably appalled by the fact that "only 12% of burglaries are investigated by the squad", in the remaining cases the police simply wait for the perpetrators to be picked up for something else, or not as the case may be. The completed system was set up in a police station for evaluation, but the burglary squad "showed little interest in it" even