defined over the state of the system and then presents some methods for controlling and managing distributed processes.

This book considers some of the fundamental issues involved in distributed computation. As such it could provide a core book of the field. The book is targeted at final year undergraduate and postgraduate students. However, the book has one major flaw, it is badly written or may be badly translated (the original being in French). In short, this book contains a great deal of information but the presentation of this information is obscured by the style in which the book is written. I cannot recommend it.

Natural language understanding and logic programming, II. Proceedings of the Second International Workshop on Natural Language Understanding and Logic Programming, Vancouver, Canada, 17–19 August 1987, Elsevier Science Publications, The Netherlands, June 1988, pp 346, Approx $84.25, ISBN 0-444-70408-6.

Natural Language (NL) parsing has been one of the main applications of Prolog since the language was developed in the early 1970s, and many logic based formalisms have been designed based on Logic Programming, such as Colmerauer's Metamorphosis Grammars, Pereira and Warren's Definite Clause Grammars, and many other Logic Grammars.

One of key ideas at the root of using Logic Programming for NL parsing consists of viewing parsing as theorem proving in a logic in which the lexicon and the grammar are interpreted as axioms and deduction rules. The Prolog language, as a theorem prover for Horn clauses, can easily be used as a parser if the grammar rules are written as Horn clauses. To take a simple example, we re-interpret the usual phrase structure rule that states that a Sentence rewrites as a Noun Phrase (NP) followed by a Verb Phrase (VP), and say instead that in order to prove that a given string is a sentence, we have to prove that it is the concatenation of two strings, the first of which is a Noun Phrase, and the second is a VP. Computationally efficient algorithms for theorem proving can then be used to parse NL.

More recently, this last decade has seen the appearance of a large number of unification-based grammars (i.e. those which use the logical operation of unification to pass information between linguistic objects), together with formalisms in which to express these, such as Shieber's PATR-II.

The general aim is to replace traditional transformational accounts of grammar (whose transformations are widely held to be computationally unmanageable), with one-level representations in which phonology, syntax and semantics have a similar status, and are assembled together at the same time. This would help to gain a better understanding of the relation between these aspects of NL.

In the past, much emphasis has been put on NL syntax and parsing, perhaps because of the unavailability of large and efficient Prolog systems. But formal logic has also an important part to play in the semantics of NL. However, it is generally acknowledged that first order logic, on which Prolog is based, is not powerful enough for NL applications, and many extensions have been proposed to cover these deficiencies.

This book may be seen as an attempt to redress the balance away from syntax and parsing. It presents 18 high quality papers, and two panel discussions, by many leading researchers in the field, covering such wide-ranging topics as syntax, parsing, generation, semantics, discourse, pragmatics and cooperative responses.

Broadly speaking, the papers on morphology and syntax aim to show how logic programming formalisms are capable of expressing well-known phenomena succinctly and effectively. The contributions on parsing and on new formalisms show how to express grammatical constraints in a declarative and, it is hoped, computationally tractable way. Those on semantics generally aim at showing how first-order logic is insufficient and/or clumsy for expressing NL semantics, and discuss ways of enriching logic programming languages in order to handle NL effectively. Finally, a group of

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.