calculus. Chapters 3 and 5 systematically confirm our expectations about the higher order extension, working out exactly as much detail as required for that.

The real 'meat' of the book lies in Chapter 4. The author first proves the completeness of a procedure for equational unification of CTT terms and shows that the problem is undecidable. He then considers higher order unifiability and pre-unifiability; these are again undecidable, but he discusses heuristics. He moves on to higher order matching (given two terms t and t' of the same type, is there a substitution γ such that $t' = \gamma t$?). This has been conjectured by G. Huet to be decidable, but the problem remains open. The author presents extensive evidence which suggests a positive result for CTT terms: he presents a terminating procedure for matching conjectured to be complete; he also discusses other approaches like the Plotkin–Statman conjecture (relating this problem to that of deciding λ-definability), some NP-hard third order matching problems, Zaionc's (1985) idea of regular unification and second-order monadic unification. Thus this chapter also constitutes an excellent survey on this subject.

The bibliography, containing 204 reference items, is quite comprehensive and ought to be of great value to researchers in this area. But excessive referencing can also be obtrusive: do we need a reference to Huet's thesis (written in French) to support a claim that composition of substitutions is associative (page 25)?

My major criticism of the book is that it is *not* self-contained. While every definition needed in any proof can be found in the book, other books and papers are needed to make sense of the material here. Andrews' book (Andrew, 1986) is a pre-requisite for understanding Chapters 2 and 3, and the important ideas of Chapter 4 need recourse to Huet's papers and Statman (1982) for a complete understanding. Material from these sources could easily have been included to add substance to this slim volume.

Further, there are very few examples. The book has 145 definitions as opposed to 24 examples, most of which highlight specific definitions used in proofs rather than illustrate concepts. Given that the book is about extending results from first order logic to a higher order one, we can reasonably expect examples of CTT formulas with specific higher order features and instances of reasoning in this logic. Indeed, in the entire book there is only *one* example giving a CTT formula (example 3.27, showing its conversion to normal form). All the rest illustrate local technical details (subsets of $\sim$ relation, applying the MATCH procedure to nodes, etc.). Such a purely formal treatment, and the lack of proofs of base material (strong normalization theorem for λ-calculus, the many theorems of Huet and Statman quoted in the book) do not make for pedagogic use. This book is more in the nature of an extended research article on CTT meant for researchers in automated theorem proving and logic programming, rather than a textbook for a graduate course (as claimed on the back cover).

The production of the book is excellent. There are very few mistakes. (Page 55, line 5 from bottom, 'equational unification' should be 'pre-unification'.)

REFERENCES

Andrews, P. B. (1986) *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press, Orlando.

Statman, R. (1982) Completeness, invariance, and λ-definability. *J. Symbolic Logic*, **47**, 17–26.

Zaionc, M. (1985) The set of unifiers in typed λ-calculus as regular expression. In *Rewriting Techniques and Applications, Lecture Notes in Computer Science 202.* Springer, Berlin.

R. RAMANUJAM
The Institute of Mathematical Sciences, Madras

GÉRARD HUET AND GORDON PLOTKIN (editors)
Logical Environments. Cambridge University Press, 1993, £35.00. 338 pp hardbound, ISBN 0-521-43312-6

This book is a collection of papers about computer systems that provide facilities for the development of formal (i.e. mathematical) proofs about hardware and software, as well as about mathematical systems proper. The concept of a logical environment can be construed in the above sense (i.e. as a software system composed of algorithms and representations) or as a formal (logical) theory within which such reasoning can take place. When implemented, the environment contains, in addition to theorem-proving software, databases of axioms and of theorems: typically, users can add to the collection of theorems available within a system.

The book reflects many of the issues raised by logical environments. It contains a section on the general concept of a logical framework and the issues raised by them, a section on the algorithms needed to support environments, a section on foundational issues, and a final section describing experiments that have been performed using two such environments (LEGO from Edinburgh and ALF from Göteborg).

The general theoretical viewpoint of logical environments is constructive logic. That is, the logics typically used and/or researched are of a constructive nature: a proof of a property is interpreted as a construction of an (abstract) object with the required property. This interpretation is clearly very close to the concept of an algorithm: proofs are ways of building things. The theory of constructive systems is an active research area, and this collection contains chapters on the foundations of such logics.

The field may be dominated by the constructivist approach, but the chapter by Matthews, Smaill and Basin outlines their work of a *classical* system due to Feferman called FS_0. In common with other work in this area (that of Constable *et al.*, and the work on the Edinburgh LF), work on FS_0 is aimed at producing a framework within which to construct other logics. The kinds of logic used to provide a framework are examples of *meta logics*—logics that describe other logics. It would appear that constructive meta logics, particularly within a computational setting, offer significant advantages over classical ones, especially as far as effective computation is concerned. However, the classical interpretation of logic may still have some relevance to the process, and the chapter on FS_0 is exciting and interesting.

The chapter by Matthews, Smaill and Basin appears in the first section of the collection. The section is concerned with logical frameworks in general and the issue of representation within them. The introductory chapter of this section, by Basin and Constable, serves as an introduction to the issues raised by the framework concept. It is worth emphasizing that logical frameworks view the logic they directly implement and manipulate as a *meta logic*: the logic used by, for example, software engineers while specifying software may have a totally different characterisation.

The second section is concerned with the development of algorithms that can be used in logical environments. For example, there is de Bruijn's chapter which contains the complete type-checking algorithm for a variant of the lambda calculus. This is, apparently, the first time such an algorithm has been presented in detail; it is important because of the close connection between type checking and deduction. In the following chapter, Bertot presents a general method for determining the justification of every part of a proof term: this allows the automated explanation of why a proof succeeds or fails.

The third section deals with foundational issues in logic, whereas the final section contains two papers on experiments with logical environments. The inclusion of reports of experiments is particularly welcome because logical environments are intended to be tools that can be used every day by working software and hardware engineers (amongst others). The experiments in this section of the book are, perhaps, a little abstract for many software or hardware people (completion of the rations and metric spaces and a proof that Ackermann's function is not Primitive Recursive), but they are long proofs and show that many representational details must be considered before proof can be attempted. The experiments were conducted using different environments, so comparisons can be made.

The book is aimed at the logically highly literate. Some may well find the material forbidding (especially the chapters that comprise section three) because of its mathematical nature. It should be remembered, though, that the collection represents the state of the art as presented at a EU ESPRIT funded workshop and represents work in progress rather than completed products and methodologies. As a consequence, the audience for this book may be a little restricted to those involved in theoretical computer science and formal methods. The various chapters of the book are, nevertheless, clear in exposition and treatment, although sometimes one would like a little more detail. For those acquainted with the concept of a logical environment and who are aware of the work in this area, the appearance of this book will be welcomed, even if the price is somewhat high.

I. CRAIG
Warwick University

PETER PADAWITZ
Deduction and Declarative Programming. Cambridge University Press, 1992, £22.95, vi+279 pp hardbound, ISBN 0-521-41723-6

To begin with declarative programming, the author argues (in Chapter 2) that both functional and logic programs can be described in the framework of *Horn formulas with equality*. Each 'clause' in these programs has premises which can be divided into a *guard* and a *generator* which has recursive function calls or predicates. The functional program returns a function value, whereas the logic program computes solutions.

A set of such guarded Horn clauses can (under specified conditions) be converted into a Gentzen clause. Computing in the declarative program now becomes trying to *prove* the Gentzen clause and the emphasis shifts to goal-directed techniques for doing this.

The *solving expansion* technique builds up a sequence of pairs of a goal and a substitution, leading up to a solution. Alternately, one can use a *proving expansion*, where one builds a sequence of goal sets (regarded as disjunctions). Thus one can work on several lemmas in parallel. Chapters 4 and 5 deal with this.

To allow proofs which use an induction hypothesis, one can work with an *inductive expansion*. Here, when working on $\gamma \Leftrightarrow \delta$, the expansion heads towards another instance of γ, and eventually ends up with δ. Clearly, dividing premises into guards and generators is of great help here. Most of Chapter 5 is concerned with inductive expansion.

A different idea is to set up a rewriting system from the specification and perform *reduction*. The author generalizes this to a *sub-p-reductive expansion* and shows how this relates to inductive expansion. (However, this is different from rewriting-based semantics, as the author is careful to point out.) This is dealt with in Chapters 6 and 7.

As can be seen from this summary, the book is an impressive compendium on the subject of proving Gentzen clauses which are relevant to functional and logic programming. Some examples are presented, including how to use EXPANDER, a Standard ML implementation of inductive expansion. The logical background required is built up in Chapters 1 and 3.

However, the book is far from being a ready reference for the practitioner. A background in algebraic specification techniques and the rudiments of theorem proving is necessary for the reader. The examples are not worked into the main theory but appear only as pages of deduction (or records of program sessions). I would