Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

"Handbook of Practical Logic and Automated Reasoning," by John R. Harrison, Cambridge University Press, 2009

Freek Wiedijk

John Harrison's *Handbook of Practical Logic and Automated Reasoning* strongly reminds me of Donald Knuth's *The Art of Computer Programming*. Both clearly are masterpieces. And both scare me. They are so comprehensive, so erudite, and the information density is so high, that one really has to pay attention to get the most from the exposition. Also, both are a bit idiosyncratic. On the other hand, reading those books is an utter pleasure, because everything is so beautifully presented and there is so much to learn.

The *Handbook of Practical Logic and Automated Reasoning* then, is about automation in mathematical logic. Theorem proving with a computer only becomes practical when mundane proof tasks are performed automatically (the two main types of automation being 'decision procedures' and 'proof search'). The best interactive theorem provers are those in which this kind of automation has been developed furthest. The *Handbook* is a comprehensive treatment of this kind of automation.

The book consists of three strains that have been woven together expertly. The first strain consists of English prose, treating the topics of the book in computer science style. The second strain is a mathematical treatment consisting of definitions and theorems with full mathematical proofs. And the third strain is the source code of a computer program in the programming language OCaml, that implements *everything* that is being treated in detailed working code.

When reading the book one can ignore the second and third strains without losing understanding of the subject. If one thinks 'Yes, I believe that that theorem holds, I am not interested in the details of the proof right now', one can skip over the proof without getting lost. Or if one thinks, 'Yes, I roughly understand how that works, I believe that it can be implemented', one can skip over the program source without problems too. However, *if* one is interested in the details of either the mathematics or the implementation, it is all there.

If one strips the English from the book, one gets the *full* source of a beautiful self-contained theorem prover. The book even includes the source for mundane things like parsing and pretty-printing. It shows the power of functional program-

Freek Wiedijk
Institute for Computing and Information Sciences, Radboud University Nijmegen
E-mail: freek@cs.ru.nl

ming that it is possible to give full implementations of so many algorithms in a book of 681 pages, which still also treats everything in a way that is understandable without looking at the code. It also shows how cleverly John Harrison has built a 'theorem proving workbench' in which various pieces are reused over and over again. In some sense the book is 'just' an annotated listing of a program. This probably will turn some readers off, but as already said, even when skipping over the code one still has a very readable and detailed handbook.

The code from the book can be downloaded from

`http://www.cl.cam.ac.uk/~jrh13/atp/`

and loaded into an interactive OCaml session. That way, one can easily experiment with everything that the book discusses while reading it.

It should be noted that the theorem prover from the book is *not* John Harrison's industrial strength theorem prover HOL Light. The prover from the book implements first order logic, while HOL Light is higher order. In fact the book only briefly mentions higher order logic on page 122. Almost all interesting proof automation apparently can be presented in the domain of first order logic. Another difference between these two theorem provers is that unlike HOL Light the program from the book is purely functional. A nice challenge for the Haskell community might be to create an attractive Haskell version.

The book seems to also be intended as a textbook of mathematical logic, as it introduces propositional logic and first order predicate logic from scratch. Still, it mostly treats highly technical subjects that will only be digestible to people who already are well-versed in mathematical logic. Still, the book *is* self-contained and a smart novice could in theory understand everything by paying close attention.

The table of contents of the *Handbook* is deceptively simple:

1. Introduction
2. Propositional logic
3. First-order logic
4. Equality
5. Decidable problems
6. Interactive theorem proving
7. Limitations

Hidden within these chapters is a wide range of subjects. For example Chapter 4 contains a comprehensive overview of term rewriting (including termination orderings and Knuth-Bendix completion). Chapter 5 treats among many other subjects Buchberger's algorithm for the calculation of Groebner bases (including applications like geometric theorem proving). Chapter 7 contains a very nice presentation of Goedel's first incompleteness theorem (including lots of code for experimenting with the machinery used in the proof). Note that proofs like the one of Goedel's theorem are not just sketched. Everything is developed and proved in full detail.

The presentation of logic in the book is sometimes a bit idiosyncratic. Occasionally a path into a subject is taken that does not follow the most 'standard' treatment. The choice of presentation then seems to have been influenced by what fits nicely in the OCaml implementation.

As an example take the treatment of first order predicate logic. The semantics of first order logic is given in Chapter 3, but a proof system only is presented in Chapter 6, and then only as a Hilbert-style calculus. Natural deduction and sequent

calculus are sketched in a few pages, but maybe too briefly for a person who does not know these systems already. Also following Gentzen the notation $\Gamma \rightarrow p$ is used for what now is usually written as $\Gamma \vdash p$. After defining the Hilbert-style calculus, a beautiful LCF-style implementation of this proof system is developed, and the completeness theorem for first order logic is proved from the fact that that implementation can execute a complete proof search procedure. Clearly, the theory of first order logic (semantics, proof system, completeness) is all there, but it is not put in one place. Like many of the subjects it is woven throughout the book.

As the semantics are treated before the proof system, most of the theory is developed in terms of the semantics. This seems one of the themes of the book.

The book refers often to the historical background of the various subjects, and the Further Reading sections contain many pointers into the literature, referring to an impressive list of references that goes on for 37 pages. Each chapter also is accompanied by many interesting and sometimes very challenging Exercises.

John Harrison is one of the foremost researchers in the field of interactive theorem proving. His HOL Light is one of the best theorem provers in existence, and his work on formalization of mathematics is at the forefront of the technology. Now he has written what clearly will be *the* book about automation in theorem proving.

People often ask me whether they should buy this book. My answer then always is: yes, *of course* you should buy this book. It is a masterpiece. But beware! It might also scare you.