JFP **30**, e6, 2 pages, 2020.
doi:10.1017/S0956796820000052

Book Review

Review of "The Little Prover" by Daniel P. Friedman and Carl Eastlund, MIT Press, 2015

Machine-checked reasoning about programs is enjoying widespread interest, and pedagogical material on the topic is flourishing. The Little Prover is a lightweight introduction to the basics of theorem proving that will likely seduce an audience that might otherwise be intimidated by the focus on type theory and ambitious breadth of other offerings.

Following the tradition of Little Books (Friedman & Felleisen, 1989, 1996; Felleisen & Friedman, 1996; Friedman *et al.*, 2005), The Little Prover focuses on few core ideas, elegantly presented in the style of a Socratic dialog, sprinkled with humor. Here, the core is reasoning about recursive functions that manipulate the beloved data structure of Lispers, Schemers, and Racketeers. No type theory, no general inductive types, no universe hierarchies. No tactics and no automation; all proofs are built using explicit focusing and rewriting, the "assembly" of mechanized proofs.

In this sense, The Little Prover might very well be a key enabler for programmers with little to no formal background, used to dynamically-typed PLs, to get a good grasp of the nuts and bolts of mechanized proofs. The Little Prover comes with a working prototype theorem prover, J-Bob, which allows the reader to get her hands dirty while reading. Excited readers will find a good list of references at the end of book, should they wish to go forward and deeper into this fascinating topic.

The book dialog starts by recalling the basics of manipulating cons cells, presenting right away its *axioms*, or fundamental laws, on top of which one can start rigorously answering questions about the meaning of little code snippets. Additionally armed with the axioms of (syntactic) equality, one can then state, and prove, theorems. The object language is progressively extended, with conditionals, recursive functions, lists, and at the end arithmetic operations. Each time, the key axioms and laws are spelled out and explored in details, using simple but insightful examples. The story is interspersed with elements of proof strategies, in order to provide guidance when facing a new proof challenge. These are called the *insights* and all 9 are summarized on the inside front cover, while the inside back cover recaps the 21 axioms used in the book.

The meat of The Little Prover is to introduce inductive reasoning. Here again, the essence of the approach is to proceed by examples, focused first on a simple, dedicated form of induction called *list induction*. Once the use of this reasoning principle is spelled out, the book calls in examples that show its limits. It then introduces another form of induction called *star induction*, for reasoning about functions that recur not only on the tail of a list, but also on its head. The exploration of induction principles continues with the presentation of a general principle, called *defun induction*, of which list induction and star induction are but specific instances. Along the way, the reader is also taught why one

needs to pay attention to the domains of functions, and progressively trained to state and prove totality claims, including using custom notions of measure.

It would not be fair to say that the matter covered by the book is lightweight. It is well delimited in scope, and much less ambitious than larger endeavors to expose full-fledged proof assistants, theorem proving, and the underlying theory. The Little Prover takes the word "Little" seriously: less than 200 pages of code-based dialog and questions, solutions, and even the complete source code of the J-Bob proof assistant!

Overall, the book is pleasant and entertaining to go through. The 10 chapters are well articulated, in progressive order of complexity. Perhaps, the interplay between the main chapters, the recess—which explains how to use J-Bob—and the proofs in appendix could have been clearer and more explicit. For instance, one is sometimes left wondering when to work on the recess, until what point, before going back to the main story from the chapters.

A Little Book is best consumed by programming hands-on in parallel with the reading. The fact that proofs are written absolutely explicitly—with only focusing and rewriting—is tedious, but (in)formative. There is no magic going on here. At the beginning, the proofs are rather small, so the overhead is not too overwhelming. As the book progresses, though, proofs become way too long, except maybe for an extremely motivated reader. But by then, one has accumulated enough experience of the underlying mechanics involved to follow the book by just reading and glancing at the full proofs in the appendix.

All in all, I would highly recommend this book to anyone with little to no formal background, as a way to understand key ideas of mechanized reasoning about programs. I also think that for anyone familiar with theorem proving in Coq or Agda, it can be enlightening to see a completely different pedagogical take on the matter. Working through The Little Prover is a lot of constructive fun!

References

Felleisen, M. & Friedman, D. P. (1996) *The Little MLer*. MIT.
Friedman, D. P. & Felleisen, M. (1989) *The Little LISPer*. MIT.
Friedman, D. P. & Felleisen, M. (1996) *The Little Schemer* (4 ed.). MIT.
Friedman, D. P., Byrd, W. E. & Kiselyov, O. (2005) *The Reasoned Schemer*. MIT.

ÉRIC TANTER
PLEIAD Lab—Computer Science Department (DCC),
University of Chile
E-mail: etanter@dcc.uchile.cl