

Concrete Semantics

Tobias Nipkow · Gerwin Klein

Concrete Semantics

With Isabelle/HOL

Tobias Nipkow
Fakultät für Informatik
Technische Universität München
Garching
Germany

Gerwin Klein
NICTA, Neville Roach Laboratory
Kensington, NSW
Australia

ISBN 978-3-319-10541-3 ISBN 978-3-319-10542-0 (eBook)
DOI 10.1007/978-3-319-10542-0

Library of Congress Control Number: 2014957301

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

I will not allow books to prove anything.

Jane Austen, *Persuasion*

Preface

This book is two books. Part I is a practical introduction to working with the Isabelle proof assistant. It teaches you how to write functional programs and inductive definitions and how to prove properties about them in Isabelle's structured proof language. Part II is an introduction to the semantics of imperative languages with an emphasis on applications like compilers and program analysers. The distinguishing features are that every bit of mathematics has been formalized in Isabelle and that much of it is executable. Part I focusses on the details of proofs in Isabelle. Part II can be read even without familiarity with Isabelle's proof language: all proofs are described in detail but informally. The Isabelle formalization, including the proofs, is available online: all the material, including accompanying slides, can be downloaded from the book's home page <http://www.concrete-semantics.org>.

Although the subject matter is semantics and applications, the not-so-hidden agenda is to teach the reader two things: the art of precise logical reasoning and the practical use of a proof assistant as a surgical tool for formal proofs about computer science artefacts. In this sense the book represents a formal approach to computer science, not just semantics.

Why?

This book is the marriage of two areas: programming languages and theorem proving. Most programmers feel that they understand the programming language they use and the programs they write. Programming language semantics replaces a warm feeling with precision in the form of mathematical definitions of the meaning of programs. Unfortunately such definitions are often still at the level of informal mathematics. They are mental tools, but their informal nature, their size, and the amount of detail makes them error prone. Since they are typically written in \LaTeX , you do not even know whether they

would type-check, let alone whether proofs about the semantics, e.g., compiler correctness, are free of bugs such as missing cases.

This is where theorem proving systems (or “proof assistants”) come in, and mathematical (im)precision is replaced by logical certainty. A proof assistant is a software system that supports the construction of mathematical theories as formal language texts that are checked for correctness. The beauty is that this includes checking the logical correctness of all proof text. No more ‘proofs’ that look more like LSD trips than coherent chains of logical arguments. Machine-checked (aka “formal”) proofs offer the degree of certainty required for reliable software but impossible to achieve with informal methods.

In research, the marriage of programming languages and proof assistants has led to remarkable success stories like a verified C compiler [53] and a verified operating system kernel [47]. This book introduces students and professionals to the foundations and applications of this marriage.

Concrete?

- The book shows that a semantics is not a collection of abstract symbols on sheets of paper but formal text that can be *checked and executed* by the computer: Isabelle is also a programming environment and most of the definitions in the book are executable and can even be exported as programs in a number of (functional) programming languages. For a computer scientist, this is as concrete as it gets.
- Much of the book deals with concrete applications of semantics: compilers, type systems, program analysers.
- The predominant formalism in the book is operational semantics, the most concrete of the various forms of semantics.
- Foundations are made of concrete.

Exercises!

The idea for this book goes back a long way [65]. But only recently have proof assistants become mature enough for inflicting them on students without causing the students too much pain. Nevertheless proof assistants still require very detailed proofs. Learning this proof style (and all the syntactic details that come with any formal language) requires practice. Therefore the book contains a large number of exercises of varying difficulty. If you want to learn Isabelle, you have to work through (some of) the exercises.

A word of warning before you proceed: theorem proving can be addictive!

Acknowledgements

This book has benefited significantly from feedback by John Backes, Harry Butterworth, Dan Dougherty, Andrew Gacek, Florian Haftmann, Peter Johnson, Yutaka Nagashima, Andrei Popescu, René Thiemann, Andrei Sabelfeld, David Sands, Sean Seefried, Helmut Seidl, Christian Sternagel and Carl Witty. Ronan Nugent provided very valuable editorial scrutiny.

The material in this book has been classroom-tested for a number of years. Sascha Böhme, Johannes Hölzl, Alex Krauss, Peter Lammich and Andrei Popescu worked out many of the exercises in the book.

Alex Krauss suggested the title *Concrete Semantics*.

NICTA, Technische Universität München and the DFG Graduiertenkolleg 1480 PUMA supported the writing of this book very generously.

We are very grateful for all these contributions.

Munich
Sydney
October 2014

TN
GK

Contents

Part I Isabelle

1	Introduction	3
2	Programming and Proving	5
2.1	Basics	5
2.2	Types <i>bool</i> , <i>nat</i> and <i>list</i>	7
2.3	Type and Function Definitions	15
2.4	Induction Heuristics	19
2.5	Simplification	21
3	Case Study: IMP Expressions	27
3.1	Arithmetic Expressions	27
3.2	Boolean Expressions	32
3.3	Stack Machine and Compilation	35
4	Logic and Proof Beyond Equality	37
4.1	Formulas	37
4.2	Sets	38
4.3	Proof Automation	39
4.4	Single Step Proofs	42
4.5	Inductive Definitions	45
5	Isar: A Language for Structured Proofs	53
5.1	Isar by Example	54
5.2	Proof Patterns	56
5.3	Streamlining Proofs	58
5.4	Case Analysis and Induction	61

Part II Semantics

6	Introduction	73
7	IMP: A Simple Imperative Language	75
7.1	IMP Commands	75
7.2	Big-Step Semantics	77
7.3	Small-Step Semantics	85
7.4	Summary and Further Reading	90
8	Compiler	95
8.1	Instructions and Stack Machine	95
8.2	Reasoning About Machine Executions	98
8.3	Compilation	99
8.4	Preservation of Semantics	102
8.5	Summary and Further Reading	112
9	Types	115
9.1	Typed IMP	117
9.2	Security Type Systems	128
9.3	Summary and Further Reading	140
10	Program Analysis	143
10.1	Definite Initialization Analysis	145
10.2	Constant Folding and Propagation	154
10.3	Live Variable Analysis	164
10.4	True Liveness	172
10.5	Summary and Further Reading	178
11	Denotational Semantics	179
11.1	A Relational Denotational Semantics	180
11.2	Summary and Further Reading	188
12	Hoare Logic	191
12.1	Proof via Operational Semantics	191
12.2	Hoare Logic for Partial Correctness	192
12.3	Soundness and Completeness	203
12.4	Verification Condition Generation	208
12.5	Hoare Logic for Total Correctness	212
12.6	Summary and Further Reading	215

13 Abstract Interpretation	219
13.1 Informal Introduction	220
13.2 Annotated Commands	224
13.3 Collecting Semantics	225
13.4 Abstract Values	236
13.5 Generic Abstract Interpreter	241
13.6 Executable Abstract States	253
13.7 Analysis of Boolean Expressions	259
13.8 Interval Analysis	264
13.9 Widening and Narrowing	270
13.10 Summary and Further Reading	279
 A Auxiliary Definitions	281
 B Symbols	283
 C Theories	285
 References	287
 Index	293