

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

2283

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Tobias Nipkow Lawrence C. Paulson
Markus Wenzel

Isabelle/HOL

A Proof Assistant for Higher-Order Logic



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Authors

Tobias Nipkow
Markus Wenzel
Technische Universität München, Institut für Informatik
Arcisstraße 21, 80333 München, Germany
E-mail: {nipkow,wenzelm}@in.tum.de

Lawrence C. Paulson
University of Cambridge, Computer Laboratory
JJ Thomson Avenue, Cambridge CB3 0FD, UK
E-mail: lcp@cl.cam

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Nipkow, Tobias:
Isabelle, HOL : a proof assistant for higher order logic / Tobias Nipkow ;
Lawrence C. Paulson ; Markus Wenzel. - Berlin ; Heidelberg ; New York ;
Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2283)
ISBN 3-540-43376-7

CR Subject Classification (1998): F.4.1, I.2.3, F.3.1, D.3.1, D.2.1

ISSN 0302-9743

ISBN 3-540-43376-7 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10846296 06/3142 5 4 3 2 1 0

In memoriam
ANNETTE SCHUMANN
1959 – 2001

Preface

This volume is a self-contained introduction to interactive proof in higher-order logic (HOL), using the proof assistant Isabelle 2002. Compared with existing Isabelle documentation, it provides a direct route into higher-order logic, which most people prefer these days. It bypasses first-order logic and minimizes discussion of meta-theory. It is written for potential users rather than for our colleagues in the research world.

Another departure from previous documentation is that we describe Markus Wenzel's proof script notation instead of ML tactic scripts. The latter make it easier to introduce new tactics on the fly, but hardly anybody does that. Wenzel's dedicated syntax is elegant, replacing for example eight simplification tactics with a single method, namely *simp*, with associated options.

The book has three parts.

- The first part, **Elementary Techniques**, shows how to model functional programs in higher-order logic. Early examples involve lists and the natural numbers. Most proofs are two steps long, consisting of induction on a chosen variable followed by the *auto* tactic. But even this elementary part covers such advanced topics as nested and mutual recursion.
- The second part, **Logic and Sets**, presents a collection of lower-level tactics that you can use to apply rules selectively. It also describes Isabelle/HOL's treatment of sets, functions, and relations and explains how to define sets inductively. One of the examples concerns the theory of model checking, and another is drawn from a classic textbook on formal languages.
- The third part, **Advanced Material**, describes a variety of other topics. Among these are the real numbers, records, and overloading. Esoteric techniques are described involving induction and recursion. A whole chapter is devoted to an extended example: the verification of a security protocol.

The typesetting relies on Wenzel's theory presentation tools. An annotated source file is run, typesetting the theory in the form of a $\text{\LaTeX}$ source file. This book is derived almost entirely from output generated in this way. The final chapter of Part I explains how users may produce their own formal documents in a similar fashion.

Isabelle's web site¹ contains links to the download area and to documentation and other information. Most Isabelle sessions are now run from within David Aspinall's wonderful user interface, Proof General², even together with the X-Symbol³ package for XEmacs. This book says very little about Proof General, which has its own documentation. In order to run Isabelle, you will need a Standard ML compiler. We recommend Poly/ML⁴, which is free and gives the best performance. The other fully supported compiler is Standard ML of New Jersey⁵.

This tutorial owes a lot to the constant discussions with and the valuable feedback from the Isabelle group at Munich: Stefan Berghofer, Olaf Müller, Wolfgang Naraschewski, David von Oheimb, Leonor Prensa Nieto, Cornelia Pusch, Norbert Schirmer, and Martin Strecker. Stephan Merz was also kind enough to read and comment on a draft version. We received comments from Stefano Bistarelli, Gergely Buday, and Tanja Vos.

The research has been funded by many sources, including the DFG grants Ni 491/2, Ni 491/3, Ni 491/4 and the EPSRC grants GR/K57381, GR/K77051, GR/M75440, GR/R01156/01, and by the ESPRIT working groups 21900 and IST-1999-29001 (the *Types* project).

¹ <http://isabelle.in.tum.de/>

² <http://www.proofgeneral.org/>

³ <http://www.fmi.uni-passau.de/~wedler/x-symbol/>

⁴ <http://www.polymml.org/>

⁵ <http://cm.bell-labs.com/cm/cs/what/smlnj/index.html>

Table of Contents

Part I. Elementary Techniques	
1. The Basics	3
1.1 Introduction	3
1.2 Theories	4
1.3 Types, Terms, and Formulae	4
1.4 Variables	7
1.5 Interaction and Interfaces	7
1.6 Getting Started	8
2. Functional Programming in HOL	9
2.1 An Introductory Theory	9
2.2 An Introductory Proof	11
2.3 Some Helpful Commands	15
2.4 Datatypes	17
2.4.1 Lists	17
2.4.2 The General Format	17
2.4.3 Primitive Recursion	18
2.4.4 Case Expressions	18
2.4.5 Structural Induction and Case Distinction	19
2.4.6 Case Study: Boolean Expressions	19
2.5 Some Basic Types	22
2.5.1 Natural Numbers	22
2.5.2 Pairs	24
2.5.3 Datatype <i>option</i>	24
2.6 Definitions	24
2.6.1 Type Synonyms	25
2.6.2 Constant Definitions	25
2.7 The Definitional Approach	26
3. More Functional Programming	27
3.1 Simplification	27
3.1.1 What Is Simplification?	27
3.1.2 Simplification Rules	28
3.1.3 The <i>simp</i> Method	28

3.1.4	Adding and Deleting Simplification Rules	29
3.1.5	Assumptions	29
3.1.6	Rewriting with Definitions	30
3.1.7	Simplifying <code>let</code> -Expressions	31
3.1.8	Conditional Simplification Rules	31
3.1.9	Automatic Case Splits	31
3.1.10	Tracing	33
3.2	Induction Heuristics	33
3.3	Case Study: Compiling Expressions	36
3.4	Advanced Datatypes	38
3.4.1	Mutual Recursion	38
3.4.2	Nested Recursion	40
3.4.3	The Limits of Nested Recursion	42
3.4.4	Case Study: Tries	43
3.5	Total Recursive Functions	46
3.5.1	Defining Recursive Functions	46
3.5.2	Proving Termination	48
3.5.3	Simplification and Recursive Functions	49
3.5.4	Induction and Recursive Functions	50
4.	Presenting Theories	53
4.1	Concrete Syntax	53
4.1.1	Infix Annotations	53
4.1.2	Mathematical Symbols	54
4.1.3	Prefix Annotations	55
4.1.4	Syntax Translations	56
4.2	Document Preparation	57
4.2.1	Isabelle Sessions	58
4.2.2	Structure Markup	59
4.2.3	Formal Comments and Antiquotations	60
4.2.4	Interpretation of Symbols	63
4.2.5	Suppressing Output	63
Part II. Logic and Sets		
5.	The Rules of the Game	67
5.1	Natural Deduction	67
5.2	Introduction Rules	68
5.3	Elimination Rules	69
5.4	Destruction Rules: Some Examples	71
5.5	Implication	72
5.6	Negation	73
5.7	Interlude: The Basic Methods for Rules	75
5.8	Unification and Substitution	76

5.8.1	Substitution and the <i>subst</i> Method	77
5.8.2	Unification and Its Pitfalls	78
5.9	Quantifiers	79
5.9.1	The Universal Introduction Rule	80
5.9.2	The Universal Elimination Rule	80
5.9.3	The Existential Quantifier	82
5.9.4	Renaming an Assumption: <i>rename_tac</i>	82
5.9.5	Reusing an Assumption: <i>frule</i>	83
5.9.6	Instantiating a Quantifier Explicitly	84
5.10	Description Operators	85
5.10.1	Definite Descriptions	85
5.10.2	Indefinite Descriptions	86
5.11	Some Proofs That Fail	87
5.12	Proving Theorems Using the <i>blast</i> Method	89
5.13	Other Classical Reasoning Methods	90
5.14	Forward Proof: Transforming Theorems	92
5.14.1	Modifying a Theorem Using <i>of</i> and <i>THEN</i>	93
5.14.2	Modifying a Theorem Using <i>OF</i>	95
5.15	Forward Reasoning in a Backward Proof	96
5.15.1	The Method <i>insert</i>	97
5.15.2	The Method <i>subgoal_tac</i>	98
5.16	Managing Large Proofs	99
5.16.1	Tacticals, or Control Structures	99
5.16.2	Subgoal Numbering	100
5.17	Proving the Correctness of Euclid's Algorithm	101
6.	Sets, Functions, and Relations	105
6.1	Sets	105
6.1.1	Finite Set Notation	107
6.1.2	Set Comprehension	107
6.1.3	Binding Operators	108
6.1.4	Finiteness and Cardinality	109
6.2	Functions	109
6.2.1	Function Basics	109
6.2.2	Injections, Surjections, Bijections	110
6.2.3	Function Image	111
6.3	Relations	111
6.3.1	Relation Basics	112
6.3.2	The Reflexive and Transitive Closure	112
6.3.3	A Sample Proof	113
6.4	Well-Founded Relations and Induction	114
6.5	Fixed Point Operators	116
6.6	Case Study: Verified Model Checking	116
6.6.1	Propositional Dynamic Logic — PDL	118
6.6.2	Computation Tree Logic — CTL	121

7. Inductively Defined Sets	127
7.1 The Set of Even Numbers	127
7.1.1 Making an Inductive Definition	127
7.1.2 Using Introduction Rules	128
7.1.3 Rule Induction	128
7.1.4 Generalization and Rule Induction	129
7.1.5 Rule Inversion	130
7.1.6 Mutually Inductive Definitions	131
7.2 The Reflexive Transitive Closure	132
7.3 Advanced Inductive Definitions	135
7.3.1 Universal Quantifiers in Introduction Rules	135
7.3.2 Alternative Definition Using a Monotone Function	137
7.3.3 A Proof of Equivalence	138
7.3.4 Another Example of Rule Inversion	139
7.4 Case Study: A Context Free Grammar	140

Part III. Advanced Material

8. More about Types	149
8.1 Numbers	149
8.1.1 Numeric Literals	150
8.1.2 The Type of Natural Numbers, nat	151
8.1.3 The Type of Integers, int	153
8.1.4 The Type of Real Numbers, real	154
8.2 Pairs and Tuples	155
8.2.1 Pattern Matching with Tuples	155
8.2.2 Theorem Proving	156
8.3 Records	158
8.3.1 Record Basics	158
8.3.2 Extensible Records and Generic Operations	159
8.3.3 Record Equality	161
8.3.4 Extending and Truncating Records	163
8.4 Axiomatic Type Classes	164
8.4.1 Overloading	164
8.4.2 Axioms	167
8.5 Introducing New Types	170
8.5.1 Declaring New Types	171
8.5.2 Defining New Types	171
9. Advanced Simplification, Recursion, and Induction	175
9.1 Simplification	175
9.1.1 Advanced Features	175
9.1.2 How the Simplifier Works	177

9.2	Advanced Forms of Recursion	178
9.2.1	Beyond Measure	178
9.2.2	Recursion over Nested Datatypes	180
9.2.3	Partial Functions	182
9.3	Advanced Induction Techniques	186
9.3.1	Massaging the Proposition	186
9.3.2	Beyond Structural and Recursion Induction	188
9.3.3	Derivation of New Induction Schemas	190
9.3.4	CTL Revisited	191
10.	Case Study: Verifying a Security Protocol	195
10.1	The Needham-Schroeder Public-Key Protocol	195
10.2	Agents and Messages	197
10.3	Modelling the Adversary	198
10.4	Event Traces	199
10.5	Modelling the Protocol	200
10.6	Proving Elementary Properties	201
10.7	Proving Secrecy Theorems	203
A.	Appendix	207
	Bibliography	209
	Index	213