

Lecture Notes in Computer Science

828

Edited by G. Goos and J. Hartmanis

Advisory Board: W. Brauer D. Gries J. Stoer



Lawrence C. Paulson

Isabelle

A Generic Theorem Prover

With Contributions by Tobias Nipkow

Springer-Verlag

Berlin Heidelberg New York

London Paris Tokyo

Hong Kong Barcelona

Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
D-76131 Karlsruhe, Germany

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Author

Lawrence C. Paulson
Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG, United Kingdom

CR Subject Classification (1991): F.4.1, F.4.3, F.3.1, D.2.4, I.2.3

ISBN 3-540-58244-4 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-58244-4 Springer-Verlag New York Berlin Heidelberg

CIP data applied for

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 1994
Printed in Germany

Typesetting: Camera-ready by author
SPIN: 10472615 45/3140-543210 - Printed on acid-free paper

Preface

Most theorem provers support a fixed logic, such as first-order or equational logic. They bring sophisticated proof procedures to bear upon the conjectured formula. The resolution prover Otter [60] is an impressive example.

ALF [30], Coq [14] and Nuprl [9] each support a fixed logic too. These are higher-order type theories, explicitly concerned with computation and capable of expressing developments in constructive mathematics. They are far removed from classical first-order logic.

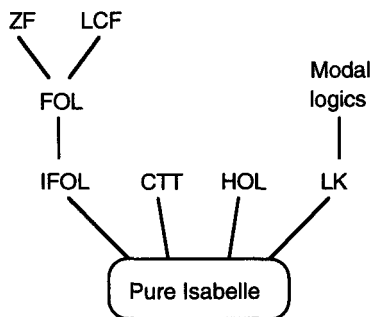
A diverse collection of logics — type theories, process calculi, λ -calculi — may be found in the Computer Science literature. Such logics require proof support. Few proof procedures are known for them, but the theorem prover can at least automate routine steps.

A **generic** theorem prover is one that supports a variety of logics. Some generic provers are noteworthy for their user interfaces [11, 29, 55]. Most of them work by implementing a syntactic framework that can express typical inference rules. Isabelle's distinctive feature is its representation of logics within a fragment of higher-order logic, called the meta-logic. The proof theory of higher-order logic may be used to demonstrate that the representation is correct [43]. The approach has much in common with the Edinburgh Logical Framework [24] and with Felty's [18] use of λ Prolog to implement logics.

An inference rule in Isabelle is a generalized Horn clause. Rules are joined to make proofs by resolving such clauses. Logical variables in goals can be instantiated incrementally. But Isabelle is not a resolution theorem prover like Otter. Isabelle's clauses are drawn from a richer language and a fully automatic search would be impractical. Isabelle does not resolve clauses automatically, but under user direction. You can conduct single-step proofs, use Isabelle's built-in proof procedures, or develop new proof procedures using tactics and tacticals.

Isabelle's meta-logic is higher-order, based on the simply typed λ -calculus. So resolution cannot use ordinary unification, but higher-order unification [27]. This complicated procedure gives Isabelle strong support for many logical formalisms involving variable binding.

The diagram below illustrates some of the logics distributed with Isabelle. These include first-order logic (intuitionistic and classical), the sequent calculus, higher-order logic, Zermelo-Fraenkel set theory [56], a version of Constructive Type Theory [39], several modal logics, and a Logic for Computable Functions [42]. Several experimental logics are being developed, such as linear logic.



How to read this book

Isabelle is a complex system, but beginners can get by with a few commands and a basic knowledge of how Isabelle works. Some knowledge of Standard ML is essential because ML is Isabelle's user interface. Advanced Isabelle theorem proving can involve writing ML code, possibly with Isabelle's sources at hand. My book on ML [46] covers much material connected with Isabelle, including a simple theorem prover.

The Isabelle documentation is divided into three parts, which serve distinct purposes:

- *Introduction to Isabelle* describes the basic features of Isabelle. This part is intended to be read through. If you are impatient to get started, you might skip the first chapter, which describes Isabelle's meta-logic in some detail. The other chapters present on-line sessions of increasing difficulty. It also explains how to derive rules define theories, and concludes with an extended example: a Prolog interpreter.
- *The Isabelle Reference Manual* provides detailed information about Isabelle's facilities, excluding the object-logics. This part would make boring reading, though browsing might be useful. Mostly you should use it to locate facts quickly.
- *Isabelle's Object-Logics* describes the various logics distributed with Isabelle. The chapters are intended for reference only; they overlap somewhat so that each chapter can be read in isolation.

This book should not be read from start to finish. Instead you might read a couple of chapters from *Introduction to Isabelle*, then try some examples referring to the other parts, return to the *Introduction*, and so forth. Starred sections discuss obscure matters and may be skipped on a first reading.

Releases of Isabelle

Isabelle was first distributed in 1986. The 1987 version introduced a higher-order meta-logic with an improved treatment of quantifiers. The 1988 version added limited polymorphism and support for natural deduction. The 1989 version included a parser and pretty printer generator. The 1992 version introduced type classes, to support many-sorted and higher-order logics. The 1993 version provides greater support for theories and is much faster.

Isabelle is still under development. Projects under consideration include better support for inductive definitions, some means of recording proofs, a graphical user interface, and developments in the standard object-logics. I hope but cannot promise to maintain upwards compatibility.

Isabelle is available by anonymous ftp:

- University of Cambridge
host `ftp.cl.cam.ac.uk`
directory `ml`
- Technical University of Munich
host `ftp.informatik.tu-muenchen.de`
directory `local/lehrstuhl/nipkow`

The electronic distribution list `isabelle-users@cl.cam.ac.uk` provides a forum for discussing problems and applications involving Isabelle. To join, send me a message via `lcp@cl.cam.ac.uk`. Please notify me of any errors you find in this book.

Acknowledgements

Tobias Nipkow has made immense contributions to Isabelle, including the parser generator, type classes, the simplifier, and several object-logics. He also arranged for several of his students to help. Carsten Clasohm implemented the theory database; Markus Wenzel implemented macros; Sonia Mahjoub and Karin Nimmermann also contributed.

Nipkow and his students wrote much of the documentation underlying this book. Nipkow wrote the first versions of Sect. 3.2, Sect. 9.1, Chap. 10, Chap. 13 and App. A. Carsten Clasohm contributed to Chap. 9. Markus Wenzel contributed to Chap. 11. Nipkow also provided the quotation at the front.

David Aspinall, Sara Kalvala, Ina Kraan, Chris Owens, Zhenyu Qian, Norbert Völker and Markus Wenzel suggested changes and corrections to the documentation.

Martin Coen, Rajeev Goré, Philippe de Groote and Philippe Noël helped to develop Isabelle's standard object-logics. David Aspinall performed some useful research into theories and implemented an Isabelle Emacs mode. Isabelle was developed using Dave Matthews's Standard ML compiler, Poly/ML.

The research has been funded by numerous SERC grants dating from the Alvey programme (grants GR/E0355.7, GR/G53279, GR/H40570) and by ESPRIT (projects 3245: Logical Frameworks and 6453: Types).

Table of Contents

I	Introduction to Isabelle	1
1	Foundations	3
1.1	Formalizing logical syntax in Isabelle	3
1.2	Formalizing logical rules in Isabelle	7
1.3	Proof construction in Isabelle	11
1.4	Lifting a rule into a context	15
1.5	Backward proof by resolution	17
1.6	Variations on resolution	21
2	Getting Started with Isabelle	25
2.1	Forward proof	25
2.2	Backward proof	30
2.3	Quantifier reasoning	34
3	Advanced Methods	41
3.1	Deriving rules in Isabelle	41
3.2	Defining theories	46
3.3	Theory example: the natural numbers	52
3.4	Refinement with explicit instantiation	55
3.5	A Prolog interpreter	58
II	The Isabelle Reference Manual	63
4	Basic Use of Isabelle	65
4.1	Basic interaction with Isabelle	65
4.2	Ending a session	66
4.3	Reading ML files	66
4.4	Printing of terms and theorems	66
4.5	Displaying exceptions as error messages	68
4.6	Shell scripts	68

5	Proof Management: The Subgoal Module	71
5.1	Basic commands	71
5.2	Shortcuts for applying tactics	74
5.3	Executing batch proofs	76
5.4	Managing multiple proofs	77
5.5	Debugging and inspecting	78
6	Tactics	81
6.1	Resolution and assumption tactics	81
6.2	Other basic tactics	83
6.3	Obscure tactics	85
6.4	Managing lots of rules	87
6.5	Programming tools for proof strategies	89
6.6	Sequences	90
7	Tacticals	93
7.1	The basic tacticals	93
7.2	Control and search tacticals	95
7.3	Tacticals for subgoal numbering	98
8	Theorems and Forward Proof	101
8.1	Basic operations on theorems	101
8.2	Primitive meta-level inference rules	105
8.3	Derived rules for goal-directed proof	109
9	Theories, Terms and Types	113
9.1	Defining theories	113
9.2	Loading a new theory	115
9.3	Reloading modified theories	116
9.4	Basic operations on theories	118
9.5	Terms	119
9.6	Variable binding	120
9.7	Certified terms	121
9.8	Types	122
9.9	Certified types	122
10	Defining Logics	125
10.1	Priority grammars	125
10.2	The Pure syntax	126
10.3	Mixfix declarations	131
10.4	Example: some minimal logics	136
11	Syntax Transformations	139
11.1	Abstract syntax trees	139
11.2	Transforming parse trees to ASTs	140
11.3	Transforming ASTs to terms	142

11.4	Printing of terms	142
11.5	Macros: Syntactic rewriting	144
11.6	Translation functions	150
12	Substitution Tactics	153
12.1	Substitution rules	153
12.2	Substitution in the hypotheses	154
12.3	Setting up <code>hyp_subst_tac</code>	155
13	Simplification	157
13.1	Simplification sets	157
13.2	The simplification tactics	160
13.3	Examples using the simplifier	161
13.4	Permutative rewrite rules	164
13.5	*Setting up the simplifier	166
14	The Classical Reasoner	171
14.1	The sequent calculus	171
14.2	Simulating sequents by natural deduction	172
14.3	Extra rules for the sequent calculus	173
14.4	Classical rule sets	174
14.5	The classical tactics	176
14.6	Setting up the classical reasoner	178

III Isabelle's Object-Logics 179

15	Basic Concepts	181
15.1	Syntax definitions	182
15.2	Proof procedures	183
16	First-Order Logic	185
16.1	Syntax and rules of inference	185
16.2	Generic packages	186
16.3	Intuitionistic proof procedures	186
16.4	Classical proof procedures	191
16.5	An intuitionistic example	192
16.6	An example of intuitionistic negation	193
16.7	A classical example	195
16.8	Derived rules and the classical tactics	196
17	Zermelo-Fraenkel Set Theory	203
17.1	Which version of axiomatic set theory?	203
17.2	The syntax of set theory	204
17.3	Binding operators	206
17.4	The Zermelo-Fraenkel axioms	208

17.5	From basic lemmas to function spaces	211
17.6	Further developments	219
17.7	Simplification rules	228
17.8	The examples directory	229
17.9	A proof about powersets	230
17.10	Monotonicity of the union operator	232
17.11	Low-level reasoning about functions	233
18	Higher-Order Logic	235
18.1	Syntax	235
18.2	Rules of inference	240
18.3	A formulation of set theory	245
18.4	Generic packages and classical reasoning	251
18.5	Types	253
18.6	The examples directories	259
18.7	Example: Cantor's Theorem	260
19	First-Order Sequent Calculus	263
19.1	Unification for lists	263
19.2	Syntax and rules of inference	265
19.3	Tactics for the cut rule	267
19.4	Tactics for sequents	268
19.5	Packaging sequent rules	269
19.6	Proof procedures	270
19.7	A simple example of classical reasoning	271
19.8	A more complex proof	272
20	Constructive Type Theory	275
20.1	Syntax	277
20.2	Rules of inference	277
20.3	Rule lists	281
20.4	Tactics for subgoal reordering	284
20.5	Rewriting tactics	284
20.6	Tactics for logical reasoning	285
20.7	A theory of arithmetic	286
20.8	The examples directory	286
20.9	Example: type inference	288
20.10	An example of logical reasoning	289
20.11	Example: deriving a currying functional	292
20.12	Example: proving the Axiom of Choice	293
A	Syntax of Isabelle Theories	297
	References	301
	Index	305

List of Figures

1.1	Intuitionistic first-order logic	4
10.1	Meta-logic syntax	127
11.1	Parsing and printing	140
11.2	Parsing examples using the Pure syntax	141
11.3	Built-in parse AST translations	141
11.4	Macro example: set theory	144
13.1	The simplifier primitives	160
16.1	Syntax of FOL	187
16.2	Rules of intuitionistic logic	188
16.3	Derived rules for intuitionistic logic	189
16.4	Derived rules for classical logic	191
17.1	Constants of ZF	205
17.2	Translations for ZF	206
17.3	Full grammar for ZF	207
17.4	Rules and axioms of ZF	209
17.5	Further definitions of ZF	210
17.6	Basic derived rules for ZF	212
17.7	Replacement and separation	214
17.8	General union and intersection	214
17.9	Unordered pairs	214
17.10	Union, intersection, difference	215
17.11	Finite and singleton sets	215
17.12	The successor function	215
17.13	Descriptions; non-circularity	215
17.14	Subset and lattice properties	216
17.15	Ordered pairs; projections; general sums	217
17.16	Domain, range and field of a relation	217
17.17	Image and inverse image	217
17.18	Functions	218
17.19	λ -abstraction	218
17.20	Constructing functions from smaller sets	219

17.21	Equalities	220
17.22	The booleans	221
17.23	Disjoint unions	221
17.24	Non-standard pairs, products and sums	222
17.25	Least and greatest fixedpoints	223
17.26	Permutations	224
17.27	The natural numbers	225
17.28	The finite set operator	226
17.29	Lists	227
17.30	Rewrite rules for set theory	228
18.1	Syntax of HOL	236
18.2	Full grammar for HOL	237
18.3	The HOL rules	239
18.4	The HOL definitions	239
18.5	Derived rules for HOL	241
18.6	More derived rules	242
18.7	Syntax of the theory Set	243
18.8	Syntax of the theory Set (continued)	244
18.9	Rules of the theory Set	246
18.10	Derived rules for set theory	247
18.11	Further derived rules for set theory	248
18.12	Derived rules involving functions	249
18.13	Derived rules involving subsets	250
18.14	Set equalities	250
18.15	Type $\alpha \times \beta$	251
18.16	Type $\alpha + \beta$	252
18.17	Defining nat , the type of natural numbers	254
18.18	Derived rules for nat	255
18.19	The theory List	257
18.20	Rewrite rules for lists	258
19.1	Syntax of LK	264
19.2	Grammar of LK	265
19.3	Rules of LK	266
19.4	Derived rules for LK	267
20.1	The constants of CTT	276
20.2	Syntax of CTT	278
20.3	General equality rules	279
20.4	Rules for type N	280
20.5	Rules for the product type $\prod_{x \in A} B[x]$	280
20.6	Rules for the sum type $\sum_{x \in A} B[x]$	281
20.7	Rules for the binary sum type $A + B$	282
20.8	Rules for types F and T	282
20.9	Rules for the equality type $Eq(A, a, b)$	283

20.10	Derived rules for CTT	283
20.11	The theory of arithmetic	287