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

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.

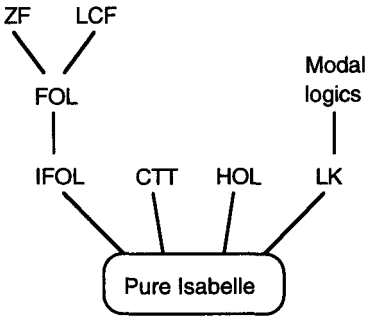
A diverse collection of logics — type theories, process calculi, λ-calculi — may be found in the Computer Science literature. Such logics require proof support. Few proof procedures are known for them, but the theorem prover can at least automate routine steps.

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 λProlog 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

List of Figures