The Isabelle Framework

Makarius Wenzel[1], Lawrence C. Paulson[2], and Tobias Nipkow[1]

[1] Technische Universität München, Institut für Informatik
[2] University of Cambridge, Computer Laboratory

1 Overview

Isabelle, which is available from `http://isabelle.in.tum.de`, is a generic framework for interactive theorem proving. The *Isabelle/Pure* meta-logic allows the formalization of the syntax and inference rules of a broad range of object-logics following the general idea of natural deduction [32, 33]. The logical core is implemented according to the well-known "LCF approach" of secure inferences as abstract datatype constructors in ML [16]; explicit proof terms are also available [8]. *Isabelle/Isar* provides sophisticated extra-logical infrastructure supporting structured proofs and specifications, including concepts for modular theory development. *Isabelle/HOL* is a large application within the generic framework, with plenty of logic-specific add-on tools and a large theory library. Other notable object-logics are *Isabelle/ZF* (Zermelo-Fraenkel set-theory, see [34, 36]) and *Isabelle/HOLCF* [26] (Scott's domain theory within HOL). Users can build further formal-methods tools on top, e.g. see [53].

Beginners are advised to start working with Isabelle/HOL; see the tutorial volume [30], and the companion tutorial [28] covering structured proofs. A general impression of Isabelle/HOL and ZF compared to other systems like Coq, PVS, Mizar etc. is given in [52]. The Proof General Emacs interface [3] is still the de-facto standard for interaction with Isabelle. The Isabelle document preparation system enables one to generate high-quality PDF-LaTeX documents from the original theory sources, with full checking of the formal content.

The *Archive of Formal Proofs* `http://afp.sf.net` collects proof libraries, examples, and larger scientific developments, mechanically checked with Isabelle. AFP is organized like a journal everybody can contribute to. Submitting formal theories there helps to maintain applications in the longer term, synchronized with the ongoing development of Isabelle itself.

2 Specification Mechanisms

Isabelle/Pure is a minimal version of higher-order logic; object-logics are specified by stating their characteristic rules as new axioms. Any later additions in application theories are usually restricted to *definitional specifications*, and the desired properties are being proven explicitly. Working directly from primitive definitions can be tedious, and higher-level specification mechanisms have emerged over the years, implemented as derived concepts within the existing background logic. This includes (co)inductive sets [35], inductive datatypes [11], and recursive functions [42, 23].

3 Structured Proofs

The Isar proof language [49] continues the natural deduction principles of Isabelle/Pure, working with human readable proof texts instead of primitive inferences. "*Isar*" abbreviates "*Intelligible semi-automated reasoning*"; the language is also related to Mizar, but many underlying principles are quite different [54].

The Isabelle/Isar design also follows the generic framework idea [51]. Starting with a small selection of common principles of natural deduction, various advanced concepts are defined as derived elements (e.g. for calculational reasoning [7] and complex induction proofs [50]). The demands for structured proof composition have also influenced the way of writing definitions and statements, using extra language elements corresponding to Isar proofs, instead of going through the object-logic again [12].

4 Modular Theory Development

Isabelle theories are organized as a graph, with monotonic operations to extend and to merge logical environments. Smaller units of context elements are managed by separate mechanisms for modular theory development, notably *axiomatic type-classes* [27, 48, 40] and *locales* [21, 5, 6]. More recent work integrates type-classes and locales [18], joining the simplicity of classes with the flexibility of locales.

The generic *local theory* concept [19] integrates user-defined module mechanisms smoothly into the Isabelle/Isar framework. The Isabelle distribution already incorporates locales, classes, and class instantiation contexts into the local theory infrastructure. Other approaches to modular theories like AWE [13] could be integrated as well.

Internally, all aspects of locality in Isabelle are centered around the notions of *proof context* and *morphism* — to transfer entities from one context into another. This covers primitive types / terms / theorems of Isabelle/Pure, and any extra-logical context data defined in Isabelle/Isar. This idea of "local everything" allows us to implement tools within an abstract theory and apply them in concrete application contexts later on. One example is an implementation [15] of algebraic methods on abstract rings that can be used for concrete rings.

5 Reasoning Tools

Isabelle has traditionally supported a fair amount of automated reasoning tools. The basic framework is centered around higher-order unification. The Simplifier supports higher-order rewriting, with plug-in interfaces for extra simplification procedures written in ML. The Classical Reasoner [37] and Classical Tableau Prover [38] have been recently complemented by the Metis prover due to Joe Hurd. Various arithmetic proof procedures are available as well. Sledghammer [41] uses external automated provers (E, Vampire, SPASS) as untrusted search tools to find the necessary lemmas for a particular goal; the actual proof is then performed internally with Metis.

6 Counterexample Search

Because much of the time one (unwittingly) tries to prove non-theorems, Isabelle/HOL offers two facilities to find counterexamples: *Quickcheck* [10] tries randomized instantiation of the free variables and is restricted to executable formulae (see §7). *Refute* [47] searches for small finite countermodels by translating (unrestricted) HOL formulae into propositional logic and hitting them with a SAT solver.

7 Code Generation

Executable HOL theories, including recursive functions and inductive definitions, can be translated into various functional languages, notably SML, OCaml, Haskell [9, 17]. Efficient imperative code can be generated from functions written in monadic style [14]. Results of ML-level computations can be re-imported as theorems ("reflection") to allow efficient computations in proofs. These code generators are restricted to evaluation of closed terms. Efficient evaluation of terms with free variables is supported by a compiled implementation of "normalization by evaluation" [1].

8 Major Applications

In the past 20 years, Isabelle has been used by numerous researchers and students of computer-science and mathematics world wide. Below we summarize some representative large-scale applications.

Pure Mathematics. Here the largest applications are: a) The verification by Bauer, Nipkow [29] and Obua [31] of two of the algorithmic parts of Hales' proof of the *Kepler Conjecture* (What is the densest arrangement of spheres in space?). This is part of Hales' *Flyspeck* project, the complete verification of his proof. b) Avigad's verification of the *Prime Number Theorem* [4] (about the distribution of primes). c) Paulson's proof [39] of the relative consistency of the axiom of choice in ZF, formalized in Isabelle/ZF.

Systems verification. The *Verisoft* project `http://www.verisoft.de` formalized a whole computer system from the hardware up to an operating system kernel [2] and a compiler for C-dialect [24]. The *L4.verified* project [20, 43] verifies the L4 operating system microkernel, relating an abstract specification, a Haskell model, and the C code.

Programming languages. A large amount of work has gone into formalizations of a sequential Java-like language *Jinja* [22], including bytecode-verification, virtual machine and compiler. Jinja has become the basis for further extensions like multithreading [25] and multiple inheritance [46]. *Isabelle/HOL-Nominal* [44] extends Isabelle/HOL with a unique infrastructure for defining and reasoning about languages with bound variables. Many case studies have been carried out, for example about the meta theory of LF [45].

References

[1] Aehlig, K., Haftmann, F., Nipkow, T.: A compiled implementation of normalization by evaluation. In: Theorem Proving in Higher Order Logics (TPHOLs 2008), Lecture Notes in Computer Science. Springer-Verlag (2008)

[2] Alkassar, E., Schirmer, N., Starostin, A.: Formal pervasive verification of a paging mechanism. In: C.R. Ramakrishnan, J. Rehof (eds.) Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), *Lecture Notes in Computer Science*, vol. 4963, pp. 109–123. Springer-Verlag (2008)

[3] Aspinall, D.: Proof General: A generic tool for proof development. In: European Joint Conferences on Theory and Practice of Software (ETAPS) (2000)

[4] Avigad, J., Donnelly, K., Gray, D., Raff, P.: A formally verified proof of the prime number theorem. ACM Trans. Comput. Logic **9**(1:2), 1–23 (2007)

[5] Ballarin, C.: Locales and locale expressions in Isabelle/Isar. In: S. Berardi, et al. (eds.) Types for Proofs and Programs (TYPES 2003), *Lecture Notes in Computer Science*, vol. 3085. Springer-Verlag (2004)

[6] Ballarin, C.: Interpretation of locales in Isabelle: Theories and proof contexts. In: J.M. Borwein, W.M. Farmer (eds.) Mathematical Knowledge Management (MKM 2006), *Lecture Notes in Artificial Intelligence*, vol. 4108. Springer-Verlag (2006)

[7] Bauer, G., Wenzel, M.: Calculational reasoning revisited — an Isabelle/Isar experience. In: R.J. Boulton, P.B. Jackson (eds.) Theorem Proving in Higher Order Logics: TPHOLs 2001, *Lecture Notes in Computer Science*, vol. 2152. Springer-Verlag (2001)

[8] Berghofer, S., Nipkow, T.: Proof terms for simply typed higher order logic. In: J. Harrison, M. Aagaard (eds.) Theorem Proving in Higher Order Logics (TPHOLs 2000), *Lecture Notes in Computer Science*, vol. 1869. Springer-Verlag (2000)

[9] Berghofer, S., Nipkow, T.: Executing higher order logic. In: P. Callaghan, Z. Luo, J. McKinna, R. Pollack (eds.) Types for Proofs and Programs (TYPES 2000), *Lecture Notes in Computer Science*, vol. 2277, pp. 24–40. Springer-Verlag (2002)

[10] Berghofer, S., Nipkow, T.: Random testing in Isabelle/HOL. In: J. Cuellar, Z. Liu (eds.) Software Engineering and Formal Methods (SEFM 2004), pp. 230–239. IEEE Computer Society (2004)

[11] Berghofer, S., Wenzel, M.: Inductive datatypes in HOL — lessons learned in Formal-Logic Engineering. In: Y. Bertot, et al. (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1999), *Lecture Notes in Computer Science*, vol. 1690 (1999)

[12] Berghofer, S., Wenzel, M.: Logic-free reasoning in Isabelle/Isar. In: Mathematical Knowledge Management (MKM 2008), Lecture Notes in Artificial Intelligence. Springer-Verlag (2008)

[13] Bortin, M., Broch Johnsen, E., Lüth, C.: Structured formal development in Isabelle. Nordic Journal of Computing **13** (2006)

[14] Bulwahn, L., Krauss, A., Haftmann, F., Erkök, L., Matthews, J.: Imperative functional programming in Isabelle/HOL. In: Theorem Proving in Higher Order Logics (TPHOLs 2008), Lecture Notes in Computer Science. Springer-Verlag (2008)

[15] Chaieb, A., Wenzel, M.: Context aware calculation and deduction — ring equalities via Gröbner Bases in Isabelle. In: M. Kauers, et al. (eds.) Towards Mechanized Mathematical Assistants (CALCULEMUS and MKM 2007), *Lecture Notes in Artificial Intelligence*, vol. 4573. Springer-Verlag (2007)

[16] Gordon, M.J.C., Milner, R., Wadsworth, C.P.: Edinburgh LCF: A Mechanized Logic of Computation, *Lecture Notes in Computer Science*, vol. 78. Springer-Verlag (1979)

[17] Haftmann, F., Nipkow, T.: A code generator framework for Isabelle/HOL. In: K. Schneider, J. Brandt (eds.) Theorem Proving in Higher Order Logics: Emerging Trends Proceedings. Dept. Comp. Sci., U. Kaiserslautern (2007)

[18] Haftmann, F., Wenzel, M.: Constructive type classes in Isabelle. In: T. Altenkirch, C. McBride (eds.) Types for Proofs and Programs (TYPES 2006), *Lecture Notes in Computer Science*, vol. 4502. Springer-Verlag (2007)

[19] Haftmann, F., Wenzel, M.: Local theory specifications in Isabelle/Isar (2008). `http://www.in.tum.de/~wenzelm/papers/local-theory.pdf`

[20] Heiser, G., Elphinstone, K., Kuz, I., Klein, G., Petters, S.M.: Towards trustworthy computing systems: taking microkernels to the next level. SIGOPS Operating Systems Review **41**(4), 3–11 (2007)

[21] Kammüller, F., Wenzel, M., Paulson, L.C.: Locales: A sectioning concept for Isabelle. In: Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, L. Thery (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1999), *Lecture Notes in Computer Science*, vol. 1690. Springer-Verlag (1999)

[22] Klein, G., Nipkow, T.: A machine-checked model for a Java-like language, virtual machine and compiler. ACM Trans. Progr. Lang. Syst. **28**(4), 619–695 (2006). DOI http://doi.acm.org/10.1145/1146809.1146811

[23] Krauss, A.: Partial recursive functions in Higher-Order Logic. In: U. Furbach, N. Shankar (eds.) International Joint Conference on Automated Reasoning (IJCAR 2006), *Lecture Notes in Computer Science*, vol. 4130. Springer-Verlag (2006)

[24] Leinenbach, D., Petrova, E.: Pervasive compiler verification — from verified programs to verified systems. In: Workshop on Systems Software Verification (SSV 2008). Elsevier (2008)

[25] Lochbihler, A.: Type safe nondeterminism — a formal semantics of Java threads. In: Foundations of Object-Oriented Languages (FOOL 2008) (2008)

[26] Müller, O., Nipkow, T., von Oheimb, D., Slotosch, O.: HOLCF = HOL + LCF. Journal of Functional Programming **9**, 191–223 (1999)

[27] Nipkow, T.: Order-sorted polymorphism in Isabelle. In: G. Huet, G. Plotkin (eds.) Logical Environments. Cambridge University Press (1993)

[28] Nipkow, T.: Structured proofs in Isar/HOL. In: H. Geuvers, F. Wiedijk (eds.) Types for Proofs and Programs (TYPES 2002), *Lecture Notes in Computer Science*, vol. 2646. Springer-Verlag (2003)

[29] Nipkow, T., Bauer, G., Schultz, P.: Flyspeck I: Tame graphs. In: U. Furbach, N. Shankar (eds.) Automated Reasoning (IJCAR 2006), *Lecture Notes in Computer Science*, vol. 4130, pp. 21–35. Springer-Verlag (2006)

[30] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, *Lecture Notes in Computer Science*, vol. 2283. Springer-Verlag (2002)

[31] Obua, S.: Flyspeck II: The basic linear programs. Ph.D. thesis, Technische Universität München (2008)

[32] Paulson, L.C.: Natural deduction as higher-order resolution. Journal of Logic Programming **3** (1986)

[33] Paulson, L.C.: Isabelle: the next 700 theorem provers. In: P. Odifreddi (ed.) Logic and Computer Science. Academic Press (1990)

[34] Paulson, L.C.: Set theory for verification: I. From foundations to functions. Journal of Automated Reasoning **11**(3) (1993)

[35] Paulson, L.C.: A fixedpoint approach to implementing (co)inductive definitions. In: A. Bundy (ed.) Automated Deduction (CADE-12), *Lecture Notes in Artificial Intelligence*, vol. 814. Springer-Verlag (1994)

[36] Paulson, L.C.: Set theory for verification: II. Induction and recursion. Journal of Automated Reasoning **15**(2) (1995)

[37] Paulson, L.C.: Generic automatic proof tools. In: R. Veroff (ed.) Automated Reasoning and its Applications: Essays in Honor of Larry Wos. MIT Press (1997)

[38] Paulson, L.C.: A generic tableau prover and its integration with Isabelle. Journal of Universal Computer Science **5**(3) (1999)
[39] Paulson, L.C.: The relative consistency of the axiom of choice — mechanized using Isabelle/ZF. LMS Journal of Computation and Mathematics **6**, 198–248 (2003)
[40] Paulson, L.C.: Organizing numerical theories using axiomatic type classes. Journal of Automated Reasoning **33**(1) (2004)
[41] Paulson, L.C., Susanto, K.W.: Source-level proof reconstruction for interactive theorem proving. In: K. Schneider, J. Brandt (eds.) Theorem Proving in Higher Order Logics (TPHOLs 2007), *Lecture Notes in Computer Science*, vol. 4732. Springer-Verlag (2007)
[42] Slind, K.: Function definition in higher order logic. In: J. Wright, J. Grundy, J. Harrison (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1996), *Lecture Notes in Computer Science*, vol. 1125 (1996)
[43] Tuch, H., Klein, G., Norrish, M.: Types, bytes, and separation logic. In: Principles of Programming Languages (POPL 2007), pp. 97–108. ACM Press (2007)
[44] Urban, C.: Nominal techniques in Isabelle/HOL. Journal of Automated Reasoning **40**, 327–356 (2008)
[45] Urban, C., Cheney, J., Berghofer, S.: Mechanizing the metatheory of LF. In: 23rd IEEE Symp. Logic in Computer Science (LICS) (2008)
[46] Wasserrab, D., Nipkow, T., Snelting, G., Tip, F.: An operational semantics and type safety proof for multiple inheritance in C++. In: Object Oriented Programming, Systems, Languages, and Applications (OOPSLA 2006), pp. 345–362. ACM Press (2006)
[47] Weber, T.: Bounded model generation for Isabelle/HOL. In: W. Ahrendt, P. Baumgartner, H. de Nivelle, S. Ranise, C. Tinelli (eds.) Workshops Disproving and Pragmatics of Decision Procedures (PDPAR 2004), *ENTCS*, vol. 125, pp. 103–116. Elsevier (2005)
[48] Wenzel, M.: Type classes and overloading in higher-order logic. In: E.L. Gunter, A. Felty (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1997), *Lecture Notes in Computer Science*, vol. 1275 (1997)
[49] Wenzel, M.: Isar — a generic interpretative approach to readable formal proof documents. In: Y. Bertot, et al. (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1999), *Lecture Notes in Computer Science*, vol. 1690. Springer-Verlag (1999)
[50] Wenzel, M.: Structured induction proofs in Isabelle/Isar. In: J. Borwein, W. Farmer (eds.) Mathematical Knowledge Management (MKM 2006), *Lecture Notes in Artificial Intelligence*, vol. 4108. Springer-Verlag (2006)
[51] Wenzel, M.: Isabelle/Isar — a generic framework for human-readable proof documents. In: R. Matuszewski, A. Zalewska (eds.) From Insight to Proof — Festschrift in Honour of Andrzej Trybulec, *Studies in Logic, Grammar, and Rhetoric*, vol. 10(23). University of Białystok (2007). `http://www.in.tum.de/~wenzelm/papers/isar-framework.pdf`
[52] Wenzel, M., Paulson, L.C.: Isabelle/Isar. In: F. Wiedijk (ed.) The Seventeen Provers of the World, *Lecture Notes in Artificial Intelligence*, vol. 3600. Springer-Verlag (2006)
[53] Wenzel, M., Wolff, B.: Building formal method tools in the Isabelle/Isar framework. In: K. Schneider, J. Brandt (eds.) Theorem Proving in Higher Order Logics (TPHOLs 2007), *Lecture Notes in Computer Science*, vol. 4732. Springer-Verlag (2007)
[54] Wiedijk, F., Wenzel, M.: A comparison of the mathematical proof languages Mizar and Isar. Journal of Automated Reasoning **29**(3-4) (2002)