# Isabelle:
# The next seven hundred theorem provers*

*Lawrence C Paulson*

Computer Laboratory, University of Cambridge
Cambridge CB2 3QG, England

*Isabelle* [2] is a theorem prover for a large class of logics. The object-logics are formalized within Isabelle's meta-logic, which is intuitionistic higher-order logic with implication, universal quantifiers, and equality.[1] The implication $\phi \Longrightarrow \psi$ means '$\phi$ implies $\psi$', and expresses logical entailment. The quantification $\bigwedge x.\phi$ means '$\phi$ is true for all $x$', and expresses generality in rules and axiom schemes. The equality $a \equiv b$ means '$a$ equals $b$', and allows new symbols to be defined as abbreviations.

Isabelle takes many ideas from LCF [1]. Formulae are manipulated through the meta-language Standard ML; proofs can be developed in the backwards direction via tactics and tacticals. But LCF represents the inference rule $\frac{A \quad B}{A \& B}$ by a function that maps the theorems $A$ and $B$ to the theorem $A \,\&\, B$, while Isabelle represents this rule by an axiom in the meta-logic:

$$\bigwedge A.\ \bigwedge B.\ [\![A]\!] \Longrightarrow ([\![B]\!] \Longrightarrow [\![A \,\&\, B]\!])$$

Observe how object-logic formulae are enclosed in brackets: $[\![A]\!]$.

Higher-order logic uses the typed $\lambda$-calculus, whose notions of free and bound variables handle quantifiers. So $\forall x.A$ can be represented by $\mathrm{All}(\lambda x.A)$, where All is a new constant and $A$ is a formula containing $x$. More precisely, $\forall x.F(x)$ can be represented by $\mathrm{All}(F)$, where the variable $F$ denotes a truth-valued function. Isabelle represents the rule $\frac{A}{\forall x.A}$ by the axiom

$$\bigwedge F.\,(\bigwedge x.\,[\![F(x)]\!]) \Longrightarrow [\![\mathrm{All}(F)]\!]$$

The introduction rule is subject to the proviso that $x$ is not free in the assumptions. Any use of the axiom involves proving $F(x)$ for arbitrary $x$, enforcing the proviso [3]. Similar techniques handle existential quantifiers, the $\Pi$ and $\Sigma$ operators of Type

*Appeared in E. Lusk and R. Overbeek (editors), *9th International Conf. on Automated Deduction*, Springer LNCS 310 (1988), pages 772–773.

[1] An early version called Isabelle-86 uses a naive calculus of proof trees as its meta-logic.

Theory, the indexed union operator of set theory, and so forth. Isabelle easily handles induction rules and axiom schemes, like set theory's Axiom of Separation.

Proof trees are derived rules, built by putting rules together. This gives forwards and backwards proof at the same time. Backwards proof is matching a goal with the conclusion of a rule; the premises become the subgoals. Forwards proof is matching theorems to the premises of a rule, making a new theorem.

Isabelle uses unification when joining rules. *Higher-order* unification is solving equations in the typed $\lambda$-calculus with respect to $\alpha$, $\beta$, and $\eta$-conversion. Unifying $f(x)$ with the constant $A$ gives the two unifiers $\{f = \lambda y.A\}$ and $\{f = \lambda y.y,\ x = A\}$. Multiple unifiers are a reflection of ambiguity: the four unifiers of $f(0)$ with $P(0,0)$ reflect the four different ways that $P(0,0)$ can be regarded as depending upon 0. Isabelle uses Huet's unification procedure.

Logics are proliferating at an alarming rate; there are seven theorem provers descended from Edinburgh LCF. With Isabelle, you need only specify the logic's syntax and rules. To go beyond proof checking, you can implement search procedures using built-in tools. Isabelle consists of 4000 lines of Standard ML. On this base stand object-logics such as Martin-Löf's Type Theory, intuitionistic first-order logic, and classical logic together with Zermelo-Fraenkel set theory.

Constructive Type Theory examples include the derivation of a choice principle and simple number theory: proofs of commutative, associative, and distributive laws for the arithmetic operations, culminating with $(m \bmod n) + (m/n) \times n = m$.

For first-order logic, an automatic procedure can prove many theorems involving quantifiers. The set theory examples include properties of union, intersection, and Cartesian products. One example is a proof that the standard definition of ordered pairs works: define $(a,b) \equiv \{\{a\},\{a,b\}\}$; if $(a,b) = (c,d)$ then $a = c$ and $b = d$. Two interesting properties of indexed intersection include

$$A \neq \emptyset \,\&\, B \neq \emptyset \quad \rightarrow \quad \bigcap(A \cup B) = (\bigcap A) \cap (\bigcap B)$$

$$C \neq \emptyset \quad \rightarrow \quad \bigcap_{x \in C} (A(x) \cap B(x)) = (\bigcap_{x \in C} A(x)) \cap (\bigcap_{x \in C} B(x))$$

# References


[1] L. C. Paulson, *Logic and Computation: Interactive Proof with Cambridge LCF* (Cambridge University Press, 1987).

[2] L. C. Paulson, Natural deduction as higher-order resolution, *Journal of Logic Programming* **3** (1986), pages 237–258.

[3] L. C. Paulson, The foundation of a generic theorem prover, Report 130, Computer Lab., University of Cambridge (1987).