

Interactive Theorem Provers

from the perspective of Isabelle/Isar

Makarius Wenzel
Univ. Paris-Sud, LRI

July 2014



1 Introduction

Notable ITP systems

LISP based:

ACL2 <http://www.cs.utexas.edu/users/moore/acl2>

PVS <http://pvs.csl.sri.com>

ML based:

HOL family: HOL4, HOL-Light, ProofPower, . . .

Coq <http://coq.inria.fr>

Isabelle/Isar <http://isabelle.in.tum.de>

Other:

Mizar <http://www.mizar.org>

Agda <http://wiki.portal.chalmers.se/agda>



See also: *The Seventeen Provers of the World*, F. Wiedijk (ed.), LNAI 3600, 2006.

The LCF family

LCF

Edinburgh LCF (1979)

Cambridge LCF (1985)

HOL (1984/1988)  

Coq 

Coc (1985/1988)

⋮

Coq 8.4pl4 (May 2014)

Isabelle    

Isabelle (1986/1989)

Isabelle/Isar (1999)

⋮

Isabelle2013-2 (December 2013)

TTY interaction

```
Terminal
File Edit View Terminal Tabs Help
Welcome to Isabelle/HOL (Isabelle2013: February 2013)
> theory A imports Main begin
theory A
> lemma "x = x";
proof (prove): step 0

goal (1 subgoal):
  1. x = x
> █

Terminal
File Edit View Terminal Tabs Help
Welcome to Coq 8.4pl2 (September 2013)

Coq < Lemma test: forall (A: Type) (x: A), x = x .
1 subgoal

=====
  forall (A : Type) (x : A), x = x
test < █
```

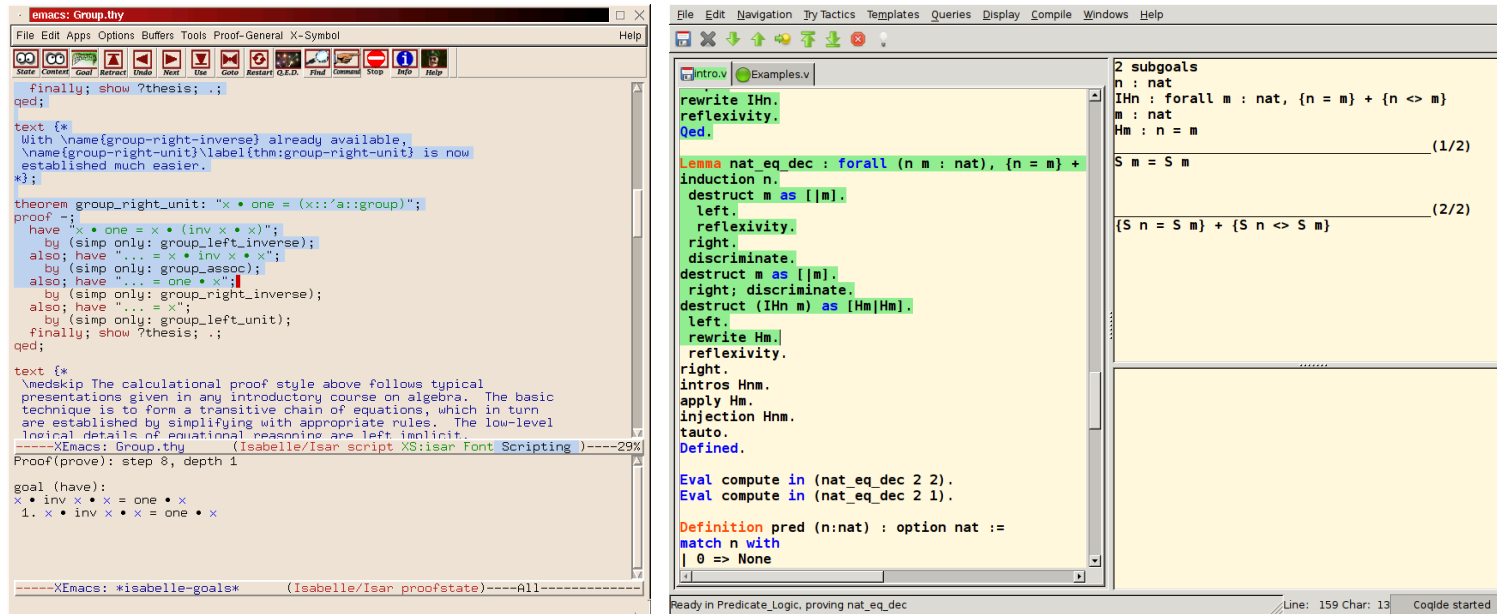


(Wikipedia: K. Thompson and D. Ritchie at PDP-11)

Interaction model:

manual copy-paste from editor window into prover process

Proof General (and clones)



Interaction model: automated copy-paste and undo in the editor, prover process in background

Isabelle today: document-oriented interaction

The screenshot shows the Isabelle IDE interface. The main window displays the source code for a theory named 'Seq'. The code defines a datatype for finite sequences and functions for concatenation and reversal. A tooltip is visible over the 'conc' function definition, showing its type signature. The right sidebar shows the 'Documentation' view for the 'Seq.thy' file, listing the theory's components. The bottom status bar indicates the current position in the file and system information.

```
header {* Finite sequences *}

theory Seq
imports Main
begin

datatype 'a seq = Empty | Seq 'a "'a seq"

fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
where
  "conc Empty ys = ys"
| "conc (Seq x xs) ys = Seq x (conc xs ys)"

fun reverse
where
  "reverse Empty = Empty"
| "reverse (Seq x xs) = conc (reverse xs) (Seq x Empty)"

lemma conc_empty: "conc xs Empty = xs"
by (induct xs) simp_all

constants
  conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
```

Example: functional specifications with proofs

datatype $'a \text{ seq} = \text{Empty} \mid \text{Seq } 'a ('a \text{ seq})$

fun $\text{concat} :: 'a \text{ seq} \Rightarrow 'a \text{ seq} \Rightarrow 'a \text{ seq}$

where

$\text{concat } \text{Empty } ys = ys$
 $\mid \text{concat } (\text{Seq } x \ xs) \ ys = \text{Seq } x \ (\text{concat } xs \ ys)$

theorem $\text{concat_empty}: \text{concat } xs \ \text{Empty} = xs$

by $(\text{induct } xs) \ \text{simp_all}$

theorem $\text{conc_assoc}: \text{concat } (\text{concat } xs \ ys) \ zs = \text{concat } xs \ (\text{concat } ys \ zs)$

by $(\text{induct } xs) \ \text{simp_all}$

Example: unstructured proof “scripts”

```
theorem concat_empty': concat xs Empty = xs  
  apply (induct xs)  
    apply simp  
    apply simp  
  done
```

```
theorem conc_assoc': concat (concat xs ys) zs = concat xs (concat ys zs)  
  apply (induct xs)  
    apply simp  
    apply simp  
  done
```

Example: abstract specifications and calculations

```
class group = times + one + inverse +  
  assumes group_assoc:  $(x * y) * z = x * (y * z)$   
  and group_left_one:  $1 * x = x$   
  and group_left_inverse:  $inverse\ x * x = 1$ 
```

```
theorem (in group) group_right_inverse:  $x * inverse\ x = 1$   
   $\langle proof \rangle$ 
```

```
theorem (in group) group_right_one:  $x * 1 = x$ 
```

```
proof —
```

```
  have  $x * 1 = x * (inverse\ x * x)$  by (simp only: group_left_inverse)
```

```
  also have  $\dots = x * inverse\ x * x$  by (simp only: group_assoc)
```

```
  also have  $\dots = 1 * x$  by (simp only: group_right_inverse)
```

```
  also have  $\dots = x$  by (simp only: group_left_one)
```

```
  finally show ?thesis .
```

```
qed
```

2 Proof Systems

Isabelle/Pure: formal context

Logical judgement:

$$\boxed{\Theta, \Gamma \vdash \varphi}$$

- **background theory** Θ
(polymorphic types, constants, axioms; **global data**)
- **proof context** Γ (fixed variables, assumptions; **local data**)

Operations on theories:

- merge and extend: $\Theta_3 = \Theta_1 \cup \Theta_2 + \tau + c :: \tau + c \equiv t$
- symbolic sub-theory relation: $\Theta_1 \subseteq \Theta_2$
- transfer of results: if $\Theta_1 \subseteq \Theta_2$ and $\Theta_1, \Gamma \vdash \varphi$ then $\Theta_2, \Gamma \vdash \varphi$

Isabelle/Pure: primitive inferences

Syntax (types and terms):

$fun :: (type, type) type$

$all :: ('a \Rightarrow prop) \Rightarrow prop$

$imp :: prop \Rightarrow prop \Rightarrow prop$

function space $'a \Rightarrow 'b$

universal quantification $\bigwedge x. B x$

implication $A \Longrightarrow B$

Derivations (theorems): implicit theory Θ

$$\frac{A \in \Theta}{\vdash A} \text{ (axiom)} \quad \frac{}{A \vdash A} \text{ (assume)}$$

$$\frac{\Gamma \vdash B[x] \quad x \notin \Gamma}{\Gamma \vdash \bigwedge x. B[x]} \text{ (\(\wedge\)-intro)} \quad \frac{\Gamma \vdash \bigwedge x. B[x]}{\Gamma \vdash B[a]} \text{ (\(\wedge\)-elim)}$$

$$\frac{}{\Gamma \vdash A \Longrightarrow B} \text{ (\(\Longrightarrow\)-intro)} \quad \frac{\Gamma_1 \vdash A \Longrightarrow B \quad \Gamma_2 \vdash A}{\Gamma_1 \cup \Gamma_2 \vdash B} \text{ (\(\Longrightarrow\)-elim)}$$

Isabelle/Isar: block-structured reasoning

Universal context: **fix** and **assume**

```
{  
  fix  $x$   
  have  $B\ x$  <proof>  
}  
have  $\bigwedge x. B\ x$  by fact
```

```
{  
  assume  $A$   
  have  $B$  <proof>  
}  
have  $A \implies B$  by fact
```

Existential context: **obtain**

```
{  
  obtain  $a$  where  $B\ a$  <proof>  
  have  $C$  <proof>  
}  
have  $C$  by fact
```

3 Proof Search

Isabelle/HOL proof methods

- *rule*: generic Natural Deduction (with HO unification)
- *cases*: elimination, syntactic representation of datatypes, inversion of inductive sets and predicates
- *induct* and *coinduct*: induction and coinduction of types, sets, predicates
- *simp*: equational reasoning by the Simplifier (HO rewriting), with possibilities for add-on tools
- *fast* and *blast*: classical reasoning (tableau)
- *auto* and *force*: combined simplification and classical reasoning
- *arith*, *presburger*: specific theories
- *smt*: Z3 with proof reconstruction

Sledgehammer

Idea:

- heavy external ATPs / SMTs for proof search
- light internal ATP (Metis) for proof reconstruction

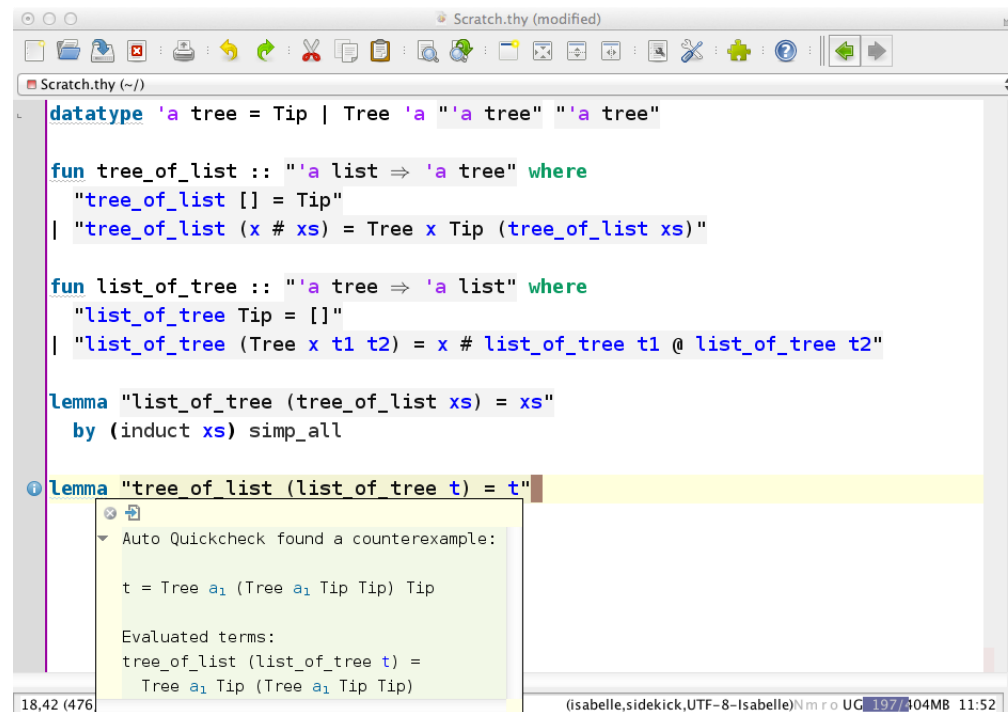
```
theory Scratch
imports Main
begin

Lemma "[x] = [y] ==> x = y" by (metis list.inject)

Provers: e spass z3
"e": Try this: by (metis the_elem_set) (6 ms).
"z3": Try this: by (metis list.inject) (7 ms).
"spass": Try this: by (metis list.inject) (11 ms).
```

Automated disprovers — counter examples

- **quickcheck** based on random functional evaluation
- **nitpick** based on relational model finder



The screenshot shows a window titled "Scratch.thy (modified)" with a toolbar and a text editor. The text editor contains the following code:

```
datatype 'a tree = Tip | Tree 'a "'a tree" "'a tree"

fun tree_of_list :: "'a list ⇒ 'a tree" where
  "tree_of_list [] = Tip"
| "tree_of_list (x # xs) = Tree x Tip (tree_of_list xs)"

fun list_of_tree :: "'a tree ⇒ 'a list" where
  "list_of_tree Tip = []"
| "list_of_tree (Tree x t1 t2) = x # list_of_tree t1 @ list_of_tree t2"

lemma "list_of_tree (tree_of_list xs) = xs"
  by (induct xs) simp_all

lemma "tree_of_list (list_of_tree t) = t"
```

A yellow highlight is under the last lemma. A tooltip is displayed over it, containing the following text:

```
Auto Quickcheck found a counterexample:

t = Tree a1 (Tree a1 Tip Tip) Tip

Evaluated terms:
tree_of_list (list_of_tree t) =
  Tree a1 Tip (Tree a1 Tip Tip)
```

The status bar at the bottom shows "18,42 (476)" on the left and "(isabelle,sidekick,UTF-8-Isabelle)Nm r o UC 197/04MB 11:52" on the right.

4 Proof Formats

Proof formats: open-ended, no standards

De-facto formats:

LCF and HOL: ML source as input and output

Coq: tactic scripts, e.g. Ltac, SSReflect

Isabelle/Isar:

- structured proof documents (Isar language)
- unstructured apply scripts (tactic emulation)

General LCF approach:

use ML to implement your own application-specific proof formats

5 Proof Production

The “LCF approach”

Correctness by construction: (R. Milner, 1979)

1. abstract datatype *thm* in ML (the “meta language”),
constructors are the rules of the logic (the “object language”)
2. implementation of arbitrary proof tools in ML,
with explicit *thm* construction at run-time

Notes:

- need to distinguish proof search from actual *thm* inferences
- *thm* values are abstract: proofs are not stored in memory,
but: optional proof trace or proof term
- goal-directed LCF-approach fits well to shared-memory multiprocessing (multicore hardware)

6 Proof Consumption

Proof consumption in Isabelle/HOL

HOL-Light importer:

replay of primitive inferences from other LCF-kernel (huge trace)

SMT proof method:

connection to Z3, with proof reconstruction by standard proof tools of Isabelle/HOL: *simp*, *blast*, *auto* etc.

Sledgehammer:

- heavy external ATPs / SMTs for proof search
- light internal ATP (Metis) for proof reconstruction

7 Proof Applications

Big formalization projects

Flyspeck <https://code.google.com/p/flyspeck> (T. Hales, HOL-Light): formal proof of Kepler's Conjecture

L4.verified <http://ertos.nicta.com.au/research/l4.verified> (G. Klein, Isabelle/HOL): formally correct operating system kernel

Feit-Thompson Odd Order Theorem <http://www.msr-inria.fr/news/feit-thomson-proved-in-coq> (G. Gonthier, Coq/SSReflect)

CompCert verified compiler <http://compcert.inria.fr/doc> (X. Leroy, Coq): optimizing C-compiler for various assembly languages, written and proven in the functional language of Coq

Libraries of formalized mathematics

Archive of Formal Proofs (AFP)

<http://afp.sf.net>

Isabelle/HOL

Mathematical Components

<http://www.msr-inria.fr/projects/mathematical-components-2>

Coq/SSReflect

Mizar Mathematical Library

<http://www.mizar.org/library>

Mizar

8 Conclusions

What is ITP? What is Isabelle/Isar?



Hanabusa Itchō: "Blind monks examining an elephant"

Helpful hints

New users:

- Spend time to develop a sense for more than one accidental candidate, before making a commitment.
- Spend substantial time to become proficient with the system of your choice.

Old users:

- Learn how other proof assistants work, and what are their specific strengths and weaknesses.

Isabelle users:

- Submit your finished applications to AFP <http://afp.sf.net>

Happy proving!