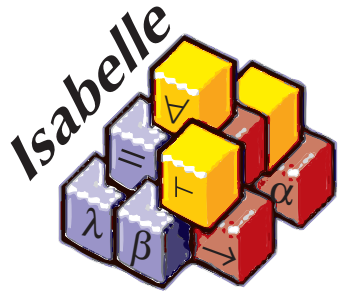
Programs and Proofs in Isabelle/HOL

Makarius Wenzel
http://sketis.net

March 2016



Introduction

What is Isabelle?

Hanabusa Itchō: “Blind monks examining an elephant”

History: LCF Prover Family

LCF: Logic + Meta Language (since 1979)
- Edinburgh LCF
- Cambridge LCF

HOL (since 1988)
- ProofPower
- HOL4
- HOL-Light
- HOL Zero

Coq (since 1985)
- Coq 8.5 (January 2016)

Isabelle (since 1986)
- Isabelle2016 (February 2016)

Isabelle: framework of domain-specific formal languages

Logic:

Isabelle/Pure: Logical framework and bootstrap environment

Isabelle/HOL: Theories and tools for applications

Programming:

Isabelle/ML: Tool implementation (Poly/ML)

Isabelle/Scala: System integration (JVM)

Proof:

Isabelle/Isar: Intelligible semi-automated reasoning

Document preparation: LaTeX type-setting of proof text

Document preparation

Structure markup

- section headings: **chapter**, **section**, **subsection**, etc.
- text blocks: **text**
- implicit lists (cf. Markdown): itemize, enumerate, description
- free-form LaTeX macros

Antiquotations

full form: `@{`*name* [*options*] *arguments* ...`}`

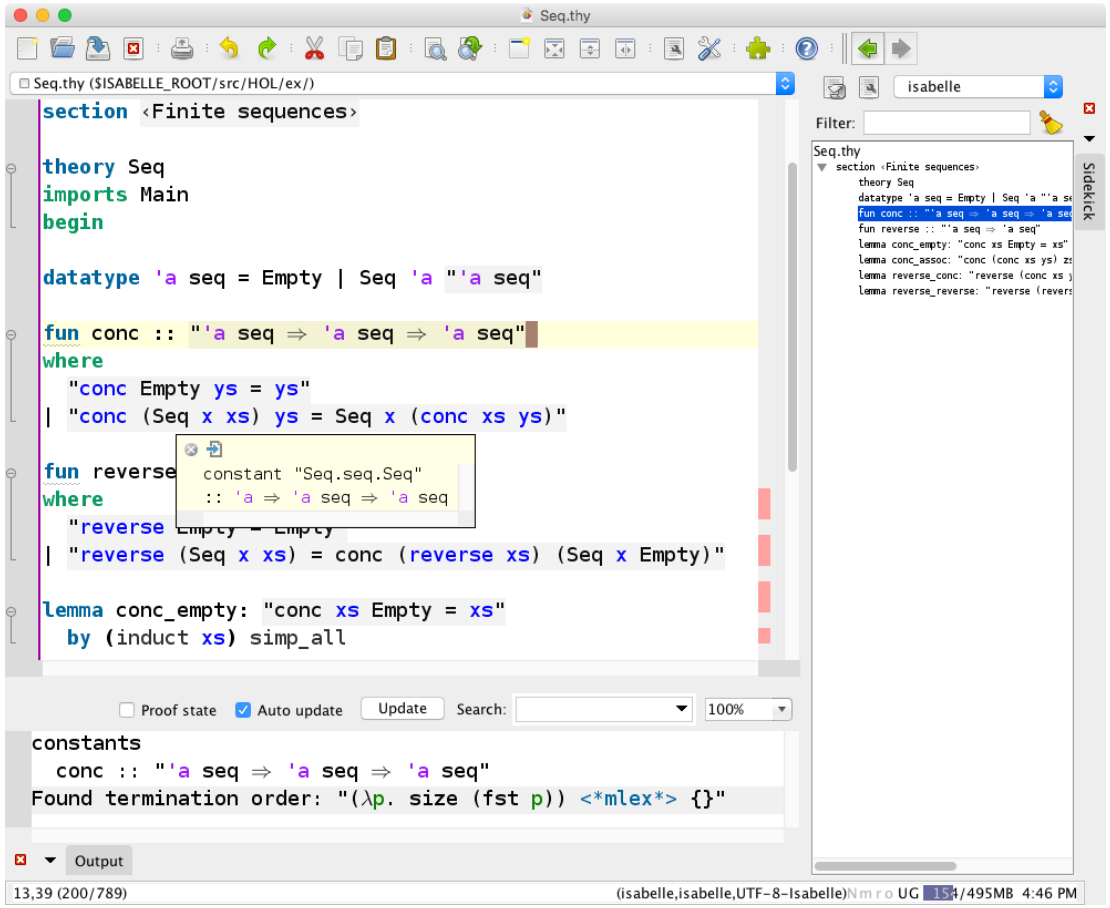
short form:

1. cartouche argument: `\<^`*name*`>`⟨*argument*⟩
2. no argument: `\<^`*name*`>`

Example: ≫*this document*≪

Prover IDE: Isabelle/jEdit

- asynchronous interaction
- continuous checking
- parallel processing



Programs in Isabelle

Isabelle/ML: tool implementation language

- based on Poly/ML (David Matthews, Edinburgh)
- SML'97: strict functional programming + exceptions
- SML'90: interrupts
- parallel evaluation via futures (implemented via Poly/ML threads)
- immutable data managed within logical context
- statically checked antiquotations

Example

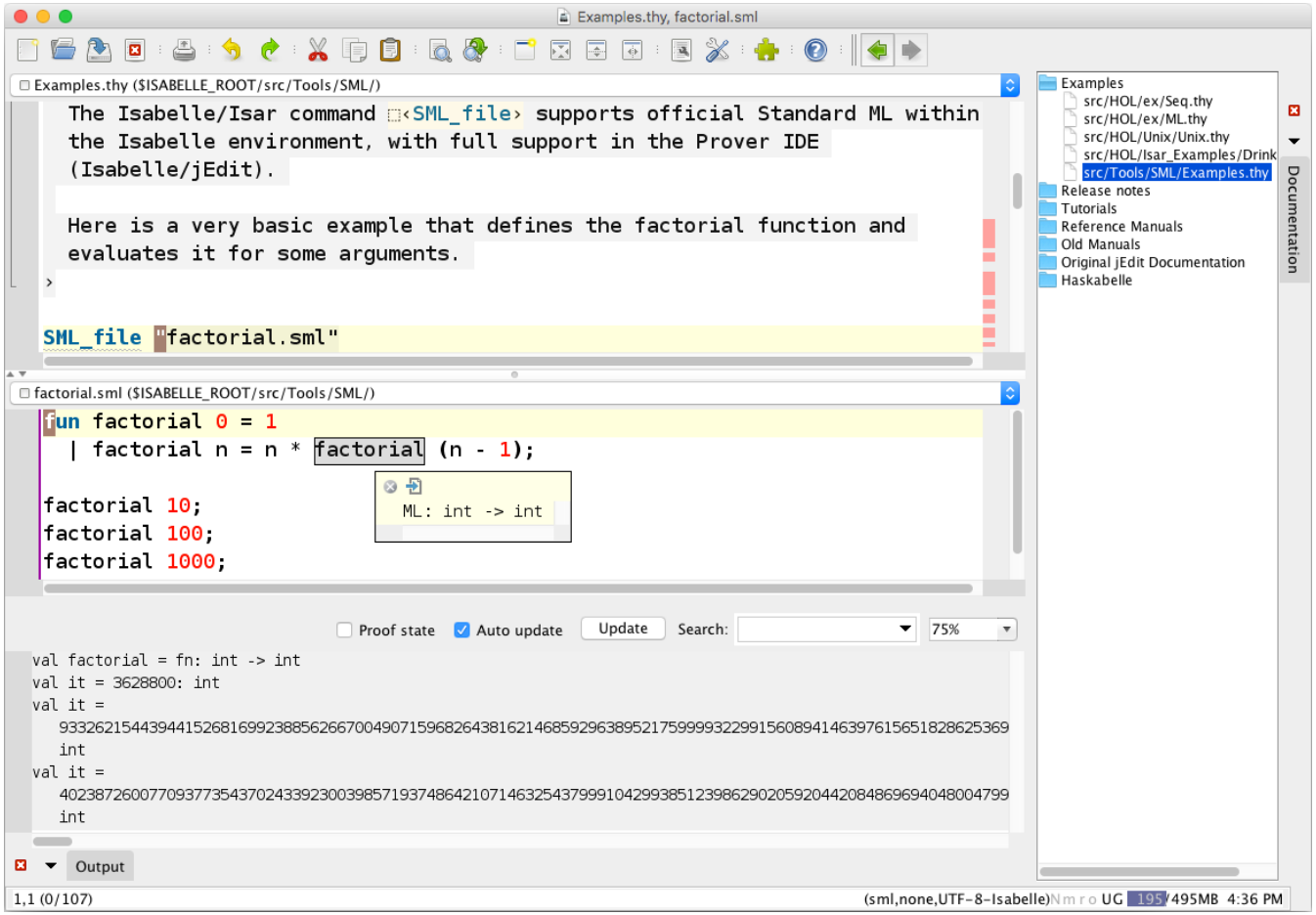
ML ⟨

```
fun conj 0 = @{term True}
  | conj 1 = @{term A :: bool}
  | conj n = @{term op ∧} $ conj 1 $ conj (n − 1);
```

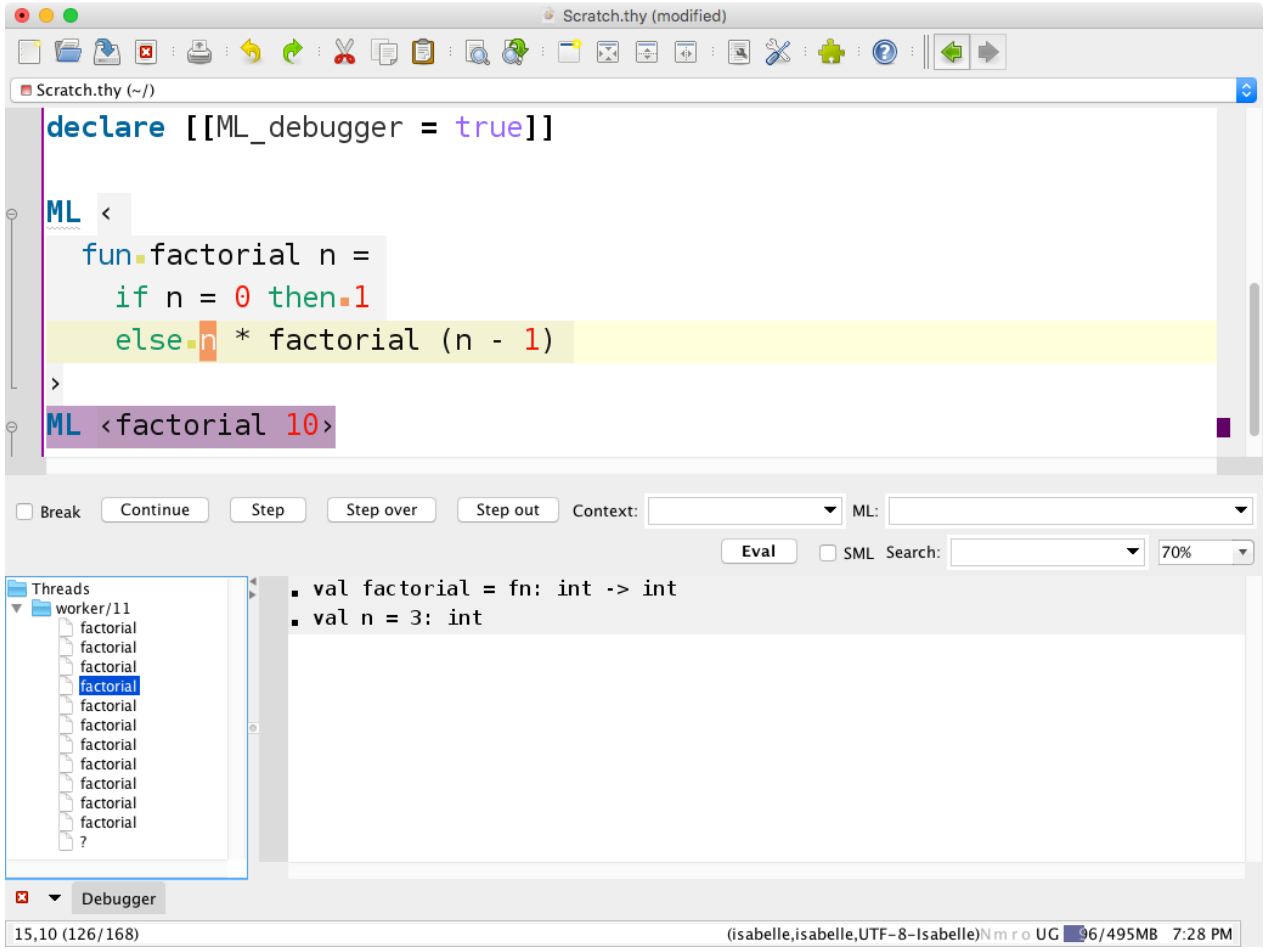
⟩

ML ⟨*writeln* (*Syntax.string_of_term* @{*context*} (*conj* 7))⟩

Isabelle/ML: IDE

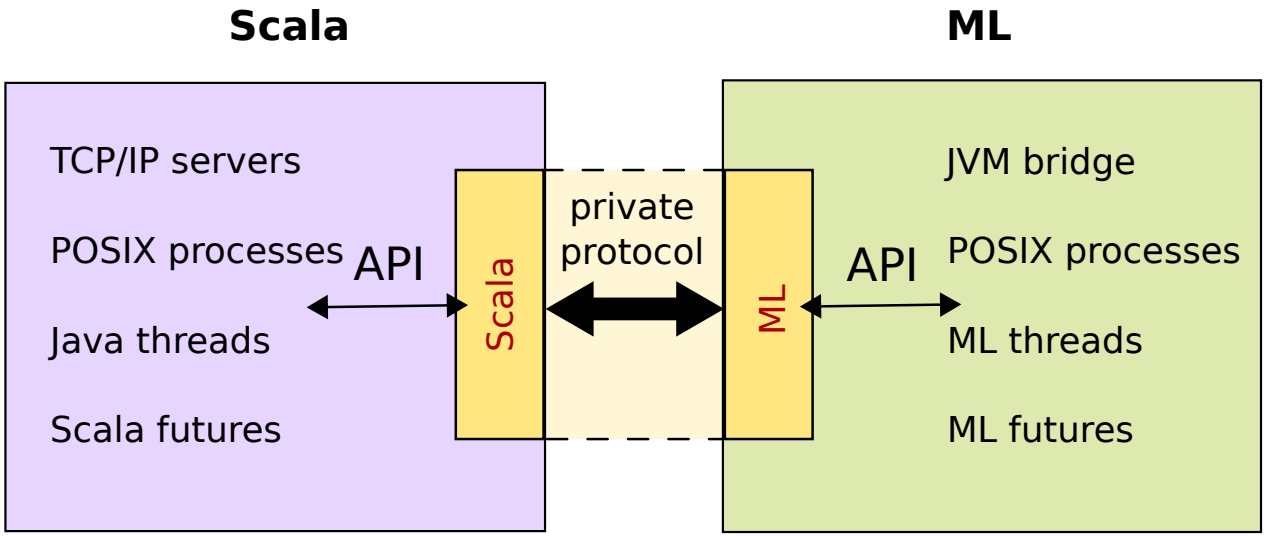


Isabelle/ML: debugger



Isabelle/Scala: system programming language

- Standard Scala 2.11 (running on Java 8)
- Isabelle/Scala programming style imitates Isabelle/ML
- duplication of some libraries on both sides
- differentiation for system front-end technology



"Programming" in Isabelle/HOL

Warning:

- Not every computer language is a programming language!
- HOL is classic set-theory — more than a programming language.
- HOL is based on total functions — less convenient than common programming languages.

Quasi-programming in HOL:

1. define conventional types: tuples, records, recursive datatypes
2. define recursive functions over types (with well-formedness proofs)
3. simulate computation via equational reasoning:
   (a) term rewriting within the logic (Simplifier)
   (b) symbolic normalization by evaluation (NBE)
   (c) actual evaluation via code-generator:
   HOL subset is translated to SML, OCaml, Scala, Haskell

Examples

- `$ISABELLE_HOME/src/HOL/ex/Seq.thy`

  export_code *conc* **in** *SML*
  export_code *conc* **in** *OCaml*
  export_code *conc* **in** *Scala*
  export_code *conc* **in** *Haskell*

- Run Length Encoding: `RLE.thy`

- See also documentation *prog−prove*:
  "Programming and Proving in Isabelle/HOL" (Tobias Nipkow)

Proofs in Isabelle

Structured Proofs

Natural Deduction: (Gentzen 1935)

$$\frac{A \longrightarrow B \quad A}{B} \qquad \frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \longrightarrow B}$$

Isabelle/Pure rules: (Paulson 1989)

$$(A \longrightarrow B) \Longrightarrow A \Longrightarrow B \qquad (A \Longrightarrow B) \Longrightarrow A \longrightarrow B$$

Isabelle/Isar proofs: (Wenzel 1999)

have $A \longrightarrow B$ $\langle proof \rangle$
also have A $\langle proof \rangle$
finally have B **.**

have $A \longrightarrow B$
proof
  assume A
  then show B $\langle proof \rangle$
qed

Rules for predicate logic (1)

theorem $impI$: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$
theorem mp: $A \longrightarrow B \Longrightarrow A \Longrightarrow B$

theorem $allI$: $(\bigwedge x.\ B\ x) \Longrightarrow \forall\, x.\ B\ x$
theorem $spec$: $\forall\, x.\ B\ x \Longrightarrow B\ a$

theorem exI: $B\ a \Longrightarrow \exists\, x.\ B\ x$
theorem exE: $\exists\, x.\ B\ x \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C$

theorem $conjI$: $A \Longrightarrow B \Longrightarrow A \wedge B$
theorem $conjE$: $A \wedge B \Longrightarrow (A \Longrightarrow B \Longrightarrow C) \Longrightarrow C$

theorem $disjI1$: $A \Longrightarrow A \vee B$
theorem $disjI2$: $B \Longrightarrow A \vee B$
theorem $disjE$: $A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$

Rules for predicate logic (2)

theorem $TrueI$: $True$
theorem $FalseE$: $False \Longrightarrow C$

theorem $notI$: $(A \Longrightarrow False) \Longrightarrow \neg\ A$
theorem $notE$: $\neg\ A \Longrightarrow A \Longrightarrow C$

theorem $iffI$: $(A \Longrightarrow B) \Longrightarrow (B \Longrightarrow A) \Longrightarrow A \longleftrightarrow B$
theorem $iffD1$: $A \longleftrightarrow B \Longrightarrow A \Longrightarrow B$
theorem $iffD2$: $A \longleftrightarrow B \Longrightarrow B \Longrightarrow A$

Examples

theorem $curry\colon (A \wedge B \longrightarrow C) \longrightarrow (A \longrightarrow B \longrightarrow C)$ $\langle proof \rangle$

theorem $iff\_contradiction\colon \neg A \longleftrightarrow A \Longrightarrow C$ $\langle proof \rangle$

Example proof patterns: induction and calculation

theorem fixes $n :: nat$ **shows** $P\ n$
proof $(induct\ n)$
  case 0
  show $P\ 0$ $\langle proof \rangle$
next
  case $(Suc\ n)$
  then show $P\ (Suc\ n)$ $\langle proof \rangle$
qed

notepad
begin
  have $a = b$ $\langle proof \rangle$
  also have $\ldots = c$ $\langle proof \rangle$
  also have $\ldots = d$ $\langle proof \rangle$
  finally have $a = d$ **.**
end

Example proof: induction × calculation

theorem
  fixes $n :: nat$
  shows $(\sum i{=}0..n.\ i) = n * (n + 1)\ div\ 2$
proof $(induct\ n)$
  case 0
  have $(\sum i{=}0..0.\ i) = (0{::}nat)$ **by** $simp$
  also have $\ldots = 0 * (0 + 1)\ div\ 2$ **by** $simp$
  finally show $?case$ **.**
next
  case $(Suc\ n)$
  have $(\sum i{=}0..Suc\ n.\ i) = (\sum i{=}0..n.\ i) + (n + 1)$ **by** $simp$
  also have $\ldots = n * (n + 1)\ div\ 2 + (n + 1)$ **by** $(simp\ add{:}\ Suc.hyps)$
  also have $\ldots = (n * (n + 1) + 2 * (n + 1))\ div\ 2$ **by** $simp$
  also have $\ldots = (Suc\ n * (Suc\ n + 1))\ div\ 2$ **by** $simp$
  finally show $?case$ **.**
qed

Examples

- `~~/src/HOL/Isar_Examples/First_Order_Logic.thy`
- `~~/src/HOL/Isar_Examples/Higher_Order_Logic.thy`
- `~~/src/HOL/Isar_Examples/Cantor.thy`
- `~~/src/HOL/Isar_Examples/Schroeder_Bernstein.thy`

Conclusion

Happy proving!

See you on http://afp.sf.net

The Archive of Formal Proofs