Isabelle/HOL and the UTP

Part 1: Isabelle basics

Simon Foster

University of York

January 29, 2013

Isabelle/HOL

- Isabelle – a generic proof assistant
  - proof checking (decidable)
  - proof automation (undecidable)
- HOL – Higher Order Logic
  - Functional Programming: $f = reverse \cdot map\ g$
  - Logic: $\forall xs.\ map\ f\ (map\ g\ xs) = map\ (f \cdot g)\ xs$
  - similar syntax to ML and Haskell
- LCF-style: proofs correct by construction wrt. a small core
- Large library of theories (Sets, Lists, Lattices, Automata etc.)
- Robust technology ($>$ 25 years in the making)

Logics

- **Two levels of logic**
- Meta-logic (Pure):
  - Types: $\sigma, \rho, \sigma \Rightarrow \rho$
  - λ-calculus: λ f. λ x. g f x
  - Propositions: $\bigwedge, \Longrightarrow, \equiv$
- Object-logic (HOL): $\wedge, \vee, \rightarrow, \forall, \exists, = \cdots$

- **Two levels of syntax**
- Outer-syntax: Isabelle language stuctural elements
- Inner-syntax: Terms of the logic (types, propositions etc.) (usually quoted)
- Unicode syntax; e.g. type "=>" for $\Rightarrow$ (or type LaTeX)

Use of Isabelle

- Two components: the proof engine and user interface
- Several interfaces available:
  - Proof General (emacs) [included]
  - PIDE (jEdit) [included]
  - I3P (Netbeans)
- User interacts with a proof-state, targetting a goal
- Proofs take the form of scripts which manipulate this state
- Commands update the global state with new facts

Useful references

Isabelle Documentation:
`http://isabelle.in.tum.de/documentation.html`

- **Tutorial on Isabelle/HOL**
- **Isabelle/Isar Reference Manual**
- **What's in Main** - useful resource for contents of HOL

Course material:
`http://www-users.cs.york.ac.uk/~simonf/Isabelle/`

A functional programming language

- HOL contains a ML/Haskell style programming language
- Allows usual functional programming constructs
- Functions - which are always pure (mathematical)
  - should be total
  - can be recursive (when a suitable termination order exists)
- Datatypes
  - synonyms, algebraic datatypes, records, subtypes
  - algebraic datatypes support induction as usual
  - subtypes must be accompanied by a membership proof
  - types in Isabelle must be non-empty
- **Tree Example**

Isabelle as Proof Checker

- proof in Isabelle is goal-directed
- user proposes a logical goal and then must provide a proof
- a proof is a sequence of commands acting on the proof-state
- these proof scripts are like Isabelle machine-code
- invalid commands are rejected by Isabelle
- user variously splits and simplifies goal (divide & conquer)
- when no more goals remain - QED
- results is then added to Isabelle's properties
- proofs can use properties already established
- HOL contains a large library of existing proofs

Isabelle as Theorem Prover

- manual application of rules very tedious
- Isabelle contains various automated proof tools
- the most important is the simplifier
- at its simplest performs recursive equational rewriting
- can also solve simple statements of logic
- driven by Isabelle's powerful higher-order unifier
- has a database of verified rewrite rules, which can be extended

if we know that $x + 0 = x$ for any x then

$$x * (y + 0) \Longrightarrow x * y$$

Isabelle Syntax

- Most statements of logic (inner-syntax) should be places in speech-marks
- Isabelle employs LaTeX-like tags for unicode syntax
- e.g. $\forall x.\exists y.y > x$ is written as per LaTeX.
- see the cheatsheet for codes

Practical: Dates

- A date at its most basic, is a triple of natural number
- But this allows invalid combinations
- We are going to build an Isabelle type which represents only valid ones
- This includes leap-years

Summary

- Getting started with Isabelle/HOL
- Isabelle/HOL functional programming
- basic proof scripts
- the simplifier

Next time

- Deductive reasoning
- Inductive proofs
- The Isar natural proof scripting language
- Automating proofs (auto, blast, sledgehammer etc.)