Programs and Proofs in Isabelle/HOL
Example: Run-Length Encoding

Makarius Wenzel
http://sketis.net

May 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

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 $\langle$

$\quad$ $\mathit{fun\ conj}\ 0 = @\{\mathit{term\ True}\}$

$\qquad$ $|\ \mathit{conj}\ 1 = @\{\mathit{term}\ A :: \mathit{bool}\}$

$\qquad$ $|\ \mathit{conj}\ n = @\{\mathit{term\ op}\ \wedge\}\ \$\ \mathit{conj}\ 1\ \$\ \mathit{conj}\ (n - 1);$

$\rangle$

ML $\langle \mathit{writeln}\ (\mathit{Syntax.string\_of\_term}\ @\{\mathit{context}\}\ (\mathit{conj}\ 7))\rangle$

“Programming” in Isabelle/HOL

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

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.

Examples

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

- `$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`