Aspects of locality in Isabelle — local proofs, local theories, and local everything

Makarius Wenzel

February 2008

Technology develops from the primitive
via the complicated
to the simple.

Antoine de Saint-Exupéry

Introduction

What is locality anyway?

Locality means . . .

- working relatively to a *context*
  (theory or proof environment)
- moving results between contexts via *morphisms*
  e.g. from abstract theory to concrete application
- replacing logical encodings by *native elements* of Isabelle/Isar

Consequences:

- reduced complexity of logical statements, proofs etc.
- improved flexibility and scalability
- simplified construction and composition of add-on tools

PART I

Local proofs

Rules and proofs

Textbook inferences: proof trees

$$\frac{B(a)}{\exists\, x.\ B(x)} \qquad \frac{\exists\, x.\ B(x) \qquad \begin{array}{c}[x][B(x)] \\ \vdots \\ C\end{array}}{C}$$

Isabelle/Pure rules: framework formulae

$$B\ a \Longrightarrow (\exists\, x.\ B\ x) \qquad (\exists\, x.\ B\ x) \Longrightarrow (\bigwedge x.\ B\ x \Longrightarrow C) \Longrightarrow C$$

Isabelle/Isar proofs: proof texts

assume $B\ a$	**assume** $\exists\, x.\ B\ x$
then have $\exists\, x.\ B\ x$ **..**	**then obtain** a **where** $B\ a$ **..**

Isabelle/Isar proof decomposition

have $\bigwedge x.\ A\ x \Longrightarrow B\ x$
proof $-$
  fix x
  assume $A\ x$
  show $B\ x$
    $\langle proof \rangle$
qed

Decomposition in two parts:

1. Rule extracted from proof body
   (conclusion within a local context)
2. Result retrofitted into pending goal
   (very flexible due to higher-order unification etc.)

Isabelle/Pure rule composition

$$\frac{\begin{array}{rl} \mathit{rule}: & \overline{A}\ \overline{a} \Longrightarrow B\ \overline{a} \\ \mathit{goal}: & (\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\ \overline{x}) \Longrightarrow C \\ \mathit{goal\ unifier}: & (\lambda \overline{x}.\ B\ (\overline{a}\ \overline{x}))\ \theta = B' \theta \end{array}}{(\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow \overline{A}\ (\overline{a}\ \overline{x}))\ \theta \Longrightarrow C\ \theta}\ (\mathit{resolution})$$

$$\frac{\begin{array}{rl} \mathit{goal}: & (\bigwedge \overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow A\ \overline{x}) \Longrightarrow C \\ \mathit{assm\ unifier}: & A\ \theta = H_i\ \theta\ \ \text{(for some } H_i\text{)} \end{array}}{C\ \theta}\ (\mathit{assumption})$$

Flexibility in proof composition:

- rename / permute parameters (**fix**)
- permute assumptions (**assume**) and goals (**show**)
- generalize claim (**fix** / **assume** / **show**)

Isabelle/Isar proof contexts

{
  fix x
  have $B\ x$ $\langle proof \rangle$
}
note $\langle \bigwedge x.\ B\ x \rangle$

{
  assume A
  have B $\langle proof \rangle$
}
note $\langle A \Longrightarrow B \rangle$

{
  def $x \equiv a$
  have $B\ x$ $\langle proof \rangle$
}
note $\langle B\ a \rangle$

{
  obtain a **where** $B\ a$ $\langle proof \rangle$
  have C $\langle proof \rangle$
}
note $\langle C \rangle$

From contexts to statements

Idea:

- Avoid unwieldy logical formula, i.e.
  no object-logic: $\forall x.\ A\ x \longrightarrow B\ x$
  no meta-logic: $\bigwedge x.\ A\ x \Longrightarrow B\ x$
- Use native Isar context & conclusion elements (for x, $A\ x \vdash B\ x$)

  fixes x
  assumes $A\ x$
  shows $B\ x$

  or enriched version:

  fixes x_1 **and** x_2 **and** $\ldots$
  assumes $a_1\ [att]$: $A_1\ \overline{x}$ (**is** $?P_1$) **and** $a_2\ [att]$: $A_2\ \overline{x}$ (**is** $?P_2$) **and** $\ldots$
  shows $b_1\ [att]$: $B_1\ \overline{x}$ (**is** $?Q_1$) **and** $b_2\ [att]$: $B_2\ \overline{x}$ (**is** $?Q_2$) **and** $\ldots$

Universal contexts — introduction rules

Context elements **fix** / **assume** in structured proofs:

{
  fix x
  have $B\ x\ \langle proof \rangle$
}
note $\langle \bigwedge x.\ B\ x \rangle$

{
  assume A
  have $B\ \langle proof \rangle$
}
note $\langle A \Longrightarrow B \rangle$

Corresponding top-level statement for rule $\bigwedge x.\ A\ x \Longrightarrow B\ x$:

theorem
  fixes x
  assumes $A\ x$
  shows $B\ x$

Existential contexts — elimination rules

Derived context element **obtain** in structured proofs:

{
  obtain a **where** $B\ a$ $\langle proof \rangle$
  have C $\langle proof \rangle$
}
note $\langle C \rangle$

Corresponding top-level statement for $\exists\,/\wedge$ elimination rules etc.:

theorem
  assumes $\exists\, x.\ B\ x$
  obtains a **where** $B\ a$

theorem
  assumes $A \wedge B$
  obtains A **and** B

theorem
  assumes $A \vee B$
  obtains $(left)\ A \mid (right)\ B$

theorem
  fixes $x\ y :: nat$
  obtains $(lt)\ x < y \mid (eq)\ x = y \mid (gt)\ x > y$

Example: Classical FOL

$conjI$: **assumes** A **and** B **shows** $A \wedge B$
$conjE$: **assumes** $A \wedge B$ **obtains** A **and** B

$disjI_1$: **assumes** A **shows** $A \vee B$
$disjI_2$: **assumes** B **shows** $A \vee B$
$disjE$: **assumes** $A \vee B$ **obtains** A **|** B

$impI$: **assumes** $A \Longrightarrow B$ **shows** $A \longrightarrow B$
$impE$: **assumes** $A \longrightarrow B$ **and** A **obtains** B

$allI$: **assumes** $\bigwedge x.\ B\ x$ **shows** $\forall\, x.\ B\ x$
$allE$: **assumes** $\forall\, x.\ B\ x$ **obtains** $B\ a$

exI: **assumes** $B\ a$ **shows** $\exists\, x.\ B\ x$
exE: **assumes** $\exists\, x.\ B\ x$ **obtains** x **where** $B\ x$

$classical$: **obtains** $\neg\ thesis$
$Peirce$: **obtains** $thesis \Longrightarrow A$

Advantages of native Isar statements

- Simple re-use of more sophisticated Isar proof elements
- Scalable goal specifications
- Reduced complexity for formal proofs in
  1. proving / using the result
  2. structured Isar proof / tactic scripts / internal proof objects

Consequences:

- Reduced "formality" — towards "logic-free reasoning"
- May have to unlearn first-order logic!

PART II

Local theories

Motivation

- Infrastructure for organizing definitions and proofs
- Separation of concerns:
  1. definitional packages (e.g. **inductive**, **function**)
  2. target mechanisms (e.g. **locale**, **class**)

  $\rightarrow$ large product space: $\mathit{definitions} \times \mathit{targets}$
- Simplification and generalization of Isabelle/Isar concepts

Definitional elements within universal contexts

	λ	let
terms	**fix** $x :: \tau$	**define** $c \equiv t$
thms	**assume** a: A	**note** $b = \langle B \rangle$

Note: separation of axiomatic vs. definitional specifications!

Hindley-Milner polymorphism:

- Assumptions: fixed types
  fix $id :: \alpha \Rightarrow \alpha$
  assume $id\text{-}def$: $id \equiv \lambda x :: \alpha.\ x$
- Conclusions: arbitrary types
  define $id \equiv \lambda x :: \alpha.\ x$ (for arbitrary α)
  note $refl = \langle \bigwedge x :: \alpha.\ x = x \rangle$ (for arbitrary α)

Example (1): global definitions

theory *Ex1* **imports** *Main*
begin

inductive $path :: ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
  for $rel :: 'a \Rightarrow 'a \Rightarrow bool$
where $base\colon path\ rel\ x\ x$
  $|\ step\colon rel\ x\ y \Longrightarrow path\ rel\ y\ z \Longrightarrow path\ rel\ x\ z$

theorem assumes $path\ rel\ x\ z$ **shows** $P\ x\ z$
using $\langle path\ rel\ x\ z\rangle$
proof *induct*
  { case $(base\ x)$ **then show** $?case$ $\langle proof\rangle$ **}**
  { case $(step\ x\ y\ z)$ **then show** $?case$ $\langle proof\rangle$ **}**
qed

end

Example (2a): local definitions

theory *Ex2* **imports** *Main* **begin**

locale *rel* = **fixes** *rel* :: $'a \Rightarrow 'a \Rightarrow bool$
begin

inductive *path* :: $'a \Rightarrow 'a \Rightarrow bool$
where $base$: $path\ x\ x$ | $step$: $rel\ x\ y \Longrightarrow path\ y\ z \Longrightarrow path\ x\ z$

theorem assumes $path\ x\ z$ **shows** $P\ x\ z$
using $\langle path\ x\ z \rangle$
proof *induct*
  { case $(base\ x)$ **then show** *?case* $\langle proof \rangle$ **}**
  { case $(step\ x\ y\ z)$ **then show** *?case* $\langle proof \rangle$ **}**
qed

end

Example (2b): adding local results

context *rel*
begin

theorem
  assumes *path* x z
  obtains $x = z$
    $|$ y **where** $x \neq z$ **and** *rel* x y **and** *path* y z
  using *assms* **by** *cases auto*

end

end

Local theory infrastructure

theory	background environment (abstract certificate)
context	main working environment (contains *theory*)
local-theory	$\approx$ *target-context* $\times$ *virtual-context*
	$+$ interpretation of conclusions

Standard interpretation by λ-lifting (over **fix** x **assume** $A\ x$):

$thy.c \equiv \lambda x.\ t$	$loc.c \equiv thy.c\ x$	**define** $c \equiv t$
$thy.b = \langle \bigwedge x.\ A\ x \Longrightarrow B \rangle$	$loc.b = \langle B \rangle$	**note** $b = \langle B \rangle$

Applications (1): specification packages

Local specifications:

- **definition** and **theorem** (wrapper for **define** and **note** primitives) [M. Wenzel, 2006]
- **abbreviation** (abstract syntax) and **notation** (concrete syntax) [M. Wenzel, 2006]
- **inductive** (inductive predicates defined as least-fixed point) [S. Berghofer, 2006/2007]
- **function** (general recursive functions defined as inductive graph) [A. Krauss, 2006/2007]

Global type constructions (non-dependent):

- **datatype** (infinitary tree types) [really soon]
- **record** (nested tuples) [really soon]

Applications (2): target mechanisms

- **locale** loc = **fixes** x **assumes** $A\ x$:
  interpret **define** and **note** via λ-lifting
  [C. Ballarin, M. Wenzel, 2006/2007]
- **class** $\approx$ **locale** + **interpretation**:
  second interpretation in terms of polymorphic **consts**
  (dictionary construction)
  [F. Haftmann, M. Wenzel, 2006/2007]
- **instantiation** $c :: (\overline{S})\ S$ [really soon]
  replace dependencies on **fixes** by instances of polymorphic **consts**
  (internal overloading)
- **statespace** $s = imports + fields$
  modular statespaces (with merge, rename)
  [N. Schirmer, 2007]

PART III

Local everything

Generic context data

Internally record of data-slots (dynamically typed disjoint sums)

Programming interface recovers strongly static typing

functor $ProofDataFun(ARGS)$: $RESULT$, where

$$\begin{array}{rcl} ARGS & = & \textit{sig} \;\; \textit{type}\ T \;\; \textit{val}\ init\colon theory \longrightarrow T \;\; \textit{end} \\ RESULT & = & \textit{sig} \;\; \textit{val}\ put\colon T \longrightarrow context \longrightarrow context \;\; \textit{val}\ get\colon context \longrightarrow T \;\; \textit{end} \end{array}$$

Example content:

- Logical declarations (variables, assumptions)
- Definitions (terms, theorems)
- Type-inference information
- Syntax annotations (mixfix grammar)
- Hints for proof tools (simpset, claset, arithmetic setup etc.)

Generic declarations

	λ	*let*	*generic*
terms	**fix** $x :: \tau$	**define** $c \equiv t$	**term-syntax** $\langle\!\langle d \rangle\!\rangle$
thms	**assume** a: A	**note** $b = \langle B \rangle$	**declaration** $\langle\!\langle d \rangle\!\rangle$

where d: $morphism \rightarrow context \rightarrow context$

Logical transformations: (for $type$, $term$, thm)

$transform\text{-}thm$: $morphism \rightarrow thm \rightarrow thm$

Arbitrary transformations: (for $morphism \rightarrow \alpha$)

$transform$: $morphism \rightarrow (morphism \rightarrow \alpha) \rightarrow (morphism \rightarrow \alpha)$
$transform\ \varphi\ f \equiv \lambda\psi.\ f\ (\psi \circ \varphi)$

$form$: $(morphism \rightarrow \alpha) \rightarrow \alpha$
$form\ f \equiv f\ identity$

Application: localized proof tools

Implementation pattern [A. Chaieb, M. Wenzel, 2007]
e.g. method $algebra$ in Isabelle/HOL:

1. *abstract theory context* defined as **locale** or **class**
   e.g. ring structures for Gröbner Bases
2. *extra-logical context data* maintained within the context
   e.g. ML functions to detect canonical ring constants, prove conversions etc.
3. *tool implementation* depending on a morphism that transfers all data into concrete application context

Note:

- require "polymorphic" tool $\leftrightarrow$ tool-compliant morphisms
- type-classes more robust than arbitrary locales

More locality: local executions and concurrency

Idea:

- independent executions *relative* to explicit theory/proof context

Benefits:

- faster loading of *theory subgraphs* (available in Isabelle2007)
- much faster loading of single theories due to *proof irrelevance* [future]
- much faster interactive development due to *asynchronous checking* [future]

$\longrightarrow$ towards *stateless* interactive theorem proving

Example: irrelevant proofs

lemma $[simp]$: $attributes\ (Val\ (att,\ text)) = att$

by $(simp\ add$: $attributes\text{-}def)$

lemma $[simp]$: $attributes\ (Env\ att\ dir) = att$

by $(simp\ add$: $attributes\text{-}def)$

lemma $[simp]$: $attributes\ (map\text{-}attributes\ f\ file) = f\ (attributes\ file)$

by $(cases\ file)\ (simp\text{-}all\ add$: $attributes\text{-}def\ map\text{-}attributes\text{-}def\ split\text{-}tupled\text{-}all)$

lemma $[simp]$: $map\text{-}attributes\ f\ (Val\ (att,\ text)) = Val\ (f\ att,\ text)$

by $(simp\ add$: $map\text{-}attributes\text{-}def)$

lemma $[simp]$: $map\text{-}attributes\ f\ (Env\ att\ dir) = Env\ (f\ att)\ dir$

by $(simp\ add$: $map\text{-}attributes\text{-}def)$

Example: sub-structured proofs

theorem *transition-uniq*:
  assumes $root'$: $root\ {-x\rightarrow}\ root'$ **and** $root''$: $root\ {-x\rightarrow}\ root''$
  shows $root' = root''$ **using** $root''$

```
proof cases
  case read
  with root' show ?thesis   by cases auto
next
  case write
  with root' show ?thesis   by cases auto
next
  case chmod
  with root' show ?thesis   by cases auto
next
...
qed
```

Conclusion

General Isabelle trends

- From a pure *logical framework* towards a general *software framework* for logic applications
- Isabelle/Isar as "logical operating system" to integrate tools for formal logic (centered around *local context*)
- Isabelle2007 greatly improves upon this infrastructure
- More to come . . .