Local Theories

Makarius Wenzel
TU München

August 2009

Motivation

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

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

Context-dependent definitions

	λ-binding	let-binding
types	fixed α	arbitrary β
terms	**fix** x	**define** $c \equiv b\ x$
theorems	**assume** $A\ x$	**note** $c = \langle A\ x \vdash B\ x \rangle$

Note: clear 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 $id \equiv \lambda x :: \beta.\ x$ (for arbitrary β)
  note $refl = \langle \bigwedge x :: \beta.\ x = x \rangle$ (for arbitrary β)

Examples

See `Slides3/Ex1.thy`

Local theory infrastructure

theory	background environment (abstract certificate)
context	main working environment (contains *theory*)
local-theory	*auxiliary-context* $\times$ *target-context*
	$+$ interpretation of specification elements

auxiliary context

target context

background theory

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

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

Morphisms

Idea: moving formal entities between contexts

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

$transform\text{-}type$: $morphism \to type \to type$
$transform\text{-}term$: $morphism \to term \to term$
$transform\text{-}thm$: $morphism \to thm \to thm$

Arbitrary transformations: for $morphism \to \alpha$

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

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

Generic declarations

	λ-binding	let-binding
types	fixed α	arbitrary β
terms	**fix** x	**define** $c \equiv b\ x$
theorems	**assume** $A\ x$	**note** $c = \langle A\ x \vdash B\ x \rangle$
data		**declaration** $\ll d \gg$

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

Note:

- System transforms data declaration functions, *not* data
- User receives morphism on types/terms/theorems, and applies it to his aggregated data

Examples

See `Slides3/Ex2.thy`