The Isabelle/Isar framework as a "logical operating system"

Makarius Wenzel
TU München

July 2009

1. Isabelle/Pure framework
2. Isabelle/Isar language environment
3. Some system infrastructure

Introduction

Isabelle characteristics

- Fully-foundational mathematics:
  formal checking of definition—statement—proof
- Interactive proof development
- Explicit deduction with "correctness by construction"
  the "LCF approach" (Robin Milner 1979)
- Integration with SML
- Managed transactions and parallel processing
- Forthcoming integration into Scala/JVM

Isabelle/Pure

Pure logical framework (Paulson 1989)

Formal system: 3 levels of λ-calculus

$\alpha \Rightarrow \beta$	terms depending on terms
$\bigwedge x.\ B\ x$	proofs depending on terms
$A \Longrightarrow B$	proofs depending on proofs

Rule composition: via higher-order unification

$resolution$: mixed forward-back chaining

$assumption$: closing branches

Note: arbitrary nesting of rules, e.g. mathematical induction

$$\frac{P\ 0 \qquad \begin{array}{c}[n][P\ n] \\ \vdots \\ P\ (Suc\ n)\end{array}}{P\ n} \qquad\qquad P\ 0 \Longrightarrow (\bigwedge n.\ P\ n \Longrightarrow P\ (Suc\ n)) \Longrightarrow P\ n$$

Example

Proof script:

theorem $example\colon A \wedge B \longrightarrow B \wedge A$
  apply $(rule\ impI)$
  apply $(erule\ conjE)$
  apply $(rule\ conjI)$
   apply $assumption$
  apply $assumption$
  done

Proof term:

$$impI \cdot A \wedge B \cdot B \wedge A \bullet$$
$$(\boldsymbol{\lambda} H\colon A \wedge B.$$
$$conjE \cdot A \cdot B \cdot B \wedge A \bullet H \bullet (\boldsymbol{\lambda}(H\colon A)\ Ha\colon B.\ conjI \cdot B \cdot A \bullet Ha \bullet H))$$

Example — ML version

ML ⟪ `val goal = Goal.init` $@\{cprop\ A \wedge B \longrightarrow B \wedge A\}$ ⟫

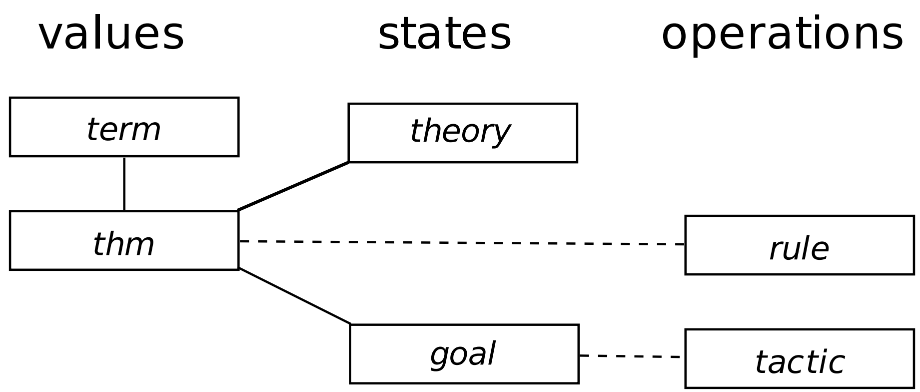
ML ⟪

```
val results = goal |>
 (rtac @{thm impI} 1 THEN
  etac @{thm conjE} 1 THEN
  rtac @{thm conjI} 1 THEN
  atac 1 THEN
  atac 1)
```
⟫

ML ⟪

```
val thm =
  (case Seq.pull results of
    NONE => error "Proof failed"
  | SOME (result, _) => Goal.finish result)
```
⟫

Isabelle/Pure architecture ($\approx$ LCF)



- $term$: simple-typed λ-calculus (modulo $\alpha\beta\eta$ conversion)
- $theory$: monotonic environment, formal certificates
- thm: derivable propositions, relative to theory
- $rule$: forward inferences $thm \to thm$ or $thm \to thm \to thm$ etc.
- $goal$, $tactic$: Prolog-style backward reasoning

Isabelle/Isar

Example

theorem $A \wedge B \longrightarrow B \wedge A$
proof
  assume $A \wedge B$
  show $B \wedge A$
  proof
    show B **using** $\langle A \wedge B \rangle$ **..**
    show A **using** $\langle A \wedge B \rangle$ **..**
  qed
qed

theorem $A \wedge B \longrightarrow B \wedge A$
proof
  assume $A \wedge B$
  then obtain B **and** A **..**
  then show $B \wedge A$ **..**
qed

Isar proof language (Wenzel 1999)

Main idea: Pure rules turned into proof schemes

from $facts_1$ **have** $props$ **using** $facts_2$
proof $(rule)$
    $body$
qed $(finish)$

Solving sub-problems: within $body$

fix $vars$
assume $props$
show $props$ $\langle proof \rangle$

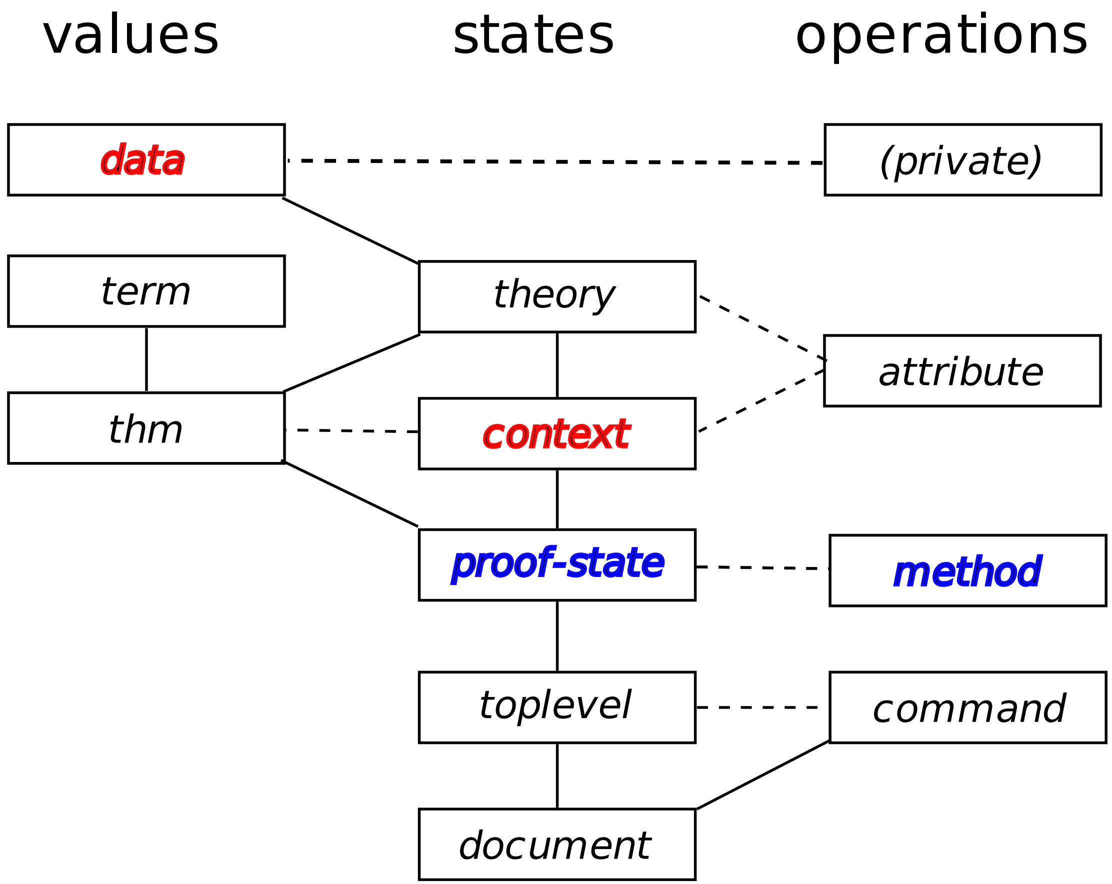
Abbreviations: for example

by $(method_1)$ $(method_2)$	$\equiv$	**proof** $(method_1)$ **qed** $(method_2)$
..	$\equiv$	**by** $(rule)$
then	$\equiv$	**from** $this$

Example

```
theorem Knaster-Tarski:
  fixes f :: 'a::complete-lattice ⇒ 'a
  assumes mono f shows ∃ a. f a = a
proof
  let ?H = {u. f u ≤ u}  let ?a = ⨅ ?H
  show f ?a = ?a
  proof (rule order-antisym)
    show f ?a ≤ ?a
    proof (rule Inf-greatest)
      fix x assume x ∈ ?H then have ?a ≤ x by (rule Inf-lower)
      with ‹mono f› have f ?a ≤ f x ..
      also from ‹x ∈ ?H› have ... ≤ x .. finally show f ?a ≤ x .
    qed
    show ?a ≤ f ?a
    proof (rule Inf-lower)
      from ‹mono f› and ‹f ?a ≤ ?a› have f (f ?a) ≤ f ?a ..
      then show f ?a ∈ ?H ..
    qed
  qed
qed
```

Main Isar concepts



Derived language elements (Pure + HOL library)

Definitions:

- simple definitions: **definition**, **abbreviation**
- (co)inductive sets and predicates: **inductive**, **coinductive**
- recusive functions: **primrec**, **function**
- datatypes: **datatype**, **record**

Statements:

- introductions: **theorem fixes** x **assumes** $A\ x$ **shows** $B\ x$
- eliminations: **theorem obtains** x **where** $B\ x$

Proofs:

- generalized elimination: **obtain** x **where** $B\ x$
- calculational reasoning: **also**, **finally**, **moreover**, **ultimately**
- structured induction: **case**, $induct$ method

Example: derived elements

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

theorem
  fixes rel **and** $x\ z$
  assumes $path\ rel\ x\ z$
  shows $P\ x\ z$ **using** $assms$
proof $induct$
  case $(base\ x)$
  show $P\ x\ x$ $\langle proof \rangle$
next
  case $(step\ x\ y\ z)$
  note $\langle rel\ x\ y \rangle$ **and** $\langle path\ rel\ y\ z \rangle$ **and** $\langle P\ y\ z \rangle$
  then show $P\ x\ z$ $\langle proof \rangle$
qed

Some system infrastructure

Local theory specifications

Motivation:

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

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

Example

locale $relation$ $=$
  fixes $rel :: {}'a \Rightarrow {}'a \Rightarrow bool$
  assumes sym: $rel\ x\ y \Longrightarrow rel\ y\ x$
begin

inductive $path$ **where**
  $base$: $path\ rel\ x\ x$
$|$ $step$: $rel\ x\ y \Longrightarrow path\ rel\ y\ z \Longrightarrow path\ rel\ x\ z$

theorem
  fixes $x\ z$
  assumes $path\ rel\ x\ z$
  shows $path\ rel\ z\ x$
  $\langle proof \rangle$

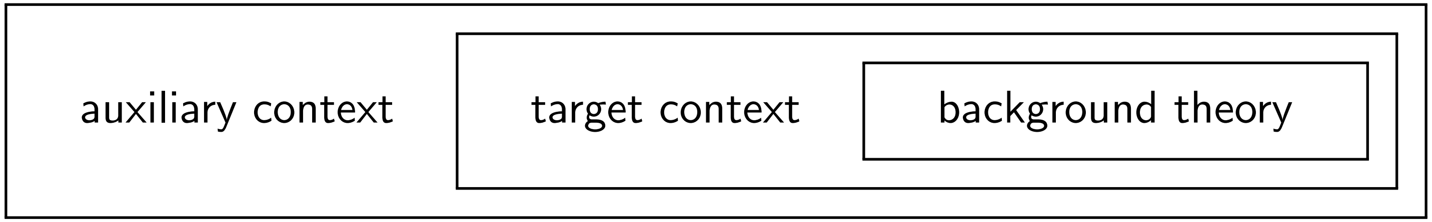
end

Local theory infrastructure

Context-dependent specifications:

	λ-binding	*let*-binding
types	fixed α	arbitrary β
terms	**fix** $x :: \tau$	**define** $c \equiv t[x]$
theorems	**assume** a: A	**note** $b = \langle B[x] \rangle$

Local theory infrastructure:



- target mechanism moves specifications between contexts
- target can modify type-discipline ("user-space type system")

Parallel proof checking

Proof document structure:

1. definitions and statements
   - fast checking (1%)
   - sequential dependency (worst case)
2. proofs
   - slow checking (99%)
   - irrelevant $\longrightarrow$ independent $\longrightarrow$ parallel checking (best case)

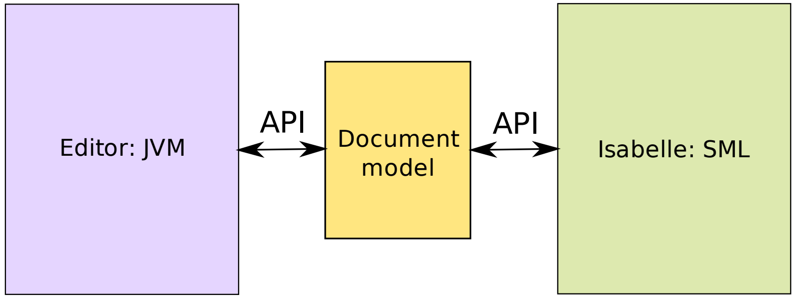
lemma a: A $\langle \textit{proof} \rangle$
lemma b: B $\langle \textit{proof} \rangle$
lemma c: C $\langle \textit{proof} \rangle$

Practical speedup: max. 3.2 on 4 cores

Future impact: asynchronous interaction model

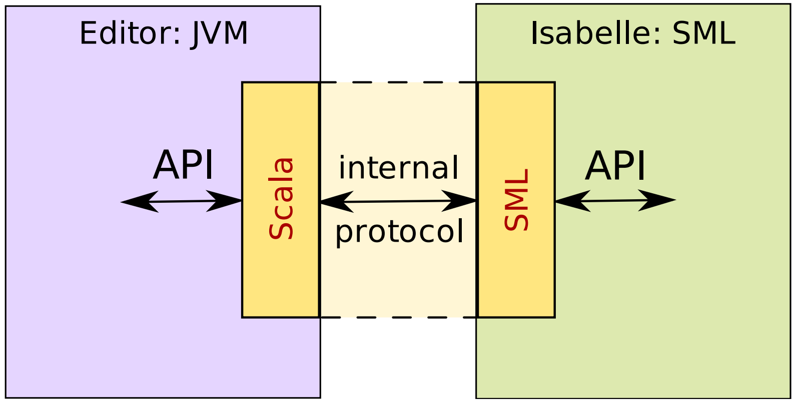
Scala/JVM system integration

Conceptual view:



- bridge SML — Scala/JVM
- support GUIs, IDEs, application servers etc.
- advanced document model: parallel checking, asynchronous interaction

Implementation view:



- public API, private protocol
- integral part of future Isabelle distributions

Conclusion

Isabelle 1989: Pure logical framework

Isabelle 2009: general system framework for logic-based applications (example: Isabelle/HOL)

After 20 years still a lot of potential for further development . . .