Intelligible semi-automated reasoning in December 2015

Makarius Wenzel
http://sketis.net

Abstract

Isabelle was introduced in 1989 (by L. Paulson) as a generic logical framework for higher-order natural deduction. Intelligible semi-automated reasoning (Isar) was introduced in 1999 (by M. Wenzel) as a structured proof language for human-readable formal proof documents. Today, in December 2015, we see large applications of Isabelle/Isar in the Isabelle/HOL object-logic, e.g. in the Archive of Formal Proofs (http://afp.sf.net) with more than 240 entries.

After so many years, development of Isar is still not finished. Recent refinements of old concepts and additions of new concepts include: structured rule statements (Eigen-contexts), multi-branch elimination (case-splitting), structured backwards refinement. The new Eisbach language (by D. Matichuk et al) allows to define complex proof methods in their usual syntax, instead of traditional Isabelle/ML. Sledgehammer (by J. Blanchette et al) allows to generate semi-intelligible Isar proofs from machine-generated proofs (via external ATPs and SMTs).

The ultimate aim of Isabelle/Isar is to turn the results of formal proof production into mathematical documents with nice type-setting. Document source was mainly written in LaTeX in the past, but is now moving towards Markdown, with specific GUI support in the Prover IDE (Isabelle/jEdit).

Introduction

What is Isabelle?

Hanabusa Itchō: “Blind monks examining an elephant”

History: LCF Prover Family

LCF (since 1979)
- Edinburgh LCF
- Cambridge LCF

HOL (since 1988)
- ProofPower
- HOL4
- HOL-Light
- HOL Zero

Coq (since 1985)
- Coq 8.4pl6 (April 2015)

Isabelle (since 1986)
- Isabelle2015 (May 2015)

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

Structured Proofs

Natural Deduction: (Gentzen, 1935)

$$\frac{A \longrightarrow B \quad A}{B} \qquad \frac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \longrightarrow B}$$

Isabelle/Pure rules: (Paulson 1989)

$$(A \longrightarrow B) \Longrightarrow A \Longrightarrow B \qquad (A \Longrightarrow B) \Longrightarrow A \longrightarrow B$$

Isabelle/Isar proofs: (Wenzel 1999)

assume $A \longrightarrow B$
also have A $\langle proof \rangle$
finally have B **.**

have $A \longrightarrow B$
proof
  assume A
  then show B $\langle proof \rangle$
qed

Isar principles

Ultimate goal: human-readable formal proof documents

Notions of proof:

Primitive proof term (internal inferences)

Primary proof text (concrete syntax and structure)

Presentation proof document (pretty-printing and type-setting)

Isar language characteristics:

- interpreted language of proof expressions:
  context elements, flow of facts towards goals
- built-in reduction to Pure rule composition
- add-on proof methods (defined in library)

$\longrightarrow$ bottom-up emergence from existing Isabelle concepts

Example proof patterns: induction and calculation

theorem fixes $n :: nat$ **shows** $P\ n$
proof $(induct\ n)$
  case 0
  show $P\ 0\ \ \langle proof\rangle$
next
  case $(Suc\ n)$
  then show $P\ (Suc\ n)\ \ \langle proof\rangle$
qed

notepad
begin
  have $a = b\ \ \langle proof\rangle$
  also have $\ldots = c\ \ \langle proof\rangle$
  also have $\ldots = d\ \ \langle proof\rangle$
  finally have $a = d$ **.**
end

Example proof: induction $\times$ calculation

theorem
  fixes $n :: nat$
  shows $(\sum i{=}0..n.\ i) = n * (n + 1)\ div\ 2$
proof $(induct\ n)$
  case 0
  have $(\sum i{=}0..0.\ i) = (0{::}nat)$ **by** $simp$
  also have $\ldots = 0 * (0 + 1)\ div\ 2$ **by** $simp$
  finally show $?case$ **.**
next
  case $(Suc\ n)$
  have $(\sum i{=}0..Suc\ n.\ i) = (\sum i{=}0..n.\ i) + (n + 1)$ **by** $simp$
  also have $\ldots = n * (n + 1)\ div\ 2 + (n + 1)$ **by** $(simp\ add{:}\ Suc.hyps)$
  also have $\ldots = (n * (n + 1) + 2 * (n + 1))\ div\ 2$ **by** $simp$
  also have $\ldots = (Suc\ n * (Suc\ n + 1))\ div\ 2$ **by** $simp$
  finally show $?case$ **.**
qed

Isabelle/Pure: declarative rules

Theory and proof context

Logical judgement:

$$\boxed{\Theta, \Gamma \vdash \varphi}$$

- background theory Θ
  (polymorphic types, constants, axioms; global data)
- proof context Γ (fixed variables, assumptions; local data)

Operations on theories:

- merge and extend: $\Theta_3 = \Theta_1 \cup \Theta_2 + \tau + c :: \tau + c \equiv t$
- symbolic sub-theory relation: $\Theta_1 \subseteq \Theta_2$
- transfer of results: if $\Theta_1 \subseteq \Theta_2$ and $\Theta_1, \Gamma \vdash \varphi$ then $\Theta_2, \Gamma \vdash \varphi$

Primitive inferences

Syntax (types and terms):

$fun :: (type, type)type$ — function space $'a \Rightarrow 'b$

$all :: ('a \Rightarrow prop) \Rightarrow prop$ — universal quantification $\bigwedge x{::}'a.\ B\ x$

$imp :: prop \Rightarrow prop \Rightarrow prop$ — implication $A \Longrightarrow B$

Derivations (theorems): implicit theory Θ

$$\frac{A \in \Theta}{\vdash A}\,(axiom) \qquad \frac{}{A \vdash A}\,(assume)$$

$$\frac{\Gamma \vdash B[x] \quad x \notin \Gamma}{\Gamma \vdash \bigwedge x.\ B[x]}\,(\bigwedge\text{-}intro) \qquad \frac{\Gamma \vdash \bigwedge x.\ B[x]}{\Gamma \vdash B[a]}\,(\bigwedge\text{-}elim)$$

$$\frac{\Gamma \vdash B}{\Gamma - A \vdash A \Longrightarrow B}\,(\Longrightarrow\text{-}intro) \qquad \frac{\Gamma_1 \vdash A \Longrightarrow B \quad \Gamma_2 \vdash A}{\Gamma_1 \cup \Gamma_2 \vdash B}\,(\Longrightarrow\text{-}elim)$$

Object-logic rules

Main principles: (Paulson 1989)

- Pure syntax is notation for rules in Natural Deduction

  logical entailment: $A_1 \Longrightarrow \ldots A_n \Longrightarrow B$ represents $\dfrac{A_1 \ \ldots \ A_n}{B}$
  local parameter: $\bigwedge x.\ B\ x$ represents "eigen-variable" condition

- Declarative rule composition via:
  - back-chaining
  - lifting into subgoal context
  - higher-order unification (G. Huet, D. Miller).

$\longrightarrow$ Isabelle/Pure reasoning similar to λ-Prolog

Inferences for rule composition

$$\frac{\overline{A} \Longrightarrow B \quad B' \Longrightarrow C \quad B\,\theta = B'\theta}{\overline{A}\,\theta \Longrightarrow C\,\theta}\ (compose)$$

$$\frac{\overline{A} \Longrightarrow B}{(\overline{H} \Longrightarrow \overline{A}) \Longrightarrow (\overline{H} \Longrightarrow B)}\ (\Longrightarrow\text{-}lift)$$

$$\frac{\overline{A}\ \overline{a} \Longrightarrow B\ \overline{a}}{(\bigwedge \overline{x}.\ \overline{A}\ (\overline{a}\ \overline{x})) \Longrightarrow (\bigwedge \overline{x}.\ B\ (\overline{a}\ \overline{x}))}\ (\bigwedge\text{-}lift)$$

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

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

Isabelle/Isar: proof context

Notepad for logical entities

— background theory Θ
notepad
begin
— proof context Γ

Terms:

let $?f = \lambda x.\ x$ — term binding (abbreviation)
let $\_ + ?b = ?f\ a + b$ — pattern matching
let $?g = ?f\ ?f$ — Hindler-Milner polymorphism

Facts:

note $rules = sym\ refl\ trans$ — collective facts
note $a = rules(2)$ — selection
note $b = this$ — implicit result $this$

end

Context elements: rules from text

Universal context: **fix** and **assume**

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

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

Existential context: **obtain**

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

Structured rule statements (within theory)

Long theorem statement:

- context elements: **fixes** a_1 **and** a_2 $\ldots$ **assumes** A_1 **and** A_2 $\ldots$
- conclusion: **shows** B
- indirect conclusion: **obtains** a **where** $B\ a$
  "may assume that $B\ a$ holds for some abstract a"

theorem $exists\_intro$:
  fixes $B :: {'a} \Rightarrow bool$ **and** $a :: {'a}$
  assumes $B\ a$
  shows $\exists\, x.\ B\ x$

theorem $exists\_elim$:
  fixes $B :: {'a} \Rightarrow bool$
  assumes $\exists\, x.\ B\ x$
  obtains $a :: {'a}$ **where** $B\ a$

Structured rule statements (within proof)

Horn-clause post-fix notation: “Eigen-context”

- premises: **have** B **if** A_1 **and** A_2 $\ldots$
- parameters: **have** B **for** a_1 **and** a_2 $\ldots$

Examples:

- conjunction introduction:
  have $A \wedge B$ **if** A **and** B
- existential introduction:
  have $\exists x.\ B\ x$ **if** $B\ a$ **for** a
- disjunction elimination:
  from $\langle A \vee B \rangle$ **have** C **if** $A \Longrightarrow C$ **and** $B \Longrightarrow C$ **for** C
- existential elimination:
  from $\langle \exists x.\ B\ x \rangle$ **have** C **if** $\bigwedge x.\ B\ x \Longrightarrow C$ **for** C

Elimination statements (within proof)

consider $\overline{x}$ **where** $\overline{A}\ \overline{x}$ $\mid$ $\overline{y}$ **where** $\overline{B}\ \overline{y}$ $\mid$ $\ldots$ $\equiv$

have $thesis$

if $\bigwedge\overline{x}.\ \overline{A}\ \overline{x} \Longrightarrow thesis$

and $\bigwedge\overline{y}.\ \overline{B}\ \overline{y} \Longrightarrow thesis$

for $thesis$

Examples:

- existential elimination:
  from $\langle\exists\, x.\ B\ x\rangle$ **consider** x **where** $B\ x$
- conjunction elimination:
  from $\langle A \wedge B\rangle$ **consider** A **and** B
- disjunction elimination:
  from $\langle A \vee B\rangle$ **consider** $A \mid B$

Examples:

notepad
begin
  assume $A_1 \vee A_2 \vee A_3 \vee A_4$
  then consider $A_1 \mid A_2 \mid A_3 \mid A_4$
    by $blast$
next
  assume $A_1 \wedge B_1 \vee A_2 \wedge B_2 \wedge C_2$
  then consider A_1 **and** $B_1 \mid A_2$ **and** B_2 **and** C_2
    by $blast$
next
  assume $(\exists\, x\ y.\ A\ x \wedge B\ y) \vee (\exists\, z.\ C\ z)$
  then consider $a\ b$ **where** $A\ a$ **and** $B\ b \mid z$ **where** $C\ z$
    by $blast$
end

Isabelle/Isar: structured proofs

Proof decomposition

Structured proof outline:

from $facts_1$ **have** $props$ **using** $facts_2$
proof $(initial\_method)$
  $body$
qed $(terminal\_method)$

Solving sub-problems: within $body$

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

or:

show $props$ **if** $props$ **for** $vars$ $\langle proof \rangle$

Proof decomposition as Pure inference

have $\bigwedge\overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\,\overline{x}$
proof $-$
  fix $\overline{a}$
  assume $\overline{G}\ \overline{a}$
  show $\overline{B}\ \overline{a}$ $\langle proof \rangle$
qed

$$\frac{\begin{array}{rll} subgoal\!: & (\bigwedge\overline{x}.\ \overline{H}\ \overline{x} \Longrightarrow B'\,\overline{x}) \Longrightarrow C & \\ subproof\!: & \overline{G}\ \overline{a} \Longrightarrow B\ \overline{a} & \text{for schematic } \overline{a} \\ concl\ unifier\!: & (\lambda\overline{x}.\ B\ (\overline{a}\ \overline{x}))\,\theta = B'\theta & \\ assm\ unifiers\!: & (\lambda\overline{x}.\ G_j\ (\overline{a}\ \overline{x}))\,\theta = H_i\,\theta & \text{for each } G_j \text{ some } H_i \end{array}}{C\,\theta}$$

Example

notepad
begin
  fix $A\ B :: bool$
  have $A \wedge B \longrightarrow B \wedge A$
  proof $(rule\ impI)$
    assume $*$: $A \wedge B$
    show $B \wedge A$
    proof $(rule\ conjI)$
      show B **proof** $(rule\ conjunct2\ [OF\ *])$ **qed**
      show A **proof** $(rule\ conjunct1\ [OF\ *])$ **qed**
    qed
  qed
end

Terminal proofs

Canonical double-step proofs:

have $prop$
proof $(initial\_method)$
qed $(terminal\_method)$

or: **have** $prop$ **by** $(initial\_method)$ $(terminal\_method)$

Single-step proofs:

by $fact$
by $this$ $\equiv$ **.**
by $rule$ $\equiv$ **..**

Skipped proofs:

sorry

Using facts

Goal statement with facts:

from $facts_1$ **have** $prop$ **using** $facts_2$
proof $(initial\_method)$
  $body$
qed $(terminal\_method)$

- $initial\_method$ sees $facts_1$ $facts_2$ as primary argument
- actual use of facts depends on proof method, e.g. $rule$, $cases$, $induct$, $auto$

Abbreviations and synonyms:

from $this$	$\equiv$	**then**
from a	$\equiv$	**note** a **then**
with a	$\equiv$	**note** a **and** $this$ **then**

Mixed forward-backward reasoning (1)

notepad
begin
  assume r: $A_1 \Longrightarrow A_2 \Longrightarrow B_1 \Longrightarrow B_2 \Longrightarrow B_3 \Longrightarrow C$

  have A_1 **and** A_2 $\langle proof \rangle$
  then have C
  proof $(rule\ r)$
    show B_1 $\langle proof \rangle$
    show B_2 $\langle proof \rangle$
    show B_3 $\langle proof \rangle$
  qed
end

Mixed forward-backward reasoning (2)

notepad
begin
  assume r: $A \Longrightarrow (\bigwedge x.\ H_1\ x \Longrightarrow B_1\ x) \Longrightarrow (\bigwedge y.\ H_2\ y \Longrightarrow B_2\ y) \Longrightarrow C$

  have A $\langle proof \rangle$
  then have C
  proof $(rule\ r)$
    show $B_1\ x$ **if** $H_1\ x$ **for** x $\langle proof \rangle$
    show $B_2\ y$ **if** $H_2\ y$ **for** y $\langle proof \rangle$
  qed
end

Calculational reasoning

also$_0$	$\equiv$	**note** *calculation* $=$ *this*
also$_{n+1}$	$\equiv$	**note** *calculation* $=$ *trans* [*OF calculation this*]
finally	$\equiv$	**also from** *calculation*
moreover	$\equiv$	**note** *calculation* $=$ *calculation this*
ultimately	$\equiv$	**moreover from** *calculation*

Example:

notepad
begin
  have $a = b$ $\langle proof \rangle$
  also have $\ldots = c$ $\langle proof \rangle$
  also have $\ldots = d$ $\langle proof \rangle$
  finally have $a = d$ **.**
end

notepad
begin
  have A $\langle proof \rangle$
  moreover have B $\langle proof \rangle$
  moreover have C $\langle proof \rangle$
  ultimately have A **and** B **and** C **.**
end

Note:

term "$\ldots$" abbreviates the argument of the last statement

Induction

using *facts*
proof $(induct\ insts\ arbitrary{:}\ vars\ rule{:}\ fact)$

Example:

notepad
begin
  fix $n :: nat$ **and** $x :: {}'a$ **have** $P\ n\ x$
  proof $(induct\ n\ arbitrary{:}\ x)$
    case 0
    show $P\ 0\ x\ \ \langle proof\rangle$
  next
    case $(Suc\ n)$
    from $\langle P\ n\ a\rangle$ **show** $P\ (Suc\ n)\ x\ \ \langle proof\rangle$
  qed
end

Generalized elimination

obtain $\overline{x}$ **where** $\overline{B}\ \overline{x}$ $\langle proof \rangle$ $\equiv$

have $reduction$: $\bigwedge thesis.\ (\bigwedge \overline{x}.\ \overline{B}\ \overline{x} \Longrightarrow thesis) \Longrightarrow thesis$ $\langle proof \rangle$
fix $\overline{x}$ **assume** $\ll eliminate\ reduction \gg\ \overline{B}\ \overline{x}$

$$\frac{\begin{array}{l}\Gamma \vdash \bigwedge thesis.\ (\bigwedge \overline{x}.\ \overline{B}\ \overline{x} \Longrightarrow thesis) \Longrightarrow thesis \\ \Gamma \cup \overline{B}\ \overline{x} \vdash C\end{array}}{\Gamma \vdash C}\ (eliminate)$$

Example:

notepad
begin
  assume $\exists\, x.\ B\ x$
  then obtain x **where** $B\ x$ ..
end

notepad
begin
  assume $A \wedge B$
  then obtain A **and** B ..
end

Structured backwards refinement

$\langle goal \rangle$
subgoal premises $prems$ **for** $x_1\ x_2\ \ldots$
  $\langle proof \rangle$

Example: structured proof scripts

$\langle goal \rangle$
subgoal by $method_1$
subgoal by $method_2$
done

$\langle goal \rangle$
subgoal premises $prems$ **for** $x\ y$
  using $prems$ $\langle proof_1 \rangle$
subgoal premises $prems$ **for** $u\ v\ w$
  using $prems$ $\langle proof_2 \rangle$
done

Isabelle/HOL: theory specifications

Types

- augmented version of Simple Theory of Types (Church 1940)
- schematic polymorphism (weaker than ML let-polymorphism)
- basic types: $bool$, nat, $'a \Rightarrow {}'b$ (full function space)

Type specifications:

- **typedef** semantic subtype of existing type
- **quotient_type** wrt. equivalence relation or PER
- **record** extensible records (glorified tuples)
- **datatype** and **codatatype** (Bounded Natural Functors)

  datatype $'a\ list = Nil \mid Cons\ (hd\colon {}'a)\ (tl\colon {}'a\ list)$
  codatatype $'a\ stream = Stream\ (HD\colon {}'a)\ (TL\colon {}'a\ stream)$

Functions and constants

- **abbreviation**: abstract syntax definitions

  abbreviation $(input)$ $double :: nat \Rightarrow nat$
  where $double\ n \equiv 2 * n$

- **definition**: simple non-recursive definitions

  definition $square :: nat \Rightarrow nat$
  where $square\ n = n * n$

- **fun** and **function** / **termination**: general recursion with implicit or explicit termination proof

  fun $fibonacci :: nat \Rightarrow nat$
  where
  $fibonacci\ 0 = 0$
  $\mid fibonacci\ (Suc\ 0) = 1$
  $\mid fibonacci\ (Suc\ (Suc\ n)) = fibonacci\ n + fibonacci\ (Suc\ n)$

Inductive predicates and sets

- **inductive** and **coinductive**: Knaster-Tarski fixed-points over predicates or sets

  inductive_set $star$ ($\_\star$ [100] 100) **for** R :: $('a \times 'a)$ set
  where
  $base$: $(x,\ x) \in R\star$
  | $step$: $(x,\ y) \in R \Longrightarrow (y,\ z) \in R\star \Longrightarrow (x,\ z) \in R\star$

This means $R\star$ is the least relation (set of pairs) that is closed under the introduction rules above. The following induction rule is provided:

$(x_1,\ x_2) \in R\star \Longrightarrow$
$(\bigwedge x.\ P\ x\ x) \Longrightarrow$
$(\bigwedge x\ y\ z.\ (x,\ y) \in R \Longrightarrow (y,\ z) \in R\star \Longrightarrow P\ y\ z \Longrightarrow P\ x\ z) \Longrightarrow P\ x_1\ x_2$

Type classes

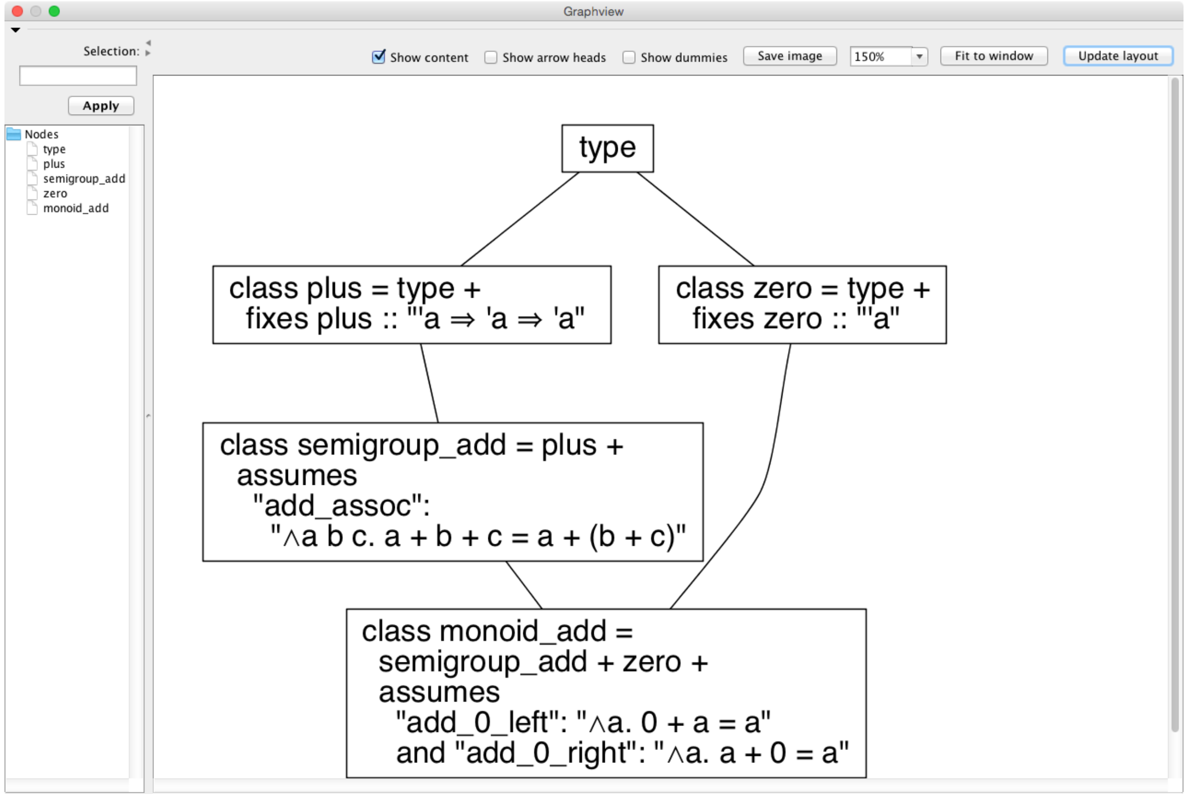
- Predicate over constant signature with single type-variable
- Integrated into type-system: order-sorted algebra of constraints
- Class intersections are called sorts
- Class inclusion hierarchy: by definition or proof
- Class instantiation by concrete types

class $zero$ = **fixes** $zero :: {}'a \ \ (\mathbf{0})$
class one = **fixes** $one :: {}'a \ \ (\mathbf{1})$
class $times$ = **fixes** $times :: {}'a \Rightarrow {}'a \Rightarrow {}'a$ $($**infixl** $* \ 70)$

class $group = times + one + inverse +$
  assumes $group\_assoc$: $(x * y) * z = x * (y * z)$
    and $group\_left\_one$: $\mathbf{1} * x = x$
    and $group\_left\_inverse$: $inverse\ x * x = \mathbf{1}$

Example: class hierarchy

class_deps *type monoid_add*



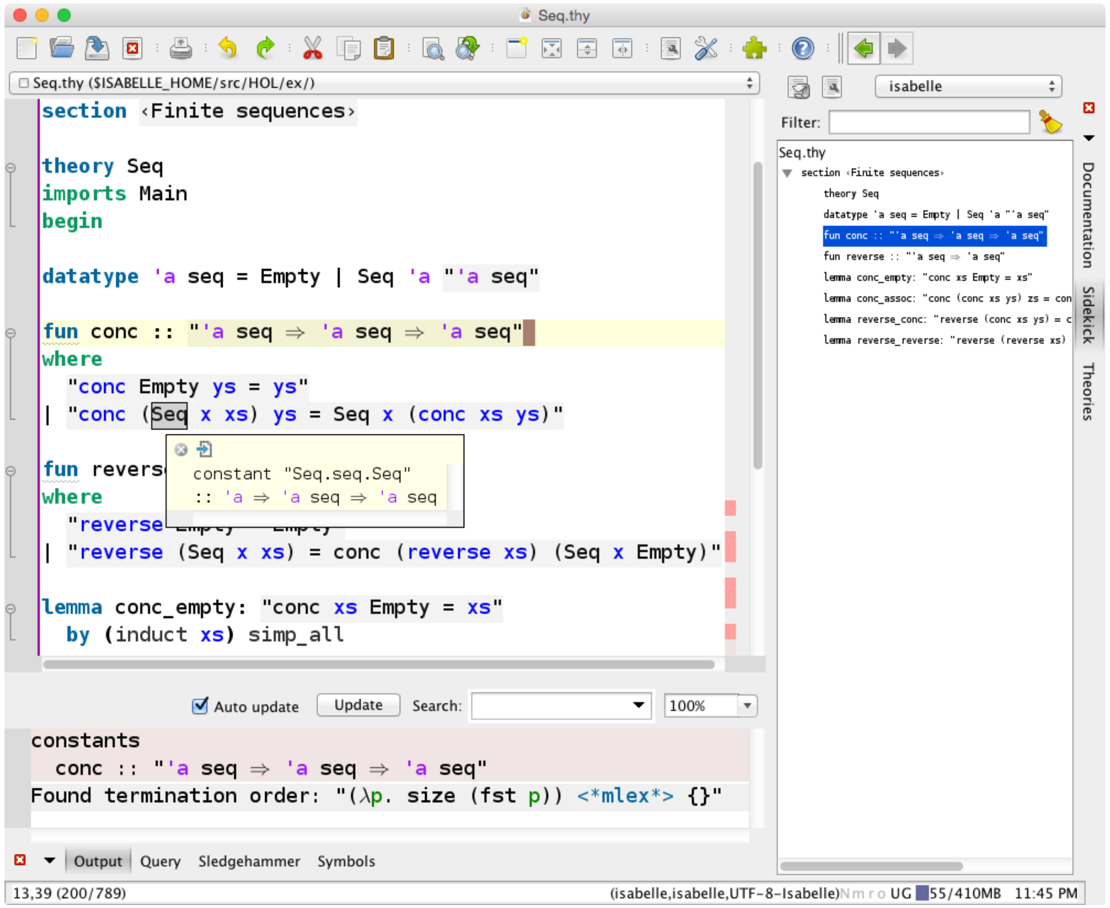
Isabelle/HOL proof methods

- rule: generic Natural Deduction (with HO unification)
- cases: elimination, syntactic representation of datatypes, inversion of inductive sets and predicates
- induct and coinduct: induction and coinduction of types, sets, predicates
- simp: equational reasoning by the Simplifier (HO rewriting), with possibilities for add-on tools
- fast and blast: classical reasoning (tableau)
- auto and force: combined simplification and classical reasoning
- arith, presburger: specific theories
- smt: Z3 with proof reconstruction

Additional tool support

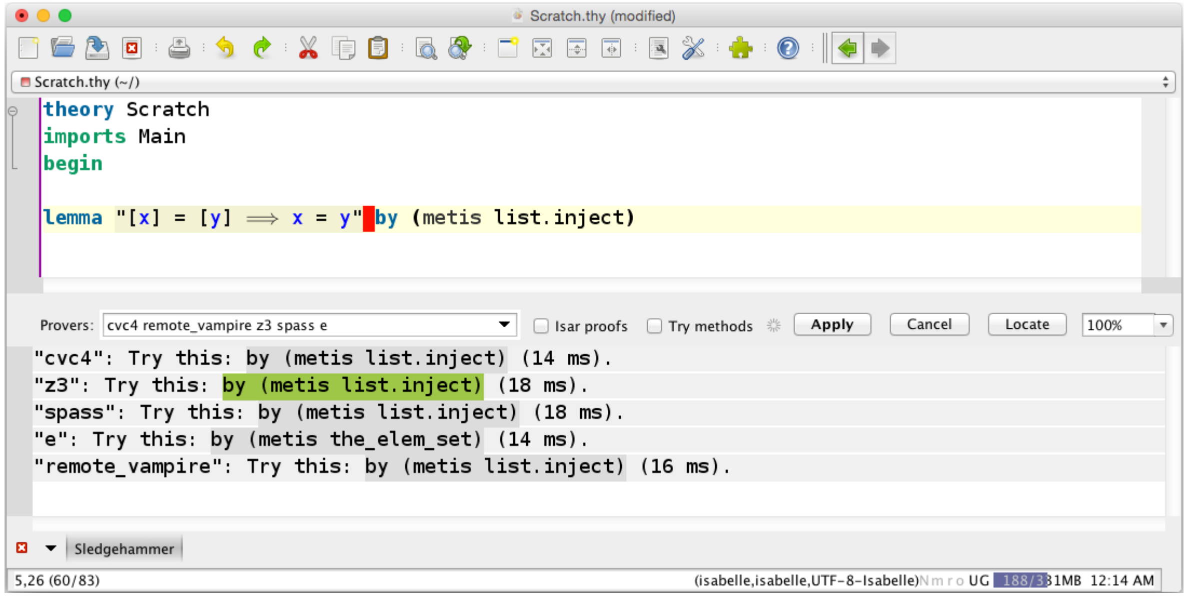
Isabelle/jEdit Prover IDE (2015)

- asynchronous interaction
- continuous checking
- parallel processing



Sledgehammer (J. Blanchette et al)

- heavy external ATPs / SMTs for proof search
- light internal ATP (Metis) for proof reconstruction



Example: semi-intelligible automated reasoning

$\boxed{\textbf{sledgehammer}\ [\mathit{isar\_proofs}]}$

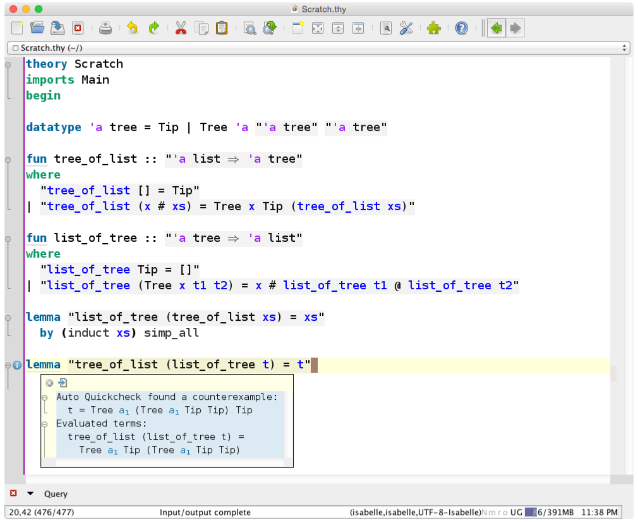
- proof redirection: classical contradiction of negated conclusion
  $\leadsto$ proof of conclusion
- treatment of Skolemization vs. Isar **obtain** x **where** $B\ x$
- post-processing for legibility and efficiency of proof-checking

$\longrightarrow$ some high-level tracing of ATPs

$\longrightarrow$ truly intelligible proofs require manual rewriting

Automated disprovers — counter examples

- **nitpick** based on relational model finder
- **quickcheck** based on random functional evaluation



Eisbach: high-level proof procedures (D. Matichuk et al)

Proof method definitions:

- abstraction over terms and facts:
  method m **for** $x\ y$ **uses** $a\ b = method\_body[m,\ x,\ y,\ a,\ b]$
- abstraction over facts, with declaration in the context:
  method m **declares** $simp = method\_body[m]$
- abstraction over other methods:
  method m **methods** $m_1\ m_2 = method\_body[m,\ m_1,\ m_2]$

Method $match$:

- goal introspection with pattern matching
- subgoal focus (similar to **subgoal** command)
- control of backtracking

Document preparation

Structure markup

- section headings: **chapter**, **section**, **subsection**, etc.
- text blocks: **text**
- implicit lists (cf. Markdown): itemize, enumerate, description
- free-form LaTeX macros

Antiquotations

full form: `@{`*name* [*options*] *arguments* ...`}`

short form:

1. cartouche argument: `\<^`*name*`>`⟨*argument*⟩
2. no argument: `\<^`*name*`>`

Example: ≫*this document*≪

Isabelle tool implementation

Isabelle/ML

Characteristics:

- SML'97: strict functional programming + exceptions
- SML'90: interrupts
- Poly/ML (by David Matthews) as main implementation;
  SML/NJ now impractical
- parallel evaluation via futures (implemented via Poly/ML threads)
- immutable data managed within logical context

Notes:

- Isabelle/ML library useful for advanced functional programming
- Isabelle/jEdit serves as IDE for Isabelle/ML and Standard ML

Isabelle/ML IDE support

Isabelle/ML/PIDE:

- precise tokenization (syntax-highlighting etc.)
- spell-checking inside comments
- antiquotations
- text cartouches with formal position
- source-level debugger

Poly/ML IDE support

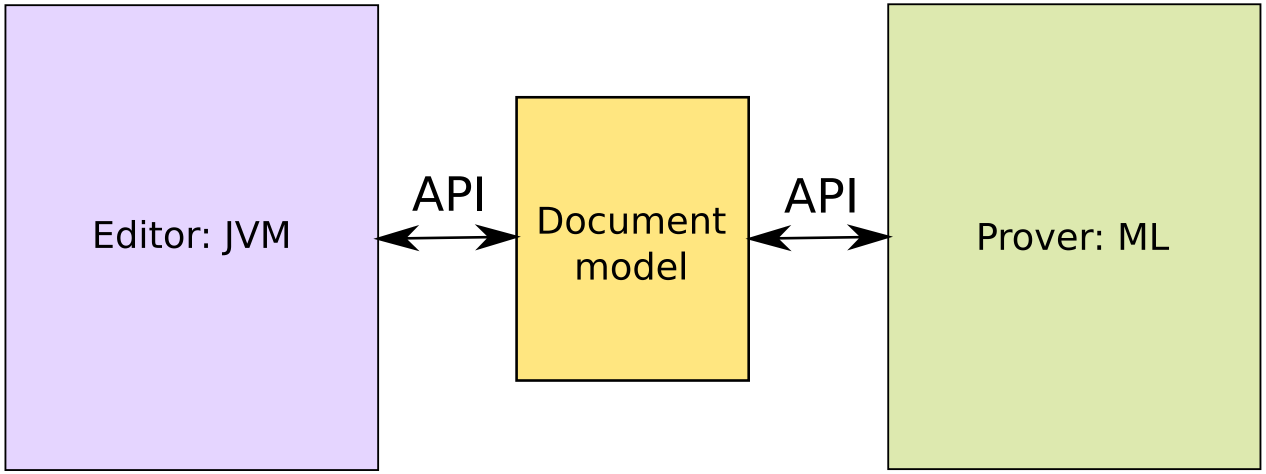
Poly/ML 5.5.2:

- inferred types for sub-expressions
- defining positions of referenced entities
- information about ML structures and open scopes
- pretty-printing of ML values with markup

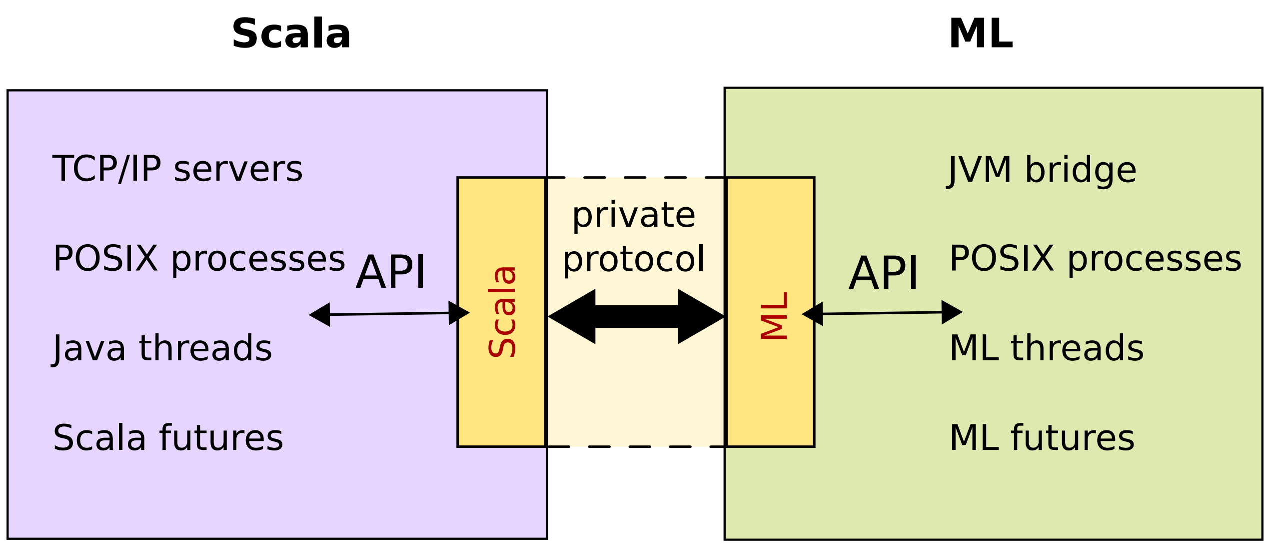
Poly/ML repository (November 2015):

- completion
- breakpoints for debugging

Isabelle/Scala/PIDE architecture: conceptual view



PIDE architecture: implementation view



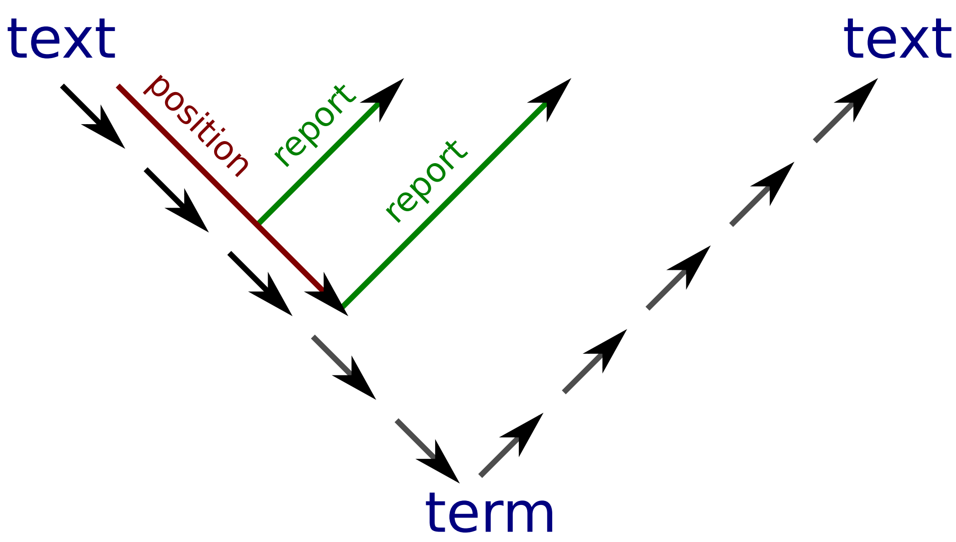
Design principles:

- private protocol for prover connectivity
  (asynchronous interaction, parallel evaluation)
- public Scala API
  (timeless, stateless, static typing)

Markup reports

Problem: round-trip through several sophisticated syntax layers

Solution: execution trace with markup reports



Conclusion

What is Isabelle?

The more it advances, . . .

. . . the less it is finished!