The Isar Proof Language after Two Decades

Makarius Wenzel, Augsburg
https://sketis.net

February 2020

Introduction

Isar language: Philosophy

Isar: *Intelligible semi-automated reasoning*

- human-readable and machine-checkable proofs by simple interpretation process
- extensible automation via proof methods (not "tactics")
- language to write proof texts (not "scripts", not "code")
- source text close to presented document via Isabelle symbols (not "Unicode")
- syntax stylistically inspired by SML'90, Haskell'98, Perl 4
- many add-on tools: notably Sledgehammer
- advanced document editor: Prover IDE (PIDE)
- Isar is the primary language of Isabelle, all others are embedded sublanguages (e.g. ML, type, term, document)

History of Structured Proof Languages

- Mizar (Trybulec $\approx$ 1973, published 1993)
- Mizar-MSE (Trybulec / Rudnicki 1982, published 1993)
- experimental "Mizar modes", e.g. for HOL-Light (Harrison 1996) (Wiedijk 2001)
- experimental "declarative modes", e.g. for Coq, Matita
- DECLARE language and system (Syme 1997/1998)
- Isabelle/Isar (Wenzel 1999–2001, 2015/2016)
- SSReflect proof language for Coq (Gonthier $\approx$ 2005)
- Lean (De Moura 2013, published 2015)

History of the Isar Proof Language (1)

1999: first usable version

- primary notion of proof document (not "proof script")
- secondary notion of proof method (not "tactic")
- subproofs with refinement: **proof** m_1 ... **qed** m_2
- nested proof refinement: **fix** x **assume** $A\ x$ **show** $B\ x$
- local facts: **note**, **have**
- chaining of facts: **then**, **with**, **from**, **using**

2000–2001: various refinements

- generalized elimination: **obtain**
- support for induction: **case** and $induct$ method
- calculations: **also**, **finally**, **moreover**, **ultimately**

History of the Isar Proof Language (2)

2006: minor reforms

- **unfolding**, **obtains**
- literal facts: $\langle prop \rangle$
- advanced $induct$ method

2015/2016: major renovations

- structured statements: **have** $B\ x$ **if** $A\ x$ **for** x
- elimination statements: **consider** x **where** $A\ x \mid B\ x \mid C\ x$
- refined $cases$ method
- structured goal refinement: **subgoal premises** $prems$ **for** $x \ldots$
- explicit facts for proof methods: $(use * \textbf{ in } m)$

History of keywords (1)

- `have`
  - origin: DECLARE
  - re-used in Isar, but independent goal statement
  - later re-used in SSReflect and Lean
- `hence` / `thus`
  - origin: Mizar (slightly odd English)
  - re-interpreted in HOL-Light Mizar mode and DECLARE
  - re-used in Isar (1999), but legacy since 2000
- `fix` / `assume` / `show`
  - origin: Isar (central concept)
  - note: `assume` in Mizar and others has different meaning

History of keywords (2)

- `obtain`
  - origin: Isar (2000)
  - re-used in Lean (phased out?)
- `sorry`
  - origin: Mizar-MSE (as output message)
  - re-interpreted in Isar
  - re-used in Lean

Examples

Elementary proofs in Isar (1)

```
lemma iff_contradiction:
  assumes *: ¬ A ⟷ A
  shows False
proof −
  have **: ¬ A
  proof
    assume A
    with * have ¬ A ..
    from this and ⟨A⟩ show False ..
  qed
  with * have A ..
  with ** show False ..
qed
```

Elementary proofs in Isar (2)

theorem — Cantor: $\nexists f :: 'a \Rightarrow 'a \Rightarrow bool.\ \forall A.\ \exists x.\ A = f\ x$
proof
  assume $\exists f :: 'a \Rightarrow 'a \Rightarrow bool.\ \forall A.\ \exists x.\ A = f\ x$
  then obtain $f :: 'a \Rightarrow 'a \Rightarrow bool$ **where** $*$: $\forall A.\ \exists x.\ A = f\ x$ **..**
  let $?D = \lambda x.\ \neg\ f\ x\ x$
  from $*$ **have** $\exists x.\ ?D = f\ x$ **..**
  then obtain a **where** $?D = f\ a$ **..**
  then have $?D\ a \longleftrightarrow f\ a\ a$ **by** $(rule\ arg\_cong)$
  then have $\neg\ f\ a\ a \longleftrightarrow f\ a\ a$ **.**
  then show $False$ **by** $(rule\ iff\_contradiction)$
qed

Automated proof tools in Isar (3)

theorem — Cantor: $\nexists f :: {}'a \Rightarrow {}'a\ set.\ \forall A.\ \exists x.\ A = f\ x$
proof
  assume $\exists f :: {}'a \Rightarrow {}'a\ set.\ \forall A.\ \exists x.\ A = f\ x$
  then obtain $f :: {}'a \Rightarrow {}'a\ set$ **where** $*$: $\forall A.\ \exists x.\ A = f\ x$ **..**
  let $?D = \{x.\ x \notin f\ x\}$
  from $*$ **obtain** a **where** $?D = f\ a$ **by** *blast*
  moreover have $a \in\ ?D \longleftrightarrow a \notin f\ a$ **by** *blast*
  ultimately show *False* **by** *blast*
qed

Notes:

- adequate tools: weaker automation is usually faster, more stable, more informative
- adequate facts: indicate upper bound of local facts for each step

Proof context without goal statement

notepad
begin
  fix $A\ B\ C :: bool$
  assume $A \wedge B$
  then obtain B **and** A **..**
  then have $B \wedge A$ **..**
end

Notes:

- implicit "thesis reduction" does not exist in Isar
- explicit goal refinement works via **show**
  usually in the context of **fix** / **assume**

Implicit context for local statements

notepad
begin
  have $P\ n$ **for** $n :: nat$
  proof $(induct\ n)$
    show $P\ 0\ \ \langle proof \rangle$
    show $P\ (Suc\ n)$ **if** $P\ n$ **for** $n\ \ \langle proof \rangle$
  qed
end

Proof via cases rule

notepad
begin
  fix $x\ y :: nat$
  consider $x = 0 \mid x = 1 \mid x \geq 2$ **and** $even\ x \mid x \geq 3$ **and** $odd\ x$
    by $(fastforce\ dest: antisym\ iff: not\_less\_eq\_eq)$
  then have C
  proof $cases$
    case 1 **show** $?thesis$ **using** $\langle x = 0\rangle$ $\langle proof \rangle$
  next
    case 2 **show** $?thesis$ **using** $\langle x = 1\rangle$ $\langle proof \rangle$
  next
    case 3 **show** $?thesis$ **using** $\langle x \geq 2\rangle$ **and** $\langle even\ x\rangle$ $\langle proof \rangle$
  next
    case 4 **show** $?thesis$ **using** $\langle x \geq 3\rangle$ **and** $\langle odd\ x\rangle$ $\langle proof \rangle$
  qed
end

Structured statements

Structured assumptions

Postfix notation for Horn-clauses:

- **assume** B **if** A_1 **and** A_2 **for** $a_1\ a_2$
  - corresponds to **assume** $\bigwedge a_1\ a_2.\ A_1 \Longrightarrow A_2 \Longrightarrow B$
  - vacuous quantifiers are omitted
- similar for **inductive**, **definition**, **function** etc.

Example: structured specifications

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

function $gcd :: nat \Rightarrow nat \Rightarrow nat$
  where
    $gcd\ x\ 0 = x$
  $|$ $gcd\ 0\ y = y$
  $|$ $gcd\ (Suc\ x)\ (Suc\ y) = gcd\ (Suc\ x)\ (y - x)$ **if** $x < y$
  $|$ $gcd\ (Suc\ x)\ (Suc\ y) = gcd\ (x - y)\ (Suc\ y)$ **if** $\neg\ x < y$

Structured conclusions (goals)

Notation for Isar "eigen-context":

- premises: **have** B **if** A_1 A_2
- parameters: **have** B **for** a_1 a_2

- corresponds to $\{$ **fix** a_1 a_2 **assume** that: A_1 A_2 **have** B $\}$
- analogous to **lemma fixes** a_1 a_2 **assumes** that: A_1 A_2 **shows** B

Example: Natural Deduction with structured conclusions

- 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

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

$$\begin{array}{l} \textbf{have } \mathit{thesis} \\ \quad \textbf{if } \bigwedge\overline{x}.\ \overline{A}\ \overline{x} \Longrightarrow \mathit{thesis} \\ \quad \textbf{and } \bigwedge\overline{y}.\ \overline{B}\ \overline{y} \Longrightarrow \mathit{thesis} \\ \quad \textbf{for } \mathit{thesis} \end{array}$$

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** $A \mid B$

Elimination and cases

- method "$cases$" detects its rule from chained facts
- command "**case**" allows name and attribute specification

Example:

consider x **where** $A\ x \mid y$ **where** $B\ y$ $\langle proof \rangle$
then have $something$
proof $cases$
  case $prems$: 1
  show $?thesis$ **using** $prems$ $\langle proof \rangle$
next
  case $prems$: 2
  show $?thesis$ **using** $prems$ $\langle proof \rangle$
qed

Obtain

$$\begin{array}{l} \textbf{obtain}\ \overline{x}\ \textbf{where}\ \overline{A\ \overline{x}}\ \ \langle proof \rangle \ \ \equiv \\ \quad \textbf{consider}\ \overline{x}\ \textbf{where}\ \overline{A\ \overline{x}}\ \ \langle proof \rangle \\ \quad \textbf{fix}\ \overline{x}\ \textbf{assume}^{*}\ \overline{A\ \overline{x}} \end{array}$$

- old meaning is unchanged, but foundation simplified
- **is** patterns now supported (with λ-lifting over the parameters)
- **if** / **for** notation available as well

Isar Proof Documents

Common syntax for embedded languages

Outer theory syntax:

- keywords: user-defined commands (e.g. **definition**, **inductive**)
- identifiers, numerals etc.
- quoted strings "$source$": nesting requires backslash-escapes
- cartouches $\langle source \rangle$: arbitrary nesting without no escapes

Example:

ML $\langle val\ t = \boldsymbol{term}\, \langle \lambda x.\ x \leq y + z$ — comment in term$\rangle$ — comment in ML$\rangle$

Isabelle symbols

- plain-text representation of infinitely many named symbols: `\<NAME>` or `\<^NAME>`, e.g. `\<alpha>` or `\<^bold>`
- default rendering of finitely many symbols in LaTeX, HTML, GUI
- bundled Isabelle fonts for quality and reliability of display

Notes:

- Isabelle symbols are conceptually closer to LaTeX than to Unicode
- Unicode cannot be "trusted": complexity, confusion, drop-outs

Document text structure

Markup

- section headings (6 levels like in HTML):
  chapter, **section**, **subsection**, . . . , **subparagraph**
- text blocks: **text**, **txt**, **text_raw**
- raw LaTeX macros (rare)

Markdown

- implicit paragraphs and lists: itemize, enumerate, description

Formal comments

- marginal comments: — ⟨*text*⟩
- canceled text: ***cancel*** ⟨*text*⟩ e.g. $\cancel{bad}$
- raw LaTeX: ***latex*** ⟨*text*⟩ e.g. $\lim_{n \to \infty} \sum_{i=0}^{n} q^i$

Document antiquotations

full form: `@{`$name$ $[options]$ $arguments$ $\ldots$`}`

e.g. @{$term$ $[show\_types]$ ‹$Suc\ n$›} for $Suc\ (n{::}nat)$

short form:

1. cartouche argument: `\<^`$name$`>`‹$argument$›

   e.g. ***term*** ‹$Suc\ n$› for $Suc\ n$
2. implicit standard name: ‹$argument$›

   e.g. ‹$Suc\ n$› for $Suc\ n$ (unchecked)

   e.g. ‹$Suc\ Suc$› for $Suc\ Suc$ (unchecked)
3. no argument: `\<^`$name$`>`

Notable antiquotations:

- *bold*, *emph*, *verbatim*, *footnote*: text styles (with proper nesting)
- *cite*: formal BibTEX items
- *path*, *file*, *dir*, *url*, *doc*: system resources

Discussion

Good versus Bad Ideas

Good:

- named Isabelle symbols: closer to $\LaTeX$ than Unicode
- control symbols and text cartouches
- proof methods as parameter to the language

Bad:

- hybrid attributes: joint syntax for
  - *declaration attributes*, e.g. $[simp]$
  - *rule attributes*, e.g. $[rule\_format]$
- instantiation as rule attributes, e.g. $[where]$, $[of]$
- alternative ASCII syntax, e.g. $A --> B$ or $A \longrightarrow B$
  better: ASCII as input method only