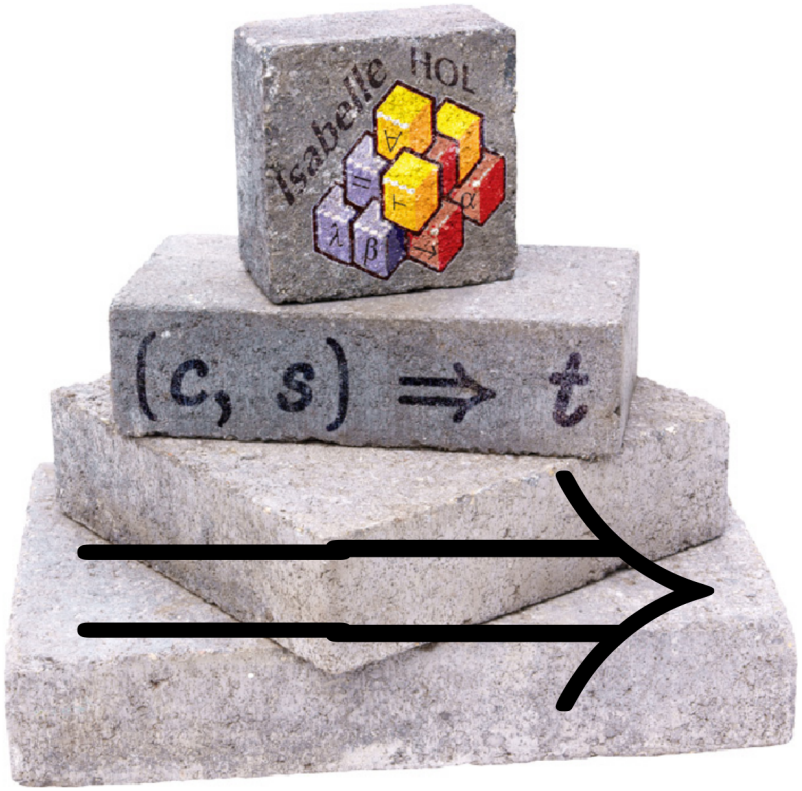


Higher-Order Logics and Interactive Theorem Proving with Isabelle/HOL II

Formal Systems II: Application

Bernhard Beckert · Mattias Ulbrich · Michael Kirsten | Summer Term 2025

More on Function Definitions

Non-recursive definitions

- **definition** $sq :: nat \Rightarrow nat$ **where** $sqn = n * n$
- No pattern matching, just $fx_1 \ldots x_n = \ldots$

Institute of Information Security
and Dependability (KASTEL)

More on Function Definitions

Non-recursive definitions

- **definition** $sq :: nat \Rightarrow nat$ **where** $sqn = n * n$
- No pattern matching, just $fx_1 \ldots x_n = \ldots$

(Possibly) recursive functions: fun

- Pattern-matching over datatype constructors
- Order of equation matters
- Termination must be provable automatically by size measures
- Proves customized induction schema
- Example: **fun** $ack :: nat \Rightarrow nat \Rightarrow nat$ **where**

$$ack\ 0\ n = Suc\ n \mid$$
$$ack\ (Suc\ m)\ 0 = ack\ m\ (Suc\ 0) \mid$$
$$ack\ (Suc\ m)\ (Suc\ n) = ack\ m\ (ack\ (Suc\ m)\ n)$$

Institute of Information Security and Dependability (KASTEL)

Proof method *simp*

Goal: 1. $[\![P_1; \ldots; P_m]\!] \Longrightarrow C$

apply$(simp\ add\text{:}\ eq_1\ \ldots\ eq_n)$

Simplify $P_1\ \ldots\ P_m$ and C using

- lemmas with attribute $simp$
- rules from **fun** and **datatype**
- additional lemmas $eq_1\ \ldots\ eq_n$
- assumptions $P_1\ \ldots\ P_m$

Variations:

- $(simp\ \ldots\ del\text{:}\ \ldots)$ removes $simp$-lemmas
- add and del are optional

Institute of Information Security and Dependability (KASTEL)

auto versus *simp*

- $auto$ acts on all subgoals
- $simp$ acts only on subgoal 1

- $auto$ applies $simp$ and more

- $auto$ can also be modified:
  $(auto\ simp\ add\text{: } \ldots\ \ simp\ del\text{: } \ldots)$

Institute of Information Security
and Dependability (KASTEL)

Rewriting with definitions

Definitions (**definition**) must be used explicitly:

$$(simp\ add\text{: } f\_def \ldots)$$

f is the function whose definition is to be unfolded.

Institute of Information Security
and Dependability (KASTEL)

Case Splitting with *simp/auto*

Automatic:

$$P\ (\mathit{if}\ A\ \mathit{then}\ s\ \mathit{else}\ t)$$
$$=$$
$$(A \longrightarrow P(s)) \wedge (\neg A \longrightarrow P(t))$$

By hand:

$$P\ (\mathit{case}\ e\ \mathit{of}\ 0 \Rightarrow a \mid \mathit{Suc}\ n \Rightarrow b)$$
$$=$$
$$(e = 0 \longrightarrow P(a)) \wedge (\forall n.\ e = \mathit{Suc}\ n \longrightarrow P(b))$$

Proof method: $(\mathit{simp}\ \mathit{split}\colon\ \mathit{nat.split})$
Or auto. Similar for any datatype t: $\mathit{t.split}$

Institute of Information Security and Dependability (KASTEL)

More Proof Methods

intro ⟨*intro-rules*⟩	repeatedly applies intro rules
elim ⟨*elim-rules*⟩	repeatedly applies elim rules
clarify	applies all safe rules that do not split the goal
safe	applies all safe rules
blast	an automatic tableaux prover (works well on predicate logic)
fast	another automatic search tactic
fastforce	rewriting, logic, sets, relations and a bit of arithmetic
arith	proves linear formulas (no "$*$"), complete for quantifier-free real and Presburger arithmetic

Institute of Information Security
and Dependability (KASTEL)

Download

https://www21.in.tum.de/teaching/fds/SS22/assets/Demos/Simp_Demo.thy

$\Longrightarrow$ versus $\longrightarrow$

$\Longrightarrow$ is part of the Isabelle framework. It structures theorems and proof states: $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$

$\longrightarrow$ is part of HOL and can occur inside the logical formulas A_i and A.

Phrase theorems like this $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$

not like this $A_1 \wedge \ldots \wedge A_n \longrightarrow A$

Institute of Information Security and Dependability (KASTEL)

Sets over type *'a*

$$'a\ set$$

- $\{\}, \quad \{e_1, \ldots, e_n\}$
- $e \in A, \quad A \subseteq B$
- $A \cup B, \quad A \cap B, \quad A - B, \quad - A$
- $\ldots$

$\in$	`\<in>` :
$\subseteq$	`\<subseteq>` `<=`
$\cup$	`\<union>` `Un`
$\cap$	`\<inter>` `Int`

Set Comprehension

- $\{x.\ P\}$ where x is a variable
- But not $\{t.\ P\}$ where t is a proper term
- Instead: $\{t \mid x\ y\ z.\ P\}$
  is short for $\{v.\ \exists\, x\ y\ z.\ v = t \wedge P\}$
  where x, y, z are the free variables in t

Institute of Information Security
and Dependability (KASTEL)

Isar Core Syntax

proof = **proof** [method] step* **qed**
| **by** method

method = $(simp \ldots)$ | $(blast \ldots)$ | $(induction \ldots)$

step = **fix** variables $(\bigwedge)$
| **assume** prop $(\Longrightarrow)$
| [**from** fact+] (**have** | **show**) prop proof

prop = [name:] "formula"

fact = name | ...

Institute of Information Security
and Dependability (KASTEL)

Example: Cantor's Theorem

lemma $\neg\ surj(f :: 'a \Rightarrow 'a\ set)$
proof default proof: assume $surj$, show $False$
  assume a: $surj\ f$
  from a **have** b: $\forall\ A.\ \exists\ a.\ A = f\ a$
    by$(simp\ add:\ surj\_def)$
  from b **have** c: $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$
    by $blast$
  from c **show** $False$
    by $blast$
qed

Institute of Information Security
and Dependability (KASTEL)

Download

https://www21.in.tum.de/teaching/fds/SS22/assets/Demos/Isar_Demo.thy

Abbreviations

$this$	$=$	the previous proposition proved or assumed
then	$=$	**from** $this$
thus	$=$	**then show**
hence	$=$	**then have**

Institute of Information Security
and Dependability (KASTEL)

using and with

(**have**|**show**) prop **using** facts

=

from facts (**have**|**show**) prop

with facts

=

from facts *this*

Structured Lemma Statement

lemma
  fixes $f :: 'a \Rightarrow 'a\ set$
  assumes s: $surj\ f$
  shows $False$
proof $-$ no automatic proof step
  have $\exists\ a.\ \{x.\ x \notin f\ x\} = f\ a$ **using** s
    by$(auto\ simp$: $surj\_def)$
  thus $False$ **by** $blast$
qed

Proves $surj\ f \Longrightarrow False$
but $surj\ f$ *becomes local fact* s *in proof.*

Institute of Information Security and Dependability (KASTEL)

The Essence of Structured Proofs

Assumptions and intermediate facts
can be named and referred to explicitly and selectively

Institute of Information Security
and Dependability (KASTEL)

Structured Lemma Statements

fixes $x :: \tau_1$ **and** $y :: \tau_2$ ...
assumes *a:* P **and** *b:* Q ...
shows R

- **fixes** and **assumes** sections optional
- **shows** optional if no **fixes** and **assumes**

Institute of Information Security
and Dependability (KASTEL)

Case Distinction

show R
proof $cases$
  assume P
  $\vdots$
  show R $\langle proof \rangle$
next
  assume $\neg$ P
  $\vdots$
  show R $\langle proof \rangle$
qed

have $P \vee Q$ $\langle proof \rangle$
then show R
proof
  assume P
  $\vdots$
  show R $\langle proof \rangle$
next
  assume Q
  $\vdots$
  show R $\langle proof \rangle$
qed

Institute of Information Security
and Dependability (KASTEL)

Set Equality and Subset

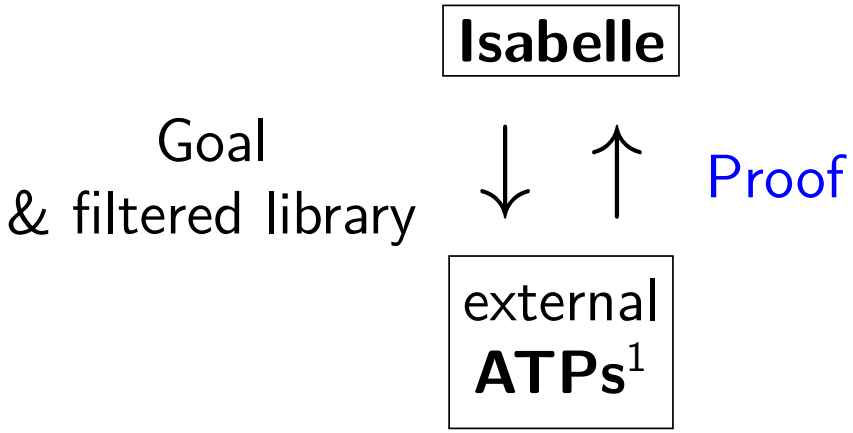
show $A = B$
proof
  show $A \subseteq B$ $\langle proof \rangle$
next
  show $B \subseteq A$ $\langle proof \rangle$
qed

show $A \subseteq B$
proof
  fix x
  assume $x \in A$
  $\vdots$
  show $x \in B$ $\langle proof \rangle$
qed

Institute of Information Security
and Dependability (KASTEL)

Sledgehammer

Architecture:



Characteristics:

- Sometimes it works,
- sometimes it doesn't.

Do you feel lucky?

[1] Automatic Theorem Provers

Sledgehammer

Sledgehammer:

- ➜ *Connects Isabelle with ATPs and SMT solvers:*
  E, SPASS, Vampire, CVC3, Yices, Z3
- ➜ *Simple invocation:*
  - ➜ *Users don't need to select or know facts*
  - ➜ *or ensure the problem is first-order*
  - ➜ *or know anything about the automated prover*
- ➜ *Exploits local parallelism and remote servers*

Institute of Information Security
and Dependability (KASTEL)

Download

https://www21.in.tum.de/teaching/fds/SS22/assets/Demos/Auto_Proof_Demo.thy

Download `Chapter2.thy` from
`http://concrete-semantics.org/Exercises/templates.tar`
and try to do some more exercises.

Institute of Information Security
and Dependability (KASTEL)