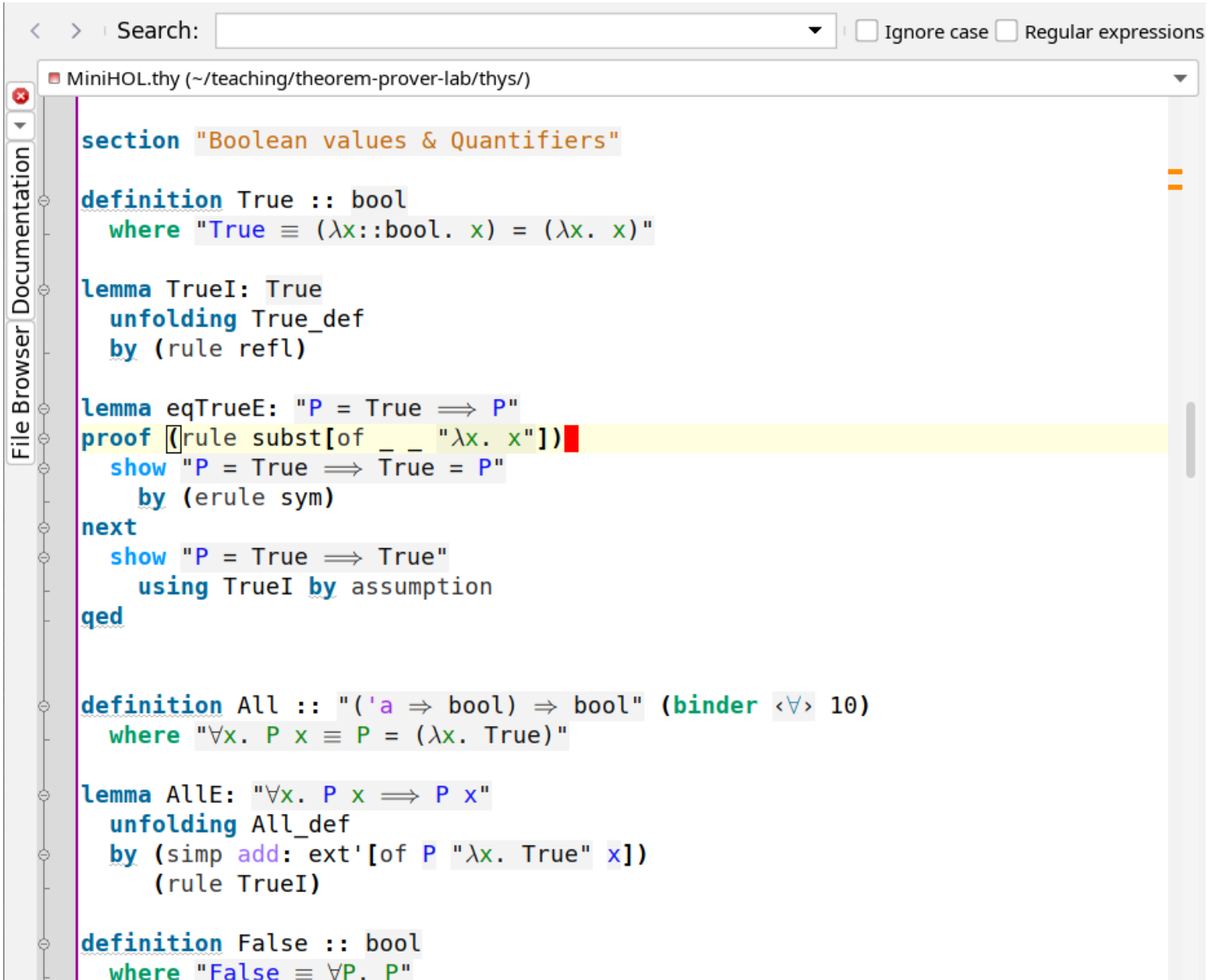
MiniHOL

Terru Stübinger Henriette Färber

2026-07-18
TheDu Workshop 2026

MiniHOL

A minimal object logic for teaching Isabelle/HOL's logic



Situating

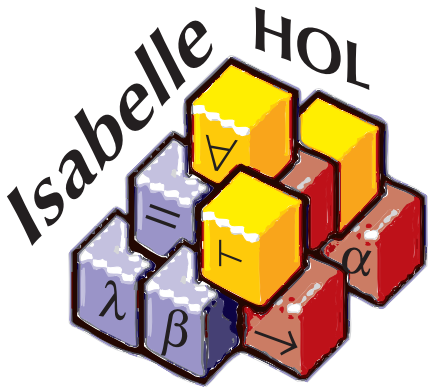
- around 3000 computer science students
- No lab course introducing any ITP
- (last in 2022)
- but: used in many student projects

- Group of Bernhard Beckert
- e.g. KeY, a Java verifier
- Verification of programs, modelling constraints, neural networks, ...
- Lots of Isabelle
- Other groups: Isabelle, Lean, ...

"Learning by doing" works but consumes (limited) project time

$\Longrightarrow$ Motivation for a new course (using Isabelle)

Isabelle/HOL



Isabelle is *generic*: Isabelle/HOL, Isabelle/FOL, Isabelle/ZF, ...

... but usually used with Higher-Order Logic [Gor88]

In active use & development for multiple decades

- heavy automation: no need to write proof trees!
- huge existing developments to build on
- many quality-of-life tools: **quickcheck**, **nitpick**, **try0**, ...
- but also: idiosyncrasies, historical artifacts, ...
- a lot of "magic" black-box automation

Unfortunately, opaque magic is hard to teach!

Course Idea "Theorem Prover Lab"

Inherited from previous course: practical course with 3 ECTS, one session per week

Basic Introduction

Data types, recursive functions, inductives, ...

Standard Ressource: First part of *Concrete Semantics* [NK14]

↓

Application Case studies

Examples of practical use

Language semantics, Refinement of Voting Systems (Michael Kirsten)

↓

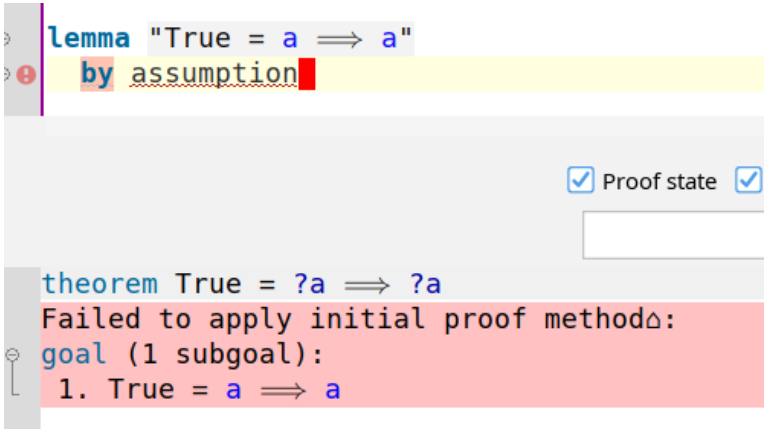
Project phase

Student projects in groups

Self-chosen topics

Hurdles

Opaque behaviour



- Automation is nice, but ...
- ... sometimes confusing
- `simp` can be given facts with `add:` and `using`
- `auto` takes arguments `intro` and `intro!`

Why teach what is underneath?

AfP alone contains over 13000 uses of `erule`, over 100000 uses of `rule`

Hurdles: previous experience

A bit of pre-history ...
"Formal Systems 2: Application" lecture in Summer 2025

- Idea: present tools used in current research
- included: short introduction to Isabelle (Michael Kirsten & myself)
- and higher-order logic

Isabelle is a tool, and teaching a tool theory-first is hard

$\Longrightarrow$ Student confusion

Questions

"I find it hard to map what was on the slides onto what one sees when using Isabelle"

> *Why are there so many more axiomatizations in the HOL folder?*
>
> *Why do we have to axiomatize logical connectives like* $p \longrightarrow q$*?*
>
> *Where is the "*$\Rightarrow$*" base type defined in HOL.thy?*

Questions

"I find it hard to map what was on the slides onto what one sees when using Isabelle"

Why and how do Isabelle/HOL and Gordon's HOL differ?

Connection between meta logic and object logic?

Built-in constructs

Gordon's HOL $\neq$ Isabelle's HOL: Axioms

Gordon's HOL famously has five axioms:

bool_cases: "$\forall$ *t. t = True* $\vee$ *t = False"* **and**
imp_antisym: "$\forall$ t_1 t_2. *(*t_1 $\longrightarrow$ t_2*)* $\longrightarrow$ *(*t_2 $\longrightarrow$ t_1*)* $\longrightarrow$ *(*t_1 *=* t_2*)"* **and**
eta: "$\forall$ *t. (*λ*x. t x) = t"* **and**
select: "$\forall$ *P::'a* $\Rightarrow$ *bool. P x* $\longrightarrow$ *P (SOME x. P x)"* **and**
infinity: "$\exists$ *(z :: ind) (f::ind* $\Rightarrow$ *ind).* $\forall$ *x. f x* $\neq$ *z)* $\wedge$ *(*$\forall$ *x y. f x = f y* $\longrightarrow$ *x = y)"*

Gordon's HOL $\neq$ Isabelle's HOL: Axioms

```
refl: "t = t" and
subst: "s = t ⟹ P s ⟹ P t" and
ext: "(⋀x. f x = g x) ⟹ (λx. f x) = (λx. g x)" and
eq_reflection: "a = b ⟹ a ≡ b" and
impI: "(P ⟹ Q) ⟹ P ⟶ Q" and
mp: "⟦P ⟶ Q; P⟧ ⟹ Q" and
True_or_False: "P = True ∨ P = False" and
the_eq_trivial: "(THE x. x = a) = (a::'a)" and
Suc_Rep_inject: "Suc_Rep x = Suc_Rep y ⟹ x = y" and
Suc_Rep_not_Zero_Rep: "Suc_Rep x ≠ Zero_Rep"
```

- No distinction between inference rules and axioms
- but: the meta logic already handles some things
- SOME has disappeared into its own file `Hilbert_Choice.thy`

Gordon's HOL $\neq$ Isabelle's HOL: Axioms

Definitions also differ, most obviously for $\exists$

```
definition Ex :: "('a ⇒ bool) ⇒ bool" (binder <∃ > 10)
  where "∃x. P x ≡ P (SOME x. P x)
```

Now instead (since 2001):

```
definition Ex :: "('a ⇒ bool) ⇒ bool" (binder <∃ > 10)
  where "∃x. P x ≡ ∀Q. (∀x. P x ⟶ Q) ⟶ Q"
```

... and intermediate derivations also differ

For understanding Isabelle/HOL, none of the common formulations [NK14; Gor88; LV24; VFB22] are "quite right"

Revised Structure

Basic Introduction

↓

Application Case studies

↓

Project phase

Student projects in groups

Revised Structure

Basic Introduction

↓

Pause: what does this logic look like we find ourselves in?

↓

Application Case studies

↓

Project phase

Student projects in groups

Revised Structure

Basic Introduction

↓

MiniHOL

A minimal object logic for teaching HOL

"We have shown you how to use the surface, now let's look underneath"

↓

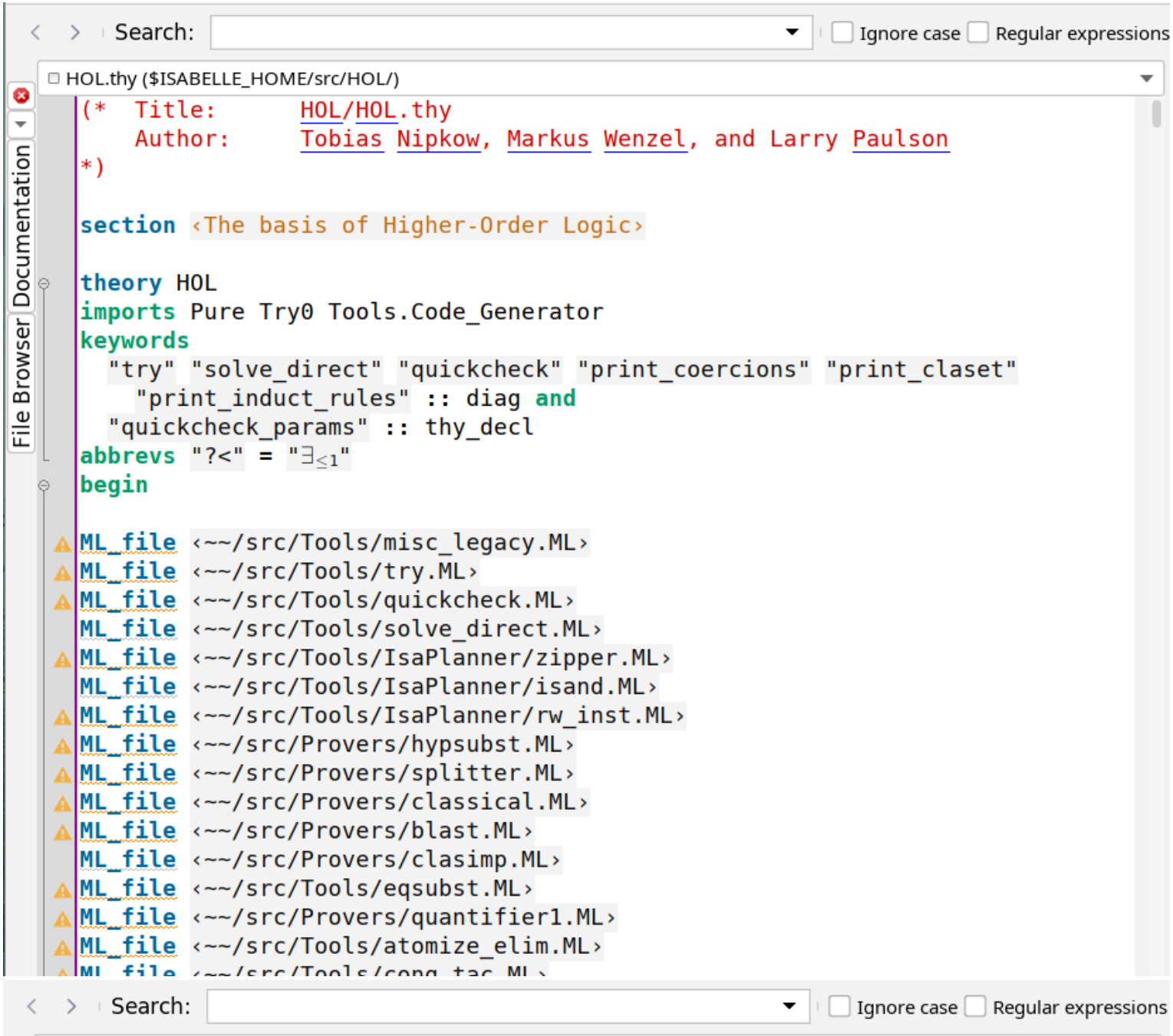
Application Case studies

↓

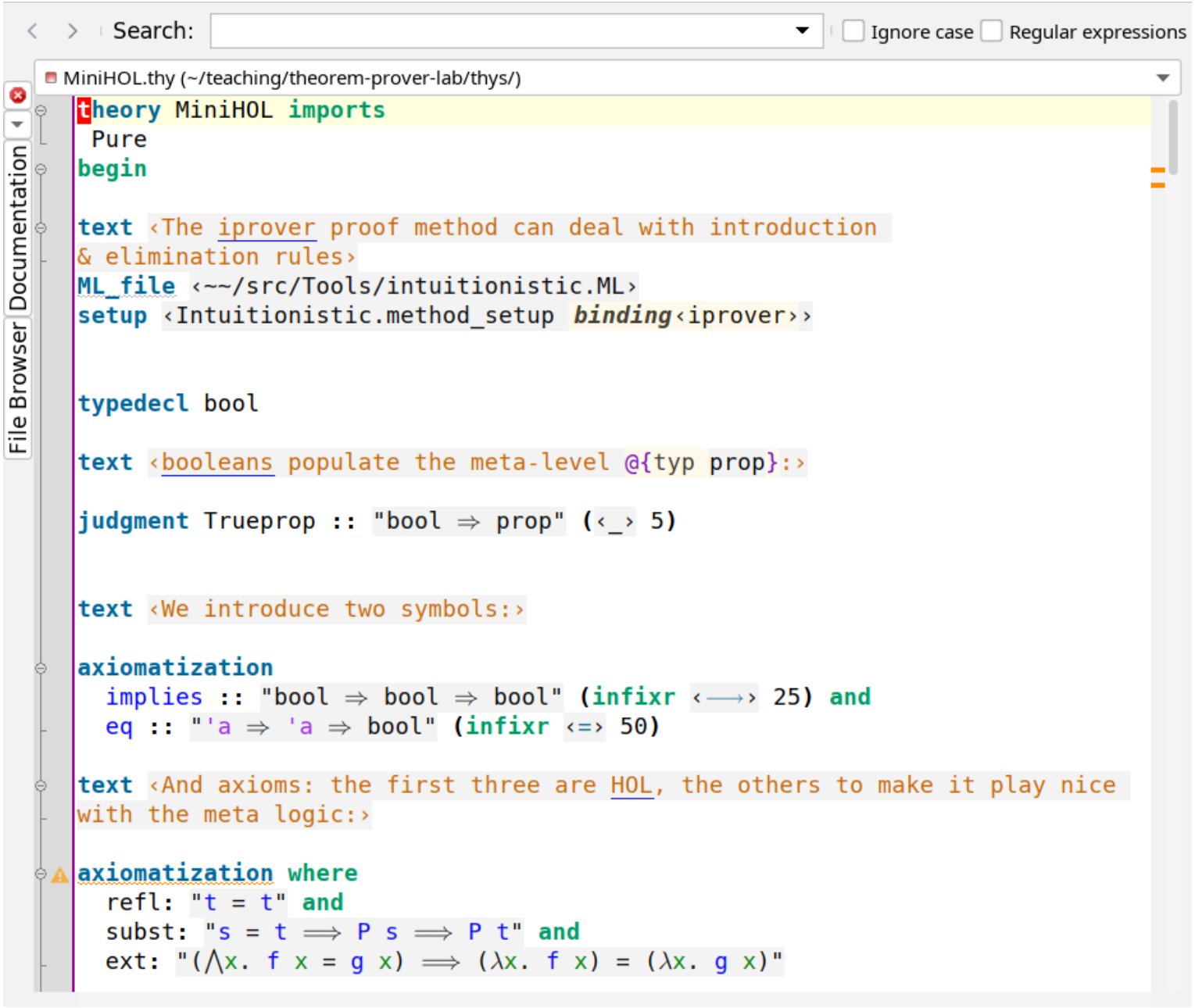
Project phase

Student projects in groups

Why not the "real" HOL.thy?



MiniHOL.thy


Search:
Ignore case
Regular expressions
MiniHOL.thy (~/teaching/theorem-prover-lab/thys/)
File Browser
Documentation
theory MiniHOL imports
Pure
begin
text ‹The iprover proof method can deal with introduction
& elimination rules›
ML_file ‹~~/src/Tools/intuitionistic.ML›
setup ‹Intuitionistic.method_setup binding‹iprover››
typedecl bool
text ‹booleans populate the meta-level @{typ prop}:›
judgment Trueprop :: "bool ⇒ prop" (‹_› 5)
text ‹We introduce two symbols:›
axiomatization
implies :: "bool ⇒ bool ⇒ bool" (infixr ‹⟶› 25) and
eq :: "'a ⇒ 'a ⇒ bool" (infixr ‹=› 50)
text ‹And axioms: the first three are HOL, the others to make it play nice
with the meta logic:›
axiomatization where
refl: "t = t" and
subst: "s = t ⟹ P s ⟹ P t" and
ext: "(⋀x. f x = g x) ⟹ (λx. f x) = (λx. g x)"

MiniHOL

Idea:

> "`HOL.thy` with everything beginner-confusing removed"

- only use **(e|d|f)?rule** and **iprover**
- use the same axioms, definitions, intermediate lemmas as `HOL.thy`
- Minimal use of syntax rules, no tool setup steps, ...
- no type classes (could be added)
- have explicit, readable proofs

Side effect: forces engagement with proof primitives

MiniHOL: Booleans

Two axioms characterise equality, one to work with it:

$$(x = x) :: bool \qquad \text{(refl)}$$

$$(\bigwedge x.fx = gy) \Longrightarrow (\lambda x.\ f\ x) = (\lambda x.\ g\ x) \qquad \text{(ext)}$$

$$s = t \Longrightarrow Ps \Longrightarrow Pt \qquad \text{(subst)}$$

Then we define:

$$\text{True} \equiv \lambda(x :: \text{bool}).\ x = \lambda x.\ x$$

And can prove:

```
1  lemma TrueI: True
2    unfolding True_def
3    by (rule refl)
```

Further axioms: Relating to the metalogic

HOL's implication and equality should behave identical to those of the meta logic

$$(P \Longrightarrow Q) \Longrightarrow P \longrightarrow Q \qquad \text{(impl)}$$

$$P \longrightarrow Q \Longrightarrow P \Longrightarrow Q \qquad \text{(mp)}$$

$$a = b \Longrightarrow a \equiv b \qquad \text{(eq\_reflection)}$$

What about $a \equiv b \Longrightarrow a = b$?

```
1 lemma meta_eq_to_obj_eq: "a ≡ b⟹ a = b"
2 proof -
3   assume "a ≡ b"
4   show "a = b"
5     unfolding <a ≡ b> by (rule refl)
6 qed
```

Eliminating equality with true

```
lemma sym: "a = b ⟹ b = a"
proof (rule subst[of a b "λx. x = a"])
  show "a = b ⟹ a = b" by assumption
next
  show "a = a" by (rule refl)
qed
```

```
lemma eqTrueE: "P = True ⟹ P"
proof (erule subst)
  show "P = True ⟹ True = P"
    by (erule sym)
next
  show "P = True ⟹ True"
    using TrueI by assumption
qed
```

Variants

MiniHOL fits into one file

$\Longrightarrow$ easy to manipulate

Allows ...

... answering questions

- Is the definition of $\exists$ equivalent to Gordon's?
- Do we need definite *and* indefinite description?

... looking at variants

- What about removing excluded middle?
- Does `undefined` need its own axiom?

Did it actually help?

The course had 12 students who actively participated

- University course evaluation: 7 respondents, very positive

  What I liked most: The background information and occasional deep dive into Isabelle internals

- Informal discussions with participants: course was well-received

Of course, obvious self-selection caveats here!

Conclusion

It's worth lifting the veil of magic automation!

MiniHOL could be extended, e.g.

- Type classes, proof methods
- More variants, proving equivalence of Gordon's definitions
- ...

You all have much more experience teaching than I do, so:

Questions / Suggestions?

References

[Gor88] Michael J. C. Gordon. "HOL: A Proof Generating System for Higher-Order Logic". In: *VLSI Specification, Verification and Synthesis*. Ed. by Graham Birtwistle and P. A. Subrahmanyam. Springer, 1988, pp. 73–128. DOI: 10.1007/978-1-4613-2007-4_3.

[LV24] Simon Tobias Lund and Jørgen Villadsen. "Teaching higher-order logic using Isabelle". In: *Theorem Proving Components for Educational Software 2021 (ThEdu'21)*. 2024. DOI: 10.4204/EPTCS.400.5.

[NK14] Tobias Nipkow and Gerwin Klein. *Concrete semantics: with Isabelle/HOL*. Springer, 2014. DOI: 10.1007/978-3-319-10542-0. URL: http://concrete-semantics.org/.

[VFB22] Jørgen Villadsen, Asta Halkjær From, and Patrick Blackburn. "Teaching Intuitionistic and Classical Propositional Logic Using Isabelle". In: *Theorem Proving Components for Educational Software 2021 (ThEdu'21)*. 2022. DOI: 10.4204/EPTCS.354.6.