Teaching Intuitionistic Propositional Logic Using Isabelle

Jørgen Villadsen[1], Asta Halkjær From[1], and Patrick Blackburn[2]

[1] Technical University of Denmark, Kongens Lyngby, Denmark
[2] Roskilde University, Roskilde, Denmark

Abstract

We describe a formalization of intuitionistic propositional logic in the Isabelle/Pure framework. In contrast to earlier work (where we explored the pedagogical benefits of using a deep embedding approach to logical modelling) here we employ a simpler *axiomatic instance* modelling. This gives rise to simple and natural teaching examples and we report on the role it played in teaching our Automated Reasoning course in 2020 and 2021.

A step towards first-order logic, higher-order logic & classical logic

Adding excluded middle p ∨ ¬ p changes “the game” a lot… :-)

New course on automated reasoning

50+ master students in 2020 & 2021

Many with international bachelor

How to teach Isabelle/HOL?

Most do not have the prerequisites in logic and functional programming (any language)

Focus:
Natural Deduction + The Isabelle/HOL Tutorial (Programming and Proving in Isabelle/HOL)

NaDeA:
A Natural Deduction Assistant with a Formalization in Isabelle

Jørgen Villadsen, Asta Halkjær From, Alexander Birch Jensen & Anders Schlichtkrull

Technical University of Denmark – DTU Compute

Natural Deduction Assistant (NaDeA)

Natural Deduction Assistant

1	Imp_I	[ ] $A \wedge B \rightarrow A$
2	Con_E1	[$A \wedge B$] A
3	Assume	[$A \wedge B$] $A \wedge B$

Load Code Help ProofJudge Login

https://nadea.compute.dtu.dk/

Natural Deduction Assistant

1 ¤ [ ] $(\forall x.R(x, x)) \rightarrow (\forall x.\exists y.R(x, y))$

Boole
Imp_E
Imp_I
Dis_E
Con_E1
Con_E2
Exi_E

X

Natural Deduction Assistant

1	Imp_I	[ ] $(\forall x.R(x, x)) \rightarrow (\forall x.\exists y.R(x, y))$
2	¤	$[\forall x.R(x, x)]$ $\forall x.\exists y.R(x, y)$

Boole
Imp_E
Dis_E
Con_E1
Con_E2
Exi_E
Uni_I

X

Natural Deduction Assistant

1	Imp_I	[ ] $(\forall x.R(x, x)) \rightarrow (\forall x.\exists y.R(x, y))$
2	Uni_I	[$\forall x.R(x, x)$] $\forall x.\exists y.R(x, y)$
3	¤	[$\forall x.R(x, x)$] $\exists x.R(c', x)$
4	*	(c')

Natural Deduction Assistant

1	Imp_I	$[\,]\ (\forall x.R(x, x)) \rightarrow (\forall x.\exists y.R(x, y))$
2	Uni_I	$[\forall x.R(x, x)]\ \forall x.\exists y.R(x, y)$
3	Exi_I	$[\forall x.R(x, x)]\ \exists x.R(c', x)$
4	Uni_E	$[\forall x.R(x, x)]\ R(c', c')$
5	Assume	$[\forall x.R(x, x)]\ \forall x.R(x, x)$
6	*	(c')

```
theorem "(∀x. r x x) ⟶ (∀x. ∃y. r x y)"
proof
  assume a: "∀x. r x x"
  show "∀x. ∃y. r x y"
  proof
    fix c
    show "∃y. r c y"
    proof
      show "r c c"
        using a ..
    qed
  qed
qed
```

```
theorem "(∀x. r x x) ⟶ (∀x. ∃y. r x y)"
proof (rule Imp_I)
  assume a: "∀x. r x x"
  show "∀x. ∃y. r x y"
  proof (rule Uni_I)
    fix c
    show "∃y. r c y"
    proof (rule Exi_I)
      show "r c c"
        using a by (rule Uni_E)
    qed
  qed
qed
```

Names of natural deduction rules are omitted to the left for brevity (works unchanged in Isabelle/HOL)

Natural Deduction Assistant

1	Imp_I	[] (∀x.R(x, x))→(∀x.∃y.R(x, y))
2	Uni_I	[∀x.R(x, x)] ∀x.∃y.R(x, y)
3	Exi_I	[∀x.R(x, x)] ∃x.R(c', x)
4	Uni_E	[∀x.R(x, x)] R(c', c')
5	Assume	[∀x.R(x, x)] ∀x.R(x, x)
6	*	(c')

Quote from reviewer:

The authors have chosen to systematically use the two periods .. to omit the justification of the steps in a proof.

The pedagogical advantage of this seems obscure to me.

How does the automation of reasoning help students to understand the inner structure of derivations produced within a given deductive system?

Quite to the contrary, I would have thought that forcing students to be explicit about the choices of rules would produce a more lasting impact on their proficiency as users of the given deductive system.

Would the authors convince me of the contrary?

Let us examine some examples. The **proposition** command is an alternative name for the **theorem** command and we use it for student exercises and examples. There is no difference to the system, only to the human reader.

proposition $\langle p \longrightarrow \neg\ \neg\ p \rangle$
proof
  assume $\langle p \rangle$
  show $\langle \neg\ \neg\ p \rangle$
  proof
    assume $\langle \neg\ p \rangle$
    from $\langle \neg\ p \rangle$ **and** $\langle p \rangle$ **show** $\langle \bot \rangle$ **..**
  qed
qed

Note the way the Isabelle/Pure code matches the way we would explain to students how they should go about proving this in natural deduction. It essentially spells out a constructive process for building the proof tree required — with the formula being proved if and only if such a proof tree exists.

Here is a proof of modus tollens. Once again, note how Isabelle/Pure tracks the way a teacher would explain to a beginner how to construct the relevant proof tree:

proposition $\langle(p \longrightarrow q) \wedge \neg\, q \longrightarrow \neg\, p\rangle$
proof
  assume $\langle(p \longrightarrow q) \wedge \neg\, q\rangle$
  show $\langle\neg\, p\rangle$
  proof
    assume $\langle p\rangle$
    from $\langle(p \longrightarrow q) \wedge \neg\, q\rangle$ **have** $\langle p \longrightarrow q\rangle$ **..**
    from $\langle p \longrightarrow q\rangle$ **and** $\langle p\rangle$ **have** $\langle q\rangle$ **..**
    from $\langle(p \longrightarrow q) \wedge \neg\, q\rangle$ **have** $\langle\neg\, q\rangle$ **..**
    from $\langle\neg\, q\rangle$ **and** $\langle q\rangle$ **show** $\langle\bot\rangle$ **..**
  qed
qed

Here is a more complex example:

proposition $\langle(p \longleftrightarrow q) \longleftrightarrow q \longleftrightarrow p\rangle$
proof
  assume $\langle p \longleftrightarrow q\rangle$
  show $\langle q \longleftrightarrow p\rangle$
  proof
    from $\langle p \longleftrightarrow q\rangle$ **show** $\langle q \Longrightarrow p\rangle$ **..**
  next
    from $\langle p \longleftrightarrow q\rangle$ **show** $\langle p \Longrightarrow q\rangle$ **..**
  qed
next
  assume $\langle q \longleftrightarrow p\rangle$
  show $\langle p \longleftrightarrow q\rangle$
  proof
    from $\langle q \longleftrightarrow p\rangle$ **show** $\langle p \Longrightarrow q\rangle$ **..**
  next
    from $\langle q \longleftrightarrow p\rangle$ **show** $\langle q \Longrightarrow p\rangle$ **..**
  qed
qed

Our work is based on the Pure/Examples by Makarius in the Isabelle Sources:

https://isabelle.in.tum.de/dist/library/Pure/Pure-Examples/document.pdf

That document, however, contains few comments and we have polished the formalization and tested it in class. Our formalization is available online in various formats:

https://hol.compute.dtu.dk/Pure_I/

We have used our approach in our Automated Reasoning course in 2020 and 2021.

https://kurser.dtu.dk/course/02256

```
axiomatization Imp (infixr ‹⟶› 3)
  where Imp_I [intro]: ‹(p ⟹ q) ⟹ p ⟶ q›
    and Imp_E [elim]: ‹p ⟶ q ⟹ p ⟹ q›

axiomatization Dis (infixr ‹∨› 4)
  where Dis_E [elim]: ‹p ∨ q ⟹ (p ⟹ r) ⟹ (q ⟹ r) ⟹ r›
    and Dis_I1 [intro]: ‹p ⟹ p ∨ q›
    and Dis_I2 [intro]: ‹q ⟹ p ∨ q›

axiomatization Con (infixr ‹∧› 5)
  where Con_I [intro]: ‹p ⟹ q ⟹ p ∧ q›
    and Con_E1 [elim]: ‹p ∧ q ⟹ p›
    and Con_E2 [elim]: ‹p ∧ q ⟹ q›

axiomatization Falsity (‹⊥›)
  where Falsity_E [elim]: ‹⊥ ⟹ p›
```

```
definition Truth (‹⊤›) where ‹⊤ ≡ ⊥ ⟶ ⊥›

theorem Truth_I [intro]: ‹⊤›
  unfolding Truth_def ..

definition Neg (‹¬ _› [6] 6) where ‹¬ p ≡ p ⟶ ⊥›

theorem Neg_I [intro]: ‹(p ⟹ ⊥) ⟹ ¬ p›
  unfolding Neg_def ..

theorem Neg_E [elim]: ‹¬ p ⟹ p ⟹ q›
  unfolding Neg_def
proof -
  assume ‹p ⟶ ⊥› and ‹p›
  then have ‹⊥› ..
  then show ‹q› ..
qed
```

```
definition Iff (infixr ‹⟷› 3) where ‹p ⟷ q ≡ (p ⟶ q) ∧ (q ⟶ p)›

theorem Iff_I [intro]: ‹(p ⟹ q) ⟹ (q ⟹ p) ⟹ p ⟷ q›
  unfolding Iff_def
proof -
  assume ‹p ⟹ q› and ‹q ⟹ p›
  from ‹p ⟹ q› have ‹p ⟶ q› ..
  from ‹q ⟹ p› have ‹q ⟶ p› ..
  from ‹p ⟶ q› and ‹q ⟶ p› show ‹(p ⟶ q) ∧ (q ⟶ p)› ..
qed

theorem Iff_E1 [elim]: ‹p ⟷ q ⟹ p ⟹ q›
  unfolding Iff_def
proof -
  assume ‹(p ⟶ q) ∧ (q ⟶ p)›
  then have ‹p ⟶ q› ..
  then show ‹p ⟹ q› ..
qed

theorem Iff_E2 [elim]: ‹p ⟷ q ⟹ q ⟹ p›
  unfolding Iff_def
proof -
  assume ‹(p ⟶ q) ∧ (q ⟶ p)›
  then have ‹q ⟶ p› ..
  then show ‹q ⟹ p› ..
qed
```

Beyond Propositional Logic:

Extensible to Intuitionistic Higher-Order Logic – Main Example

```
text ‹Cantor's Theorem: Every set has more subsets than members›

theorem Cantor: ‹¬ (∃f. ∀s :: 'a ⇒ bool. ∃x :: 'a. s = f x)›
proof
  assume ‹∃f. ∀s :: 'a ⇒ bool. ∃x :: 'a. s = f x›
  then obtain f where ‹∀s :: 'a ⇒ bool. ∃x :: 'a. s = f x› ..
  let ?D = ‹λx. ¬ f x x›
  from ‹∀s. ∃x. s = f x› have ‹∃x. ?D = f x› ..
  then obtain c where ‹?D = f c› ..
  from subst [of ?D] and this and refl have ‹¬ f c c ⟷ f c c› .
  with contr show ⊥ .
qed
```

Classical Logic from Extensionality and Choice (Epsilon Operator)

Actual 2021 Exam Problem – Peirce's Law from Classical Propositional Logic

Weighted as 15 Minutes of the 2-Hour Exam

```
section ‹Question 4.1›

proposition ‹((p ⟶ q) ⟶ p) ⟶ p›
proof
  assume ‹(p ⟶ q) ⟶ p›
  show p
  proof (rule Boole)
    assume ‹p ⟶ ⊥›
    have ‹p ⟶ q›
    proof
      assume p
      with ‹p ⟶ ⊥› have ⊥ ..
      then show q ..
    qed
    with ‹(p ⟶ q) ⟶ p› have p ..
    with ‹p ⟶ ⊥› show ⊥ ..
  qed
qed
```

```
theorem Diaconescu: ‹p ∨ ¬ p›
  \<proof>

corollary classical: ‹(¬ p ⟹ p) ⟹ p›
  \<proof>

theorem Boole: ‹(p ⟶ ⊥ ⟹ ⊥) ⟹ p›
  \<proof>
```

Conclusions

We presented a simple and direct way of teaching intuitionistic propositional logic, namely by “axiomatizing” the Isabelle/Pure system in the most direct way possible

The pedagogical benefits of this approach are summed up in the words ‘simple’ and ‘natural’

The object and metalevel deductions mirror each other clearly — and the way that teachers would typically instruct students to construct proofs is directly reflected in the code the students learn to write to get Isabelle to prove things

Thanks!