AR: Proving and Reasoning in Isabelle/HOL

Victor Dumitrescu & Jacques Fleuriot

2 February 2016

Learning Outcomes

- Be able to write Natural Deduction proofs in propositional and first-order logic
- Learn to use an interactive proof assistant
- Mechanically verify natural deduction proofs
- Formalise theorems in first-order logic
- Mechanically verify theorems about data-structures

Assignment Overview

1. Natural Deduction Proofs
    - Propositional Logic
    - First-Order Logic
    - Reasoning with Equality
2. Inductive Proofs on Binary Trees

Some Natural Deduction Proofs

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

Some Natural Deduction Proofs

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

Some Natural Deduction Proofs

lemma $"P \wedge Q \longrightarrow Q \wedge P"$

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(rule conjE)*

1. $P \wedge Q \Longrightarrow ?P2 \wedge ?Q2$
2. $[\![P \wedge Q;\ ?P2;\ ?Q2]\!] \Longrightarrow Q \wedge P$

Some Natural Deduction Proofs

lemma $"P \wedge Q \longrightarrow Q \wedge P"$

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(rule conjE)*

1. $P \wedge Q \Longrightarrow ?P2 \wedge ?Q2$
2. $[\![P \wedge Q;\ ?P2;\ ?Q2]\!] \Longrightarrow Q \wedge P$

apply *assumption*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q \wedge P$

Some Natural Deduction Proofs

lemma $"P \wedge Q \longrightarrow Q \wedge P"$

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(rule conjE)*

1. $P \wedge Q \Longrightarrow ?P2 \wedge ?Q2$
2. $[\![P \wedge Q;\ ?P2;\ ?Q2]\!] \Longrightarrow Q \wedge P$

apply *assumption*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q \wedge P$

apply *(rule conjI)*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q$
2. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

Some Natural Deduction Proofs (continued)

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q$
2. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

Some Natural Deduction Proofs (continued)

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q$
2. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

apply *assumption*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

Some Natural Deduction Proofs (continued)

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q$
2. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

apply *assumption*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

apply *assumption*

No subgoals!

Some Natural Deduction Proofs (continued)

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow Q$
2. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

apply *assumption*

1. $[\![P \wedge Q;\ P;\ Q]\!] \Longrightarrow P$

apply *assumption*

No subgoals!

done

Using *erule*

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

Using *erule*

lemma *"P ∧ Q ⟶ Q ∧ P"*

1. P ∧ Q ⟶ Q ∧ P

apply *(rule impI)*

1. P ∧ Q ⟹ Q ∧ P

Using *erule*

lemma $"P \wedge Q \longrightarrow Q \wedge P"$

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(erule conjE)*

1. $[\![P;\ Q]\!] \Longrightarrow Q \wedge P$

Using *erule*

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(erule conjE)*

1. $[\![P;\ Q]\!] \Longrightarrow Q \wedge P$

apply *(rule conjI)*

1. $[\![P;\ Q]\!] \Longrightarrow Q$
2. $[\![P;\ Q]\!] \Longrightarrow P$

Using *erule*

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(erule conjE)*

1. $[\![P;\ Q]\!] \Longrightarrow Q \wedge P$

apply *(rule conjI)*

1. $[\![P;\ Q]\!] \Longrightarrow Q$
2. $[\![P;\ Q]\!] \Longrightarrow P$

apply *assumption+*

Using *erule*

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(erule conjE)*

1. $[\![P;\ Q]\!] \Longrightarrow Q \wedge P$

apply *(rule conjI)*

1. $[\![P;\ Q]\!] \Longrightarrow Q$
2. $[\![P;\ Q]\!] \Longrightarrow P$

apply *assumption+*

No subgoals!

done

Using *erule*

lemma "$P \wedge Q \longrightarrow Q \wedge P$"

1. $P \wedge Q \longrightarrow Q \wedge P$

apply *(rule impI)*

1. $P \wedge Q \Longrightarrow Q \wedge P$

apply *(erule conjE)*

1. $[\![P; Q]\!] \Longrightarrow Q \wedge P$

by *(rule conjI)*

Negation

lemma "(P ⟶ Q) ⟶ ¬Q ⟶ ¬P"

1. (P ⟶ Q) ⟶ ¬ Q ⟶ ¬ P

Negation

lemma "$(P \longrightarrow Q) \longrightarrow \neg Q \longrightarrow \neg P$"

1. $(P \longrightarrow Q) \longrightarrow \neg\ Q \longrightarrow \neg\ P$

apply *(rule impI)+*

1. $[\![P \longrightarrow Q;\ \neg\ Q]\!] \Longrightarrow \neg\ P$

Negation

lemma "$(P \longrightarrow Q) \longrightarrow \neg Q \longrightarrow \neg P$"

1. $(P \longrightarrow Q) \longrightarrow \neg\ Q \longrightarrow \neg\ P$

apply *(rule impI)+*

1. $[\![P \longrightarrow Q;\ \neg\ Q]\!] \Longrightarrow \neg\ P$

apply *(rule notI)*

1. $[\![P \longrightarrow Q;\ \neg\ Q;\ P]\!] \Longrightarrow False$

Negation

lemma "$(P \longrightarrow Q) \longrightarrow \neg Q \longrightarrow \neg P$"

1. $(P \longrightarrow Q) \longrightarrow \neg\, Q \longrightarrow \neg\, P$

apply *(rule impI)+*

1. $[\![P \longrightarrow Q;\ \neg\, Q]\!] \Longrightarrow \neg\, P$

apply *(rule notI)*

1. $[\![P \longrightarrow Q;\ \neg\, Q;\ P]\!] \Longrightarrow False$

apply *(erule notE)*

1. $[\![P \longrightarrow Q;\ P]\!] \Longrightarrow Q$

Negation

lemma "$(P \longrightarrow Q) \longrightarrow \neg Q \longrightarrow \neg P$"

1. $(P \longrightarrow Q) \longrightarrow \neg\ Q \longrightarrow \neg\ P$

apply *(rule impI)+*

1. $[\![P \longrightarrow Q;\ \neg\ Q]\!] \Longrightarrow \neg\ P$

apply *(rule notI)*

1. $[\![P \longrightarrow Q;\ \neg\ Q;\ P]\!] \Longrightarrow False$

apply *(erule notE)*

1. $[\![P \longrightarrow Q;\ P]\!] \Longrightarrow Q$

by *(erule mp)*

First-order Reasoning

lemma $"(\forall x.\ P\ x \lor Q\ x) \land (\exists x.\ \neg Q\ x) \longrightarrow (\exists x.\ P\ x)"$

1. $(\forall x.\ P\ x \lor Q\ x) \land (\exists x.\ \neg\ Q\ x) \longrightarrow (\exists x.\ P\ x)$

lemma "$(\forall x.\ P\ x \lor Q\ x) \land (\exists x.\ \neg Q\ x) \longrightarrow (\exists x.\ P\ x)$"

1. $(\forall x.\ P\ x \lor Q\ x) \land (\exists x.\ \neg\ Q\ x) \longrightarrow (\exists x.\ P\ x)$

apply *(rule impI)*

1. $(\forall x.\ P\ x \lor Q\ x) \land (\exists x.\ \neg\ Q\ x) \Longrightarrow \exists x.\ P\ x$

First-order Reasoning

lemma *"(∀x. P x ∨ Q x) ∧ (∃x. ¬Q x) ⟶ (∃x. P x)"*

1. (∀x. P x ∨ Q x) ∧ (∃x. ¬ Q x) ⟶ (∃x. P x)

apply *(rule impI)*

1. (∀x. P x ∨ Q x) ∧ (∃x. ¬ Q x) ⟹ ∃x. P x

apply *(erule conjE)*

1. ⟦∀x. P x ∨ Q x; ∃x. ¬ Q x⟧ ⟹ ∃x. P x

First-order Reasoning

lemma $"(\forall x.\ P\ x \vee Q\ x) \wedge (\exists x.\ \neg Q\ x) \longrightarrow (\exists x.\ P\ x)"$

1. $(\forall x.\ P\ x \vee Q\ x) \wedge (\exists x.\ \neg\ Q\ x) \longrightarrow (\exists x.\ P\ x)$

apply *(rule impI)*

1. $(\forall x.\ P\ x \vee Q\ x) \wedge (\exists x.\ \neg\ Q\ x) \Longrightarrow \exists x.\ P\ x$

apply *(erule conjE)*

1. $[\![\forall x.\ P\ x \vee Q\ x;\ \exists x.\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule exE)*

1. $\bigwedge x.\ [\![\forall x.\ P\ x \vee Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

First-order Reasoning (continued)

1. $\bigwedge x.\ [\![\forall x.\ P\ x \lor Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

First-order Reasoning (continued)

1. $\bigwedge x.\ [\![\forall x.\ P\ x \vee Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

apply *(drule spec)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x) \vee Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

First-order Reasoning (continued)

1. $\bigwedge x.\ [\![\forall x.\ P\ x \lor Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

apply *(drule spec)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x) \lor Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule disjE)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$
2. $\bigwedge x.\ [\![\neg\ Q\ x;\ Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

First-order Reasoning (continued)

1. $\bigwedge x.\ [\![\forall x.\ P\ x \vee Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

apply *(drule spec)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x) \vee Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule disjE)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$
2. $\bigwedge x.\ [\![\neg\ Q\ x;\ Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule exI)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

First-order Reasoning (continued)

1. $\bigwedge x.\ [\![\forall x.\ P\ x \vee Q\ x;\ \neg\ Q\ x]\!] \Longrightarrow \exists x.\ P\ x$

apply *(drule spec)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x) \vee Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule disjE)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ P\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$
2. $\bigwedge x.\ [\![\neg\ Q\ x;\ Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

apply *(erule exI)*

1. $\bigwedge x.\ [\![\neg\ Q\ x;\ Q\ (?x6\ x)]\!] \Longrightarrow \exists x.\ P\ x$

by *(erule notE)*

Substitution

lemma $"a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c"$

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

Substitution

lemma $"a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c"$

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

Substitution

lemma $"a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c"$

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply *(drule trans)*

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

Substitution

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply (*rule impI*)+

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply (*drule trans*)

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

apply *assumption*

1. $[\![b = c;\ P\ a;\ a = c]\!] \Longrightarrow P\ c$

Substitution

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply *(drule trans)*

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

apply *assumption*

1. $[\![b = c;\ P\ a;\ a = c]\!] \Longrightarrow P\ c$

by *(erule_tac s = a* in *subst)*

Substitution (rotating)

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

Substitution (rotating)

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

Substitution (rotating)

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply (rule impI)+

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply (drule trans)

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

Substitution (rotating)

lemma *"a = b ⟶ b = c ⟶ P a ⟶ P c"*

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply *(drule trans)*

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

apply *assumption*

1. $[\![b = c;\ P\ a;\ a = c]\!] \Longrightarrow P\ c$

Substitution (rotating)

lemma "$a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$"

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply (rule impI)+

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply (drule trans)

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

apply assumption

1. $[\![b = c;\ P\ a;\ a = c]\!] \Longrightarrow P\ c$

apply rotate_tac

1. $[\![P\ a;\ a = c;\ b = c]\!] \Longrightarrow P\ c$

Substitution (rotating)

lemma $"a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c"$

1. $a = b \longrightarrow b = c \longrightarrow P\ a \longrightarrow P\ c$

apply *(rule impI)+*

1. $[\![a = b;\ b = c;\ P\ a]\!] \Longrightarrow P\ c$

apply *(drule trans)*

1. $[\![b = c;\ P\ a]\!] \Longrightarrow b = ?t6$
2. $[\![b = c;\ P\ a;\ a = ?t6]\!] \Longrightarrow P\ c$

apply *assumption*

1. $[\![b = c;\ P\ a;\ a = c]\!] \Longrightarrow P\ c$

apply *rotate_tac*

1. $[\![P\ a;\ a = c;\ b = c]\!] \Longrightarrow P\ c$

by *(erule subst)*

Summary

Syntax

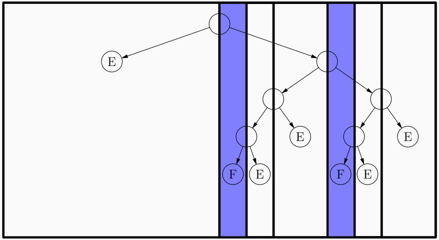
`\<longrightarrow>`	$\longrightarrow$
`\<and>`	$\wedge$
`\<or>`	$\vee$
`\<not>`	$\neg$
`\<longleftrightarrow>`	$\longleftrightarrow$
`\<forall>`	$\forall$
`\<exists>`	$\exists$

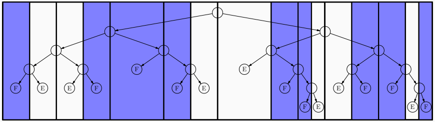
Commands apply, by, done

Methods *rule*, *erule*, *drule*, *frule*, *erule_tac s = a* in *subst*, *rotate_tac*, *assumption*, *auto*, *simp*, *insert*, *simp add: field_eq_simps*.

Binary Tree Partitioning

Binary trees can describe subsets of a line-segment by recursively dividing them in two.





Inductive Definitions

We can define this data type in terms of its possible *cases*, one of which is *recursive*:

```
datatype partition =
  Empty
| Filled
| Branch partition partition
```

A *partition* is either:

- *Empty*
- *Filled*
- A *Branch* of two partitions

Recursive Functions

We define functions on our data-type by specifying *equations* for the possible *cases* of our data. The inductive cases give rise to *recursive* equations:

fun *invert :: partition ⇒ partition* where

```
invert (Empty) = Filled
| invert (Filled)  = Empty
| invert (Branch l r)   = Branch (invert l) (invert r)
```

Case-analysis and Inductive Proofs

- *`case_tac`*: Generates a subgoal corresponding to the possible cases of a data-type
- *`induct_tac`*: Performs *structural induction*. Generates a subgoal corresponding to the possible cases, with an *inductive hypothesis* for recursive cases.

Simplification

- The `simp` command performs *equational rewriting* in an attempt to normalise terms.
- The simplifier automatically rewrites using equations from function definitions.
- You can provide the simplifier with additional equations from *lemmas* and *theorems* using `add:`
- You can also add `[simp]` after the name of a theorem in order to make it available to the simplifier everywhere

Strategies and Pitfalls

- Prefer `case_tac` to `induct_tac`. But sometimes, you *need* to use `induct_tac`.
- Try to use `induct_tac` before `allI`.
- When you get stuck, inspect the goal stack to determine appropriate *lemmas*.

Quick note on Isabelle/jEdit

If you prefer the $[\![A;B]\!] \Longrightarrow C$ style to the $A \Longrightarrow B \Longrightarrow C$, which is enabled by default in Isabelle/jEdit 2015, you can easily switch.

Add the `brackets` option to *Print Mode*, under *Plugins* > *Plugin Options* > *Isabelle* > *General* and restart Isabelle.

Information

Demonstrator Victor Dumitrescu

`victor.dumitrescu@ed.ac.uk`

Drop-in Lab Sessions Thursdays 3pm - 5pm. FH 1.B31

Submission Deadline Friday 26th February, 4pm (Week 6)

Isabelle Notes

`http://www.inf.ed.ac.uk/teaching/courses/ar/FormalCheatSheet.pdf`