HOL: Propositional Logic

Overview

- Natural deduction
- Rule application in Isabelle/HOL

Rule notation

$$\frac{A_1 \ldots A_n}{A} \qquad \text{instead of} \qquad [\![A_1 \ldots A_n]\!] \Longrightarrow A$$

Natural Deduction

Natural deduction

Two kinds of rules for each logical operator $\oplus$:

Natural deduction

Two kinds of rules for each logical operator $\oplus$:

Introduction: how can I prove $A \oplus B$?

Natural deduction

Two kinds of rules for each logical operator $\oplus$:

Introduction: how can I prove $A \oplus B$?

Elimination: what can I prove from $A \oplus B$?

Natural deduction for propositional logic

$$\frac{\quad}{A \land B}\ \texttt{conjI} \qquad \frac{A \land B}{C}\ \texttt{conjE}$$

$$\frac{\quad}{A \lor B}\ \frac{\quad}{A \lor B}\ \texttt{disjI1/2} \qquad \frac{A \lor B}{C}\ \texttt{disjE}$$

$$\frac{\quad}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{\quad}{A = B}\ \texttt{iffI} \qquad \frac{\quad}{A \Longrightarrow B}\ \texttt{iffD1} \qquad \frac{\quad}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{\quad}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad \qquad}{C}\ \texttt{conjE}$$

$$\frac{}{A \vee B} \quad \frac{}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B \quad \qquad}{C}\ \texttt{disjE}$$

$$\frac{}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B \quad \qquad}{C}\ \texttt{impE}$$

$$\frac{}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B}{C}\ \texttt{disjE}$$

$$\frac{}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B} \; \texttt{conjI} \qquad \frac{A \wedge B \qquad}{C} \; \texttt{conjE}$$

$$\frac{A}{A \vee B} \; \frac{B}{A \vee B} \; \texttt{disjI1/2} \qquad \frac{A \vee B \qquad}{C} \; \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B} \; \texttt{impI} \qquad \frac{A \longrightarrow B \qquad}{C} \; \texttt{impE}$$

$$\frac{}{A = B} \; \texttt{iffI} \qquad \frac{}{A \Longrightarrow B} \; \texttt{iffD1} \quad \frac{}{B \Longrightarrow A} \; \texttt{iffD2}$$

$$\frac{}{\neg A} \; \texttt{notI} \qquad \frac{\neg A}{C} \; \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B} \quad \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \land B}\ \texttt{conjI} \qquad \frac{A \land B}{C}\ \texttt{conjE}$$

$$\frac{A}{A \lor B}\ \frac{B}{A \lor B}\ \texttt{disjI1/2} \qquad \frac{A \lor B}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad [\![A;B]\!] \Longrightarrow C}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad [\![A;B]\!] \Longrightarrow C}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad [\![A;B]\!] \Longrightarrow C}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B} \quad \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B \quad A \quad B \Longrightarrow C}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad [\![A;B]\!] \Longrightarrow C}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B \quad A \quad B \Longrightarrow C}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{A{=}B}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{A{=}B}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A}{C}\ \texttt{notE}$$

Natural deduction for propositional logic

$$\frac{A \quad B}{A \wedge B}\ \texttt{conjI} \qquad \frac{A \wedge B \quad [\![A;B]\!] \Longrightarrow C}{C}\ \texttt{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \texttt{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \texttt{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \texttt{impI} \qquad \frac{A \longrightarrow B \quad A \quad B \Longrightarrow C}{C}\ \texttt{impE}$$

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \texttt{iffI} \qquad \frac{A{=}B}{A \Longrightarrow B}\ \texttt{iffD1} \quad \frac{A{=}B}{B \Longrightarrow A}\ \texttt{iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A}\ \texttt{notI} \qquad \frac{\neg A \quad A}{C}\ \texttt{notE}$$

Operational reading

$$\frac{A_1 \ldots A_n}{A}$$

Operational reading

$$\frac{A_1 \dots A_n}{A}$$

Introduction rule:

To prove A it suffices to prove $A_1 \dots A_n$.

Operational reading

$$\frac{A_1 \ldots A_n}{A}$$

Introduction rule:

To prove A it suffices to prove $A_1 \ldots A_n$.

Elimination rule

If I know A_1 and want to prove A
it suffices to prove $A_2 \ldots A_n$.

Equality

$$\frac{}{t = t}\ \texttt{refl} \qquad \frac{s = t}{t = s}\ \texttt{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \texttt{trans}$$

Equality

$$\frac{}{t = t}\ \texttt{refl} \qquad \frac{s = t}{t = s}\ \texttt{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \texttt{trans}$$

$$\frac{s = t \quad A(s)}{A(t)}\ \texttt{subst}$$

Equality

$$\frac{}{t = t}\ \texttt{refl} \qquad \frac{s = t}{t = s}\ \texttt{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \texttt{trans}$$

$$\frac{s = t \quad A(s)}{A(t)}\ \texttt{subst}$$

Rarely needed explicitly — used implicitly by *simp*

More rules

$$\frac{A \longrightarrow B \quad A}{B}\ \mathtt{mp}$$

More rules

$$\frac{A \longrightarrow B \quad A}{B}\ \texttt{mp}$$

$$\frac{\neg A \Longrightarrow \mathit{False}}{A}\ \texttt{ccontr} \qquad \frac{\neg A \Longrightarrow A}{A}\ \texttt{classical}$$

More rules

$$\frac{A \longrightarrow B \quad A}{B}\ \mathtt{mp}$$

$$\frac{\neg\, A \Longrightarrow \mathit{False}}{A}\ \mathtt{ccontr} \qquad \frac{\neg\, A \Longrightarrow A}{A}\ \mathtt{classical}$$

Remark:

`ccontr` and `classical` are not derivable from the ND-rules.

More rules

$$\frac{A \longrightarrow B \quad A}{B}\ \texttt{mp}$$

$$\frac{\neg A \Longrightarrow \mathit{False}}{A}\ \texttt{ccontr} \qquad \frac{\neg A \Longrightarrow A}{A}\ \texttt{classical}$$

Remark:

`ccontr` and `classical` are not derivable from the ND-rules.

They make the logic “classical”, i.e. “non-constructive”.

Proof by assumption

$$\frac{A_1 \quad \ldots \quad A_n}{A_i}\ \texttt{assumption}$$

Rule application: the rough idea

Applying rule $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$ to subgoal C:

Rule application: the rough idea

Applying rule $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C

Rule application: the rough idea

Applying rule $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C
- Replace C with n new subgoals $A_1 \ldots A_n$

Rule application: the rough idea

Applying rule $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C
- Replace C with n new subgoals $A_1 \ldots A_n$

Working backwards, like in Prolog!

Rule application: the rough idea

Applying rule $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C
- Replace C with n new subgoals $A_1 \ldots A_n$

Working backwards, like in Prolog!

Example: rule: $[\![?P; ?Q]\!] \Longrightarrow ?P \wedge ?Q$

subgoal: *1.* $A \wedge B$

Rule application: the rough idea

Applying rule $[\![A_1; \dots ; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C
- Replace C with n new subgoals $A_1 \dots A_n$

Working backwards, like in Prolog!

Example: rule: $[\![?P; ?Q]\!] \Longrightarrow ?P \wedge ?Q$

subgoal: *1.* $A \wedge B$

Result: *1.* A

2. B

Rule application: the details

Rule: $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$

Subgoal: *1.* $[\![B_1; \ldots ; B_m]\!] \Longrightarrow C$

Rule application: the details

Rule:	$[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$
Subgoal:	1. $[\![B_1; \ldots ; B_m]\!] \Longrightarrow C$
Substitution:	$\sigma(A) \equiv \sigma(C)$

Rule application: the details

Rule:	$\llbracket A_1; \ldots ; A_n \rrbracket \Longrightarrow A$
Subgoal:	1. $\llbracket B_1; \ldots ; B_m \rrbracket \Longrightarrow C$
Substitution:	$\sigma(A) \equiv \sigma(C)$
New subgoals:	1. $\sigma(\llbracket B_1; \ldots ; B_m \rrbracket \Longrightarrow A_1)$
	$\vdots$
	n. $\sigma(\llbracket B_1; \ldots ; B_m \rrbracket \Longrightarrow A_n)$

Rule application: the details

Rule: $[\![A_1; \ldots ; A_n]\!] \Longrightarrow A$

Subgoal: 1. $[\![B_1; \ldots ; B_m]\!] \Longrightarrow C$

Substitution: $\sigma(A) \equiv \sigma(C)$

New subgoals: 1. $\sigma([\![B_1; \ldots ; B_m]\!] \Longrightarrow A_1)$

$\vdots$

n. $\sigma([\![B_1; \ldots ; B_m]\!] \Longrightarrow A_n)$

Command:

apply(*rule <rulename>*)

Proof by assumption

apply *assumption*

proves

$$1.\ [\![\, B_1; \ldots ; B_m \,]\!] \Longrightarrow C$$

by unifying C with one of the B_i

Proof by assumption

apply *assumption*

proves

$$1.\ \llbracket B_1; \ldots ; B_m \rrbracket \Longrightarrow C$$

by unifying C with one of the B_i (backtracking!)

Demo: application of introduction rule

Applying elimination rules

apply*(erule <elim-rule>)*

Like *rule* but also

- unifies first premise of rule with an assumption
- eliminates that assumption

Applying elimination rules

apply(*erule <elim-rule>*)

Like *rule* but also

- unifies first premise of rule with an assumption
- eliminates that assumption

Example:

Rule: $[\![?P \wedge ?Q; [\![?P; ?Q]\!] \Longrightarrow ?R]\!] \Longrightarrow ?R$

Subgoal: $1.\ [\![\ X; A \wedge B; Y\ ]\!] \Longrightarrow Z$

Applying elimination rules

apply(*erule <elim-rule>*)

Like *rule* but also

- unifies first premise of rule with an assumption
- eliminates that assumption

Example:

Rule: $[\![?P \wedge ?Q; [\![?P; ?Q]\!] \Longrightarrow ?R]\!] \Longrightarrow ?R$

Subgoal: $1.\ [\![\ X; A \wedge B; Y\ ]\!] \Longrightarrow Z$

Unification: $?P \wedge ?Q \equiv A \wedge B$ and $?R \equiv Z$

Applying elimination rules

apply(*erule <elim-rule>*)

Like *rule* but also

- unifies first premise of rule with an assumption
- eliminates that assumption

Example:

Rule:	$[\![?P \wedge ?Q; [\![?P; ?Q]\!] \Longrightarrow ?R]\!] \Longrightarrow ?R$
Subgoal:	1. $[\![X; A \wedge B; Y]\!] \Longrightarrow Z$
Unification:	$?P \wedge ?Q \equiv A \wedge B$ and $?R \equiv Z$
New subgoal:	1. $[\![X; Y]\!] \Longrightarrow [\![A; B]\!] \Longrightarrow Z$

Applying elimination rules

apply(*erule <elim-rule>*)

Like *rule* but also

- unifies first premise of rule with an assumption
- eliminates that assumption

Example:

Rule: $[\![?P \wedge ?Q; [\![?P; ?Q]\!] \Longrightarrow ?R]\!] \Longrightarrow ?R$

Subgoal: $1.\ [\![X; A \wedge B; Y]\!] \Longrightarrow Z$

Unification: $?P \wedge ?Q \equiv A \wedge B$ and $?R \equiv Z$

New subgoal: $1.\ [\![X; Y]\!] \Longrightarrow [\![A; B]\!] \Longrightarrow Z$

same as: $1.\ [\![X; Y; A; B]\!] \Longrightarrow Z$

How to prove it by natural deduction

- Intro rules decompose formulae to the right of $\Longrightarrow$.

  apply*(rule <intro-rule>)*

How to prove it by natural deduction

- Intro rules decompose formulae to the right of $\Longrightarrow$.

  apply*(rule <intro-rule>)*

- Elim rules decompose formulae on the left of $\Longrightarrow$.

  apply*(erule <elim-rule>)*

Demo: examples

$\Longrightarrow$ vs $\longrightarrow$

- Write theorems as $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$
  not as $A_1 \wedge \ldots \wedge A_n \longrightarrow A$ (to ease application)

$\Longrightarrow$ vs $\longrightarrow$

- Write theorems as $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ not as $A_1 \wedge \ldots \wedge A_n \longrightarrow A$ (to ease application)
- *Exception* (in **apply**-style): induction variable must not occur in the premises.

$\Longrightarrow$ ***vs*** $\longrightarrow$

- Write theorems as $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$
  not as $A_1 \wedge \ldots \wedge A_n \longrightarrow A$ (to ease application)
- *Exception* (in **apply**-style): induction variable must not occur in the premises.

  Example: $[\![A; B(x)\,]\!] \Longrightarrow C(x) \leadsto A \Longrightarrow\ B(x) \longrightarrow C(x)$

$\Longrightarrow$ **vs** $\longrightarrow$

- Write theorems as $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ not as $A_1 \wedge \ldots \wedge A_n \longrightarrow A$ (to ease application)
- *Exception* (in **apply**-style): induction variable must not occur in the premises.

  Example: $[\![A; B(x)]\!] \Longrightarrow C(x) \leadsto A \Longrightarrow B(x) \longrightarrow C(x)$

  Reverse transformation (after proof):
  lemma *abc[rule_format]: A* $\Longrightarrow$ *B(x)* $\longrightarrow$ *C(x)*

Demo: further techniques

HOL: Predicate Logic

Parameters

Subgoal:

1. $\bigwedge x_1 \ldots x_n$. *Formula*

The x_i are called parameters of the subgoal.
Intuition: local constants, i.e. arbitrary but fixed values.

Parameters

Subgoal:

1. $\bigwedge x_1 \ldots x_n$. *Formula*

The x_i are called parameters of the subgoal.
Intuition: local constants, i.e. arbitrary but fixed values.

Rules are automatically lifted over $\bigwedge x_1 \ldots x_n$ and applied directly to *Formula*.

Scope

- Scope of parameters: whole subgoal
- Scope of $\forall$, $\exists$, $\ldots$: ends with *;* or $\Longrightarrow$

Scope

- Scope of parameters: whole subgoal
- Scope of $\forall$, $\exists$, ...: ends with *;* or $\Longrightarrow$

$$\bigwedge x\ y.\ [\![\ \forall y.\ P\ y \longrightarrow Q\ z\ y;\ \ Q\ x\ y\ ]\!] \Longrightarrow \exists x.\ Q\ x\ y$$

means

$$\bigwedge x\ y.\ [\![\ (\forall y_1.\ P\ y_1 \longrightarrow Q\ z\ y_1);\ \ Q\ x\ y\ ]\!] \Longrightarrow \exists x_1.\ Q\ x_1\ y$$

α-Conversion

- $\forall x.\ P(x)$: x can appear in $P(x)$.

α-Conversion

- $\forall x.\ P(x)$: x can appear in $P(x)$.

  Example: $\forall x.\ x = x$ is obtained by $P \mapsto \lambda u.\ u = u$

α-Conversion

- $\forall x.\ P(x)$: x can appear in $P(x)$.

  Example: $\forall x.\ x = x$ is obtained by $P \mapsto \lambda u.\ u = u$
- $\forall x.\ P$: x cannot appear in P.

α-Conversion

- $\forall x.\ P(x)$: x can appear in $P(x)$.

  Example: $\forall x.\ x = x$ is obtained by $P \mapsto \lambda u.\ u = u$
- $\forall x.\ P$: x cannot appear in P.

  Example: $P \mapsto x = x$ yields $\forall x'.\ x = x$

α-Conversion

- $\forall x.\ P(x)$: x can appear in $P(x)$.

  Example: $\forall x.\ x = x$ is obtained by $P \mapsto \lambda u.\ u = u$

- $\forall x.\ P$: x cannot appear in P.

  Example: $P \mapsto x = x$ yields $\forall x'.\ x = x$

Bound variables are renamed automatically to avoid name clashes with other variables.

Natural deduction for quantifiers

$$\frac{}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x)}{R}\ \texttt{allE}$$

$$\frac{}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x)}{R}\ \texttt{exE}$$

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x)}{R}\ \texttt{allE}$$

$$\frac{}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x)}{R}\ \texttt{exE}$$

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x)}{R}\ \texttt{allE}$$

$$\frac{P(?x)}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x)}{R}\ \texttt{exE}$$

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x) \quad P(?x) \Longrightarrow R}{R}\ \texttt{allE}$$

$$\frac{P(?x)}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x)}{R}\ \texttt{exE}$$

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x) \quad P(?x) \Longrightarrow R}{R}\ \texttt{allE}$$

$$\frac{P(?x)}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x) \quad \bigwedge x.\ P(x) \Longrightarrow R}{R}\ \texttt{exE}$$

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x) \quad P(?x) \Longrightarrow R}{R}\ \texttt{allE}$$

$$\frac{P(?x)}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x) \quad \bigwedge x.\ P(x) \Longrightarrow R}{R}\ \texttt{exE}$$

- `allI` and `exE` introduce new parameters ($\bigwedge x$).

Natural deduction for quantifiers

$$\frac{\bigwedge x.\ P(x)}{\forall x.\ P(x)}\ \texttt{allI} \qquad \frac{\forall x.\ P(x) \quad P(?x) \Longrightarrow R}{R}\ \texttt{allE}$$

$$\frac{P(?x)}{\exists x.\ P(x)}\ \texttt{exI} \qquad \frac{\exists x.\ P(x) \quad \bigwedge x.\ P(x) \Longrightarrow R}{R}\ \texttt{exE}$$

- `allI` and `exE` introduce new parameters ($\bigwedge x$).
- `allE` and `exI` introduce new unknowns ($?x$).

Instantiating rules

apply(*rule_tac x* = $term$ *in* $rule$)

Like *rule*, but *?x* in $rule$ is instantiated by $term$ before application.

Instantiating rules

apply(*rule_tac x* = $term$ *in* $rule$)

Like *rule*, but *?x* in $rule$ is instantiated by $term$ before application.

Similar: *erule_tac*

Instantiating rules

apply(*rule_tac x* = $term$ *in* $rule$)

Like *rule*, but *?x* in $rule$ is instantiated by $term$ before application.

Similar: *erule_tac*

! *x* is in $rule$, not in the goal **!**

Two successful proofs

1. $\forall x.\ \exists y.\ x = y$

Two successful proofs

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

Two successful proofs

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1. $\bigwedge x.\ \exists y.\ x = y$

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

apply(*rule refl*)

Two successful proofs

1\. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1\. $\bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

1\. $\bigwedge x.\ x = x$

apply(*rule refl*)

exploration

apply(*rule exI*)

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

apply(*rule refl*)

exploration

apply(*rule exI*)

$1.\ \bigwedge x.\ x = ?y\ x$

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

apply(*rule refl*)

exploration

apply(*rule exI*)

$1.\ \bigwedge x.\ x = ?y\ x$

apply(*rule refl*)

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

apply(*rule refl*)

exploration

apply(*rule exI*)

$1.\ \bigwedge x.\ x = ?y\ x$

apply(*rule refl*)

$?y \mapsto$

Two successful proofs

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1. $\bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

1. $\bigwedge x.\ x = x$

apply(*rule refl*)

exploration

apply(*rule exI*)

1. $\bigwedge x.\ x = ?y\ x$

apply(*rule refl*)

$?y \mapsto \lambda u.\ u$

Two successful proofs

$1.\ \forall x.\ \exists y.\ x = y$

apply(*rule allI*)

$1.\ \bigwedge x.\ \exists y.\ x = y$

best practice

apply(*rule_tac x = x in exI*)

$1.\ \bigwedge x.\ x = x$

apply(*rule refl*)

simpler & clearer

exploration

apply(*rule exI*)

$1.\ \bigwedge x.\ x = ?y\ x$

apply(*rule refl*)

$?y \mapsto \lambda u.\ u$

shorter & trickier

Two unsuccessful proofs

1. $\exists y.\ \forall x.\ x = y$

Two unsuccessful proofs

1. $\exists y.\ \forall x.\ x = y$

apply(*rule_tac x = ??? in exI*)

Two unsuccessful proofs

1. $\exists y.\ \forall x.\ x = y$

apply(*rule_tac x = ??? in exI*)

apply(*rule exI*)

1. $\forall x.\ x = ?y$

Two unsuccessful proofs

$1.\ \exists y.\ \forall x.\ x = y$

apply$(rule\_tac\ x = ???\ in\ exI)$

apply$(rule\ exI)$

$1.\ \forall x.\ x = ?y$

apply$(rule\ allI)$

$1.\ \bigwedge x.\ x = ?y$

Two unsuccessful proofs

$1.\ \exists y.\ \forall x.\ x = y$

apply(*rule_tac x = ??? in exI*)

apply(*rule exI*)

$1.\ \forall x.\ x = ?y$

apply(*rule allI*)

$1.\ \bigwedge x.\ x = ?y$

apply(*rule refl*)

Two unsuccessful proofs

$1.\ \exists y.\ \forall x.\ x = y$

apply(*rule_tac x = ??? in exI*)

apply(*rule exI*)

$1.\ \forall x.\ x = ?y$

apply(*rule allI*)

$1.\ \bigwedge x.\ x = ?y$

apply(*rule refl*)

$?y \mapsto x$ yields $\bigwedge x'.\ x' = x$

Two unsuccessful proofs

1. $\exists y.\ \forall x.\ x = y$

apply(*rule_tac x = ??? in exI*)

apply(*rule exI*)

1. $\forall x.\ x = ?y$

apply(*rule allI*)

1. $\bigwedge x.\ x = ?y$

apply(*rule refl*)

$?y \mapsto x$ yields $\bigwedge x'.\ x' = x$

Principle:

$?f\ x_1 \ldots x_n$ can only be replaced by term t if $params(t) \subseteq \{x_1, \ldots, x_n\}$

Demo: quantifier proofs

Proof methods

Parameter names

Parameter names are chosen by Isabelle

Parameter names

Parameter names are chosen by Isabelle

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1. $\bigwedge x.\ \exists y.\ x = y$

apply(*rule_tac x = x in exI*)

Parameter names

Parameter names are chosen by Isabelle

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1. $\bigwedge x.\ \exists y.\ x = y$

apply(*rule_tac x = x in exI*)

Brittle!

Renaming parameters

1. $\forall x.\ \exists y.\ x = y$

apply(*rule allI*)

1. $\bigwedge x.\ \exists y.\ x = y$

apply(*rename_tac xxx*)

1. $\bigwedge xxx.\ \exists y.\ xxx = y$

apply(*rule_tac x = xxx in exI*)

Renaming parameters

1. $\forall$ *x.* $\exists$ *y. x = y*

apply*(rule allI)*

1. $\bigwedge$*x.* $\exists$ *y. x = y*

apply*(rename_tac xxx)*

1. $\bigwedge$*xxx.* $\exists$ *y. xxx = y*

apply*(rule_tac x = xxx in exI)*

In general:

(rename_tac $x_1 \ldots x_n$*)* renames the rightmost (inner) n parameters to $x_1 \ldots x_n$

Forward proofs: frule and drule

“Forward” rule: $A_1 \Longrightarrow A$

Subgoal: *1.* $[\![B_1; \ldots ; B_n]\!] \Longrightarrow C$

Forward proofs: frule and drule

“Forward” rule: $A_1 \Longrightarrow A$

Subgoal: *1.* $[\![B_1; \ldots ; B_n]\!] \Longrightarrow C$

Substitution: $\sigma(B_i) \equiv \sigma(A_1)$

Forward proofs: frule and drule

"Forward" rule:	$A_1 \Longrightarrow A$
Subgoal:	1. $[\![B_1; \ldots ; B_n]\!] \Longrightarrow C$
Substitution:	$\sigma(B_i) \equiv \sigma(A_1)$
New subgoal:	1. $\sigma([\![B_1; \ldots ; B_n; A]\!] \Longrightarrow C)$

Forward proofs: frule and drule

"Forward" rule:	$A_1 \Longrightarrow A$
Subgoal:	1. $[\![B_1; \ldots ; B_n]\!] \Longrightarrow C$
Substitution:	$\sigma(B_i) \equiv \sigma(A_1)$
New subgoal:	1. $\sigma([\![B_1; \ldots ; B_n; A]\!] \Longrightarrow C)$

Command:

apply(*frule* $rulename$)

Forward proofs: frule and drule

"Forward" rule:	$A_1 \Longrightarrow A$
Subgoal:	1. $[\![B_1; \ldots ; B_n]\!] \Longrightarrow C$
Substitution:	$\sigma(B_i) \equiv \sigma(A_1)$
New subgoal:	1. $\sigma([\![B_1; \ldots ; B_n; A]\!] \Longrightarrow C)$

Command:

apply(*frule* $rulename$)

Like *frule* but also deletes B_i:

apply(*drule* $rulename$)

frule and drule: the general case

Rule: $[\![A_1; \ldots ; A_m]\!] \Longrightarrow A$

Creates additional subgoals:

1. $\sigma([\![B_1; \ldots ; B_n]\!] \Longrightarrow A_2)$

$\vdots$

m-1. $\sigma([\![B_1; \ldots ; B_n]\!] \Longrightarrow A_m)$

m. $\sigma([\![B_1; \ldots ; B_n; A]\!] \Longrightarrow C)$

Forward proofs: OF

r[OF $r_1 \ldots r_n$*]*

Prove assumption 1 of theorem *r* with theorem r_1, and assumption 2 with theorem r_2, and ...

Forward proofs: OF

$r[OF\ r_1 \ldots r_n]$

Prove assumption 1 of theorem r with theorem r_1, and assumption 2 with theorem r_2, and ...

Rule r	$[\![A_1; \ldots ; A_m]\!] \Longrightarrow A$
Rule r_1	$[\![B_1; \ldots ; B_n]\!] \Longrightarrow B$
Substitution	$\sigma(B) \equiv \sigma(A_1)$
$r[OF\ r_1]$	

Forward proofs: OF

r[OF $r_1 \ldots r_n$]

Prove assumption 1 of theorem r with theorem r_1, and assumption 2 with theorem r_2, and ...

Rule r	$[\![A_1; \ldots ; A_m]\!] \Longrightarrow A$
Rule r_1	$[\![B_1; \ldots ; B_n]\!] \Longrightarrow B$
Substitution	$\sigma(B) \equiv \sigma(A_1)$
r[OF r_1]	$\sigma([\![B_1; \ldots; B_n; A_2; \ldots; A_m]\!] \Longrightarrow A)$

Forward proofs: THEN

r_1[THEN r_2] means r_2[OF r_1]

Clarifying the goal

Clarifying the goal

- **apply**(*intro* ...)
  Repeated application of intro rules
  Example: **apply**(*intro allI*)

Clarifying the goal

- **apply**(*intro* ...)
  Repeated application of intro rules
  Example: **apply**(*intro allI*)
- **apply**(*elim* ...)
  Repeated application of elim rules
  Example: **apply**(*elim conjE*)

Clarifying the goal

- **apply**(*intro* . . .)
  Repeated application of intro rules
  Example: **apply**(*intro allI*)
- **apply**(*elim* . . .)
  Repeated application of elim rules
  Example: **apply**(*elim conjE*)
- **apply**(*clarify*)
  Repeated application of safe rules
  without splitting the goal

Clarifying the goal

- **apply**(*intro* . . .)
  Repeated application of intro rules
  Example: **apply**(*intro allI*)
- **apply**(*elim* . . .)
  Repeated application of elim rules
  Example: **apply**(*elim conjE*)
- **apply**(*clarify*)
  Repeated application of safe rules
  without splitting the goal
- **apply**(*clarsimp simp add:* . . .)
  Combination of *clarify* and *simp*.

Demo: proof methods