NICTA Advanced Course

Theorem Proving

Principles, Techniques, Applications

HOL

Content

- Intro & motivation, getting started with Isabelle
- **Foundations & Principles**
  - Lambda Calculus
  - **Higher Order Logic, natural deduction**
  - Term rewriting
- Proof & Specification Techniques
  - Datatypes, recursion, induction
  - Inductively defined sets, rule induction
  - Calculational reasoning, mathematics style proofs
  - Hoare logic, proofs about programs

LAST TIME ON HOL

➜ Proof rules for propositional and predicate logic

LAST TIME ON HOL

- ➜ Proof rules for propositional and predicate logic
- ➜ Safe and unsafe rules

LAST TIME ON HOL

- ➜ Proof rules for propositional and predicate logic
- ➜ Safe and unsafe rules
- ➜ Forward Proof

LAST TIME ON HOL

- ➜ Proof rules for propositional and predicate logic
- ➜ Safe and unsafe rules
- ➜ Forward Proof
- ➜ The Epsilon Operator

Last Time on HOL

➜ Proof rules for propositional and predicate logic

➜ Safe and unsafe rules

➜ Forward Proof

➜ The Epsilon Operator

➜ Some automation

Defining Higher Order Logic

WHAT IS HIGHER ORDER LOGIC?

➜ **Propositional Logic:**

- no quantifiers
- all variables have type bool

What is Higher Order Logic?

➜ **Propositional Logic:**

- no quantifiers
- all variables have type bool

➜ **First Order Logic:**

- quantification over values, but not over functions and predicates,
- terms and formulas syntactically distinct

What is Higher Order Logic?

➜ **Propositional Logic:**

- no quantifiers
- all variables have type bool

➜ **First Order Logic:**

- quantification over values, but not over functions and predicates,
- terms and formulas syntactically distinct

➜ **Higher Order Logic:**

- quantification over everything, including predicates
- consistency by types
- formula = term of type bool
- definition built on $\lambda^{\rightarrow}$ with certain default types and constants

Default types:

bool

Defining Higher Order Logic

Default types:

bool $\quad\quad\quad$ $\_ \Rightarrow \_$

DEFINING HIGHER ORDER LOGIC

Default types:

bool $\quad$ $\_ \Rightarrow \_$ $\quad$ ind

Defining Higher Order Logic

Default types:

bool $\qquad\qquad$ $\_ \Rightarrow \_$ $\qquad\qquad$ ind

- ➜ **bool** sometimes called o
- ➜ $\Rightarrow$ sometimes called fun

DEFINING HIGHER ORDER LOGIC

Default types:

bool $\quad\quad _{-} \Rightarrow _{-} \quad\quad$ ind

- **bool** sometimes called o
- $\Rightarrow$ sometimes called fun

Default Constants:

Defining Higher Order Logic

Default types:

bool $\quad\quad\quad$ $\_ \Rightarrow \_$ $\quad\quad\quad$ ind

- **bool** sometimes called o
- $\Rightarrow$ sometimes called fun

Default Constants:

$$\longrightarrow \quad :: \quad bool \Rightarrow bool \Rightarrow bool$$

Default types:

bool $\qquad\qquad$ $\_ \Rightarrow \_$ $\qquad\qquad$ ind

- ➜ **bool** sometimes called o
- ➜ $\Rightarrow$ sometimes called fun

Default Constants:

$$\begin{array}{lll} \longrightarrow & :: & bool \Rightarrow bool \Rightarrow bool \\ = & :: & \alpha \Rightarrow \alpha \Rightarrow bool \end{array}$$

Defining Higher Order Logic

Default types:

bool $\qquad$ $\_ \Rightarrow \_$ $\qquad$ ind

- **bool** sometimes called o
- $\Rightarrow$ sometimes called fun

Default Constants:

$$
\begin{array}{lll}
\longrightarrow & :: & bool \Rightarrow bool \Rightarrow bool \\
= & :: & \alpha \Rightarrow \alpha \Rightarrow bool \\
\epsilon & :: & (\alpha \Rightarrow bool) \Rightarrow \alpha
\end{array}
$$

HIGHER ORDER ABSTRACT SYNTAX

Problem: Define syntax for binders like $\forall$, $\exists$, ε

Higher Order Abstract Syntax

Problem: Define syntax for binders like $\forall$, $\exists$, ε

One approach: $\forall :: var \Rightarrow term \Rightarrow bool$
Drawback: need to think about substitution, α conversion again.

Higher Order Abstract Syntax

Problem: Define syntax for binders like $\forall$, $\exists$, ε

One approach: $\forall :: var \Rightarrow term \Rightarrow bool$
Drawback: need to think about substitution, α conversion again.

But: Already have binder, substitution, α conversion in meta logic

$$\lambda$$

Higher Order Abstract Syntax

Problem: Define syntax for binders like $\forall$, $\exists$, ε

One approach: $\forall :: var \Rightarrow term \Rightarrow bool$
Drawback: need to think about substitution, α conversion again.

But: Already have binder, substitution, α conversion in meta logic

$$\lambda$$

So: Use λ to encode all other binders.

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**

HIGHER ORDER ABSTRACT SYNTAX

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
$\text{ALL}\ (\lambda x.\ x = 2)$	

Higher Order Abstract Syntax

Example:

$$\mathsf{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
$\mathsf{ALL}\ (\lambda x.\ x = 2)$	$\forall x.\ x = 2$

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
ALL $(\lambda x.\ x = 2)$	$\forall x.\ x = 2$
ALL P	

Higher Order Abstract Syntax

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
ALL $(\lambda x.\ x = 2)$	$\forall x.\ x = 2$
ALL P	$\forall x.\ P\ x$

Higher Order Abstract Syntax

Example:

$$\text{ALL} :: (\alpha \Rightarrow bool) \Rightarrow bool$$

HOAS	**usual syntax**
$\text{ALL}\ (\lambda x.\ x = 2)$	$\forall x.\ x = 2$
$\text{ALL}\ P$	$\forall x.\ P\ x$

Isabelle can translate usual binder syntax into HOAS.

Side Track: Syntax Declarations in Isabelle

➜ **mixfix:**

consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(\text{"}\_, \_ \vdash \ \_\text{"})$

Legal syntax now: $\Gamma, \Pi \vdash F$

SIDE TRACK: SYNTAX DECLARATIONS IN ISABELLE

➜ **mixfix:**

consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(" \_, \_ \vdash \ \_ ")$

Legal syntax now: $\Gamma, \Pi \vdash F$

➜ **priorities:**

pattern can be annotated with priorities to indicate binding strength

Example: drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(" \_, \_ \vdash \ \_ " \ [30, 0, 20] \ 60)$

SIDE TRACK: SYNTAX DECLARATIONS IN ISABELLE

- **mixfix:**

  consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(\text{"}\_, \_ \vdash \ \_\text{"})$

  Legal syntax now: $\Gamma, \Pi \vdash F$

- **priorities:**

  pattern can be annotated with priorities to indicate binding strength

  Example: drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(\text{"}\_, \_ \vdash \ \_\text{"}\ [30, 0, 20]\ 60)$

- **infixl/infixr**: short form for left/right associative binary operators

  Example: or :: $bool \Rightarrow bool \Rightarrow bool$ $(\text{infixr "} \vee \text{" } 30)$

SIDE TRACK: SYNTAX DECLARATIONS IN ISABELLE

➜ **mixfix:**

consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(\text{"}\_, \_ \vdash \ \_\text{"})$

Legal syntax now: $\Gamma, \Pi \vdash F$

➜ **priorities:**

pattern can be annotated with priorities to indicate binding strength

Example: drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(\text{"}\_, \_ \vdash \ \_\text{"}\ [30, 0, 20]\ 60)$

➜ **infixl/infixr**: short form for left/right associative binary operators

Example: or :: $bool \Rightarrow bool \Rightarrow bool$ (infixr " $\vee$ " 30)

➜ **binders:** declaration must be of the form

$c :: (\tau_1 \Rightarrow \tau_2) \Rightarrow \tau_3$ (binder "B" $< p >$)

$B\ x.\ P\ x$ translated into $c\ P$ (and vice versa)

Example ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool$ (binder "$\forall$" 10)

Side Track: Syntax Declarations in Isabelle

➜ **mixfix:**
consts drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ ("_, _ ⊢ _")
Legal syntax now: $\Gamma, \Pi \vdash F$

➜ **priorities:**
pattern can be annotated with priorities to indicate binding strength
Example: drvbl :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ ("_, _ ⊢ _" $[30, 0, 20]$ 60)

➜ **infixl/infixr**: short form for left/right associative binary operators
Example: or :: $bool \Rightarrow bool \Rightarrow bool$ (infixr " ∨ " 30)

➜ **binders:** declaration must be of the form
$c :: (\tau_1 \Rightarrow \tau_2) \Rightarrow \tau_3$ (binder "B" $< p >$)
$B\ x.\ P\ x$ translated into $c\ P$ (and vice versa)
Example ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool$ (binder "∀" 10)

More (including pretty printing) in Isabelle Reference Manual (7.3)

BACK TO HOL

Base: $bool, \Rightarrow, ind$ $\quad =, \longrightarrow, \varepsilon$

And the rest is

Back to HOL

Base: $bool$, $\Rightarrow$, ind $\quad$ $=$, $\longrightarrow$, ε

And the rest is definitions:

$$
\begin{array}{ll}
\text{True} & \equiv (\lambda x :: bool.\ x) = (\lambda x.\ x) \\
\text{All } P & \equiv P = (\lambda x.\ \text{True}) \\
\text{Ex } P & \equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q \\
\text{False} & \equiv \forall P.\ P \\
\neg P & \equiv P \longrightarrow \text{False} \\
P \wedge Q & \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R \\
P \vee Q & \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R \\
\text{If } P\ x\ y & \equiv \text{SOME } z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y) \\
\text{inj } f & \equiv \forall x\ y.\ f\ x = f\ y \longrightarrow x = y \\
\text{surj } f & \equiv \forall y.\ \exists x.\ y = f\ x
\end{array}
$$

The Axioms of HOL

$$\frac{}{t = t} \text{ refl} \qquad \frac{s = t \quad P\ s}{P\ t} \text{ subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)} \text{ ext}$$

THE AXIOMS OF HOL

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \text{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \text{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \text{mp}$$

The Axioms of HOL

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \text{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \text{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \text{mp}$$

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)}\ \text{iff}$$

The Axioms of HOL

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \text{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \text{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \text{mp}$$

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)}\ \text{iff}$$

$$\frac{}{P = \mathsf{True} \lor P = \mathsf{False}}\ \text{True\_or\_False}$$

The Axioms of HOL

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \text{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \text{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \text{mp}$$

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)}\ \text{iff}$$

$$\frac{}{P = \mathsf{True} \vee P = \mathsf{False}}\ \text{True\_or\_False}$$

$$\frac{P\ ?x}{P\ (\mathsf{SOME}\ x.\ P\ x)}\ \text{someI}$$

The Axioms of HOL

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t \quad P\ s}{P\ t}\ \text{subst} \qquad \frac{\bigwedge x.\ f\ x = g\ x}{(\lambda x.\ f\ x) = (\lambda x.\ g\ x)}\ \text{ext}$$

$$\frac{P \Longrightarrow Q}{P \longrightarrow Q}\ \text{impI} \qquad \frac{P \longrightarrow Q \quad P}{Q}\ \text{mp}$$

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)}\ \text{iff}$$

$$\frac{}{P = \mathsf{True} \vee P = \mathsf{False}}\ \text{True\_or\_False}$$

$$\frac{P\ ?x}{P\ (\mathsf{SOME}\ x.\ P\ x)}\ \text{someI}$$

$$\frac{}{\exists f :: ind \Rightarrow ind.\ \mathsf{inj}\ f \wedge \neg \mathsf{surj}\ f}\ \text{infty}$$

THAT'S IT.

- ➜ 3 basic constants
- ➜ 3 basic types
- ➜ 9 axioms

THAT'S IT.

➜ 3 basic constants

➜ 3 basic types

➜ 9 axioms

With this you can define and derive all the rest.

THAT'S IT.

- ➜ 3 basic constants
- ➜ 3 basic types
- ➜ 9 axioms

With this you can define and derive all the rest.

Isabelle knows 2 more axioms:

$$\frac{x = y}{x \equiv y} \text{ eq\_reflection} \qquad \frac{}{(\text{THE } x.\ x = a) = a} \text{ the\_eq\_trivial}$$

DEMO: THE DEFINITIONS IN ISABELLE

Deriving Proof Rules

In the following, we will

DERIVING PROOF RULES

In the following, we will

➜ look at the definitions in more detail

DERIVING PROOF RULES

In the following, we will

- ➜ look at the definitions in more detail
- ➜ derive the traditional proof rules from the axioms in Isabelle

DERIVING PROOF RULES

In the following, we will

➜ look at the definitions in more detail

➜ derive the traditional proof rules from the axioms in Isabelle

Convenient for deriving rules: **named assumptions in lemmas**

lemma $[name :]$
assumes $[name_1 :]$ "$< prop >_1$"
assumes $[name_2 :]$ "$< prop >_2$"
⋮
shows $" < prop > " \quad < proof >$

Deriving Proof Rules

In the following, we will

- ➜ look at the definitions in more detail
- ➜ derive the traditional proof rules from the axioms in Isabelle

Convenient for deriving rules: **named assumptions in lemmas**

lemma $[name :]$
assumes $[name_1 :]$ "$< prop >_1$"
assumes $[name_2 :]$ "$< prop >_2$"
$\vdots$
shows "$< prop >$" $< proof >$

proves: $[\![< prop >_1; < prop >_2; \ldots]\!] \Longrightarrow < prop >$

TRUE

consts True $:: bool$

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

Intuition:

right hand side is always true

TRUE

consts True :: $bool$

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

Intuition:

right hand side is always true

Proof Rules:

$$\frac{}{\text{True}}\ \text{TrueI}$$

Proof:

$$\cfrac{\cfrac{}{(\lambda x :: bool.\ x) = (\lambda x.\ x)}\ \text{refl}}{\text{True}}\ \text{unfold True\_def}$$

DEMO

Universal Quantifier

consts ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \quad \equiv \quad P = (\lambda x.\ \mathsf{True})$

Universal Quantifier

consts ALL $:: (\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \quad \equiv \quad P = (\lambda x.$ True$)$

Intuition:

- ALL P is Higher Order Abstract Syntax for $\forall x.\ P\ x$.

UNIVERSIAL QUANTIFIER

consts ALL $:: (\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \quad \equiv \quad P = (\lambda x.\ \text{True})$

Intuition:

- ➜ ALL P is Higher Order Abstract Syntax for $\forall x.\ P\ x$.
- ➜ P is a function that takes an x and yields a truth values.

Universal Quantifier

consts ALL $:: (\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \equiv P = (\lambda x.$ True$)$

Intuition:

- ➜ ALL P is Higher Order Abstract Syntax for $\forall x.\ P\ x$.
- ➜ P is a function that takes an x and yields a truth values.
- ➜ ALL P should be true iff P yields true for all x, i.e.
  if it is equivalent to the function $\lambda x.$ True.

Universial Quantifier

consts ALL $:: (\alpha \Rightarrow bool) \Rightarrow bool$

ALL $P \quad \equiv \quad P = (\lambda x.\ \text{True})$

Intuition:

- ➜ ALL P is Higher Order Abstract Syntax for $\forall x.\ P\ x$.
- ➜ P is a function that takes an x and yields a truth values.
- ➜ ALL P should be true iff P yields true for all x, i.e.
  if it is equivalent to the function $\lambda x.$ True.

Proof Rules:

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

Proof: Isabelle Demo

FALSE

consts False $:: bool$

False $\equiv \forall P. P$

FALSE

consts False :: $bool$

False $\equiv$ $\forall P. P$

Intuition:

Everything can be derived from *False*.

FALSE

consts False :: $bool$

False $\equiv$ $\forall P. P$

Intuition:

Everything can be derived from *False*.

Proof Rules:

$$\frac{\text{False}}{P}\ \text{FalseE} \qquad \frac{}{\text{True} \neq \text{False}}$$

Proof: Isabelle Demo

NEGATION

consts Not :: $bool \Rightarrow bool$ $(\neg\ \_)$

$\neg P \equiv P \longrightarrow$ False

NEGATION

consts Not $:: bool \Rightarrow bool$ $(\neg\ \_)$

$\neg P \equiv P \longrightarrow \mathsf{False}$

Intuition:

Try $P = True$ and $P = False$ and the traditional truth table for $\longrightarrow$.

NEGATION

consts Not :: $bool \Rightarrow bool$ $(\neg$ _$)$

$\neg P \equiv P \longrightarrow$ False

Intuition:

Try $P = True$ and $P = False$ and the traditional truth table for $\longrightarrow$.

Proof Rules:

$$\frac{A \Longrightarrow False}{\neg A}\ \text{notI} \qquad \frac{\neg A \quad A}{P}\ \text{notE}$$

Proof: Isabelle Demo

Existential Quantifier

consts EX $:: (\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \quad \equiv \quad \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Existential Quantifier

consts EX $:: (\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \quad \equiv \quad \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)

EXISTENTIAL QUANTIFIER

consts EX $:: (\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \quad \equiv \quad \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)
- Right hand side is characterization of $\exists$ with $\forall$ and $\longrightarrow$

EXISTENTIAL QUANTIFIER

consts EX :: $(\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \;\; \equiv \;\; \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)
- Right hand side is characterization of $\exists$ with $\forall$ and $\longrightarrow$
- Note that inner $\forall$ binds wide: $(\forall x.\ P\ x \longrightarrow Q)$

EXISTENTIAL QUANTIFIER

consts EX $:: (\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \;\; \equiv \;\; \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)
- Right hand side is characterization of $\exists$ with $\forall$ and $\longrightarrow$
- Note that inner $\forall$ binds wide: $(\forall x.\ P\ x \longrightarrow Q)$
- Remember lemma from last time:
  $(\forall x.\ P\ x \longrightarrow Q) = ((\exists x.\ P\ x) \longrightarrow Q)$

Existential Quantifier

consts EX :: $(\alpha \Rightarrow bool) \Rightarrow bool$

EX $P \quad \equiv \quad \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

Intuition:

- EX P is HOAS for $\exists x.\ P\ x$. (like $\forall$)
- Right hand side is characterization of $\exists$ with $\forall$ and $\longrightarrow$
- Note that inner $\forall$ binds wide: $(\forall x.\ P\ x \longrightarrow Q)$
- Remember lemma from last time:
  $(\forall x.\ P\ x \longrightarrow Q) = ((\exists x.\ P\ x) \longrightarrow Q)$

Proof Rules:

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

Proof: Isabelle Demo

Conjunction

consts And $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \wedge \_)$

$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

CONJUNCTION

consts And $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \wedge \_)$

$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

Intuition:

➜ Mirrors proof rules for $\wedge$

CONJUNCTION

consts And $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \wedge \_)$

$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

Intuition:

- ➜ Mirrors proof rules for $\wedge$
- ➜ Try truth table for P, Q, and R

CONJUNCTION

consts And $:: bool \Rightarrow bool \Rightarrow bool\ (\_ \wedge \_)$

$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

Intuition:

- ➜ Mirrors proof rules for $\wedge$
- ➜ Try truth table for P, Q, and R

Proof Rules:

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

Proof: Isabelle Demo

DISJUNCTION

consts Or $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \vee \_)$

$P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

DISJUNCTION

consts Or $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \vee \_)$

$P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

Intuition:

➜ Mirrors proof rules for $\vee$ (case distinction)

DISJUNCTION

consts Or :: $bool \Rightarrow bool \Rightarrow bool$ $(\_ \vee \_)$

$P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

Intuition:

- ➜ Mirrors proof rules for $\vee$ (case distinction)
- ➜ Try truth table for P, Q, and R

DISJUNCTION

consts Or $:: bool \Rightarrow bool \Rightarrow bool$ $(\_ \lor \_)$

$P \lor Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

Intuition:

- ➜ Mirrors proof rules for $\lor$ (case distinction)
- ➜ Try truth table for P, Q, and R

Proof Rules:

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

Proof: Isabelle Demo

IF-THEN-ELSE

consts If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if _ then _ else _)

If $P\ x\ y \ \equiv\ $ SOME $z.\ (P =$ True $\longrightarrow z = x) \wedge (P =$ False $\longrightarrow z = y)$

IF-THEN-ELSE

consts If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if _ then _ else _)

If $P\ x\ y\ \equiv$ SOME $z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y)$

Intuition:

➜ for $P =$ True, right hand side collapses to SOME $z.\ z = x$

IF-THEN-ELSE

consts If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if_ then _ else _)

If $P\ x\ y\ \equiv$ SOME $z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y)$

Intuition:

- ➜ for $P =$ True, right hand side collapses to SOME $z.\ z = x$
- ➜ for $P =$ False, right hand side collapses to SOME $z.\ z = y$

IF-THEN-ELSE

consts If $:: bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if_ then _ else _)

If $P\ x\ y\ \equiv$ SOME $z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y)$

Intuition:

- ➜ for $P =$ True, right hand side collapses to SOME $z.\ z = x$
- ➜ for $P =$ False, right hand side collapses to SOME $z.\ z = y$

Proof Rules:

$$\frac{}{\text{if True then } s \text{ else } t = s}\ \text{ifTrue} \qquad \frac{}{\text{if False then } s \text{ else } t = t}\ \text{ifFalse}$$

Proof: Isabelle Demo

THAT WAS HOL

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

This can be automated

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

This can be automated

Syntax:

[<kind>!]	for safe rules (<kind> one of intro, elim, dest)
[<kind>]	for unsafe rules

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

This can be automated

Syntax:

[<kind>!]	for safe rules (<kind> one of intro, elim, dest)
[<kind>]	for unsafe rules

Application (roughly):

do safe rules first, search/backtrack on unsafe rules only

More on Automation

Last time: safe and unsafe rule, heuristics: use safe before unsafe

This can be automated

Syntax:

[<kind>!]	for safe rules (<kind> one of intro, elim, dest)
[<kind>]	for unsafe rules

Application (roughly):

do safe rules first, search/backtrack on unsafe rules only

Example:

declare attribute globally	**declare** conjI [intro!] allE [elim]
remove attribute gloabllay	**declare** allE [rule del]
use locally	**apply** (blast intro: someI)
delete locally	**apply** (blast del: conjI)

DEMO: AUTOMATION

We have learned today ...

➜ Defining HOL

We have learned today ...

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax

WE HAVE LEARNED TODAY ...

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax
- ➜ Deriving proof rules

We have learned today ...

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax
- ➜ Deriving proof rules
- ➜ More automation

EXERCISES

- derive the classical contradiction rule $(\neg P \Longrightarrow False) \Longrightarrow P$ in Isabelle
- define **nor** and **nand** in Isabelle
- show $\;\textsf{nor}\ x\ x\ =\ \textsf{nand}\ x\ x$
- derive safe intro and elim rules for them
- use these in an automated proof of $\;\textsf{nor}\ x\ x\ =\ \textsf{nand}\ x\ x$