COMP4161: Advanced Topics in Software Verification

# HOL

Gerwin Klein, June Andronick, Ramana Kumar

S2/2016

# Content

- Intro & motivation, getting started [1]
- Foundations & Principles
  - Lambda Calculus, natural deduction [1,2]
  - Higher Order Logic [3[a]]
  - Term rewriting [4]
- Proof & Specification Techniques
  - Inductively defined sets, rule induction [5]
  - Datatypes, recursion, induction [6, 7]
  - Hoare logic, proofs about programs, C verification [8[b],9]
  - (mid-semester break)
  - Writing Automated Proof Methods [10]
  - Isar, codegen, typeclasses, locales [11[c],12]

[a] a1 due; [b] a2 due; [c] a3 due

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

# What is Higher Order Logic?

➜ **Propositional Logic:**
- no quantifiers
- all variables have type bool

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

© Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

bool

© Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad\qquad \_ \Rightarrow \_$$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad \_ \Rightarrow \_ \qquad \text{ind}$$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad\qquad \_ \Rightarrow \_ \qquad\qquad \text{ind}$$

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad \_ \Rightarrow \_ \qquad \text{ind}$$

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

**Default Constants:**

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad \_ \Rightarrow \_ \qquad \text{ind}$$

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

**Default Constants:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad\qquad \_ \Rightarrow \_ \qquad\qquad \text{ind}$$

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

**Default Constants:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Defining Higher Order Logic

**Default types:**

$$\text{bool} \qquad\qquad \_ \Rightarrow \_ \qquad\qquad \text{ind}$$

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

**Default Constants:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

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

**One approach:** $\forall :: \mathit{var} \Rightarrow \mathit{term} \Rightarrow \mathit{bool}$
**Drawback:** need to think about substitution, $\alpha$ conversion again.

© Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

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

**One approach:** $\forall :: \mathit{var} \Rightarrow \mathit{term} \Rightarrow \mathit{bool}$
**Drawback:** need to think about substitution, $\alpha$ conversion again.

**But:** Already have binder, substitution, $\alpha$ conversion in meta logic

$$\lambda$$

© Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

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

**One approach:** $\forall :: \mathit{var} \Rightarrow \mathit{term} \Rightarrow \mathit{bool}$
**Drawback:** need to think about substitution, $\alpha$ conversion again.

**But:** Already have binder, substitution, $\alpha$ conversion in meta logic

$$\lambda$$

**So:** Use $\lambda$ to encode all other binders.

© Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

| **HOAS** | **usual syntax** |
| --- | --- |

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Higher Order Abstract Syntax

**Example:**

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

| **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.

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Side Track: Syntax Declarations in Isabelle

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 :: $ct \Rightarrow ct \Rightarrow fm \Rightarrow bool$ $(''\_, \_ \ \vdash \ \_'' \ [30, 0, 20] \ 60)$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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)

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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)

© Data61, CSIRO: provided under Creative Commons Attribution License

# 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 in Isabelle/Isar Reference Manual (7.2)

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Back to HOL

**Base:** $bool, \Rightarrow, ind \qquad =, \longrightarrow, \varepsilon$

**And the rest is**

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Back to HOL

**Base:** $\mathit{bool}, \Rightarrow, \mathit{ind} \qquad =, \longrightarrow, \varepsilon$

**And the rest is definitions:**

$$
\begin{array}{ll}
\text{True} & \equiv \\
\text{All } P & \equiv \\
\text{Ex } P & \equiv \\
\text{False} & \equiv \\
\neg P & \equiv \\
P \wedge Q & \equiv \\
P \vee Q & \equiv \\
\text{If } P\ x\ y & \equiv \\
\text{inj } f & \equiv \\
\text{surj } f & \equiv
\end{array}
$$

© Data61, CSIRO: provided under Creative Commons Attribution License

# Back to HOL

**Base:** $bool, \Rightarrow, ind \qquad =, \longrightarrow, \varepsilon$

**And the rest is definitions:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Back to HOL

**Base:** $bool, \Rightarrow, ind \qquad =, \longrightarrow, \varepsilon$

**And the rest is definitions:**

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# Back to HOL

**Base:** $bool, \Rightarrow, ind \qquad =, \longrightarrow, \varepsilon$

**And the rest is definitions:**

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# The Axioms of HOL

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# The Axioms of HOL

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

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

# The Axioms of HOL

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

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

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# The Axioms of HOL

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

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

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

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# The Axioms of HOL

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

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

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

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

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

# The Axioms of HOL

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

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

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

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

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

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# That's it.

- 3 basic constants
- 3 basic types
- 9 axioms

 © Data61, CSIRO: provided under Creative Commons Attribution License

# That's it.

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

**With this you can define and derive all the rest.**

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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:

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# Demo: The Definitions in Isabelle

# Deriving Proof Rules

In the following, we will

 Data61, CSIRO: provided under Creative Commons Attribution License

# Deriving Proof Rules

In the following, we will

➜ look at the definitions in more detail

© Data61, CSIRO: provided under Creative Commons Attribution License

# 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

© Data61, CSIRO: provided under Creative Commons Attribution License

# 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* $>$" $<$ *proof* $>$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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* >" < *proof* >

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# True

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

**Intuition:**
right hand side is always true

© Data61, CSIRO: provided under Creative Commons Attribution License

# 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**:

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Demo

# Universal Quantifier

$$\textbf{consts}\ \text{ALL} :: (\alpha \Rightarrow \textit{bool}) \Rightarrow \textit{bool}$$
$$\text{ALL}\ P \quad \equiv \quad P = (\lambda x.\ \text{True})$$

© Data61, CSIRO: provided under Creative Commons Attribution License

# Universal Quantifier

**consts** ALL :: $(\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$

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

**Intuition:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Universal Quantifier

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

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

**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 value.

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Universal 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 value.
- ALL $P$ should be true iff $P$ yields true for all $x$, i.e.
  if it is equivalent to the function $\lambda x.$ True.

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Universal Quantifier

**consts** ALL :: $(\alpha \Rightarrow bool) \Rightarrow bool$
ALL $P \equiv P = (\lambda x.\ \text{True})$

**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 value.
- ➜ 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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# False

**consts** False :: *bool*

$$\text{False} \equiv \forall P. P$$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# False

**consts** False :: *bool*

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

**Intuition:**

Everything can be derived from *False*.

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 :: $bool \Rightarrow bool$ $(\neg\ \_)$

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# Negation

**consts** Not :: *bool* $\Rightarrow$ *bool* $(\neg\ \_)$
$\neg P \equiv P \longrightarrow \text{False}$

**Intuition:**
Try $P = \mathit{True}$ and $P = \mathit{False}$ and the traditional truth table for $\longrightarrow$.

© Data61, CSIRO: provided under Creative Commons Attribution License

# Negation

**consts** Not :: $bool \Rightarrow bool$ $(\neg\ \_)$
$\neg P \equiv P \longrightarrow \text{False}$

**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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** EX :: $(\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$

$\text{EX}\ P \ \equiv\ \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** EX :: $(\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$
EX $P \;\; \equiv \;\; \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

**Intuition:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** $\text{EX} :: (\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$

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

**Intuition:**

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** $\text{EX} :: (\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$

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

**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: $(\forall x.\ P\ x \longrightarrow Q)$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** $\text{EX} :: (\alpha \Rightarrow \mathit{bool}) \Rightarrow \mathit{bool}$

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

**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: $(\forall x.\ P\ x \longrightarrow Q) = ((\exists x.\ P\ x) \longrightarrow Q)$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Existential Quantifier

**consts** EX :: $(\alpha \Rightarrow \textit{bool}) \Rightarrow \textit{bool}$

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

**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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Conjunction

**consts** And :: $bool \Rightarrow bool \Rightarrow bool$ $(\_ \wedge \_)$
$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Conjunction

**consts** And :: $\mathit{bool} \Rightarrow \mathit{bool} \Rightarrow \mathit{bool}$ $(\_ \wedge \_)$
$P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

**Intuition:**

➜ Mirrors proof rules for $\wedge$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 $R$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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

© Data61, CSIRO: provided under Creative Commons Attribution License

# Disjunction

**consts** Or :: $bool \Rightarrow bool \Rightarrow bool$ (_ $\vee$ _)
$P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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)

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 $R$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 $R$

**Proof Rules**:

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

**Proof**: Isabelle Demo

© Data61, CSIRO: provided under Creative Commons Attribution License

# If-Then-Else

**consts** If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if_ then _ else _)
If $P\ x\ y\ \equiv$ SOME $z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y)$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 $P = \text{True}$, right hand side collapses to SOME $z.\ z = x$

 © Data61, CSIRO: provided under Creative Commons Attribution License

# 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 $P = \text{True}$, right hand side collapses to SOME $z.\ z = x$
- for $P = \text{False}$, right hand side collapses to SOME $z.\ z = y$

© Data61, CSIRO: provided under Creative Commons Attribution License

# If-Then-Else

**consts** If :: $bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (if_ then _ else _)
If $P\ x\ y \ \equiv\ \text{SOME}\ z.\ (P = \text{True} \longrightarrow z = x) \wedge (P = \text{False} \longrightarrow z = y)$

**Intuition:**

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

**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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# That was HOL

# More on Automation

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# More on Automation

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

**This can be automated**

 © Data61, CSIRO: provided under Creative Commons Attribution License

# More on Automation

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

**This can be automated**

© Data61, CSIRO: provided under Creative Commons Attribution License

# More on Automation

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

**This can be automated**

**Syntax**:

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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# More on Automation

**Last time**: safe and unsafe, 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

 © Data61, CSIRO: provided under Creative Commons Attribution License

# More on Automation

**Last time**: safe and unsafe, 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 gloablly | **declare** allE [rule del] |
| use locally | **apply** (blast intro: someI) |
| delete locally | **apply** (blast del: conjI) |

 © Data61, CSIRO: provided under Creative Commons Attribution License

# Demo: Automation

# We have learned today ...

➜ Defining HOL

 © Data61, CSIRO: provided under Creative Commons Attribution License

# We have learned today ...

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax

© Data61, CSIRO: provided under Creative Commons Attribution License

# We have learned today ...

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

© Data61, CSIRO: provided under Creative Commons Attribution License

# We have learned today ...

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

© Data61, CSIRO: provided under Creative Commons Attribution License