Exercises from last time

- We said that \( \varepsilon \) implies the Axiom of Choice:
  \[ \forall x. \exists y. Q x y \Rightarrow \exists f. \forall x. Q x (f x) \]
- Prove the axiom of choice as a lemma, using only the introduction and elimination rules for \( \forall \) and \( \exists \), namely allI, exI, allE, exE, and the introduction rule for \( \varepsilon \), someI, using only the proof methods rule, rule_sac, erule, erule_sac and assumption.
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 \( \lambda \) with certain default types and constants

Defining Higher Order Logic

Default types:

- bool
- \( \to \)
- \( \text{ind} \)

- bool sometimes called \( o \)
- \( \to \) sometimes called \( \text{fun} \)

Default Constants:

- \( \to \) :: bool \to bool \to bool
- = :: \( \alpha \to \alpha \to \text{bool} \)
- \( \epsilon \) :: (\( \alpha \to \text{bool} \)) \to \( \alpha \)

Higher Order Abstract Syntax

Problem: Define syntax for binders like \( \forall \), \( \exists \), \( \epsilon \)

One approach: \( \forall : \text{var} \Rightarrow \text{term} \Rightarrow \text{bool} \)

Drawback: need to think about substitution, \( \alpha \) conversion again.

But: Already have binder, substitution, \( \alpha \) conversion in meta logic

So: Use \( \lambda \) to encode all other binders.

Example:

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

\( \lambda \)

Isabelle can translate usual binder syntax into HOAS.
Side Track: Syntax Declarations in Isabelle

- mixfix:
  - const: drvbl :: ct ⇒ ct ⇒ f m ⇒ bool ("_ _")
  Legal syntax now: Γ, Π ⊢ F

- priorities:
  - pattern can be annotated with priorities to indicate binding strength
  Example: drvbl :: ct ⇒ ct ⇒ f m ⇒ bool ("_ _") [30, 0, 20] 60

- infixl/infixr: short form for left/right associative binary operators
  Example: or :: bool ⇒ bool ⇒ bool (infixr "∨" 30)

- binders: declaration must be of the form
  Example: ALL :: (α ⇒ bool) ⇒ bool (binder "∀" 10)

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

Back to HOL

Base: bool, ⊢, ind =, →, ε

And the rest is definitions:

True ≡ (λx::bool. x) = (λx. x)

All P ≡ P = (λx. True)

Ex P ≡ ∀Q. (∀x. P x → Q) → Q

False ≡ ∀P. P → False

¬P ≡ ∀Q. (P → Q) → R

P ∧ Q ≡ ∀R. (P → Q) → R

P ∨ Q ≡ ∀R. (P → R) → (Q → R) → R

If P x y = SOME z. (P = True → z = x) ∧ (P = False → z = y)

inj f ≡ ∀x y. f x = f y → x = y

surj f ≡ ∀y. ∃x. y = f x

The Axioms of HOL

x = x refl

P = P subst

λx. f x = g x ext

P → Q P → Q impl

P → Q Q → P mp

(P → Q) → (Q → P) → (P = Q) iff

P = True ∨ P = False True_or_False

P + x somel

∃f :: ind ⇒ ind. inj f ∧ ¬surj f infly

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:

x = y eq reflection

THE x. x = a eq trivial
Slide 13

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**

```isabelle
lemma [name: ]
  assumes [name1: ] "< prop >1"
  assumes [name2: ] "< prop >2"
  shows "< prop >" < proof >

proves: [ < prop >1; < prop >2; ... ] \implies < prop >
```

Slide 14

Slide 15

```
consts True :: bool
True = (\lambda x :: bool. x) = (\lambda x. x)
```

Intuition:
right hand side is always true

Proof Rules:

```isabelle
True TrueI
```

Proof:

```isabelle
(\lambda x :: bool. x) = (\lambda x. x) \Rightarrow True
```

unfold True\_def

Slide 16
Universal Quantifier

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

Intuition:
\(\Rightarrow\) ALL \(P\) is Higher Order Abstract Syntax for \(\forall x. P \, x\).
\(\Rightarrow\) \(P\) is a function that takes an \(x\) and yields a truth values.
\(\Rightarrow\) ALL \(P\) should be true if \(P\) yields true for all \(x\), i.e.
if it is equivalent to the function \(\lambda x. \text{True}\).

Proof Rules:
\[
\frac{\prod x. P \, x}{\forall x. P \, x} \text{ allI}
\]
\[
\frac{\forall x. P \, x \implies R}{R} \text{ allE}
\]

Proof: Isabelle Demo

Slide 17

Negation

consts Not :: bool \Rightarrow bool \rightarrow (\sim)
\(\text{not} \ P = P \rightarrow \text{False}\)

Intuition:
Try \(P = \text{True}\) and \(P = \text{False}\) and the traditional truth table for \(\sim\).

Proof Rules:
\[
\frac{A \rightarrow \text{False}}{\sim A} \text{ notI}
\]
\[
\frac{A}{\sim A} \text{ notE}
\]

Proof: Isabelle Demo

Slide 19

Existential Quantifier

consts EX :: \(\alpha \Rightarrow \text{bool} \Rightarrow \text{bool}\)
\(\text{EX} \ P = \forall Q. (\prod x. P \, x \implies Q) \implies Q\)

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

Proof Rules:
\[
\frac{P \rightarrow Q}{\exists x. P \, x} \text{ exI}
\]
\[
\frac{\exists x. P \, x \rightarrow R}{\forall x. P \, x \rightarrow R} \text{ allE}
\]

Proof: Isabelle Demo

Slide 20
Conjunction

cons And :: bool ⇒ bool ⇒ bool (\( ∧ \))
\( P \land Q \equiv \forall R. (P \rightarrow Q \rightarrow R) \rightarrow R \)

Intuition:

- Mirrors proof rules for \( \land \)
- Try truth table for \( P, Q, \) and \( R \)

Proof Rules:

\[
\frac{A \quad B}{A \land B} \quad \frac{A \land B \quad [A, B] \rightarrow C}{C} \quad \text{conjE}
\]

Proof: Isabelle Demo

If-Then-Else

cons If :: bool ⇒ α ⇒ α ⇒ α (if \( P \) then \( x \) else \( y \))
If \( P \ x \ y \equiv \text{SOME } z. \ (P \rightarrow z = x) \land (P = \text{False} \rightarrow z = y) \)

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:

\[
\frac{\text{if True then } x \text{ else } t = s}{\text{if True}} \quad \frac{\text{if False then } x \text{ else } t = t}{\text{if False}}
\]

Proof: Isabelle Demo

Disjunction

cons Or :: bool ⇒ bool ⇒ bool (\( \lor \))
\( P \lor Q \equiv \forall R. (P \rightarrow R) \rightarrow (Q \rightarrow R) \rightarrow R \)

Intuition:

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

Proof Rules:

\[
\frac{A \quad B}{A \lor B} \quad \frac{A \lor B \quad A \rightarrow C \quad B \rightarrow C}{C} \quad \text{disjE}
\]

Proof: Isabelle Demo
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
remove attribute globally
use locally
delete locally

We have learned today ...

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

---

Demo: Automation

---

Slide 25

Slide 26

Slide 27