Today we will learn...

- Existential operator
- Mock Mini-Test 1
- Formal language
- Language operators
- Language equivalence
From proposition to proof state

Goal for all \( (a \ b \ c : \text{nat}), \ a = b \rightarrow b = c \).

Proof.

\text{intros}.

What is the expected proof state?
From proposition to proof state

Goal for all (a b c: nat), a = b → b = c.

Proof.
intros.

What is the expected proof state?

Solution

1 subgoal
a, b, c : nat
H : a = b
______________________________________(1/1)
b = c

- Each parameter of a theorem is an assumption
- Each variable in the forall is one parameter becomes an assumption
- Each pre-condition of an implication becomes an assumption
- Variables and pre-conditions are parameters
You can name assumptions in a forall

Goal forall (a b c:nat) (eq_a_b: a = b),
   b = c.
Proof.
   intros.

What is the expected proof state?
You can name assumptions in a forall

Goal forall (a b c:nat) (eq_a_b: a = b),
  b = c.
Proof.
  intros.

What is the expected proof state?

Solution

1 subgoal
a, b, c : nat
eq_a_b : a = b
---------------------------(1/1)
b = c

• Implications are just anonymous parameters (name will be generated automatically)
• Think assert (x = y) versus assert (Ha: x = y)
From proof state to proposition:

What is the lemma that originates the following proof state?

\[ a, b, c : \text{nat} \]
\[ P, Q : \text{Prop} \]
\[ H : P \rightarrow a = b \]
\[ H0 : Q \lor P \]
\[ H1 : b = c \]

\[ \text{------------------ (1/1)} \]
\[ a = c \]
From proof state to proposition:

What is the lemma that originates the following proof state?

\[\begin{align*}
a, b, c & : \text{nat} \\
P, Q & : \text{Prop} \\
H & : P \rightarrow a = b \\
H0 & : Q \lor P \\
H1 & : b = c \\
\hline \\
a = c
\end{align*}\]

Solution 1:

Goal for all (a b c : nat) (P Q : Prop) (H : P → a = b) (H0 : Q \lor P) (H1 : b = c), a = c.

Solution 2:

Goal for all (a b c : nat) (P Q : Prop), (P → a = b) → (Q \lor P) → (b = c) → a = c.
Existential quantification

$\exists x. P$
Existential quantification

\[
\text{Inductive} \ ex \ (A : \text{Type}) \ (P : A \to \text{Prop}) : \text{Prop} := \\
| \text{ex_intro} : \forall (x : A) \ (\_ : P x), \ ex \ P.
\]

Notation:

\[
\textit{exists} \ x : A, \ P \ x
\]

- To conclude a goal \(\text{exists} \ x : A, \ P \ x\) we can use tactics \text{exist} \ x. which yields \(P \ x\). Alternatively, we can use \text{apply} \ ex\_intro.

\[
\forall n, \text{exists} \ z, \ z + n = n
\]

- To use a hypothesis of type \(H:\exists x : A, \ P \ x\), you can use \text{destruct} \ H \ as \ (x, H), \ or \ \text{inversion} \ H

\[
\forall n, (\text{exists} \ m, \ m < n) \to n \not< 0.
\]
Defining arbitrary logical relations
Defining less-than-equal

Inductive definition of $\leq$

\[
\begin{align*}
\text{le}_n & : n \leq n \\
\text{le}_S & : n \leq S m
\end{align*}
\]

Inductive \(\text{le} : \text{nat} \to \text{nat} \to \text{Prop} := \)

\[
\begin{align*}
\text{le}_n & : \forall n : \text{nat}, \\
& \quad n \leq n \\
\text{le}_S & : \forall (n \ m : \text{nat}), \\
& \quad n \leq m \\
& \quad n \leq S m.
\end{align*}
\]

- Any pre-condition will appear above the line
- Preconditions are separated by whitespace
How do we know that less-than-equal was defined correctly?
How do we know that less-than-equal was defined correctly?

With theorems!

(* Simple tests *)
Goal 1 ≤ 1. Proof. Admitted.

(* More interesting properties *)
Theorem le_is_reflexive: forall x,
  x ≤ x.
Proof. Admitted. (* Proved in class *)
Theorem le_is_anti_symmetric: forall x y,
  x ≤ y →
  y ≤ x →
  x = y.
Proof. Admitted. (* Proved in class *)
Theorem le_is_transitive: forall x y z,
  x ≤ y →
  y ≤ z →
  x ≤ z.
Proof. Admitted.
Mock Mini-Test 1
Q1.1

All functions defined in Coq via Fixpoint must terminate on all inputs.
Q1.1

All functions defined in Coq via Fixpoint must terminate on all inputs.
Solution: True

All functions must terminate.
If \( S(n + m) = n + S\ m \) is the goal in the current proof state, then \texttt{reflexivity} will solve the goal.
Q1.2

If $S (n + m) = n + S m$ is the goal in the current proof state, then \texttt{reflexivity} will solve the goal.

Solution: False
A *polymorphic* type is one that is parameterized by a type argument by using the universal quantifier `forall`. For instance: `forall (X:Type), list X → list X` is a polymorphic type.
Q1.3

A polymorphic type is one that is parameterized by a type argument by using the universal quantifier forall. For instance: forall (X:Type), list X → list X is a polymorphic type.

Solution: True
Q1.4

If \( E \) has type \( \text{beq_nat } m \ n = \text{true} \), then \( E \) also has type \( m = n \).
Q1.4

If $E$ has type $\text{beq_nat\ m\ n = true}$, then $E$ also has type $m = n$.
Solution: False

Goal

forall n m (E:Nat.eqb n m = true),
m = n.

Proof.

intros.
Fail apply E.
Abort.
Q1.5

The proposition \( \forall n, S\ n \not\leftrightarrow n \) is provable in Coq.
The proposition \( \forall n, S n \not\equiv n \) is provable in Coq.

Solution: True

```coq
Goal
  \forall n, S n \not\equiv n.

Proof.
  intros.
  intros N.
  induction n. { 
    inversion N.
  }
  inversion N.
  apply IHn.
  assumption.
Qed.
```
Q2.1

What is the type of the following expression?

Nat.eqb 28
Q2.1

What is the type of the following expression?

Nat.eqb 28

Answer: nat → bool
Q2.2

What is the type of the following expression?

14 = 68
Q2.2

What is the type of the following expression?

14 = 68

Answer: Prop
Q3.1

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\[
\forall n, \ n \not= S\ n
\]
Q3.1

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\( \forall n, n \neq S\ n \)

**Answer:** induction
Q3.2

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

forall (n m:nat), n = m \/ n <> m
Q3.2

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\[ \forall (n \ m: \text{nat}), n = m \lor n \not= m \]

**Answer:** BY INDUCTION
Q3.3

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

forall A B:Type, forall (f g: A → B), f = g → forall x, f x = g x
Q3.3

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\[ \forall A \ B : \text{Type}, \forall (f \ g : A \to B), f = g \to \forall x, f \ x = g \ x \]

**Answer: EASY**

**Goal**
\[ \forall A \ B : \text{Type}, \forall (f \ g : A \to B), f = g \to \forall x, f \ x = g \ x. \]

**Proof.**
- *intros.*
- *rewrite H.*
- *reflexivity.*

Qed.
Q3.4

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\texttt{forall \ P : Prop, P}
The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

forall P : Prop, P

Answer: NOT PROVABLE

Goal
  forall P : Prop, P.
Proof.
  intros X.
  Fail apply X.
Abort.
Q3.5

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\[ \forall n, n+5 \leq n+6 \]
Q3.5

The proof of this goal is: EASY / BY INDUCTION / NOT PROVABLE

\[ \forall n, n+5 \leq n+6 \]

Answer: INDUCTION
Q4.1

Prove this goal:

\[ H : \sim \sim P \]
\[ H_0 : P \lor \sim P \]

\[ \text{---------------(1/1)} \]
\[ P \]
Q4.1

Prove this goal:

\[ H : \sim \sim P \]
\[ H_0 : P \lor \sim P \]

\[ \frac{}{}(1/1) \]

\[ P \]

destruct \( H_0 \). { 
  assumption. 
}
apply \( H \) in \( H_0 \).
contradiction.
Q4.2

Prove this goal:

\[ H : P \rightarrow Q \]
\[ H_0 : P \lor \sim P \]

\[ \sim P \lor Q \]
Q4.2

Prove this goal:

\( H : P \to Q \)
\( H0 : P \lor \neg P \)

\[ \neg P \lor Q \] (1/1)

\( \neg P \lor Q \)

\text{destruct } H0. \{ 
  \text{apply } H \text{ in } H0. 
  \text{right.} 
  \text{assumption.} 
\} 

\text{left.} 
\text{assumption.}
Q4.3

Prove this goal:

\[ \begin{align*}
P, Q &: \text{Prop} \\
PQ &: P \rightarrow Q \\
NQ &: \sim Q \\
HP &: P \\
\text{-------------------------------------------(1/1)} \\
\text{False} \\
\end{align*} \]
Q4.3

Prove this goal:

\begin{align*}
P, Q : \text{Prop} \\
PQ : P \rightarrow Q \\
NQ : \sim Q \\
HP : P \\
\hline
\end{align*}

False

apply \ PQ \ in \ HP. \ contradiction.
Q4.4

forall (A:Type) (l:list A), l = [] \rightarrow l = []
forall \ A : Type, l : list A, l = [] → l = []

intros. assumption.