Today we will learn...

- Existential operator
- Mock Mini-Test 1
- Formal language
- Language operators
- Language equivalence
Goal for all (a b c:nat), a = b → b = c.
Proof.
    intros.

What is the expected 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?

\[
\begin{align*}
a, b, c & \colon \text{nat} \\
P, Q & \colon \text{Prop} \\
H & \colon P \implies a = b \\
H_0 & \colon Q \lor P \\
H_1 & \colon b = c \\
\hline
a = c
\end{align*}
\]
From proof state to proposition:

What is the lemma that originates the following proof state?

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

\[\text{------------------------------------ (1/1)}\]

\[a = c\]

**Solution 1:**

Goal forall (a b c : nat) (P Q : Prop) (H : P \rightarrow a = b) (H0 : Q \lor P) (H1 : b = c), a = c.

**Solution 2:**

Goal forall (a b c : nat) (P Q : Prop), (P \rightarrow a = b) \rightarrow (Q \lor P) \rightarrow (b = c) \rightarrow a = c.
Existential quantification

$\exists x. P$
Existential quantification

Inductive ex (A : Type) (P : A → Prop) : Prop :=
    | ex_intro : forall (x : A) (_ : P x), ex P.

Notation:

exists x:A, P x

- To conclude a goal exists x:A, P x we can use tactics exist x. which yields P x. Alternatively, we can use apply ex_intro.

forall n, exists z, z + n = n

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

forall n, (exists m, m < n) → n <> 0.
Defining arbitrary logical relations
Defining less-than-equal

Inductive definition of \( \leq \)

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

\[
\text{Inductive } \text{le} : \text{nat} \to \text{nat} \to \text{Prop} := \\
| \text{le}_n : \forall n : \text{nat}, \quad \text{le}_n \, n \, n \\
| \text{le}_S : \forall (n \, m : \text{nat}), \quad \text{le} \, n \, m \to \quad \text{le} \, n \, (S \, m).
\]

- 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 \leq 1. Proof. Admitted.
Goal 1 \leq 10. Proof. Admitted.

(* More interesting properties *)
Theorem le_is_reflexive: forall x, 
  x \leq x.
Proof. Admitted. (* Proved in class *)

Theorem le_is_anti_symmetric: forall x y, 
  x \leq y \to
  y \leq x \to
  x = y.
Proof. Admitted. (* Proved in class *)

Theorem le_is_transitive: forall x y z, 
  x \leq y \to
  y \leq z \to
  x \leq 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.
Q1.2

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

Goal
\[
\forall n \ m, \quad S(n + m) = n + S\ m.
\]

Proof.
\begin{itemize}
  \item intros.
  \item Fail \texttt{reflexivity}.
  \item Abort.
\end{itemize}
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.
A polymorphic type is one that is parameterized by a type argument by using the universal quantifier \( \forall \). For instance: \( \forall (X:\text{Type}), \text{list } X \rightarrow \text{list } X \) 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 $\text{beq_nat} \ m \ n = \text{true}$, 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 $\forall n, S\ n \neq n$ 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 <> 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

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

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

forall n, n <> S n

Answer: induction
Q3.2

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

\[ \text{forall } (n \ m: \text{nat}), \ n = m \lor n \neq m \]
Q3.2

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

forall (n m:nat), n = m \lor n <> m

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

\[
\text{forall } A \text{ B:Type, forall } (f \ g: A \rightarrow B), f = g \rightarrow \text{forall } x, f \ x = g \ x
\]
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.
\[
\text{intros.}
\]
\[
\text{rewrite H.}
\]
\[
\text{reflexivity.}
\]

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

\[ \forall P : \text{Prop}, P \]
Q3.4

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 : ~ ~ P
H0 : P \lor \sim P

(1/1)
P
Prove this goal:

H : ~ ~ P
H₀ : P \lor ~ P

-----------------------------(1/1)
P

destruct H₀. {
  assumption.
}
apply H in H₀.
contradiction.
Q4.2

Prove this goal:

\[ \begin{align*}
H : & \quad P \rightarrow Q \\
H0 : & \quad P \lor \sim P \\
\sim P \lor Q & \quad (1/1) \\
\sim P \lor Q & 
\end{align*} \]
Q4.2

Prove this goal:

\[ H : P \rightarrow Q \]
\[ H_0 : P \lor \sim P \]
\[ \underline{\sim P \lor Q} (1/1) \]
\[ \sim P \lor Q \]

destruct \( H_0 \). {
  apply \( H \) in \( H_0 \).
  right.
  assumption.
}
left.
assumption.
Q4.3

Prove this goal:

\[
\begin{align*}
P, Q : \text{Prop} \\
PQ : P \rightarrow Q \\
NQ : \neg Q \\
HP : P \\
\end{align*}
\]

\[\text{False}\]
Q4.3

Prove this goal:

P, Q : Prop
PQ : P → Q
NQ : ~ Q
HP : P

____________________(1/1)
False

apply PQ in HP. contradiction.
Q4.4

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

intros. assumption.