CS420

Logical Foundations of Computer Science

Lecture 7: Mock mini-test 1

Tiago Cogumbreiro

Today we will learn...

- Existential operator
- Mock Mini-Test 1
- Formal language
- Language operators
- Language equivalence

From proposition to proof state

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

What is the expected proof state?

From proposition to proof state

```
Goal forall (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: nat
P, Q: Prop
H: P → a = b
H0: Q \/ P
H1: b = c
____________________________ (1/1)
a = c
```

From proof state to proposition:

What is the lemma that originates the following proof state?

```
a, b, c: nat
P, Q: Prop
H: P → a = b
H0: Q \/ P
H1: b = c
____________________________ (1/1)
a = c
```

Solution 1:

```
Goal forall (a b c: nat) (P Q: Prop) (H: P → a = b) (H0: Q \/ P) (H1: b = c), a = c.
```

Solution 2:

```
Goal forall (a b c: nat) (P Q: Prop), (P → a = b) → (Q \/ P) → (b = c) → 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 $\leq$

$$\frac{}{n \leq n}\,\texttt{le\_n} \qquad \frac{n \leq m}{n \leq \mathtt{S}\; m}\,\texttt{le\_S}$$

```
Inductive le : nat → nat → Prop :=
  | le_n : forall n:nat,
    n ≤ n
  | le_S : forall (n m : nat),
    n ≤ m →
    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 ≤ 1. Proof. Admitted.
Goal 1 ≤ 10. 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.

Q1.2

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

Q1.2

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

Solution: False

```
Goal
  forall n m,
  S (n + m) = n + S m.
Proof.
  intros.
  Fail reflexivity.
Abort.
```

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.

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 `beq_nat m n = true`, then `E` also has type `m = n`.

Q1.4

If E has type `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 <> n` is provable in Coq.

Q1.5

The proposition `forall n, S n <> n` is provable in Coq.

Solution: True

```
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

```
forall n, n <> 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

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

Q3.2

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

```
forall (n m:nat), n = m \/ n <> 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:Type, forall (f g: A -> B), f = g -> forall x, f x = g x
```

Answer: EASY

```
Goal
  forall A B:Type, forall (f g: A -> B), f = g -> 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

```
forall P : 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 ≤ n+6
```

Q3.5

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

```
forall n, n+5 ≤ n+6
```

Answer: INDUCTION

Q4.1

Prove this goal:

```
H : ~ ~ P
H0 : P \/ ~ P
______________________________________(1/1)
P
```

Q4.1

Prove this goal:

```
H : ~ ~ P
H0 : P \/ ~ P
______________________________________(1/1)
P

destruct H0. {
  assumption.
}
apply H in H0.
contradiction.
```

Q4.2

Prove this goal:

```
H : P -> Q
H0 : P \/ ~ P
______________________________________(1/1)
~ P \/ Q
```

Q4.2

Prove this goal:

```
H : P -> Q
H0 : P \/ ~ P
______________________________________(1/1)
~ P \/ Q

destruct H0. {
  apply H in H0.
  right.
  assumption.
}
left.
assumption.
```

Q4.3

Prove this goal:

```
P, Q : Prop
PQ : P → Q
NQ : ~ Q
HP : P
______________________________________(1/1)
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

```
______________________________________(1/1)
forall (A:Type) (l:list A), l = [] → l = []
```

Q4.4

```
----------------------------------------(1/1)
forall (A:Type) (l:list A), l = [] → l = []

intros. assumption.
```