CS720

Logical Foundations of Computer Science

Lecture 7: Logical connectives in Coq

Tiago Cogumbreiro
What have we learned so far

- Comparing if two expressions are equal syntactically: $e_1 = e_2$
- Implication $P \rightarrow Q$
- Universal quantifier $\forall x, P$

Is this all we can do?
What have we learned so far

- Comparing if two expressions are equal syntactically: $e_1 = e_2$
- Implication $P \rightarrow Q$
- Universal quantifier $\forall x, P$

Is this all we can do? No.

We encoded predicates *computationally*:
- In `Basics.v` we defined `Nat.eqb : nat -> nat -> bool` to compare if two naturals are equal.
- In `Basics.v` we defined `even : nat -> bool` to check if a natural number is even

Computational predicates are limited in what they can describe (eg, functions in Coq have to be total), and are not very easy to reason about (ie, they are meant to compute/execute, not build logic statements).
Today we will...

- Logical connectives in Coq

\[ P \land Q \quad P \lor Q \]

Why are we learning this?

- The building blocks of any interesting property
Typing equality

What is the type of an equality?

Check \( \text{beq}_\text{nat} \ 2 \ 2 = \text{true} \).

Check \( \forall (n \ m : \text{nat}), \ n + m = m + n \).
Typing equality

What is the type of an equality?

Check  beq_nat 2 2 = true.

Check  forall (n m : nat), n + m = m + n.

Both of these expressions have type Prop, for proposition.
Are all propositions provable?
Are all propositions provable?

**Obviously no.** How do you prove this proposition:

```
Check 0 = 1. (* Prints: 0 = 1: Prop *)
```

```
Goal 0 = 1.
```

**Insights**

- We can write any proposition, even **unprovable** ones. 
  
  *We can write proposition \(0 = 1\), but we cannot prove it.*

- The fact that something is **false** is not the same as unprovable!
  
  *We can prove that something is false (by showing it leads to false), eg, \(0 = 1\).*

  *We cannot prove the law of the excluded middle in Coq.*

- In Coq, we must show evidence of what holds. (*This is known as a constructive logic.*)
Propositions are still expressions (1/3)

What is the type of $\text{ex0}$:

Definition $\text{ex0} := \text{beq_nat \ 2 \ 2}$. 
Propositions are still expressions (1/3)

What is the type of \texttt{ex0}:

\texttt{Definition ex0 := beq_nat 2 2.}

What is the type of \texttt{ex1}? How can we use \texttt{ex1}?

\texttt{Definition ex1 (n:nat) := beq_nat 2 n = true.}
\texttt{Check ex1.}

\texttt{ex1} is a function that returns a proposition, a parameterized proposition. For which \texttt{n} is \texttt{ex1 \ n} provable?
Propositions are still expressions (1/3)

What is the type of $ex_0$?

**Definition**

$ex_0 := \text{beq_nat } 2 \ 2$.

What is the type of $ex_1$? How can we use $ex_1$?

**Definition**

$ex_1 (n:\text{nat}) := \text{beq_nat } 2 \ n = \text{true}$.

**Check**

$ex_1$.

$ex_1$ is a function that returns a proposition, a parameterized proposition. For which $n$ is $ex_1 \ n$ provable?

**Lemma** easy:

$\forall \ n, n = 2 \implies ex_1 \ n$.

**Proof**.

*(Done in class.)*
Propositions are still expressions (2/3)

What is the difference between `ex1` and `ex2`?

**Definition**  
`ex1 (n:nat) := beq_nat 2 n = true.`

**Theorem**  
`ex2: forall (n:nat), beq_nat 2 n = true.`
Propositions are still expressions (2/3)

What is the difference between \texttt{ex1} and \texttt{ex2}?

\textbf{Definition} \texttt{ex1} (n:nat) := beq_nat 2 n = true.

\textbf{Theorem} \texttt{ex2} : \texttt{forall} (n:nat), beq_nat 2 n = true.

\texttt{ex1} defines a position (\texttt{Prop}), \texttt{ex2} is a theorem definition and is expecting a proof.

What is the relation between \texttt{ex3} and \texttt{ex1}, \texttt{ex2}?

\textbf{Definition} \texttt{ex3} (n:nat) : beq_nat 2 n = true.
Propositions are still expressions (2/3)

What is the difference between \texttt{ex1} and \texttt{ex2}?

\textbf{Definition} \texttt{ex1 (n:nat) := beq_nat 2 n = true.}

\textbf{Theorem} \texttt{ex2 : forall (n:nat), beq_nat 2 n = true.}

\texttt{ex1} defines a position (Prop), \texttt{ex2} is a theorem definition and is expecting a proof.

What is the relation between \texttt{ex3} and \texttt{ex1, ex2}?

\textbf{Definition} \texttt{ex3 (n:nat) : beq_nat 2 n = true.}

- Recall that \textbf{Theorem} and \textbf{Definition} are synonyms!
- Thus, \texttt{ex2} and \texttt{ex3} are the same
Logical connectives
Conjunction

\[ P \land Q \]
What is $P \land Q$?

1. What is the type of $P$?
What is $P \land Q$?

1. What is the type of $P$? Prop
2. What is the type of $Q$?
What is $P \land Q$?

1. What is the type of $P$? Prop
2. What is the type of $Q$? Prop
3. What is the type of $\land$?
What is $P \land Q$?

1. What is the type of $P$? Prop
2. What is the type of $Q$? Prop
3. What is the type of $\land$? Prop $\to$ Prop $\to$ Prop
Split conjunctions in the goal

- When a logical-and appears in the goal, use split
- You need to prove both propositions

**Goal** \(3 + 4 = 7 \land 2 \times 2 = 4\).

**Proof.**

split.

*(Done in class.)*
Conjunction example 1

More generally, we can show that if we have propositions $A$ and $B$, we can conclude that we have $A \land B$.

**Goal** $\forall A B : \text{Prop}, \ A \rightarrow B \rightarrow A \land B$. 
Destruct conjunction in hypothesis

- Case analysis $A \land B$, how many proofs? how many goals?

**Goal**

$\forall x \ y, 3 + x = y \land 2 \times 2 = x \Rightarrow x = 4 \land y = 7.$

**Proof.**

```
intros x y Hconj.
destruct Hconj as [Hleft Hright].
```

*(Done in class.*)
Conjunction example 2

Lemma correct_2 : \( \forall A B : \text{Prop}, A \land B \to A \).
Proof.

Lemma correct_3 : \( \forall A B : \text{Prop}, A \land B \to B \).
Proof.

(Done in class.)
Disjunction

\[ P \lor Q \]
What is $P \lor Q$?

1. What is the type of $P$?
What is $P \lor Q$?

1. What is the type of $P$? $\text{Prop}$
2. What is the type of $Q$?
What is $P \lor Q$?

1. What is the type of $P$? Prop
2. What is the type of $Q$? Prop
3. What is the type of $\lor$?
What is $P \lor Q$?

1. What is the type of $P$? Prop
2. What is the type of $Q$? Prop
3. What is the type of $\lor$? Prop -> Prop -> Prop
Choose disjunction in goal

- Use left/right to pick what you want to prove
- Only choose when you know you can prove it

Goal

\[ \text{forall } n \text{ m : nat, Nat.beq n m = true} \lor \text{Nat.beq n m = false.} \]

Proof.
Destruct disjunction in hypothesis

- Case analysis $A \lor B$, how many proofs? how many goals?

**Lemma** or_example :

```
forall n m : nat, n = 0 \lor m = 0 -> n * m = 0.
```

**Proof.**

```
intros n m Hor.
destruct Hor as [Heq | Heq].
```
Falsehood
Find contradiction, false in goal

- False cannot be proved (we postpone how to our next lecture)
- Equality contradictions can be handled via explosion principle (discriminate)
- In this example we show that $1 = 2$ is (leads to) false.

Goal

\[
1 = 2 \rightarrow False.
\]
Destruct false in hypothesis

- Case analysis concludes any proof with False as assumption

**Theorem** ex_falso_quodlibet : forall (P:Prop), False -> P.

**Proof.**
Negation

\[ \neg P \]
Not in assumption and contradictions

**Definition** not \((\text{P:Prop}) := \text{P} \rightarrow \text{False}\).

**Notation** "\(\neg \ \text{x}\)" := (not \(\text{x}\)) : type_scope.

- apply \(\neg \ \text{P}\) in \(\text{P}\) to reach contradiction
- alternatively, use **contradiction**

**Theorem** contradiction_implies_anything : forall \(\text{P \ Q : Prop}\),
\((\text{P} \land \neg \text{P}) \rightarrow \text{Q}\).

**Proof.**
Negation in goal, proof by contradiction

- Show \( \sim P \) by assuming \( P \) and reaching contradiction

**Goal**

\( \sim \text{False} \).