

CS720

Logical Foundations of Computer Science

Lecture 8: Logical connectives in Coq

Tiago Cogumbreiro

Today we will learn...

- more logic connectives
- constructive logic (and its relation to classical logic)
- building propositions with functions
- building propositions with inductive definitions

Logic connectives

Truth

Truth

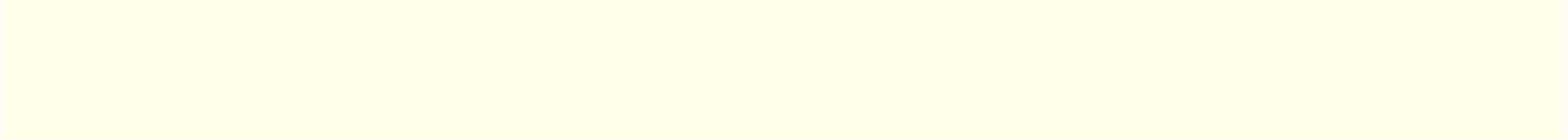
Truth can be encoded in Coq as a proposition that always holds, which can be described as a proposition type with a single constructor with 0-arity.

Truth example

(Done in class.)

Equivalence

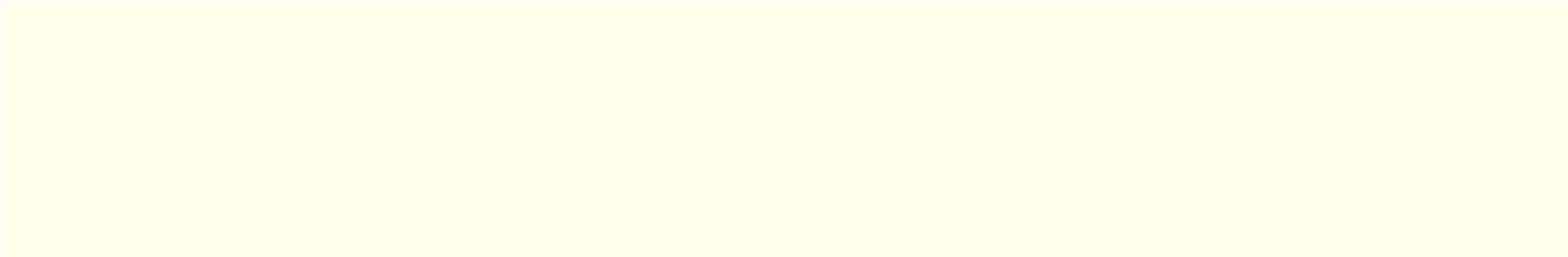
Logical equivalence



Split equivalence in goal

- When induction is required, prove each side by induction independently.
Split, and prove each side in its own theorem by induction.

Apply equivalence to assumption



Interpret equivalence as equality

The `Eq` library lets you treat an equivalence as an equals:

Tactics `congr`, `congrEq`, and `congrEq1` all handle equivalence as well.

Existential quantification

Existential quantification

Notation:

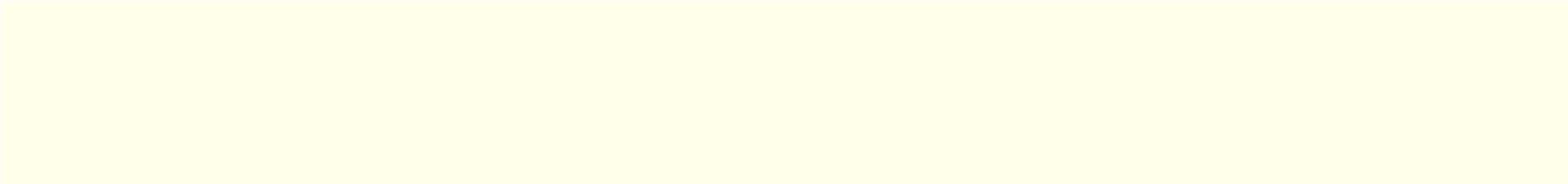
- To conclude a goal G we can use tactics t which yields G .
- To use a hypothesis of type H , you can use $\text{apply } H$.

Use exist for existential in goal

To conclude a goal $\exists x. P(x)$ we can use tactics `exists` which yields $P(a)$.

- Give the value that satisfies the equality.
- You can play around with exists to figure out what makes sense.

Destruct existential in assumption

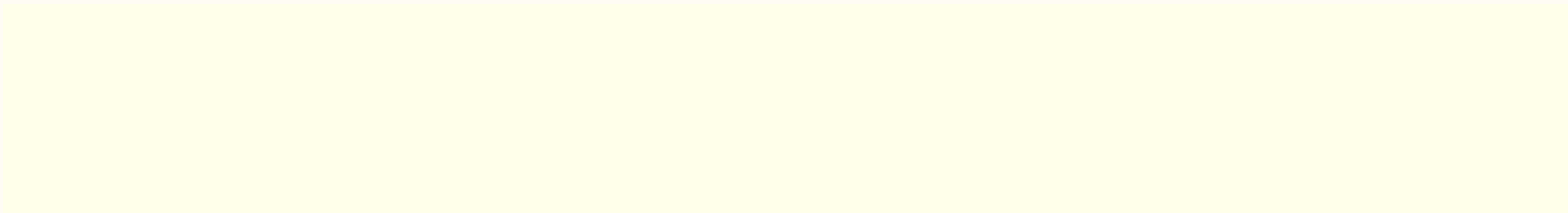


Constructive logic
is not classical logic

Constructive logic is not classical logic

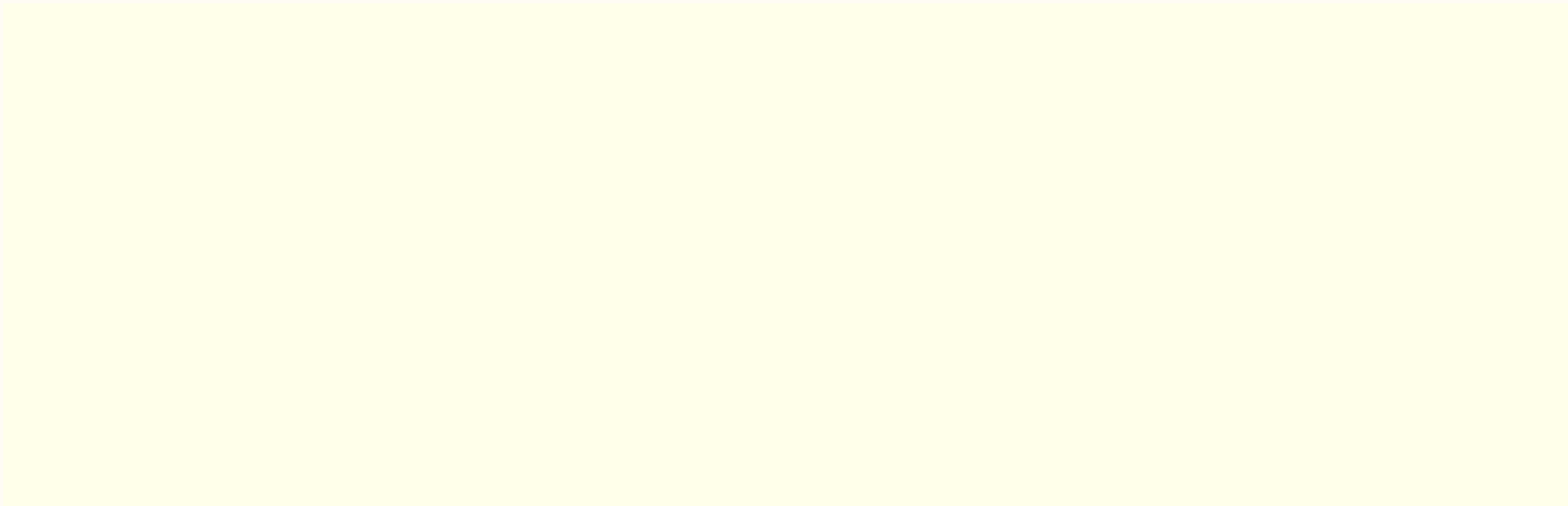
- Coq implements a constructive logic
- Every proof consists of evidence that is constructed
- You cannot assume the law of the excluded middle (proofs that appear out of thin air)
- **Truth tables may fail you!**
Especially if there are negations involved.

The following are **unprovable** in constructive logic (and therefore in Coq):



Building propositions with functions

Building propositions with functions



List membership example

- Computation cannot match on propositions
- Computations destruct types, not propositions

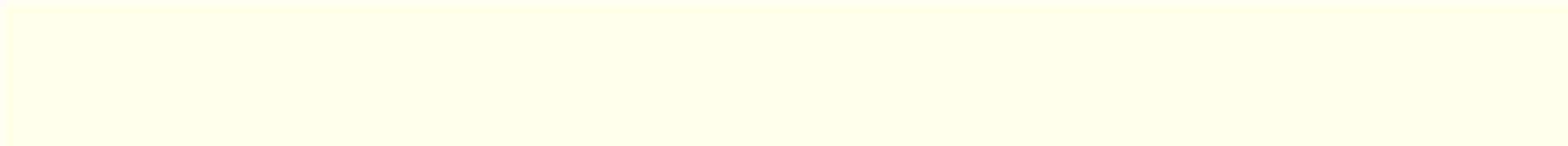
Building propositions with data structures (inductively)

Enumerated propositions

Recall enumerated types?

You can think of true as an enumerated type.

Many equivalent proofs



Many equivalent proofs

Yet, same as having one

- We can prove P with A or with B , we still just have P
- What happens when we do a case analysis on P ? Show when A holds, then show when B holds.

Falsehood

Falsehood in Coq is represented by an **empty** type.

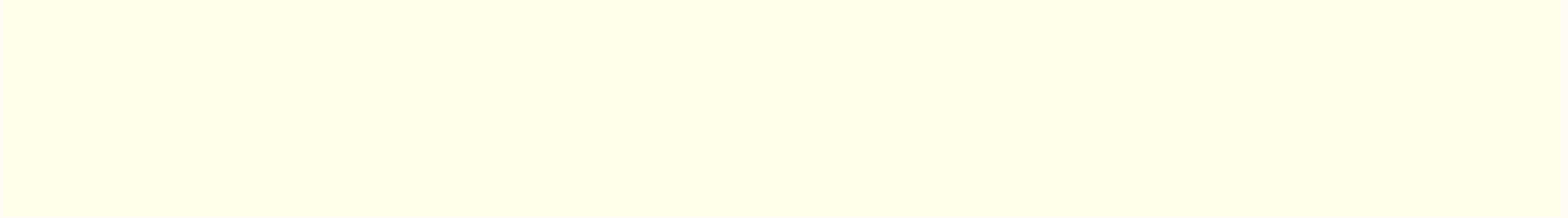
This explains why case analysis proves the following goal:

Composite inductive propositions

Disjunction

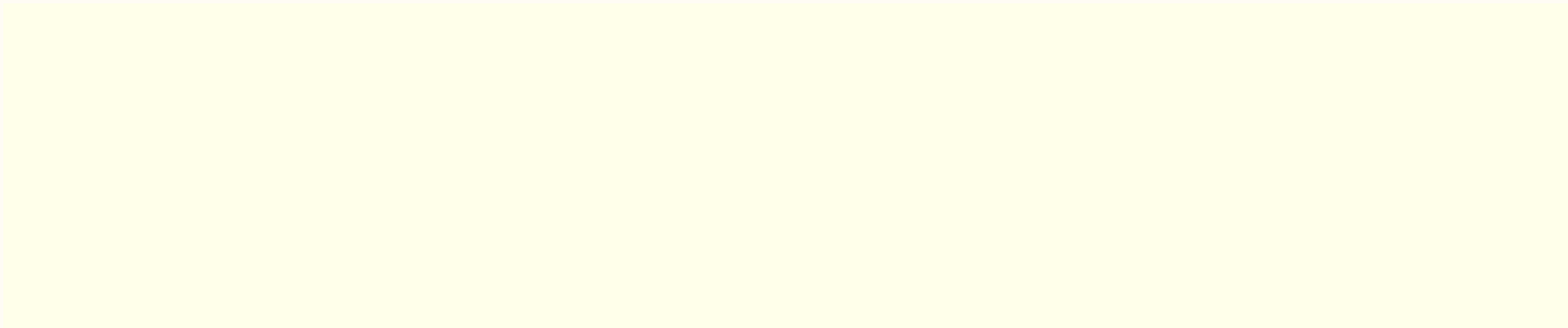


Conjunction



Adding parameters to predicates

Adding parameters to predicates



Alternative definition of Bar

Existential

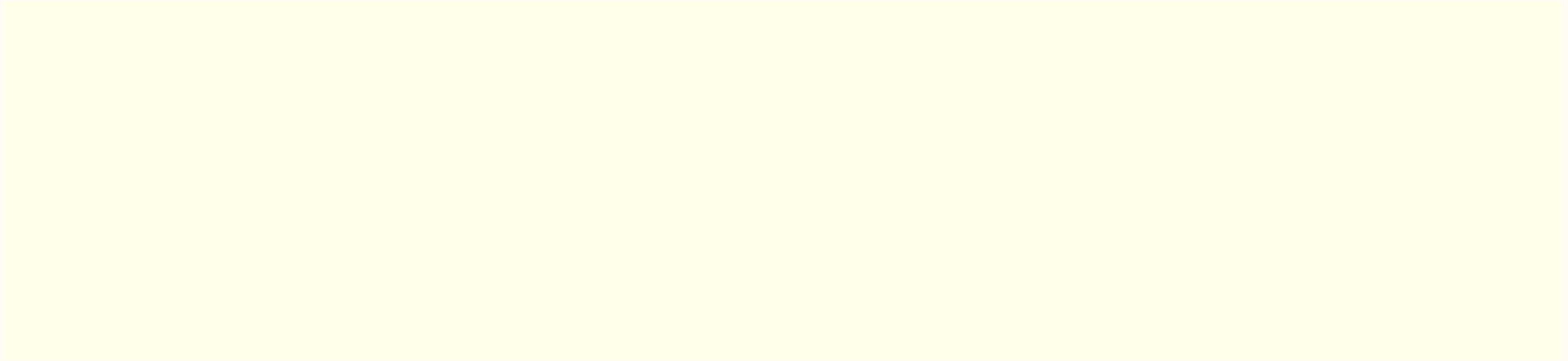


Recursive inductive propositions

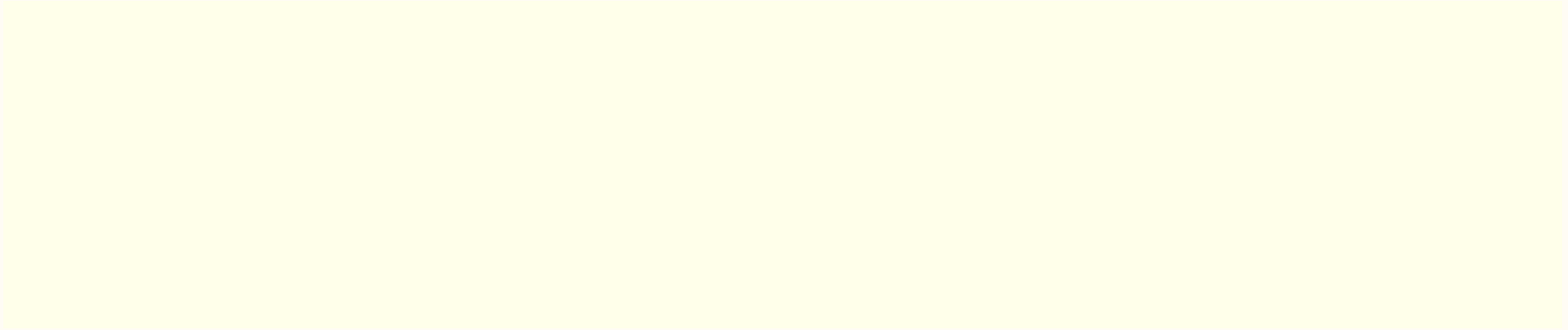
Defining inductively



Defining inductively



Fixed parameters in inductive propositions



Defining even numbers

