CS720

Logical Foundations of Computer Science

Lecture 9: Inductive propositions

Tiago Cogumbreiro
Inductive propositions

In lectures 7 and 8 we learned to write inductive definitions that compose other propositions (eg, \( \land \) takes holds two propositions)

Think about the following statement:

A product \( X \times Y \) is to a conjunction \( P \land Q \), the same way a list \( \text{list } X \) is to...?

Today we define inductive definitions that can "hold" an unbounded number of propositions.
Today we will learn...

- (recursive) inductive definitions
- implementing binary relations
- properties on binary relations
Exercise

Let us define even numbers inductively...

In the world of propositions, what is a signature of a number being even?
Exercise

Let us define even numbers inductively...

In the world of propositions, what is a signature of a number being even?

Inductive ev: nat → Prop
Exercise

Let us define even numbers inductively...

In the world of propositions, what is a signature of a number being even?

\[ \text{Inductive } \text{ev}: \text{nat} \rightarrow \text{Prop} \]

- 0 is even
- If \( n \) is even, then \( 2 + n \) is also even.
Inductively defined even

In Logic, the constructors `ev_0` and `ev_SS` of propositions can be called *inference rules*.

Which can be typeset as an inductive definition with the following notation:

\[
\begin{array}{c}
\text{ev}_0 \\
\text{ev}(0) \quad \equiv \quad \text{ev}_0 \\
\text{ev}(S(S(n))) \quad \equiv \quad \text{ev}_SS
\end{array}
\]
Proving that 4 is even

Backward style: From \textit{ev\_SS} we can conclude that 4 is even, if we can show that 2 is even, which follows from \textit{ev\_SS} and the fact that 0 is even (by \textit{ev\_0}).

Forward style: From the fact that 0 is even (\textit{ev\_0}), we use theorem \textit{ev\_SS} to show that 2 is even; so, applying theorem \textit{ev\_SS} to the latter, we conclude that 4 is even.
Reasoning about inductive propositions

Theorem evSS : forall n,
  ev (S (S n)) → ev n.

(Done in class.)
Example

\textbf{Goal} \sim ev 3.

\textit{(Done in class.)}
Proofs by induction

Goal \forall n, \text{ev } n \rightarrow \neg \text{ev } (S \ n).

(Done in class.)
Proofs by induction

Goal forall \( n \), ev \( n \rightarrow \neg ev (S\ n) \).

(Done in class.)

Notice the difference between induction on \( n \) and on judgment \( ev \ n \).
Relations in Coq

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

Notation "$n \leq m$" := (le n m).
Exercise

Goal $3 \leq 6$. 
Less-than

**Definition** \( \text{lt} \ (n \ m : \text{nat}) := \text{le} \ (S \ n) \ m. \)

How do we prove that this definition is correct?
Less-than

Definition \( \text{lt} \ (n \ m: \text{nat}) := \text{le} \ (S \ n) \ m. \)

How do we prove that this definition is correct?

Goal \( n \leq m \leftrightarrow \text{lt} \ n \ m \ \lor \ n = m. \)
Less-than

How can we define Less-Than inductively?
Less-than

How can we define Less-Than inductively?

```ocaml
Inductive lt : nat → nat → Prop :=
  | lt_base :forall n, lt n (S n)
  | lt_S :forall n m, lt n m → lt n (S m).
Notation "n < m" := (lt n m).
```

How do we prove that this definition is correct?
Exercises on Less-Than

Prove that

1. $<$ is transitive
2. $<$ is irreflexive
3. $<$ is asymmetric
4. $<$ is decidable