Logical Foundations of Computer Science

Lecture 10: Inductive propositions

Tiago Cogumbreiro
Summary

- How is Coq being used in research
- Exercises on inductive propositions
- Proofs by reflection
Projects that use Coq

- **Coq Proof of the Four Color Theorem** (Georges Gonthier, 2008) (Proposed in 1852, first proof in 1976 by Appel and Haken, proved in Coq in 2005). Four colors suffice to color any flat map.
- **CompCert** (2009): "CompCert is the first commercially available optimizing compiler that is formally verified, using machine assisted mathematical proofs, to be free from mis-compilation."
- **Verdi** (2015): Verdi is a framework from the University of Washington to implement and formally verify distributed systems.
- **Coq: The world's best macro assembler** (2013)
Projects that use Coq @ UMB

- **Deadlock Avoidance in Parallel Programs with Futures** (2017): formalized a task parallel programming model and the result that Data-Race-Freedom implies Deadlock-Freedom.
- **Checking Data-Race Freedom of GPU Kernels, Compositionally** (2021): formalized GPU program semantics and our data-race-freedom analysis.
- **Formalizing the Introduction to the Theory of Computation** (unpublished): decidability/undecidability results (e.g., halting problem, Rice's theorem, etc). Rice's Theorem was proved by Kleopatra Ginji, an undergraduate student here at UMB.
Proofs are code
Proofs by induction

Derivation versus data
Recall the definition on even numbers

```
Fixpoint evenb (n:nat) : bool :=
  match n with
  | 0       => true
  | S 0      => false
  | S (S n') => evenb n'
end.

Inductive ev : nat -> Prop :=
| ev_0   : ev 0
| ev_SS  : forall n : nat, ev n -> ev (S (S n)).
```
Let us prove that these two propositions are equivalent

**Theorem** evenb_to_ev:
  
  \[ \forall n, \quad \text{evenb } n = \text{true} \implies \text{ev } n. \]

  (* Hint: use \([\text{even_bool_prop}]\); no need for induction. *)

**Theorem** ev_to_evenb:
  
  \[ \forall n, \quad \text{ev } n \implies \text{evenb } n = \text{true}. \]

**Theorem** ev_iff_evenb:
  
  \[ \forall n, \quad \text{ev } n \iff \text{evenb } n = \text{true}. \]
Proofs by reflection
Reflection

We say that a proposition is reflected by a boolean value according to the following definition.

\[
\text{Inductive } \text{reflect} \ (P : \text{Prop}) : \text{bool} \rightarrow \text{Prop} := \\
| \text{ReflectT} : P \rightarrow \text{reflect} P \text{ true} \\
| \text{ReflectF} : \sim P \rightarrow \text{reflect} P \text{ false}.
\]

Theorem \(\text{iff\_reflect} : \forall P \ b, (P \leftrightarrow b = \text{true}) \rightarrow \text{reflect} P \ b\).

Theorem \(\text{reflect\_iff} : \forall P \ b, \text{reflect} P \ b \rightarrow (P \leftrightarrow b = \text{true}).\) (* Homework*)

Let us prove that \(\text{ev} \ n\) reflects \(\text{evenb} \ n\).

Lemma \(\text{ev\_reflect} : \forall n, \text{reflect} (\text{ev} \ n) (\text{evenb} \ n)\).
Recall proving that 6 is even

It is much easier to compute that 6 is even, than to derive a proposition for it.

Proof.
  apply ev_SS, ev_SS, ev_SS, ev_0.
Qed.

Theorem evenb_6: evenb 6 = true.
  reflexivity.
Qed.
Prove that 6 is even with reflection

Lemma reflect_true:
  forall P,
  reflect P true → P.
Proof.
  intros.
  inversion H.
  apply H0.
Qed.

Proof.
  apply (reflect_true (ev 6) (ev_reflect 6)).
Qed.
Proof by Reflection

The term reflection applies because we will need to translate Gallina propositions into values of inductive types representing syntax, so that Gallina programs may analyze them, and translating such a term back to the original form is called reflecting it.

- **Certified Programming with Dependent Types**
A bit more than what we have seen so far...
Reflecting the Logical And

**Lemma** reflect_and:

\[
\forall P \ b1 \ Q \ b2, \\
\text{reflect } P \ b1 \rightarrow \\
\text{reflect } Q \ b2 \rightarrow \\
\text{reflect } (P \land Q) \text{ (andb } b1 \ b2). 
\]
Reflecting the Logical Or

**Lemma reflect_or:**

\[
\forall P \ b1 \ Q \ b2, \\
\text{reflect } P \ b1 \rightarrow \\
\text{reflect } Q \ b2 \rightarrow \\
\text{reflect } (P \lor Q) \ (\text{orb } b1 \ b2).
\]
A mini-language of expressions

**Inductive** Lang :=

- **Eq**: nat → nat → Lang  (* x = n * )
- **Even**: nat → Lang  (* ev n * )
- **And**: Lang → Lang → Lang  (* P \( \land \) Q * )
- **Or**: Lang → Lang → Lang.  (* P \( \lor \) Q * )
Evaluate our mini-language

Fixpoint eval (exp:Lang) :=
    match exp with
    | Eq n m ⇒ beq_nat n m
    | Even n ⇒ evenb n
    | And l r ⇒ andb (eval l) (eval r)
    | Or l r ⇒ orb (eval l) (eval r)
    end.

Goal eval (Or (Even 3) (Eq 3 3)) = true.
   reflexivity.
Qed.
**Fixpoint** \( \text{as} \_ \text{prop} \ (\text{exp}:\text{Lang}) := \)  
\[
\begin{align*}
\text{match } \text{exp} \ \text{with} \\
| \text{Eq} \ n \ m & \Rightarrow n = m \\
| \text{Even} \ n & \Rightarrow \text{ev} \ n \\
| \text{And} \ l \ r & \Rightarrow \text{as} \_ \text{prop} \ l \ \text{\&\&} \ \text{as} \_ \text{prop} \ r \\
| \text{Or} \ l \ r & \Rightarrow \text{as} \_ \text{prop} \ l \ \text{\|\|} \ \text{as} \_ \text{prop} \ r
\end{align*}
\end{equation}
\]

**Goal** \( \text{as} \_ \text{prop} \ (\text{Or} \ (\text{Even} \ 3) \ (\text{Eq} \ 3 \ 3)) \).  
\[
(*) \ \text{ev} \ 3 \ \text{\|\|} \ 3 = 3 (*)
\]
\text{simp.}  
\text{right.}  
\text{reflexivity.}  
Qed.
Show that our language is reflective

**Lemma** reflect_lang:

\[
\forall p, \\
\text{reflect (as_prop } p) \text{ (eval } p). \\
\]

**Goal** ev 3 \lor 3 = 3.

assert (H:=reflect_lang (Or (Even 3) (Eq 3 3))).

apply reflect_true, H.

Qed.
Automating the translation

Ltac trans P :=

match P with
| ?P1 /\ ?P2 ⇒
  let t1 := trans P1 in
  let t2 := trans P2 in constr:(And t1 t2)
| ev ?x ⇒ constr:(Even x)
| ?P1 \/ ?P2 ⇒
  let t1 := trans P1 in
  let t2 := trans P2 in constr:(Or t1 t2)
| ?x = ?y ⇒ constr:(Eq x y)
end.

Goal ev 3 \/ 3 = 3.
let t := trans (ev 3 \/ 3 = 3) in
assert (H:= reflect_lang t).
Automating the translation

Ltac solve :=
match goal with
| [ |- ?P ] ⇒
  let t := trans P in
  let H := fresh "H" in
  assert (H := reflect_lang t);
  apply reflect_true, H
end.

Goal ev 3 ∨ 3 = 3.
solve.
Qed.
Summary on Proof by Reflection

- Reflection establishes a deep connection between a proposition and the function that decides it.
- We can leverage Ltac to automate trivial operations and build solvers (Not covered in this course.)
Exercises on Less-Than

Prove that

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

- We looked at Coq being used in research
- Exercises on inductive propositions
- A deep dive in proofs by reflection