Coqatoo

Generating Natural Language Versions of Coq Proofs

Andrew Bedford

Laval University

CoqPL 2018

Motivation

- Proofs can sometimes be hard to understand, particularly for less-experienced users

```
Lemma conj_imp_equiv : forall P Q R:Prop, (P /\ Q -> R) <-> (P -> Q -> R).
Proof.
    intros. split. intros H HP HQ. apply H. apply conj. assumption. assumption.
     intros H HPQ. inversion HPQ. apply H. assumption. assumption.
Qed.
```

Example

Input

CtCoq and its successor Pcoq are no longer available

```
conj_imp_equiv =
fun P Q R : Prop =>
conj (fun (H : P /\ Q -> R) (HP : P) (HQ : Q) => H (conj HP HQ))
    (fun (H : P -> Q -> R) (HPQ : P /\ Q) =>
    let H0 :=
        match HPQ with
        | conj H0 H1 => (fun (H2 : P) (H3 : Q) => H H2 H3) H0 H1
        end
        :
        R in
    H0)
        : forall P Q R : Prop, (P /\ Q -> R) <-> (P -> Q -> R)
```

- Verbosity

Coqatoo

Different approach

Overview of Coqatoo

Coqatoo's rewriting algorithm can be decomposed in three steps:

1. Information extraction
2. Proof tree construction
3. Tactic-based rewriting

Step 1: Information extraction

Coqatoo captures the intermediary proof states

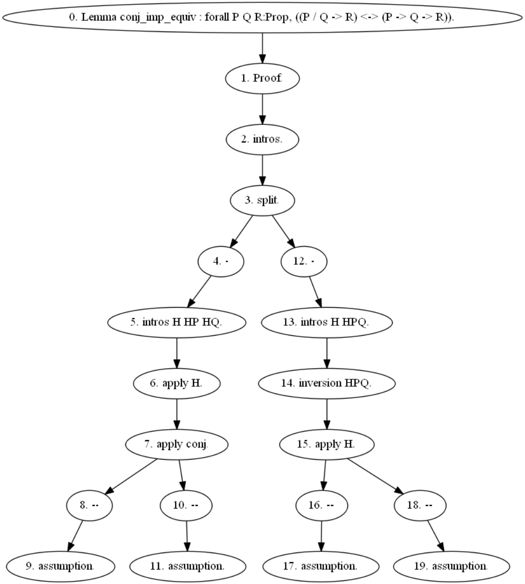
```
1 subgoal

============================
forall P Q R : Prop, (P /\ Q -> R) <-> (P -> Q -> R)
```

```
1 subgoal

P, Q, R : Prop
============================
(P /\ Q -> R) <-> (P -> Q -> R)
```

Step 2: Proof tree construction



Step 3: Tactic-based rewriting

Example

Output Modes

Example

Output (–mode plain)

Example

Output (–mode annotated)

```
Lemma conj_imp_equiv : forall P Q R:Prop, (P /\ Q -> R) <-> (P -> Q -> R).
Proof.
  (* Given any P, Q, R : Prop. Let us show that (P /\ Q -> R) <-> (P -> Q -> R)
     is true. *) intros.
  split.
  - (* Case (P /\ Q -> R) -> P -> Q -> R: *)
    (* Suppose that P, Q and P /\ Q -> R are true. Let us show that R is true.
     *) intros H HP HQ.
    (* By our hypothesis P /\ Q -> R, we know that R is true if P /\ Q  is true.
      *) apply H.
    apply conj.
    -- (* Case P: *)
       (* True, because it is one of our assumptions. *) assumption.
    -- (* Case Q: *)
       (* True, because it is one of our assumptions. *) assumption.
  - (* Case (P -> Q -> R) -> P /\ Q -> R: *)
    (* Suppose that P /\ Q and P -> Q -> R are true. Let us show that R is true.
      *) intros H HPQ.
    (* By inversion on P /\ Q, we know that P, Q are also true. *) inversion HPQ
     .
    (* By our hypothesis P -> Q -> R, we know that R is true if P and Q are true
     . *) apply H.
    -- (* Case P: *)
       (* True, because it is one of our assumptions. *) assumption.
    -- (* Case Q: *)
       (* True, because it is one of our assumptions. *) assumption.
Qed.
```

Lemma

(conj_imp_equiv) $\forall P, Q, R : Prop, (P \wedge Q \Rightarrow R) \Leftrightarrow (P \Rightarrow Q \Rightarrow R)$

Proof.

Given any $P, Q, R : Prop$. Let us show that $(P \wedge Q \Rightarrow R) \Leftrightarrow (P \Rightarrow Q \Rightarrow R)$ is true. $\square$

Demonstration

- It only works on proofs whose tactics are supported, while the approach of Coscoy et al. worked on any proof.
- It may require additional verifications to ensure that unecessary information (e.g., an assertion which isn't used) is not included in the generated proof.

- It enables us to more easily control the size and verbosity of the generated proof (one or two sentences per tactic by default).
- It maintains the order and structure of the user's original proof script; this is not necessarily the case in Coscoy et al.

Future work

- Increase the number of supported tactics
  - Goal: Software Foundations
- Add partial support for automation
- Integration with existing development environments
- Add a LaTeX output mode

Thank you!

github.com/andrew-bedford/coqatoo