# Coqatoo

## Generating Natural Language Versions of Coq Proofs

Andrew Bedford

Laval University

CoqPL 2018

# Motivation

Scenarios where having the natural-language version of a Coq proof could be useful:

1. Learning
2. Communicating
3. Documenting

You could manually write natural-language version of your proofs, but...

# Example

## Input

```
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

## Output (–mode text)

```
Given any P, Q and R : Prop. Let us show that (P /\ Q -> R) <-> (P -> Q -> R) is
      true.
- Case (P /\ Q -> R) -> P -> Q -> R:
    Suppose that P, Q, P /\ Q -> R are true. Let us show that R is true.
    By our hypothesis P /\ Q -> R, we know that R is true if P /\ Q are true.
    -- Case P:
       True, because it is one of our assumptions.
    -- Case Q:
       True, because it is one of our assumptions.
- Case (P -> Q -> R) -> P /\ Q -> R:
    Suppose that P /\ Q, P -> Q -> R are true. Let us show that R is true.
    By our hypothesis P -> Q -> R, we know that R is true if P, Q are true.
    -- Case P:
       True, because it is one of our assumptions.
    -- Case Q:
       True, because it is one of our assumptions.
```

# Example

## Output (–mode coq)

```
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.
```

# Example

## Output (–mode latex)

### Lemma

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

### Proof.

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

- Case $(P \wedge Q \Rightarrow R) \Rightarrow P \Rightarrow Q \Rightarrow R$:
  Suppose that $P, Q, P \wedge Q \Rightarrow R$ are true. Let us show that $R$ is true.
  By our hypothesis $P \wedge Q \Rightarrow R$, we know that $R$ is true if $P \wedge Q$ are true.
  – Case $P$:
    True, because it is one of our assumptions.
  – Case $Q$:
    True, because it is one of our assumptions.
- Case $(P \Rightarrow Q \Rightarrow R) \Rightarrow P \wedge Q \Rightarrow R$:
  Suppose that $P \wedge Q, P \Rightarrow Q \Rightarrow R$ are true. Let us show that $R$ is true.
  By our hypothesis $P \Rightarrow Q \Rightarrow R$, we know that $R$ is true if $P, Q$ are true.
  – Case $P$:
    True, because it is one of our assumptions.
  – Case $Q$:
    True, because it is one of our assumptions.

$\square$

- Can generate proofs in multiple languages
- Easy to add support for additional languages

```
assumption = True, because it is one of our assumptions.
bullet = %s Case <[{%s}]>:
destruct = Let us consider the different possible cases of <[{%s}]>.
intros.given = Given any <[{%s}]>.
intros.suppose = Suppose that <[{%s}]> are true.
intros.goal = Let us show that <[{%s}]> is true.
```

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

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

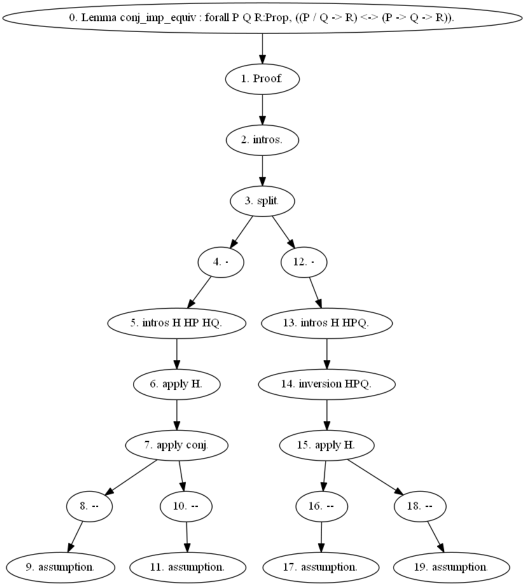
# Step 1: Information extraction

```
1 subgoal

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

intros.

```
1 subgoal

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

# Step 2: Proof tree construction



# Step 3: Tactic-based rewriting

Each supported tactic has its own set of rules. Take `intros` for example:

- If variables are introduced, then `Given any...`
- If hypotheses are introduced, then `Suppose that...`
- `Let us prove...`

Other example, if the `omega` tactic is used then `Using William Pugh's Omega algorithm, we can...`

# Related Work

Work done by Coscoy et al. in the 90's

```
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)
```

Disadvantages

- Only works on proofs whose tactics are supported, while the approach of Coscoy et al. worked on any proof.
- Won't work well if there is a lot of automation

Advantages

- More easy to control the size and verbosity of the generated proof (one or two sentences per tactic by default).
- Itaintains 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
    - Currently supports only a handful
    - Goal: Software Foundations
- Add partial support for automation
- Add support for other languages
- Integration with existing development environments

# Thank you!

github.com/andrew-bedford/coqatoo