

# Proofs in Propositional Logic <sup>1</sup>

Pierre Castéran

Suzhou, August 2011

---

<sup>1</sup>This lecture corresponds mainly to Chapter 3 : “Propositions and Proofs” and part of Chapter 5 : “Everyday Logic” of the book.

In this class and tomorrow , we introduce the reasoning techniques used in *Coq*, starting with a very simple fragments of logic, *propositional* (today) and *first order* (tomorrow) intuitionistic logic. We shall present :

- ▶ The logical formulas and the statements we want to prove,
- ▶ How to build proofs interactively.

## What is a proposition ?

- ▶  $2 < 3$
- ▶ 41 is a prime number
- ▶ 100 is a prime number
- ▶ `insertion sort` is a correct sorting algorithm
- ▶ etc.

Today, we will study a very simple class of propositions, within the frame of *propositional logic*.

## The Type `Prop`

In *Coq*, a predefined type, namely `Prop`, is inhabited by all logical propositions. For instance the true and false propositions are simply constants of type `Prop` :

Check `True`.

*True* : *Prop*

Check `False`.

*False* : *Prop*

Don't mistake the *proposition* `True` (resp. `False`) for the *boolean* `true` (resp. `false`), which belong to the *bool* *datatype*.

## Propositional Variables

We shall learn later how to build propositions for expressing such statements as  $5 \times 7 < 6^2$ , *41 is a prime number*, or *the list  $l$  is sorted*.

In this lecture we shall consider only abstract propositions build from *variables* using *connectives* :  $\vee$ ,  $\wedge$ ,  $\rightarrow$ , etc.

$it\_is\_raining \vee \sim it\_is\_raining$

$P \wedge Q \rightarrow Q \wedge P$

$\sim(P \vee Q) \rightarrow \sim(P \wedge Q)$

$it\_is\_raining$ ,  $P$  and  $Q$  are *propositional variables*.

## How to declare propositional variables

A propositional variable is just a variable of type `Prop`. So, you may just use the `Parameter` command for declaring a new propositional variable :

```
Parameter it_is_raining : Prop.  
Parameters P Q R : Prop.
```

Check `P`.

*P : Prop*

## Propositional Formulas

One can build propositions by using the following rules :

- ▶ Each variable of type **Prop** is a proposition,
- ▶ The constants **True** and **False** are propositions,
- ▶ if  $A$  and  $B$  are propositions, so are :
  - ▶  $A \leftrightarrow B$  (logical equivalence) (in ASCII :  $A \leftrightarrow B$ )
  - ▶  $A \rightarrow B$  (implication) (in ASCII :  $A \rightarrow B$ )
  - ▶  $A \vee B$  (disjunction) (in ASCII :  $A \vee B$ )
  - ▶  $A \wedge B$  (conjunction) (in ASCII :  $A \wedge B$ )
  - ▶  $\sim A$  (negation)

Like in many programming languages, connectors have *precedence* and *associativity* conventions :

The connectors  $\rightarrow$ ,  $\vee$ , and  $\wedge$  are *right-associative* : for instance  $P \rightarrow Q \rightarrow R$  is an abbreviation for  $P \rightarrow (Q \rightarrow R)$ .

The connectors are displayed below in order of increasing precedence :

$$\leftrightarrow, \rightarrow, \vee, \wedge, \sim$$

Check  $((P \rightarrow (Q \wedge P)) \rightarrow (Q \rightarrow P))$ .

$(P \rightarrow Q \wedge P) \rightarrow Q \rightarrow P : Prop$



## Logical Statements

In *Coq*, we may want to prove some *statements* like :

*“If the following propositions :*

$P \vee Q$

$\sim Q$

*hold, then the following proposition :*

$R \rightarrow R \wedge P$

*holds.”*

The propositions in blue are called *hypotheses*, and the proposition in red is the *conclusion* of the statement.

## The Sequent Notation

The (intuitionistic) sequent notation is a convenient mathematical notation for denoting a statement composed of a set of hypotheses  $\Gamma$  and a conclusion  $A$ . The notation is simply  $\Gamma \vdash A$ <sup>2</sup>

For instance, our previous statement may look like that :


$$\underbrace{P \vee Q, \sim Q}_{\text{hypotheses}} \vdash \underbrace{R \rightarrow R \wedge P}_{\text{conclusion}}$$

Another useful presentation is the following one :

$$\begin{array}{l} P \vee Q \\ \sim Q \end{array}$$

---


$$R \rightarrow R \wedge P$$

<sup>2</sup>The symbol  $\vdash$  is often called *turnstile*, or *corkscrew*. 

## Hypotheses and Goals

A *goal* is just a statement composed of a set of hypotheses  $\Gamma$  and a conclusion  $A$ . We use *Coq* for *solving the goal*, i.e. for building *interactively* a *proof* that the conclusion logically follows from the hypotheses. We shall use also the notation  $\Gamma \vdash A$ .

In *Coq* a goal is shown as below : each hypothesis is given a distinct name, and the conclusion is displayed under a bar which separates it from the hypotheses :

$H : P \vee Q$

$H0 : \sim Q$

-----  
 $R \rightarrow R \wedge P$

## A very quick demo

Let us show how to prove the previous goal :

The first step is to build a *context* from the two hypotheses. This can be done using a *section* (sort of named block).

```
Section my_first_proof.
```

```
Hypothesis H : P \ / Q.
```

```
Hypothesis H0 : ~ Q.
```

The screenshot shows the CoqIDE interface with the following content:

**CoqIDE** (Title Bar)

File Edit Navigation Try Tactics Templates Queries Display Compile Windows Help (Menu Bar)

Icons: Save, Close, Undo, Redo, Copy, Paste, Refresh, Stop, Help

Windows: \*Unnamed Buffer\* C3.v

**Left Pane (Editor):**

```
Parameters P Q R : Prop.
Section my_first_proof.
Hypothesis H : P V Q.
Hypothesis H0 : ~ Q.

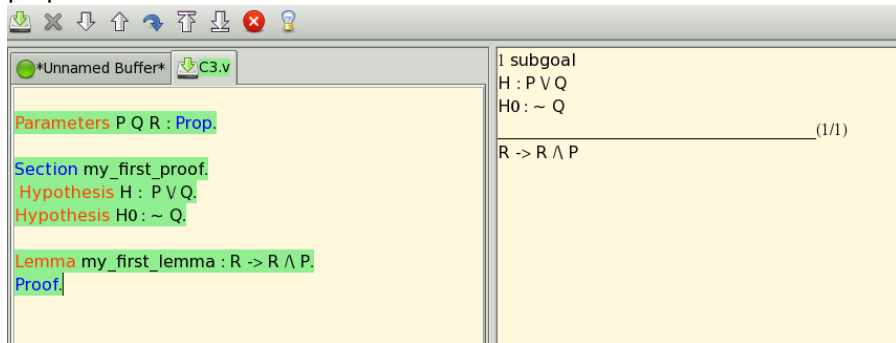
Check H.
```

**Right Pane (Goal View):**

```
H
: P V Q
```

**Status Bar:** Ready in my\_first\_proof | Line: 8 Char: 9 | CoqIDE started

Then *inside the section*, we tell *Coq* we want to prove some proposition.



The screenshot shows the Coq IDE interface. The top toolbar contains icons for file operations (save, close, undo, redo, copy, paste, delete) and a lightbulb icon. Below the toolbar, there are two tabs: '\*Unnamed Buffer\*' and 'C3.v'. The main editor area contains the following Coq code:

```
Parameters P Q R : Prop.  
  
Section my_first_proof.  
Hypothesis H : P V Q.  
Hypothesis H0 : ~ Q.  
  
Lemma my_first_lemma : R -> R ^ P.  
Proof.
```

On the right side, the 'subgoal' window displays the current goal:

```
1 subgoal  
H : P V Q  
H0 : ~ Q  
----- (1/1)  
R -> R ^ P
```

Then we use the *tactic* `intro` for introducing the hypothesis  $r : R$ .  
 The conclusion of the current goal becomes  $R \wedge P$ .

The screenshot shows a proof assistant interface with two main panels. The left panel contains code for defining a section, hypotheses, a lemma, and a proof step. The right panel shows the current goal state after the `intro r` command.

**Left Panel (Code):**

```

Parameters P Q R : Prop.

Section my_first_proof.
Hypothesis H : P V Q.
Hypothesis H0 : ~ Q.

Lemma my_first_lemma : R -> R ^ P.
Proof.
intro r.
  
```

**Right Panel (Goal State):**

```

1 subgoal
H : P V Q
H0 : ~ Q
r : R
----- (1/1)
R ^ P
  
```

For proving  $R \wedge P$ , we may prove  $R$ , *and* prove  $P$ . The tactic **split** generates two new *subgoals*.

The screenshot shows a proof editor window with a toolbar at the top. The main editor area contains the following text:

```

Parameters P Q R : Prop.

Section my_first_proof.
Hypothesis H : P V Q.
Hypothesis H0 : ~ Q.

Lemma my_first_lemma : R -> R ^ P.
Proof.
intro r.
split.
  
```

On the right side, a goal pane displays the state after the `split` tactic:

```

2 subgoals
H : P V Q
H0 : ~ Q
r : R
----- (1/2)
R
----- (2/2)
P
  
```

Note that the first subgoal is trivial, since  $R$  is assumed in the context of this subgoal. In this situation, one may use the tactic **exact r** or **assumption**.



The screenshot shows a proof assistant interface with two main panels. The left panel contains a proof script, and the right panel shows the current subgoal.

**Left Panel (Proof Script):**

```

Parameters P Q R : Prop.

Section my_first_proof.
Hypothesis H : P ∨ Q.
Hypothesis H0 : ~ Q.

Lemma my_first_lemma : R -> R ∧ P.
Proof.
intro r.
split.
exact r (* assumption *).

```

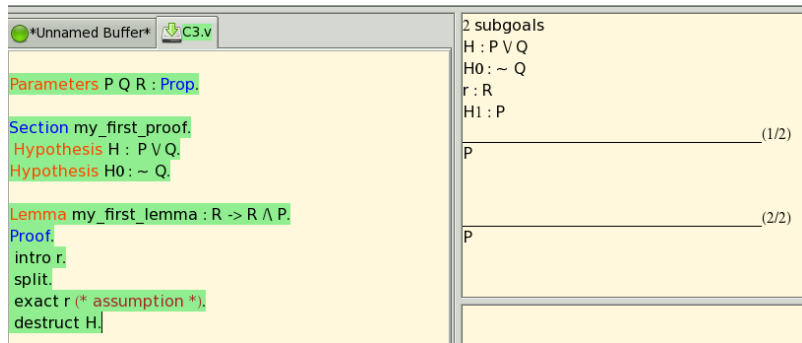
**Right Panel (Subgoal):**

```

1 subgoal
H : P ∨ Q
H0 : ~ Q
r : R
----- (1/1)
P

```

The displayed subgoal suggests to proceed to a *case analysis* on the hypothesis  $H$ . One may use the tactic call **destruct H** (or better : **destruct H as [Hp | Hq]**)



The screenshot shows a proof assistant interface with a code editor on the left and a goal pane on the right. The code editor contains the following text:

```
*Unnamed Buffer* C3.v  
Parameters P Q R : Prop.  
Section my_first_proof.  
Hypothesis H : P V Q.  
Hypothesis H0 : ~ Q.  
Lemma my_first_lemma : R -> R ^ P.  
Proof.  
  intro r.  
  split.  
  exact r (* assumption *).  
  destruct H.
```

The goal pane on the right displays the current state of the proof:

```
2 subgoals  
H : P V Q  
H0 : ~ Q  
r : R  
H1 : P  
----- (1/2)  
P  
----- (2/2)  
P
```

The first subgoal is immediately solved with **assumption**.

The screenshot shows a proof assistant interface with two main panels. The left panel contains a proof script, and the right panel shows the current goal.

**Left Panel (Proof Script):**

```

+Unnamed Buffer*  C3.v
Parameters P Q R : Prop.
Section my_first_proof.
Hypothesis H : P ∨ Q.
Hypothesis H0 : ~ Q.

Lemma my_first_lemma : R -> R ∧ P.
Proof.
intro r.
split.
exact r (* assumption *).
destruct H.
assumption.
    
```

**Right Panel (Goal Panel):**

```

1 subgoal
H : P ∨ Q
H0 : ~ Q
r : R
H1 : Q
----- (1/1)
P
    
```

The current context contains two mutually contradictory propositions :  $Q$  and  $\sim Q$ . The tactic call `absurd Q` helps to start a *proof by reduction to the absurd*.

\*Unnamed Buffer\* C3.v

Parameters P Q R : Prop.

Section my\_first\_proof.

Hypothesis H : P V Q.

Hypothesis H0 : ~ Q.

Lemma my\_first\_lemma : R -> R ^ P.

Proof.

intro r.

split.

exact r (\* assumption \*).

destruct H.

assumption.

absurd Q.

2 subgoals

H : P V Q

H0 : ~ Q

r : R

H1 : Q

---

(1/2)

~ Q

---

(2/2)

Q

```
Section my_first_proof.  
Hypothesis H : P ∨ Q.  
Hypothesis H0 : ~ Q.  
  
Lemma my_first_lemma : R -> R ∧ P.  
Proof.  
intro r.  
split.  
exact r (* assumption *).  
destruct H.  
assumption.  
absurd Q.  
assumption.  
assumption.  
Qed.
```

```
Section my_first_proof.
```

```
Hypothesis H : P ∨ Q.
```

```
Hypothesis H0 : ~ Q.
```

```
Lemma my_first_lemma : R -> R ∧ P.
```

```
Proof.
```

```
intro r.
```

```
split.
```

```
exact r (* assumption *).
```

```
destruct H.
```

```
assumption.
```

```
absurd Q.
```

```
assumption.
```

```
assumption.
```

```
Qed.
```

```
Check my_first_lemma.
```

```
my_first_lemma
: R -> R ∧ P
```

When we close the section `my_first_proof` the *local* hypotheses disappear :

```
Check my_first_lemma.  
End my_first_proof.  
Check my_first_lemma.  
Check H.
```

Error: The reference H was not found in the current environment.

**Important note** : The scope of an hypothesis is always limited to its enclosing section. If we need assumptions with *global* scope, declare them with the command

Axiom Axi : A.

Note that the statement of our lemma is enriched with the hypotheses that were used in its proof :

```
Check my_first_lemma.
```

```
End my_first_proof.
```

```
Check my_first_lemma.
```

```
my_first_lemma  
: P V Q -> ~ Q -> R -> R ∧ P
```



## Structure of an interactive proof (1)

Lemma  $L$ :  $A$ .

Proof.

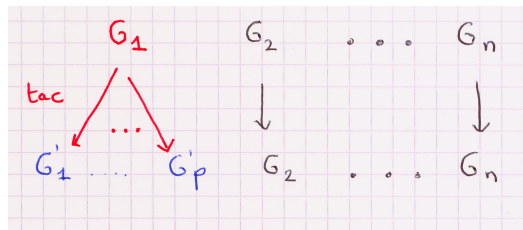
*sequence of tactic applications*

Qed.

**Notes** : The keyword Lemma may be replaced by Theorem, Fact, Remark, etc. The name  $L$  must be *fresh*.

A goal is immediately built, the conclusion of which is the proposition  $A$ , and the context of which is build from the currently active hypotheses.

## Structure of an interactive proof (2)



Note that  $p$  may be 0, 1, or any number greater or equal than 2!

## When is an interactive proof finished ?

The number of subgoals that remain to be solved decreases only when some tactic application generates 0 new subgoals.

The interactive search of a proof is finished when there remain no subgoals to solve. The `Qed` command makes *Coq* do the following actions :

1. build a proof term from the history of tactic invocations,
2. check whether this proof is correct,
3. register the proven theorem.

## Basic tactics for minimal propositional logic

In a first step, we shall consider only formulas built from propositional variables and the implication connective  $\rightarrow$ . It is a good framework for learning basic concepts on tactics in *Coq*.

## The tactic **assumption**

The tactic **assumption** can be used everytime the current goal has the following form :

...

**H**:A

...

-----

**A**

- ▶ Note that one can use **exact H**, or **trivial** in the same situation.
- ▶ This tactic is associated to the following *inference rule* :

$$\frac{A \in \Gamma}{\Gamma \vdash A} \text{ assumption}$$

## Introduction tactic for the implication

Let us consider a goal  $\Gamma \vdash A \rightarrow B$ . The tactic `intro H` (where  $H$  is a fresh name) transforms this goal into  $\Gamma, H : A \vdash B$ .

- ▶ This tactic is applicable when *the conclusion* of the goal is an implication.
- ▶ This tactic corresponds to the *implication introduction rule*

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \text{imp}_i$$

- ▶ The multiple introduction tactic `intros H1 H2 ... Hn` is a shorthand for `intro H1; intro H2; ...; intro Hn`.

## Elimination tactic for the implication (modus ponens)

Let us consider a goal of the form  $\Gamma \vdash A$ . If  $H : A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n \rightarrow A$  is an hypothesis of  $\Gamma$  or an already proven theorem, then the tactic **apply**  $H$  generates  $n$  new subgoals,  $\Gamma \vdash A_1, \dots, \Gamma \vdash A_n$ .

This tactic corresponds to the following inference rules :

$$\frac{\overline{\Gamma \vdash B \rightarrow A} \quad \overline{\Gamma \vdash B}}{\Gamma \vdash A} \text{ mp}$$

$$\frac{\overline{\Gamma \vdash A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n \rightarrow A} \quad \overline{\Gamma \vdash A_1} \quad \overline{\Gamma \vdash A_2} \quad \dots \quad \overline{\Gamma \vdash A_n}}{\Gamma \vdash A}$$

## A simple example

Section Propositional\_Logic.

Variables P Q R : Prop.

Lemma imp\_dist : (P → (Q → R)) → (P → Q) → P → R.

Proof.

*1 subgoal*

*P : Prop*

*Q : Prop*

*R : Prop*

-----  
*(P → Q → R) → (P → Q) → P → R*

intros H H0 p.



*1 subgoal:*

*$P : Prop$*

*$Q : Prop$*

*$R : Prop$*

*$H : P \rightarrow Q \rightarrow R$*

*$H0 : P \rightarrow Q$*

*$p : P$*

---

*$R$*

apply H.

2 subgoals:

$P : Prop$

$Q : Prop$

$R : Prop$

$H : P \rightarrow Q \rightarrow R$

$H0 : P \rightarrow Q$

$p : P$

---

$P$

subgoal 2 is:

$Q$

assumption.

*1 subgoal:*

*$P : Prop$*

*$Q : Prop$*

*$R : Prop$*

*$T : Prop$*

*$H : P \rightarrow Q \rightarrow R$*

*$H0 : P \rightarrow Q$*

*$p : P$*

---

*$Q$*

`apply H0;assumption.`

*Proof completed*

Qed.

*imp\_dist is defined*

Check imp\_dist.

*imp\_dist*

*: (P → Q → R) → (P → Q) → P → R*

Print imp\_dist.

*imp\_dist =*

*fun (H : P → Q → R) (H0 : P → Q) (H1 : P) ⇒ H H1 (H0 H1)*

*: (P → Q → R) → (P → Q) → P → R*

We notice that the internal representation of the proof we have just built is a term whose type is the theorem statement.

It is possible, but not usual, to build directly proof terms, considering that a proof of  $A \rightarrow B$  is just a function which maps any proof of  $A$  to a proof of  $B$ .

Definition `imp_trans`  $(H:P \rightarrow Q) (H0:Q \rightarrow R) (p:P) : R$   
`:= H0 (H p).`

Check `imp_trans`.

*$imp\_trans : (P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow P \rightarrow R.$*

## Using the section mechanism

Another way to prove an implication  $A \rightarrow B$  is to prove  $B$  inside a *section* which contains a hypothesis assuming  $A$ , *if the proof of  $B$  uses truly the hypothesis assuming  $A$* . This scheme generalizes to any number of hypotheses  $A_1, \dots, A_n$ .

```
Section Imp_trans.
```

```
Hypothesis H : P → Q.
```

```
Hypothesis H0 : Q → R.
```

```
Lemma imp_trans' : P → R.
```

```
(* Proof skipped, uses H and H0 *)
```

```
End Imp_trans.
```

```
Check imp_trans'.
```

```
imp_trans : (P → Q) → (Q → R) → P → R
```

## Introduction and Elimination Tactics

Let us consider again the goal below :

$$H : R \rightarrow P \ \wedge \ Q$$

$$H0 : \sim (R \ \wedge \ Q)$$

-----  
$$R \rightarrow P$$

We colored in **blue** the main connective of the conclusion, and in **red** the main connective of each hypothesis.

To solve this goal, we can use an **introduction tactic** associated to the **main connective** of the conclusion, or an **elimination tactic** on some hypothesis.

## Propositional Intuitionistic Logic

We will now add to Minimal Propositional Logic introduction and elimination rules and tactics for the constants **True** and **False**, and the connectives **and** ( $\wedge$ ), **or** ( $\vee$ ), **iff** ( $\leftrightarrow$ ) and **not** ( $\sim$ ).



## Introduction rule for **True**

In any context  $\Gamma$  the proposition **True** is immediately provable (thanks to a predeclared constant  $I : \text{True}$ ).

Practically, any goal  $\Gamma \vdash^2 \text{True}$  can be solved by the *tactic* **trivial** :

$H : R \rightarrow P \vee Q$

$H0 : \sim(R \wedge Q)$

-----  
**True**

trivial.

There is no useful elimination rule for **True**.

## Falsity

The elimination rule for the constant **False** implements the so-called *principle of explosion*, according to which “any proposition follows from a contradiction”.

$$\frac{\Gamma \vdash \text{False}}{\Gamma \vdash A} \text{False}_e$$

There is an elimination tactic for **False** :

*H : False*

-----

*2 = 3.*

destruct H.

In order to avoid to prove contradictions, there is no introduction rule nor introduction tactic for **False**.

## Introduction rule and tactic for conjunction

A proof of a sequent  $\Gamma \vdash A \wedge B$  is composed of a proof of  $\Gamma \vdash A$  and a proof of  $\Gamma \vdash B$ .

$$\frac{\frac{\dots}{\Gamma \vdash A} \quad \frac{\dots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B} \text{ conj}$$

Coq's tactic **split**, splits a goal  $\Gamma \vdash A \wedge B$  into two *subgoals*  $\Gamma \vdash A$  and  $\Gamma \vdash B$ .

## Conjunction elimination

**Rule :**

$$\frac{\frac{\dots}{\Gamma \vdash A \wedge B} \quad \frac{\dots}{\Gamma, A, B \vdash C}}{\Gamma \vdash C} \text{and\_e}$$

**Associated tactic :**

Let us consider a goal  $\Gamma \vdash C$ , and  $H : A \wedge B$ . Then the tactic **destruct H as [H1 H2]** generates the new goal

$$\Gamma, H1 : A, H2 : B \vdash C$$

## Example

Lemma and\_comm :  $P \wedge Q \rightarrow Q \wedge P$ .

Proof.

intro H.

*1 subgoal*

*P : Prop*

*Q : Prop*

*H : P ∧ Q*

-----  
*Q ∧ P*

destruct H as [H1 H2].

*1 subgoal*

*P : Prop*

*Q : Prop*

*H1 : P*

*H2 : Q*

---

*Q ∧ P*

split.

*2 subgoals*

*P : Prop*

*Q : Prop*

*H1 : P*

*H2 : Q*

---

*Q*

*subgoal 2 is:*

*P*

...

## Introduction rules and tactics for disjunction

There are two introduction rules for  $\vee$  :

$$\frac{\overline{\Gamma \vdash A}}{\Gamma \vdash A \vee B} \text{ or\_intro\_l}$$

$$\frac{\overline{\Gamma \vdash B}}{\Gamma \vdash A \vee B} \text{ or\_intro\_r}$$

The tactic **left** is associated to *or\_intro\_l*, and the tactic **right** to *or\_intro\_r*.



## Elimination rule and tactic for disjunction

$$\frac{\overline{\Gamma \vdash A \vee B} \quad \overline{\Gamma, A \vdash C} \quad \overline{\Gamma, B \vdash C}}{\Gamma \vdash C} \text{ or\_e}$$

Let us consider a goal  $\Gamma \vdash C$ , and  $H : A \vee B$ . Then the tactic **destruct H as [H1 | H2]** generates two new subgoals :

$$\begin{aligned} &\Gamma, H1 : A \vdash C \\ &\Gamma, H2 : B \vdash C \end{aligned}$$

This tactic implements the *proof by cases* paradigm.

## A combination of left, right and destruct

Consider the following goal :

$P : Prop$

$Q : Prop$

$H : P \vee Q$

-----  
 $Q \vee P$

We have to choose between an introduction tactic on the conclusion  $Q \vee P$ , or an elimination tactic on the hypothesis  $H$ .

If we start with an introduction tactic, we have to choose between `left` and `right`. Let us use `left` for instance :

```
left.
```

*P : Prop*

*Q : Prop*

*H : P ∨ Q*

-----  
*P*

This is clearly a dead end. Let us come back to the previous step (with command `Undo` (`coqtop` or using `Coqide`'s navigation menu)).

destruct H as [H0 | H0].

*two subgoals*

*P : Prop*

*Q : Prop*

*H : P ∨ Q*

*H0 : P*

---

*Q ∨ P*

*subgoal 2 is :*

*Q ∨ P*

right;assumption.

left;assumption.

Qed.

## Negation

In *Coq*, the negation of a proposition  $A$  is represented with the help of a constant `not`, where `not A` (also written  $\sim A$ ) is defined as the implication  $A \rightarrow \text{False}$ .

The tactic `unfold not` allows to expand the constant `not` in a goal, but is seldom used.

The introduction tactic for  $\sim A$  is the introduction tactic for  $A \rightarrow \text{False}$ , i.e. `intro H` where  $H$  is a fresh name. This tactic pushes the hypothesis  $H : A$  into the context and leaves `False` as the proposition to prove.

## Elimination tactic for the negation

The elimination tactic for negation implements some kind of reasoning by contradiction (absurd).

Let us consider a goal  $\Gamma, H : \sim B \vdash A$ . Then the tactic **destruct H** generates a new subgoal  $\Gamma \vdash B$ .

**Note :** Using **case H** instead of **destruct H** allows to keep the hypothesis H in the context (we may need to use it later in the proof).

**Note** : In situation like below :

$H : C \rightarrow B \rightarrow \sim A$

-----  
False

You can use simply **apply H** (because  $\sim A$  is just  $A \rightarrow \text{False}$ )

## Logical equivalence

Let  $A$  and  $B$  be two propositions. Then the formula  $A \leftrightarrow B$  (read “A iff B”) is defined as the conjunction  $(A \rightarrow B) \wedge (B \rightarrow A)$ .

The introduction tactic for  $\leftrightarrow$  is **split**, which associates to any goal  $\Gamma \vdash A \leftrightarrow B$  the subgoals  $\Gamma \vdash A \rightarrow B$  and  $\Gamma \vdash B \rightarrow A$ .

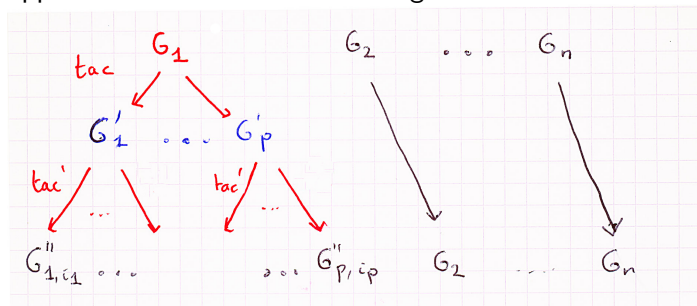
The elimination tactic for  $\leftrightarrow$  is **destruct H as [H1 H2]** where  $H$  is an hypothesis of type  $A \leftrightarrow B$  and  $H1$  and  $H2$  are “fresh” names. This tactic adds to the current context the hypotheses  $H1 : A \rightarrow B$  and  $H2 : B \rightarrow A$ .



## Simple tactic composition

Let  $tac$  and  $tac'$  be two tactics.

The tactic  $tac;tac'$  applies  $tac'$  to each subgoal generated by the application of  $tac$  to the first subgoal.



Lemma `and_comm'` :  $P \wedge Q \rightarrow Q \wedge P$ .

Proof.

```
intro H;destruct H as [H1 H2].
```

```
H1 : P
```

```
H2 : Q
```

```
-----  
Q ∧ P
```

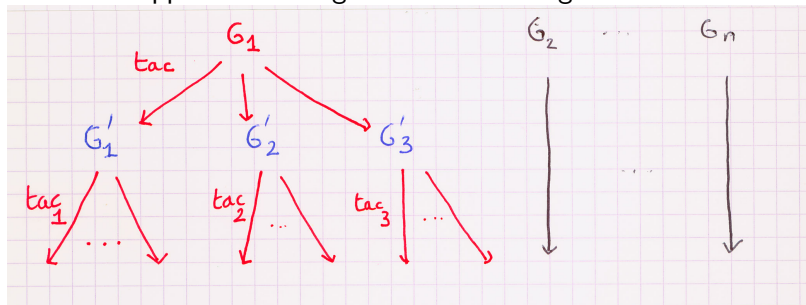
```
split;assumption.
```

```
(* assumption has been applied to each one of the  
   two subgoals generated by split *)
```

Qed.

## Another composition operator

The tactic composition  $tac;[tac1|tac2|...]$  is a generalization of the simple composition operator, in situations where the same tactic cannot be applied to each generated new subgoal.



## The `assert` tactic (forward chaining)

Let us consider some goal  $\Gamma \vdash A$ , and  $B$  be some proposition.

The tactic `assert (H : B)`, generates two subgoals :

1.  $\Gamma \vdash B$
2.  $\Gamma, H : B \vdash A$

This tactic can be useful for avoiding proof duplication inside some interactive proof. *Notice that the scope of the declaration  $H : B$  is limited to the second subgoal. If a proof of  $B$  is needed elsewhere, it would be better to prove a lemma stating  $B$ .*

**Remark :** Sometimes the overuse of `assert` may lead to verbose developments (remember that the user has to type the statement  $B!$ )

Section assert.

Hypotheses (H :  $P \rightarrow Q$ )

(H0 :  $Q \rightarrow R$ )

(H1 :  $(P \rightarrow R) \rightarrow T \rightarrow Q$ )

(H2 :  $(P \rightarrow R) \rightarrow T$ ).

Lemma L8 : Q.

(\* A direct backward proof would need to prove twice  
the proposition  $(P \rightarrow R)$  \*)

The tactic `assert (PR :  $P \rightarrow R$ )` generates two subgoals :

*2 subgoals*

$H : P \rightarrow Q$

$H0 : Q \rightarrow R$

$H1 : (P \rightarrow R) \rightarrow T \rightarrow Q$

$H2 : (P \rightarrow R) \rightarrow T$

---

$P \rightarrow R$

$Q$

`intro p;apply H0;apply H;assumption.`

$H : P \rightarrow Q$  $H0 : Q \rightarrow R$  $H1 : (P \rightarrow R) \rightarrow T \rightarrow Q$  $H2 : (P \rightarrow R) \rightarrow T$  $PR : P \rightarrow R$ 

-----

 $Q$ 

apply H1; [ assumption | apply H2; assumption].

Qed.

## A more clever use of destruct

The tactic **destruct H** works also when  $H$  is an hypothesis (or axiom, or already proven theorem), of type  $A_1 \rightarrow A_2 \dots \rightarrow A_n \rightarrow A$  where the main connective of  $A$  is  $\vee$ ,  $\wedge$ ,  $\sim$ ,  $\leftrightarrow$  or **False**.

In this case, new subgoals of the form  $\Gamma \vdash A_i$  are also generated (in addition to the behaviour we have already seen).



Section Ex5.

Hypothesis H :  $T \rightarrow R \rightarrow P \ \backslash / \ Q$ .

Hypothesis H0 :  $\sim (R \ /\ \ Q)$ .

Hypothesis H1 : T.

Lemma L5 :  $R \rightarrow P$ .

Proof.

intro r.

Destructuring H will produce four subgoals :

- ▶ prove T
- ▶ prove R
- ▶ assuming P, prove P,
- ▶ assuming Q, prove P.

(\* Let us *try* to apply *assumption*  
to each of these four subgoals \*)  
destruct H as [H2 | H2] ;try assumption.

*1 subgoal*

*$H : T \rightarrow R \rightarrow P \vee Q$*

*$H0 : \sim (R \wedge Q)$*

*$H1 : T$*

*$r : R$*

*$H2 : Q$*

-----  
 *$P$*

destruct H0; split;assumption.

Qed.

## A variant of intros

Lemma L2 : (P\Q) /\ ~P -> Q.

Proof.

```
intros [[p | q] p'].
```

*2 subgoals*

*p : P*

*p' : ~ P*

---

*Q*

*subgoal 2 is:*

*Q*

```
destruct p';trivial.
```

*1 subgoal*

*$q : Q$*

*$p' : \sim P$*

-----

*$Q$*

assumption.

Qed.

## An automatic tactic for intuitionistic propositional logic

The tactic `tauto` solves goals which are instances of intuitionistic propositional tautologies.

Lemma L5' :  $(R \rightarrow P \ \backslash / \ Q) \rightarrow \sim(R \ /\ \ Q) \rightarrow R \rightarrow P$ .

Proof.

`tauto.`

Qed.

The tactic `tauto` doesn't solve goals that are only provable in classical propositional logic (*i.e.* intuitionistic + the rule of excluded middle  $\vdash A \ \backslash / \ \sim A$ ). Here are some examples :

$$P \ \backslash / \ \sim P$$

$$(P \rightarrow Q) \leftrightarrow (\sim P \ \backslash / \ Q)$$

$$\sim(P \ /\ \ Q) \leftrightarrow \sim P \ \backslash / \ \sim Q$$

$$((P \rightarrow Q) \rightarrow P) \rightarrow P \quad (\textit{Peirce's formula})$$