Proofs in Propositional Logic[1]

Pierre Castéran

Beijing, August 2010

1. This lecture corresponds mainly to Chapter 3 : "Propositions and Proofs" and part of Chapter 5 : "Everyday Logic" of the book.

In this class, we introduce the reasoning techniques used in *Coq*, starting with a very reduced fragment of logic, *propositional intuitonistic logic*.
We shall present :

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

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 with Yves 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* : \/, /\, →, etc.

```
it_is_raining \/ ~ it_is_raining
P /\ Q → Q /\ P
~(P \/ Q) → ~(P /\ 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* <-> *B*)
  - $A \rightarrow B$ (implication) (in ASCII : *A* -> *B*)
  - *A* \/ *B* (disjunction) (in ASCII : *A* \/ *B*)
  - *A* /\ *B* (conjunction) (in ASCII : *A* /\ *B*)
  - $\sim A$ (negation)

Like in many programming languages, connectors have *precedence* and *associativity* conventions :
The connectors $\rightarrow$, \/, and /\ 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 :

$$\leftrightarrow, \quad \rightarrow, \quad \backslash/, \quad /\backslash, \quad \sim$$

```
Check ((P → (Q /\ P)) → (Q → P)).
```

(P → Q /\ P) → Q → P   : Prop

Logical Statements

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

"If the following propositions :

```
P \/ Q
 ~ Q
```

hold, then the following proposition :

```
R → R /\ 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 Γ and a conclusion A. The notation is simply $\Gamma \vdash A$ [2]
For instance, our previous statement may look like that :

$$\underbrace{P \backslash/ Q,\ \sim Q}_{hypotheses} \vdash \underbrace{R \rightarrow R /\backslash P}_{conclusion}$$

Another useful presentation is the following one :

```
P \/ Q
~Q
-----------------------------
R → R /\ P
```

2. The symbol $\vdash$ is often called *turnstyle*, or *corkscrew*.

Hypotheses and Goals

A *goal* is just a statement composed of a set of hypotheses Γ 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 $\Gamma \overset{?}{\vdash} A$.
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 :

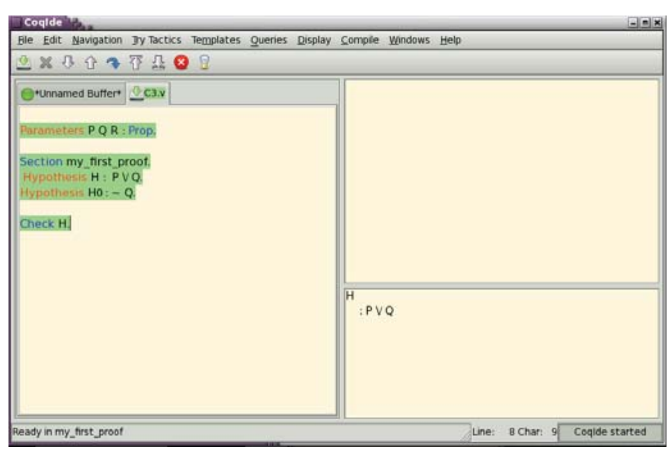
H : P \/ Q
H0 : ~ Q
\-----------------------
R → R /\ P

A very quick demo

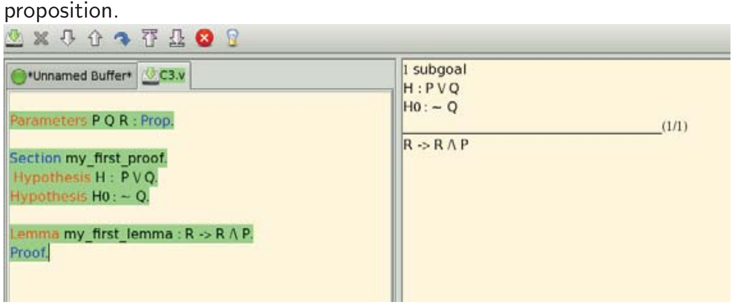
Let us show how to prove the previous goal :
The first step is to build a *context* from the two hypotheses. This can be done using a *section* (sort of named block).

```
Section my_first_proof.
 Hypothesis H : P \/ Q.
 Hypothesis H0 : ~ Q.
 Check H.
```

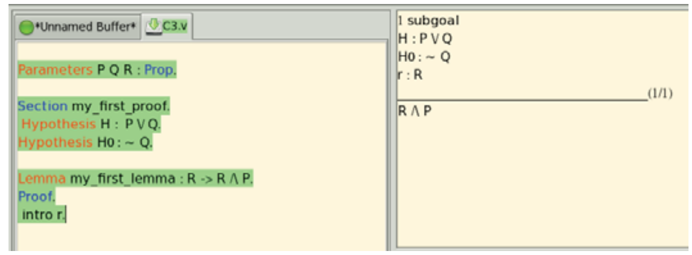
H : P \/ Q



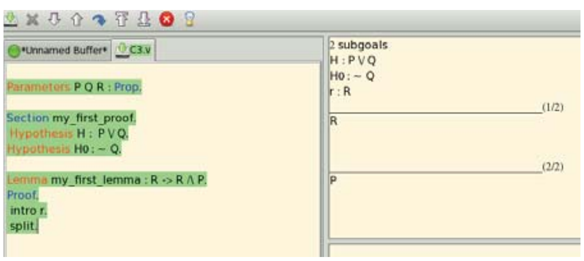
Then *inside the section*, we tell *Coq* we want to prove some proposition.



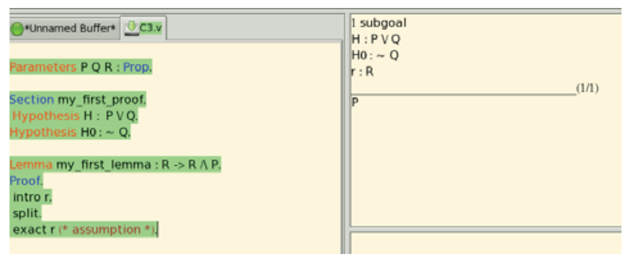
Then we use the *tactic* intro for introducing the hypothesis `r :R`. The conclusion of the current goal becomes `R /\ P`.



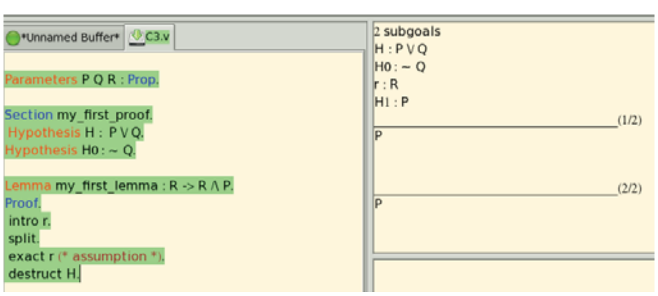
For proving `R /\ P`, we may prove `R`, *and* prove `P`. The tactic split generates two new *subgoals*.



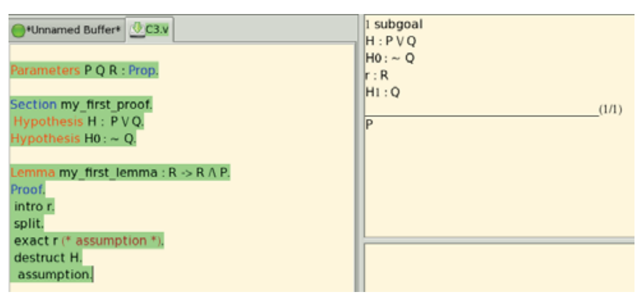
Note that the first subgoal is trivial, since `R` is assumed in the context of this subgoal. In this situation, one may use the tactic exact r or assumption.



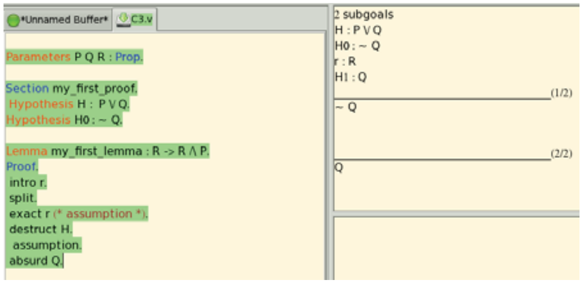
The displayed subgoal suggests to proceed to a *case analysis* on the hypothesis `H`. One may use the tactic call destruct H (or better : destruct H as [Hp | Hq])

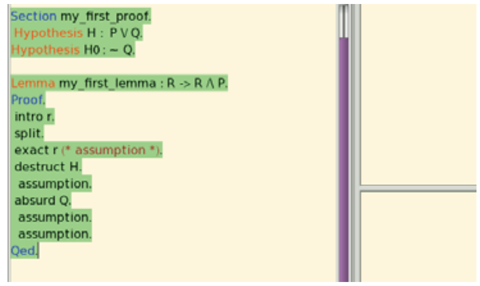


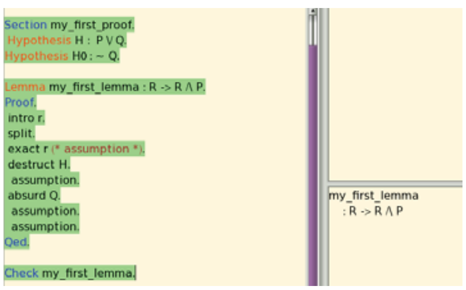
The first subgoal is immediately solved with assumption.



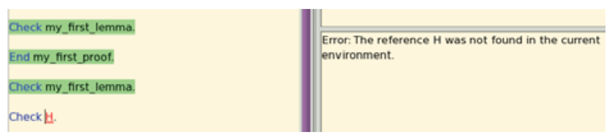
The current context contains two mutually contradictory propositions : Q and ∼Q. The tactic call absurd Q helps to start a *proof by reduction to the absurd*.







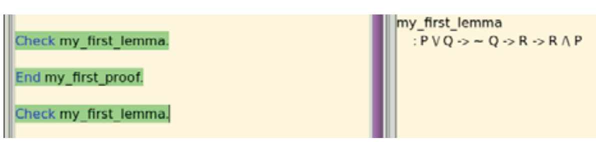
When we close the section `my_first_proof` the *local* hypotheses disappear :



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 Axm :` A.

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



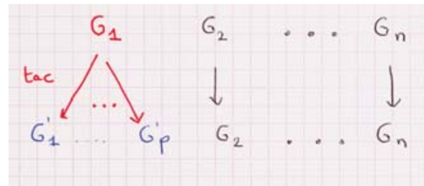
Structure of an interactive proof (1)

`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)

- In general, at each step of an interactive proof, a finite sequence of *subgoals* $G_1, G_2, \ldots, G_n$ must be solved.
- The basic tool for interactively solving a goal $G = \Gamma \vdash^{?} A$ is called a *tactic*, which is a command typed by the user.
- An elementary step of an interactive proof has the following form : The user tries to apply a tactic to (by default) the first subgoal G_1,
  - This application may fail, in which case the state of the proof doesn't change,
  - or this application generates a finite sequence (possibly empty) of new subgoals, which replaces the previous one.



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 miminal 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* :

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

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*

$$\frac{\begin{array}{c}\ldots \\ \hline \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \rightarrow B}\ \textit{imp\_i}$$

- The multiple introduction tactic intros H1 H2 ...Hn is a shortand for intro H1 ; intro H2 ; ... ; intro Hn.

Elimination tactic for the implication (modus ponens)

Let us consider a goal of the form $\Gamma \overset{?}{\vdash} A$. If $H : A_1 \to A_2 \to \ldots A_n \to A$ is an hypothesis of Γ or an already proven theorem, then the tactic apply H generates n new subgoals, $\Gamma \overset{?}{\vdash} A_1$, $\ldots$, $\Gamma \overset{?}{\vdash} A_n$.

This tactic corresponds to the following inference rules :

$$\frac{\dfrac{\ldots}{\Gamma \vdash B \to A} \quad \dfrac{\ldots}{\Gamma \vdash B}}{\Gamma \vdash A}\ mp$$

$$\frac{\dfrac{\ldots}{\Gamma \vdash A_1 \to A_2 \to \ldots \to A_n \to A} \quad \dfrac{\ldots}{\Gamma \vdash A_1} \quad \dfrac{\ldots}{\Gamma \vdash A_2} \quad \ldots \quad \dfrac{\ldots}{\Gamma \vdash A_n}}{\Gamma \vdash A}$$

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 → Q → R
  H0 : P → Q
  p : P
------------------------
   R
apply H.
```



```
 2 subgoals:
  P : Prop
  Q : Prop
  R : Prop
  H : P → Q → R
  H0 : P → Q
  p : P
------------------------
   P
subgoal 2 is:
 Q
assumption.
```



```
 1 subgoal:
  P : Prop
  Q : Prop
  R : Prop
  T : Prop
  H : P → Q → R
  H0 : P → 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->Q)(H0:Q->R)(p:P) : R
  := H0 (H p).

Check imp_trans.
```

imp_trans : (P->Q)->(Q->R)->P->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 truely the hypothesis assuming A*. This scheme generalizes to any number of hypotheses $A_1, \ldots, A_n$.

```
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 → P \/ Q
H0 : ~(R /\ 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 (/\), or (\/), iff (↔) and not (~).

Introduction rule for True

In any context Γ the proposition True is immediately provable (thanks to a predeclared constant I :True).
Practically, any goal $\Gamma \vdash^{?}$ `True` can be solved by the *tactic* trivial :

H : R → P \/ Q
H0 : ~(R /\ 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".

$$\frac{\Gamma \vdash \mathrm{False}}{\Gamma \vdash A}\ \mathit{False\_e}$$

There is an elimination tactic for False : Let us consider a goal of the form $\Gamma \vdash^{?} A$, and an hypothesis H :False. Then the tactic destruct H solves this goal immediately.

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 $\Gamma \vdash A/\backslash B$ is composed of a proof of $\Gamma \vdash A$ and a proof of $\Gamma \vdash B$.

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash A} \quad \dfrac{\cdots}{\Gamma \vdash B}}{\Gamma \vdash A/\backslash B}\ conj$$

Coq's tactic split, splits a goal $\Gamma \overset{?}{\vdash} A/\backslash B$ into two *subgoals* $\Gamma \overset{?}{\vdash} A$ and $\Gamma \overset{?}{\vdash} B$.

Conjunction elimination

Rule :

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash A/\backslash B} \quad \dfrac{\cdots}{\Gamma,\ A,\ B \vdash C}}{\Gamma \vdash C}\ and\_e$$

Associated tactic :
Let us consider a goal $\Gamma \overset{?}{\vdash} C$, and H $:A/\backslash B$. Then the tactic destruct H as [H1 H2] generates the new goal

$$\Gamma, H1 : A,\ H2 : B \overset{?}{\vdash} C$$

Example

```
Lemma and_comm :  P /\ Q → Q /\ 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 \/ :

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash A}}{\Gamma \vdash A\backslash/B}\ or\_intro\_l$$

$$\dfrac{\dfrac{\cdots}{\Gamma \vdash B}}{\Gamma \vdash A\backslash/B}\ or\_intro\_r$$

The tactic left is associated to *or_intro_l*, and the tactic right to *or_intro_r*.

Elimination rule and tactic for disjunction

$$\dfrac{\dfrac{\dots}{\Gamma \vdash A \backslash / B} \quad \dfrac{\dots}{\Gamma, A \vdash C} \quad \dfrac{\dots}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ or\_e$$

Let us consider a goal $\Gamma \overset{?}{\vdash} C$, and H $:A \backslash / B$. Then the tactic destruct H as [H1 | H2] generates two new subgoals :

$$\Gamma, H1 : A \overset{?}{\vdash} C$$
$$\Gamma, H2 : B \overset{?}{\vdash} C$$

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 \/ Q
------------------------
   Q \/ P
```

We have to choose between an introduction tactic on the conclusion Q \/ 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 $A \rightarrow$`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 $A \rightarrow$`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 $\Gamma, H : \sim B \overset{?}{\vdash} A$. Then the tactic destruct H generates a new subgoal $\Gamma \overset{?}{\vdash} B$.
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).

Justification of the previous tactic

$$\dfrac{\dfrac{\dfrac{\dfrac{\ldots}{\Gamma \vdash B}}{\Gamma, H : \sim B \vdash B} \qquad \dfrac{\overline{\Gamma, H : \sim B \vdash \sim B}}{\Gamma, H : \sim B \vdash B \to False}}{\Gamma, H : \sim B \vdash False}}{\Gamma, H : \sim B \vdash A}$$

Note : In situation like below :

```
H : C -> B -> ~A
------------------------
False
```

You can use simply apply H (because ~A is just A -> False)

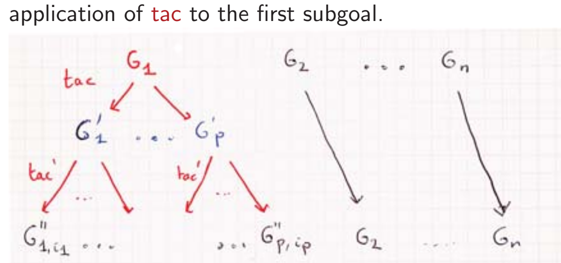
Logical equivalence

Let A and B be two propositions. Then the formula $A \leftrightarrow B$ (read "A iff B") is defined as the conjunction $(A \to B) /\backslash (A \to B)$.
The introduction tactic for $\leftrightarrow$ is split, which associates to any goal $\Gamma \vdash^? A \leftrightarrow B$ tho subgoals $\Gamma \vdash^? A \to B$ and $\Gamma \vdash^? B \to A$.

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 →B and H2 : B →A.

Simple tactic composition

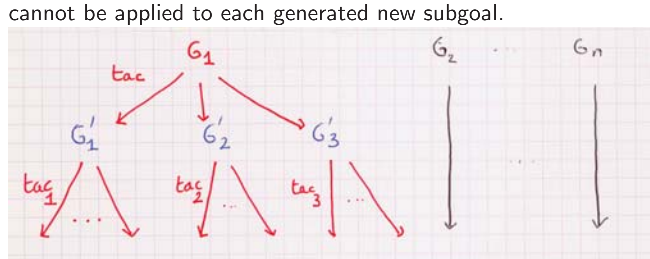
Let tac and tac' be two tactics.
The tactic *tac;tac'* applies tac' to each subgoal generated by the application of tac to the first subgoal.




```
Lemma and_comm' : P /\ Q → Q /\ 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 $\Gamma \overset{?}{\vdash} A$, and B be some proposition.
The tactic assert (H :B), generates two subgoals :

1. $\Gamma \overset{?}{\vdash} B$
2. $\Gamma, H : B \overset{?}{\vdash} A$

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 → Q)
            (H0 : Q → R)
            (H1 : (P → R) → T → Q)
            (H2 : (P → R) → T).
 Lemma L8 : Q.
 (* A direct backward proof would need to prove twice
   the proposition (P → R) *)
```

The tactic assert (PR : P → R) generates two subgoals :

2 subgoals

H : P → Q
H0 : Q → R
H1 : (P → R) → T → Q
H2 : (P → R) → T
```
------------------------
```
P → R

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


H : P → Q
H0 : Q → R
H1 : (P → R) → T → Q
H2 : (P → R) → T
PR : P → 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 $A_1 \rightarrow A_2 \ldots \rightarrow A_n \rightarrow A$ where the main connective of A is \/, /\, ~, ↔ or False.

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


```
Section Ex5.
Hypothesis H : T → R → P \/ Q.
Hypothesis H0 : ~ (R /\ Q).
Hypothesis H1 : T.

Lemma L5 : R → 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 → R → P \/ Q
  H0 : ~ (R /\ 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' : ~ P
------------------------
   Q
 assumption.
Qed.
```


An automatic tactic for intuitionistic propositional logic

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

```
Lemma L5' : (R → P \/ Q) → ~(R /\ Q) →  R →  P.
Proof.
 tauto.
Qed.
```

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

```
 P \/ ~ P
 (P → Q) ↔ (~ P \/ Q)
~(P /\ Q) ↔  ~ P \/ ~ Q
((P→ Q) → P) → P  (Peirce's formula)
```