Proofs in Proposition Logic and Predicate Logic [1]

Pierre Castéran

Beijing, August 2009

[1] This lecture corresponds mainly to the chapters 3 : "Propositions and Proofs" and 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*, then *first-order 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*.

Since Prop is a type, it is easy to declare propositional variables, using *Coq's* declaration mechanism :

```
Section Propositional_Logic.
Variables P Q R T : Prop.
```

P is assumed [2]

Q is assumed . . .

```
Check P.
```

P : Prop

[2] read "*P* is assumed to be a proposition"

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)
  - $A \vee B$ (disjunction) (in ASCII : A \/ B)
  - $A \wedge B$ (conjunction) (in ASCII : A /\ B)
  - $\sim A$ (negation)

```
Check ((P → (Q ∧ P)) → (Q → P)).
```

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

The Sequent Notation

In *Coq*, a frequent activity consists in proving a proposition A under some set Γ of hypotheses (also called a *context*.)
For instance, one would like to prove the formula $\mathsf{R} \rightarrow \mathsf{P}$ under the hypotheses $\mathsf{R} \rightarrow \mathsf{P} \vee \mathsf{Q}$ and $\sim(\mathsf{R} \wedge \mathsf{Q})$.
A structure consisting of a finite set Γ of hypotheses and a conclusion A is called an (intuitonistic) *sequent*. Its usual notation is $\Gamma \vdash A$.
In our example, the sequent we consider is written :

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

Hypotheses and Goals

The *Coq* system helps the user to build *interactively* a proof of some sequent $\Gamma \vdash A$. We also say that one wants to *solve the goal* $\Gamma \overset{?}{\vdash} A$.
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 :

$H : R \rightarrow P \lor Q$
$H0 : \sim(R \land Q)$
============================
$R \rightarrow P$

In the proof of some theorem, it is usual to have to prove several subgoals. In this case, *Coq* displays in full the first subgoal to solve, and an abbreviated view of the remaining subgoals.

2 subgoals

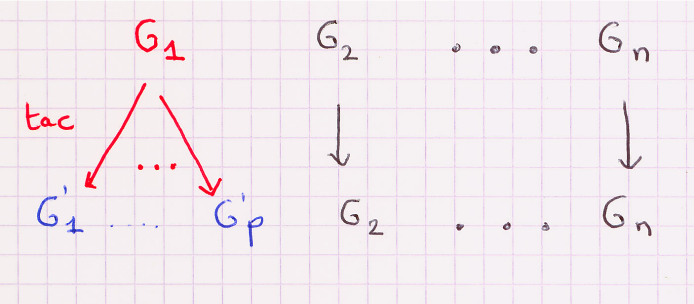
$P : Prop$
$Q : Prop$
$H : P \vee Q$
$H0 : P$
$============================$
$Q \vee P$

subgoal 2 is:
$Q \vee P$

Rules and Tactics

Let us consider some goal $\Gamma \stackrel{?}{\vdash} A$. Solving this goal consists in building a *proof* of A under the hypotheses of Γ. Formally, such a proof is a term of type A, but we won't explore deeply this aspect (see the book for more details).
The structure of a proof both depends on the form of A and the contents of Γ. We will present now some basic rules for building proofs.

Goal Directed Proofs



Tactic application can be slightly more complex in some situations (*e.g.* shared existential variables, see *Coq*'s documentation).

More details

The basic tool for interactively solving a goal $G = \Gamma \stackrel{?}{\vdash} A$ is called a *tactic*, which is a command typed by the user.

In general, at each step of an interactive proof, a finite sequence of *subgoals* $G_1, G_2, \ldots, G_n$ must be solved. 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 replace the previous one.

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.

Introduction and Elimination Tactics

Let us consider again the goal below :

```
H : R → P ∨ Q
H0 : ~(R ∧ Q)
============================
```

$R \to P$

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.

Minimal Propositional Logic

Minimal propositional logic is a very simple fragment of mathematical logic :

- Formulas are built only with propositional variables and the implication connective $\rightarrow$.
- There are only three simple inference rules.

It is a good framework for learning basic concepts on tactics in *Coq*.

The rule of assumption

The following rule builds a proof of any sequent $\Gamma \vdash A$, whenever the conclusion A is already assumed in Γ.

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

In an interactive proof with *Coq*, the tactic assumption solves any goal $\Gamma \stackrel{?}{\vdash} A$, where the context Γ contains an hypothesis assuming A.

```
H : R → P → Q
H0 :(R → Q)→ T
============================
```

$R \rightarrow P \rightarrow Q$

assumption.

Elimination rule for the implication (modus ponens)

$$\dfrac{\dfrac{\dots}{\Gamma \vdash B \rightarrow A} \quad \dfrac{\dots}{\Gamma \vdash B}}{\Gamma \vdash A}\ mp$$

Applying several times the *mp* rule, we get the following derived rule :

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

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

Introduction rule for the implication

$$\dfrac{\dfrac{\dots}{\Gamma, A \vdash B}}{\Gamma \vdash A \rightarrow B}\ imp\_i$$

Let us consider a goal $\Gamma \overset{?}{\vdash} A \rightarrow B$. The tactic intro H (where H is not the name of an hypothesis in Γ) transforms this goal into $\Gamma, H : A \overset{?}{\vdash} B$.
The multiple introduction tactic intros H1 H2 ...Hn is a shortand for intro H1 ; intro H2 ; ... ; intro Hn.

A simple example

The following proof tree represents a proof of the sequent $\vdash (P \to Q \to R) \to (P \to Q) \to (P \to R)$. The leaves of this tree are instances of rule assumption. For simplicity's sake, we use Γ as an abbreviation of the context $P \to Q \to R,\ P \to Q$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma, P \vdash P \to Q \to R} \quad \overline{\Gamma, P \vdash P}}{\Gamma, P \vdash Q \to R}\ mp \quad \dfrac{\overline{\Gamma, P \vdash P \to Q} \quad \overline{\Gamma, P \vdash P}}{\Gamma, P \vdash Q}\ mp}{\Gamma, P \vdash R}\ mp}{\Gamma \vdash P \to R}\ imp\_i}{P \to Q \to R \vdash (P \to Q) \to (P \to R)}\ imp\_i}{\vdash (P \to Q \to R) \to (P \to Q) \to (P \to R)}\ imp\_i
$$

The same proof using tactics

```
Section Propositional_Logic.
Variables P Q R : Prop.

Lemma imp_dist : (P → (Q → R)) → (P → Q) → P → R.
Proof.
```

1 subgoal

$P : Prop$
$Q : Prop$
$R : Prop$
$==============================$
$(P \to Q \to R) \to (P \to Q) \to P \to R$

```
intros H H0 p.
```

1 subgoal:

$P : Prop$

$Q : Prop$

$R : Prop$

$H : P \to Q \to R$

$H0 : P \to Q$

$p : P$

$============================$

R

```
apply H.
```

2 subgoals:

$P : Prop$

$Q : Prop$

$R : Prop$

$H : P \to Q \to R$

$H0 : P \to Q$

$p : P$

============================

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

$: (P \to Q \to R) \to (P \to Q) \to P \to R$

```
Print imp_dist.
```

imp_dist $=$

fun $(H : P \to Q \to R)\ (H0 : P \to Q)\ (H1 : P) \Rightarrow H\ H1\ (H0\ H1)$

$: (P \to Q \to R) \to (P \to Q) \to P \to R$

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.

```
Check fun p:P ⇒ p.
```

fun p:P ⇒ p
: P → P

```
Check  fun (H : P → Q → R)(q: Q)(p:P) ⇒ H p q.
```

fun (H : P → Q → R) (q : Q) (p : P) ⇒ H p q
: (P → Q → R) → Q → P → R

```
Check fun (p:P)(H: P → False) ⇒ H p.
```

fun (p : P) (H : P → False) ⇒ H p
: P → (P → False) → False

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

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 Γ the proposition True is immediately provable (thanks to a predeclared constant I :True).
Practically, any goal $\Gamma \overset{?}{\vdash}$ `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".

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

There is an elimination tactic for False : Let us consider a goal of the form $\Gamma \overset{?}{\vdash} A$, 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 $\Gamma \vdash A \wedge B$ is composed of a proof of $\Gamma \vdash A$ and a proof of $\Gamma \vdash B$.

$$\frac{\dfrac{\ldots}{\Gamma \vdash A} \quad \dfrac{\ldots}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ conj$$

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

Conjunction elimination

Rule :

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

Associated tactic :

Let us consider a goal $\Gamma \overset{?}{\vdash} C$, and H $:A \wedge B$. 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

$P : Prop$
$Q : Prop$
$H1 : P$
$H2 : Q$
$============================$
$Q \wedge P$

```
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 $\vee$:

$$\frac{\frac{\cdots}{\Gamma \vdash A}}{\Gamma \vdash A \vee B}\ \mathit{or\_intro\_l}$$

$$\frac{\frac{\cdots}{\Gamma \vdash B}}{\Gamma \vdash A \vee B}\ \mathit{or\_intro\_r}$$

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

Elimination rule and tactic for disjunction

$$\frac{\dfrac{\dots}{\Gamma \vdash A \vee B} \quad \dfrac{\dots}{\Gamma, A \vdash C} \quad \dfrac{\dots}{\Gamma, B \vdash C}}{\Gamma \vdash C} \; or\_e$$

Let us consider a goal $\Gamma \overset{?}{\vdash} C$, and H $:A \vee B$. 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 :

$P : Prop$
$Q : Prop$
$H : P \vee Q$
$============================$
$Q \vee P$

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.
```

$P : Prop$

$Q : Prop$

$H : P \vee Q$

$============================$

P

This is clearly a dead end. Let us come back to the previous step (with command `Undo`).

```
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 $\Gamma \overset{?}{\vdash} B$.
Justification (by a derived rule) :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\Gamma \vdash B}}{\Gamma, H : \sim B \vdash B} \quad \dfrac{\dfrac{}{\Gamma, H : \sim B \vdash \sim B}}{\Gamma, H : \sim B \vdash B \rightarrow False}}{\Gamma, H : \sim B \vdash False}}{\Gamma, H : \sim B \vdash A}$$

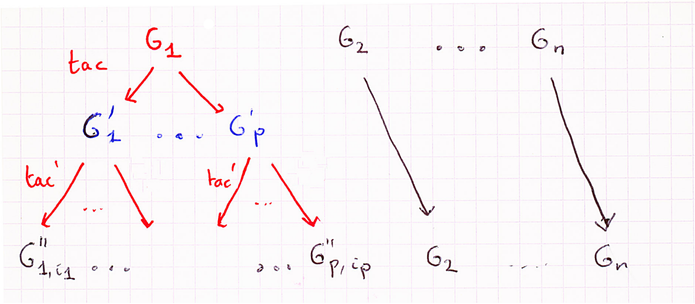
Logical equivalence

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

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 →B and H2 : B →A.

Simple tactic composition

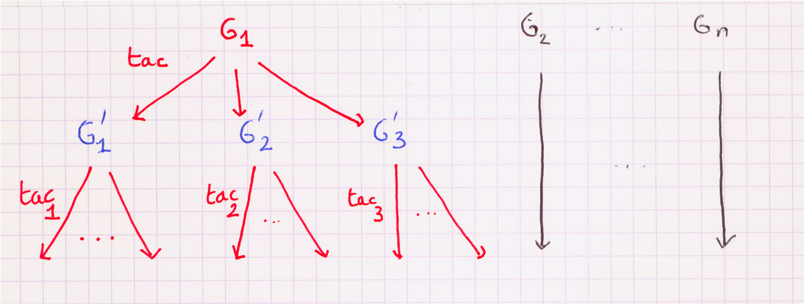
Let tac and tac' be two tactics.
The tactic *tac;tac'* applies tac' to each subgoal generated by the application of tac to the first subgoal.



```
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 $\Gamma \stackrel{?}{\vdash} A$, and B be some proposition.
The tactic assert (H :B), generates two subgoals :

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

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 $(PR : P \rightarrow R)$ generates two subgoals :

2 subgoals

$H : P \to Q$
$H0 : Q \to R$
$H1 : (P \to R) \to T \to Q$
$H2 : (P \to R) \to T$
$==============================$
$P \to R$

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 $A_1 \rightarrow A_2 \ldots \rightarrow A_n \rightarrow A$ where the main connective of A is $\vee$, $\wedge$, $\sim$, $\leftrightarrow$ or False.
In this case, new subgoals of the form $\Gamma \overset{?}{\vdash} A_i$ are also generated (in addition to the behaviour we have already seen).

In fact, this use of destruct H replaces a composition of calls to assert, applications, and destruct. Notice the use of H as a function that receives as arguments proofs of A_1, A_2, $\ldots$, A_n.

```
assert (H1:A1);[auto |
     assert(H2:A2);[auto|
       ...
       assert (Hn:An);[auto |
                          destruct (H H1 H2 ... Hn)]]]
```

```
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

$H : T \to R \to P \vee Q$
$H0 : \sim (R \wedge Q)$
$H1 : T$
$r : R$
$H2 : Q$
============================
P

```
destruct H0; split;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 $\vdash A \vee \sim A$). Here are some examples :

```
 P ∨ ∼ P
 (P → Q) ↔ (∼ P ∨ Q)
∼(P ∧ Q) ↔ ∼ P ∨ ∼ Q
((P→ Q) → P) → P
```
(Peirce's formula)

Formulas of First-Order Logic : 1

- Terms : we can build terms according to the declarations of constants and variables, using *Coq*'s typing rules.
- Predicates : a *Predicate* is just any function of type $A_1 \rightarrow A_2 \ldots A_n \rightarrow \mathtt{Prop}$ where $A_i : \mathtt{Set}$ for each i. Predicates are declared as any other function symbol.
- Atomic propositions : let $P : A_1 \rightarrow A_2 \ldots A_n \rightarrow \mathtt{Prop}$ and $t_i : A_i \ (i = 1 \ldots n)$. Then the term $f\ t_1\ t_2\ \ldots\ t_n$ of sort $\mathtt{Prop}$ is an atomic proposition.
  If t_1 and t_2 are terms *of the same type*, then $t_1 = t_2$ is an atomic proposition.

First Order formulas : 2

According to the declarations of the current context :

- Any atomic formula is a formula,
- True and False are formulas,
- If F and G are formulas, then $F \leftrightarrow G$, $F \rightarrow G$, $F \wedge G$, $F \vee G$ and $\sim F$ are formulas,
- let x be a variable, then $\forall$ `x:A, F` and $\exists x : A, F$ are formulas. x is said to be **bound** in F.

ASCII notation : The symbol $\forall$ is typed forall and $\exists$ is typed exists.

Examples

```
Section First_Order.
Variable A : Type.
Variable R : A → A → Prop.
Variable f : A → A.
Variable a : A.

Check f (f a).
```
(f (f a)) : A
```
Check R a (f (f a)).
```
R a (f (f a)): Prop
```
Check forall x :A, R a x → R a (f (f (f x))).
```
forall x :A, R a x → R a (f (f (f x))) : Prop.

Introduction rule for the universal quantifier

$$\frac{\Gamma, x : A \vdash \mathrm{F}}{\Gamma \vdash \forall\, \mathtt{x} : A,\ F}\ \mathrm{x\ not\ bound}\ \mathit{in}\ \Gamma$$

The tactic associated with this rule is the same as for the introduction of implication : intro x.

It is very usual to use intros on nested universal quantifications and implications :

. . .
====================
forall x :A, P x → forall y: A, R x y → R x (f (f (f y))).

```
intros x Hx y Hy.
```

. . .
x: A
Hx: P x
y: A
H: R x y
=======================
R x (f (f (f y)))

Elimination rule for the universal quantifier

$$\dfrac{t : A \quad \dfrac{\dots}{\Gamma \vdash \forall\, \mathtt{x{:}A,\, F}}}{\Gamma \vdash F\{x/t\}}\ \forall_e$$

The associated tactic is apply H, where H has type $\forall\, \mathtt{x{:}A,\, F}$. This tactic is generalized to the case of nested implications and universal quantifications, like, for instance :

H : ∀ x:A, P x → ∀ y:A, R x y → R x (f y)

On a goal like R a (f (f a)), the tactic apply H will generate two subgoals : P a and R a (f a).

A Small Example

```
Hypothesis Hf : forall x y:A, R x y → R x (f y).
Hypothesis R_refl : forall x:A, R x x.

Lemma Lf : forall x :A, R x (f (f (f x))).
Proof.
 intro  x;apply Hf.
```

1 subgoal

Hf : forall x y : A, R x y → R x (f y)
R_refl : forall x : A, R x x
x : A
============================
R x (f (f x))

Helping apply

Let us use the following theorems from the library `Arith` :

lt_n_Sn : forall n : nat, n < S n

```
lt_trans  : forall n m p : nat, n < m → m < p → n < p
```

```
Lemma lt_n_SSn : forall i:nat, i < S (S i).
Proof.
 intro i;apply lt_trans.
```

Error: Unable to find an instance for the variable m.

```
 intro i;apply lt_trans with (S i);apply lt_n_Sn.
```

Another possibility : use eapply (see the documentation).

Introduction rule for the existential quantifier

$$\frac{\Gamma \vdash F\{x/t\} \quad t : A}{\Gamma \vdash \exists x : A, F}\ \exists_i$$

The associated tactic is exists t.

$==========$

exists p, 3 < p.

```
exists 4.
```

$=================$

$3 < 4$

Elimination rule for the existential quantifier

$$\frac{\dfrac{\dots}{\Gamma, x : A, Hx : F \vdash G} \quad \Gamma \vdash \exists x : A,\ F}{\Gamma \vdash G}\ x \text{ not bound in } \Gamma$$

The associated tactic is destruct H as [x Hx], where H $:\exists\, x : A,\ F$ w.r.t. Γ.

H : exists n:nat, forall p: nat, p < n
====================
False

```
destruct H as [n Hn].
```

n : nat
Hn : forall p : nat, p < n
==============================
False

Rules and tactics for the equality

Introduction rule.

$$\frac{a : A}{a = a}\ \textit{refl\_equal}$$

Associated tactics : reflexivity, trivial, auto.

```
Lemma L36 : 9 * 4 = 3 * 12.
Proof.
 reflexivity.
Qed.
```

Elimination tactics for the equality

$$\frac{\Gamma, e : a = b \vdash A[a]}{\Gamma, e : a = b \vdash A[b]}$$

The associated tactic is rewrite $\rightarrow$ e.

$$\frac{\Gamma, e : a = b \vdash A[b]}{\Gamma, e : a = b \vdash A[a]}$$

The associated tactic is rewrite $\leftarrow$ e.

See also : tactics symmetry, transitivity, replace, etc.

Example

```
 Lemma eq_trans_on_A :
  forall x y z:A,  x = y → y = z → x = z.
Proof.
 intros x y z e.
```

$\ldots$

$e : x = y$

$==============================$

$y = z \rightarrow x = z$

```
 rewrite → e.
```

$\ldots$

$e : x = y$

$==============================$

$y = z \rightarrow y = z$

Autres tactiques pour l'égalité

- symmetry transforme un but $t_1 = t_2$ en $t_2 = t_1$
- transitivity t_3 transforme un but $t_1 = t_3$ en les deux sous-buts $t_2 = t_3$ et $t_3 = t_2$

Voir aussi replace, subst, etc.

Utilisation de l'application

```
Require Import Omega.
Lemma L : forall n:nat, n < 2 -> n = 0 \/ n = 1.
Proof.
 intros;omega.
Qed.

Lemma L2 : forall i:nat, i < 2 -> i*i = i.
Proof.
 intros i H; destruct (L _ H); subst i; trivial.
Qed.
```