In this class, we introduce the reasoning techniques used in Coq, starting with a very reduced fragment of logic, *propositional intuitionistic 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*: \( \lor, \land, \to \), etc.

- `it_is_raining \lor \sim it_is_raining`
- \( P \land Q \to Q \land P \)
- \( \sim(P \lor Q) \to \sim(P \land Q) \)

`it_is_raining`, `P` and `Q` are *propositional variables*.

### 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 \to B` (implication) (in ASCII: `A -> B`)
  - `A \lor B` (disjunction) (in ASCII: `A \lor B`)
  - `A \land B` (conjunction) (in ASCII: `A \land B`)
  - `\sim A` (negation)
Like in many programming languages, connectors have **precedence** and **associativity** conventions:

The connectors \( \to, \lor, \land \) are right-associative: for instance \( P \to Q \to R \) is an abbreviation for \( P \to (Q \to R) \).

The connectors are displayed below in order of increasing precedence:

\[ \leftrightarrow, \to, \lor, \land, \sim \]

Check \((P \to (Q \land P)) \to (Q \to P))\).

\((P \to Q \land P) \to Q \to P : \text{Prop}\)

### Logical Statements

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

"If the following propositions:

\[
P \lor Q
\sim Q
\]

hold, then the following proposition:

\[
R \to R \lor P
\]

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 \( \Gamma \vdash A \).

For instance, our previous statement may look like that:

\[
P \lor Q, \sim Q \vdash R \to R \lor P
\]

Another useful presentation is the following one:

\[
P \lor Q
\sim Q
------------------------
R \to R \lor P
\]

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 \( \Gamma \) 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:

\[
H : P \lor Q
H_0 : \sim Q
------------------------
R \to R \lor P
\]

### 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 \lor Q.

**Hypothesis** H0 : \sim Q.

**Check** H.

**H : P \lor Q**
Then inside the section, we tell Coq we want to prove some proposition.

For proving $R \land P$, 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 first subgoal is immediately solved with `assumption`.

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

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 current context contains two mutually contradictory propositions: $Q$ and $\neg Q$. The tactic call `absurd Q` helps to start a proof by reduction to the absurd.
Proofs in Propositional Logic

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 Axi: 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 propositional intuitionistic logic

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

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

- The multiple introduction tactic intros $H_1 \ H_2 \ \ldots \ H_n$ is a shorthand for intro $H_1$; intro $H_2$; \ldots; intro $H_n$. 

The tactic assumption

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

```

```

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

$$\begin{array}{c} A \in \Gamma \\
\Gamma \vdash A \end{array} \quad \text{assumption}$$
Elimination tactic for the implication (modus ponens)

Let us consider a goal of the form \( \Gamma \vdash A \). If \( H : A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_n \rightarrow A \)

is an hypothesis of \( \Gamma \) or an already proven theorem, then the tactic apply \( H \) generates \( n \) new subgoals, \( \Gamma \vdash A_1 \), \( \ldots \), \( \Gamma \vdash A_n \).

This tactic corresponds to the following inference rules :

\[
\begin{array}{c}
\Gamma \vdash B \rightarrow A \\
\Gamma \vdash B \\
\hline \\
\Gamma \vdash A \\
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_n \rightarrow A \\
\Gamma \vdash A_1 \\
\Gamma \vdash A_2 \\
\ldots \\
\Gamma \vdash A_n \\
\hline \\
\Gamma \vdash A \\
\end{array}
\]

A simple example

Section Propositional_Logic.

Variables \( P \), \( Q \), \( R : \text{Prop} \).

Lemma imp_dist : \( (P \rightarrow (Q \rightarrow R)) \rightarrow (P \rightarrow Q) \rightarrow P \rightarrow R \).

Proof.

1 subgoal

\( P : \text{Prop} \)
\( Q : \text{Prop} \)
\( R : \text{Prop} \)
\( H : P \rightarrow Q \rightarrow R \)
\( H0 : P \rightarrow Q \)
\( p : P \)

\[
\begin{array}{c}
\hline \\
R \\
\end{array}
\]

apply \( H \).

2 subgoals:

\( P : \text{Prop} \)
\( Q : \text{Prop} \)
\( R : \text{Prop} \)
\( H : P \rightarrow Q \rightarrow R \)
\( H0 : P \rightarrow Q \)
\( p : P \)

\[
\begin{array}{c}
\hline \\
R \\
subgoal 2 is: \\
Q \\
assumption. \\
\end{array}
\]

Proof completed

Qed.

imp_dist is defined

Check imp_dist.

imp_dist :

\( (P \rightarrow Q \rightarrow R) \rightarrow (P \rightarrow Q) \rightarrow P \rightarrow R \)

Print imp_dist.

imp_dist =

fun (H : P \rightarrow Q \rightarrow R) (H0 : P \rightarrow Q) (H1 : P) ⇒ H H1 (H0 H1) : (P \rightarrow Q \rightarrow R) \rightarrow (P \rightarrow Q) \rightarrow P \rightarrow 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.

\[
\text{imp_trans} : (P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow P \rightarrow R.
\]

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 \( A_1, \ldots, A_n \).

Section Imp_trans.

Hypothesis H : P \rightarrow Q.
Hypothesis H0 : Q \rightarrow R.

Lemma imp_trans' : P \rightarrow R.
(* Proof skipped, uses H and H0 *)

End Imp_trans.

Check imp_trans'.

\[
\text{imp_trans'} : (P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow P \rightarrow R.
\]

Introduction and Elimination Tactics

Let us consider again the goal below:

\[
H : R \rightarrow P \lor Q\nonumber \\
H0 : \sim(R \land Q)\nonumber \\
\hline
R \rightarrow P
\]

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.

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 \( \Gamma \vdash \text{True} \) can be solved by the tactic trivial:

\[
H : R \rightarrow P \lor Q\nonumber \\
H0 : \sim(R \land Q)\nonumber \\
\hline
\text{True}
\]

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: Let us consider a goal of the form \( \Gamma \vdash A \), and an hypothesis \( H : \text{False} \). 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 $\Gamma \vdash A \land B$ is composed of a proof of $\Gamma \vdash A$ and a proof of $\Gamma \vdash B$.

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

Coq’s tactic split, splits a goal $\Gamma \vdash A \land B$ into two subgoals $\Gamma \vdash A$ and $\Gamma \vdash B$.

Conjunction elimination

**Rule :**

$$\frac{\Gamma \vdash A \quad \Gamma, A, B \vdash C}{\Gamma \vdash C} \text{ and_e}$$

**Associated tactic :**

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

Example

**Lemma and_comm** : $P \land Q \rightarrow Q \land P$.
Proof.
intro $H$.
1 subgoal

- $P : \text{Prop}$
- $Q : \text{Prop}$
- $H : P \land Q$

$\quad \quad \quad \quad Q \land P$

split.
2 subgoals

- $P : \text{Prop}$
- $Q : \text{Prop}$
- $H1 : P$
- $H2 : Q$

$\quad \quad \quad \quad Q \land P$

Introduction rules and tactics for disjunction

There are two introduction rules for $\lor$:

$$\frac{\Gamma \vdash \rightarrow \quad \Gamma \vdash B}{\Gamma \vdash A \lor B} \text{ or_intro_l}$$

$$\frac{\Gamma \vdash \rightarrow \quad \Gamma \vdash A}{\Gamma \vdash A \lor B} \text{ or_intro_r}$$

The tactic left is associated to $\text{or_intro_l}$, and the tactic right to $\text{or_intro_r}$. 
Elimination rule and tactic for disjunction

\[
\frac{\Gamma \vdash A \lor B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C \lor e}
\]

Let us consider a goal \( \Gamma \vdash C \), and \( H : A \lor B \). Then the tactic \texttt{destruct} \( H \) as \([H1 \mid H2]\) generates two new subgoals:

\[
\begin{align*}
\Gamma, H1 : A &\vdash C \\
\Gamma, H2 : B &\vdash C
\end{align*}
\]

This tactic implements the proof by cases paradigm.

A combination of left, right and destruct

Consider the following goal:

\[
P : \text{Prop} \\
Q : \text{Prop} \\
H : P \lor Q
\]

\[
\begin{array}{c}
\text{---------} \\
Q \lor P
\end{array}
\]

We have to choose between an introduction tactic on the conclusion \( Q \lor P \), 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. 
\[
P : \text{Prop} \\
Q : \text{Prop} \\
H : P \lor Q
\]

\[
\begin{array}{c}
P \quad \text{---------} \\
Q \lor P
\end{array}
\]

This is clearly a dead end. Let us come back to the previous step (with command \texttt{Undo} (coqtop or using Coqide’s navigation menu).

destruct \( H \) as \([H0 \mid H0] \). 
\textit{two subgoals}
\[
P : \text{Prop} \\
Q : \text{Prop} \\
H : P \lor Q \\
H0 : P
\]

\[
\begin{array}{c}
Q \lor P \quad \text{---------} \\
Q \lor P
\end{array}
\]

\textit{subgoal 2 is:}
\[
Q \lor P \\
\text{right};\text{assumption.} \\
\text{left};\text{assumption.} \\
\text{Qed.}
\]

Negation

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

The tactic \texttt{unfold} allows to expand the constant \texttt{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. \texttt{intro} \( H \) where \( H \) is a fresh name. This tactic pushes the hypothesis \( H : A \) into the context and leaves \texttt{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 \texttt{destruct} \( H \) generates a new subgoal \( \Gamma \vdash B \).

\textbf{Note}: Using \texttt{case} \( H \) instead of \texttt{destruct} \( H \) allows to keep the hypothesis \( H \) in the context (we may need to use it later in the proof).
Proofs in Propositional Logic
Propositional Intuitionistic Logic

Justification of the previous tactic

\[ \frac{\Gamma \vdash \neg B}{\Gamma, H : \neg B \vdash \neg B} \]
\[ \frac{\Gamma, H : \neg B \vdash \neg B}{\Gamma, H : B 
\rightarrow \text{False}} \]
\[ \frac{\Gamma, H : \neg B \vdash \text{False}}{\Gamma, H : \neg B \vdash A} \]

Note: In situation like below:

\[ H : C \rightarrow B \rightarrow \neg A \]

\[ \frac{\text{False}}{\text{False}} \]

You can use simply apply H (because \(\neg A\) 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 \(\Gamma \vdash \neg A\) 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.

\(\text{intro } H;\text{destruct } H\) as \([H1 H2]\).

\[ H1 : P \]
\[ H2 : Q \]

\[ Q \wedge P \]

\[ \frac{\text{split; assumption.}}{\text{(* assumption has been applied to each one of the two subgoals generated by split *)}} \]

Qed.

Another composition operator

The tactic composition \(tac ; [tac1 ; tac2 ; \ldots]\) 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 $B$!)

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 $\lor$, $\land$, $\sim$, $\leftrightarrow$ 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).
Proofs in Propositional Logic
More on tactics

(*) 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 \lor Q
H0 : \sim (R \land Q)
H1 : T
r : R
H2 : Q

------------------------
P
destruct H0; split; assumption.
Qed.

A variant of intros

Lemma L2 : (P \lor Q) \land \sim P \rightarrow Q.
Proof.
intros [p | q] p'.
2 subgoals

p : P
p' : \sim P

------------------------
Q

subgoal 2 is:
Q
destruct p'; trivial.

An automatic tactic for intuitionistic propositional logic

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

Lemma L5' : (R \rightarrow P \lor Q) \rightarrow \sim (R \land 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 ⊢ A\lor\sim A). Here are some examples:

P \lor \sim P
(P \rightarrow Q) \leftrightarrow (\sim P \lor Q)
\sim (P \lor Q) \leftrightarrow \sim P \lor \sim Q
((P \rightarrow Q) \rightarrow P) \rightarrow P (Peirce's formula)