Initiation à la recherche

Introduction à Rocq

Jolan PHILIPPE

4 Decembre 2025

Université d'Orléans

Rocq en 2 phrases

- Rocq (anciennement *Coq*) est un assistant de preuve + un langage à types dépendants.
- Objectif : écrire des **spécifications** et des **preuves** vérifiées par une machine.

Idée clé

Une preuve est un programme que Rocq vérifie. (Correspondance Curry-Howard)

Curry–Howard (version “pratique”)

- Une proposition (`Prop`) est un **type**.
- Une preuve est un **terme** (un programme) de ce type.

Exemples

- $A \to B$ correspond à une fonction : prendre une preuve de A, produire une preuve de B.
- $A \wedge B$ correspond à une paire : `(preuveA, preuveB)`.
- $A \vee B$ correspond à une somme : `inl preuveA` ou `inr preuveB`.

Gallina : le langage fonctionnel

- Fonctions, produits, sommes, définitions inductives
- Calcul, exécution, `Compute`

```
Definition add1 (n : nat) : nat := S n.

Compute (add1 41).   (* = 42 : nat *)
```

`Prop` et `Type` (intuition)

- `Type` : types de données (programmation)
- `Prop` : propositions (logique)

Exemples

```
Check nat : Set.   (* ou Type *)
Check True : Prop.
Check False : Prop.
```

Quantification : `forall`

- $\forall x : T, P(x)$: une fonction qui prend un $x : T$ et produit une preuve de $P(x)$

```
Lemma id_nat : forall n : nat, n = n.
Proof.
  intro n.  (* on introduit n dans le contexte *)
  reflexivity.
Qed.
```

Note

`rewrite` travaille sur l'égalité = (qui est une proposition dans Rocq).

Écrire un lemme / théorème

```
Theorem ex1 : forall A B : Prop, A -> (A \/ B).
Proof.
  intros A B a.
  apply or_introl.
  exact a.
Qed.
```

À retenir

En mode preuve, on transforme un **but** (`Goal`) en sous-buts jusqu'à obtenir `Qed`.

Mode preuve : `Proof. ... Qed.`

- `Theorem T : P. Proof. ... Qed.`
- En réalité : on construit un terme t tel que $t : P$.

Deux styles

- **Script de tactiques** (`intro`, `apply`, `destruct`, ...)
- **Terme direct** (lambda-terme / expression)

LTac : les tactiques (boîte à outils minimale)

- `intro` / `intros` : introduire des hypothèses / variables
- `assumption` : fermer un but s'il est dans le contexte
- `apply` : utiliser une implication ou un lemme
- `split` : prouver une conjonction
- `destruct` : raisonnement par cas / élimination sur une donnée
- `rewrite` : réécriture via une égalité

Exemple central : élimination de $\vee$ (règle c / C)

Proposition

$(A \to C) \to (B \to C) \to (A \vee B) \to C$

```
Theorem or_elim_like :
  forall A B C : Prop, (A -> C) -> (B -> C) -> (A \/ B) -> C.
Proof.
  intros A B C f g e.
  destruct e as [a | b].
  - apply f. assumption.
  - apply g. assumption.
Qed.
```

Le même résultat : version “programme” (Curry–Howard)

Terme (style lambda / match)

```
fun (A B C : Prop) (f : A -> C) (g : B -> C) (e : A \/ B) =>
  match e with
  | or_introl a => f a
  | or_intror b => g b
  end
```

- `destruct` construit exactement ce `match`.

Règles Fig. 1 $\leftrightarrow$ tactiques (vue d'ensemble)

- **(v)** variable : `assumption` (ou `exact H`)
- **(i)** intro de $\rightarrow$: `intro` / `intros`
- **(a)** élimination de $\rightarrow$: `apply` (puis sous-buts)
- **(p)** intro de $\wedge$: `split`
- **(π_1)/(π_2)** proj. de $\wedge$: `destruct` ou `apply (proj1 H)` / `apply (proj2 H)`
- **(l)/(r)** intro de $\vee$: `apply or_introl` / `apply or_intror`
- **(c)** élimination de $\vee$: `destruct` (raisonnement par cas)

Correspondance détaillée : (v), (i), (a)

Déduction naturelle	**LTac**

▪ (v) : si $F \in \Gamma$ alors $\Gamma \vdash F$
▪ (i) : pour prouver $F_1 \to F_2$, supposer F_1 et prouver F_2
▪ (a) : de $F_1 \to F_2$ et F_1 conclure F_2
	`intros H.          (* (i) *)`
	`assumption.        (* (v) *)`
	
	`apply Himp.        (* (a) *)`
	`(* -> sous-but: prouver l'argument *)`

Correspondance détaillée : (p), (π_1), (π_2)

Déduction naturelle	LTac

• (p) : prouver F_1 et prouver F_2 pour conclure $F_1 \wedge F_2$
• (π_1), (π_2) : extraire un côté d'une conjonction
	`split.                  (* (p) *)`
	`- ...                    (* prouver F1 *)`
	`- ...                    (* prouver F2 *)`
	
	`destruct H as [H1 H2].   (* pi1/pi2 *)`

Correspondance détaillée : (l), (r), (c)

Déduction naturelle	LTac

(l) : de F_1 conclure $F_1 \vee F_2$ (r) : de F_2 conclure $F_1 \vee F_2$ (c) : prouver F en faisant deux cas (F_1 et F_2)	<pre>apply or_introl. (* (l) *)
	apply or_intror. (* (r) *)
	
	destruct H as [H1
		H2]. (* (c) *)
		- ...
		- ...</pre>

Exemple “rewrite” (égalité)

Objectif

Montrer qu’on peut remplacer une expression par une égale.

```
Theorem rewrite_example :
  forall (X : Type) (x y : X) (P : X -> Prop),
    x = y -> P x -> P y.
Proof.
  intros X x y P Hxy HPx.
  rewrite <- Hxy.
  assumption.
Qed.
```