

Leçon 3 : Logique du Premier Ordre

Quantificateurs et Égalité dans Rocq

Logique et Fondements de l'Informatique

Année 2026

Les Prédicats comme Fonctions

D'une proposition à une propriété :

- Avant : $A : \text{Prop}$

Les Prédicats comme Fonctions

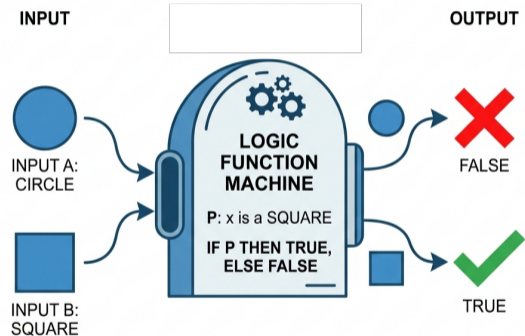
D'une proposition à une propriété :

- Avant : $A : \text{Prop}$
- Maintenant : Type domaine T
- Prédicat : $P : T \rightarrow \text{Prop}$

Les Prédicats comme Fonctions

D'une proposition à une propriété :

- Avant : $A : \text{Prop}$
- Maintenant : Type domaine T
- Prédicat : $P : T \rightarrow \text{Prop}$
- La propriété est testée sur un individu x .



Le Quantificateur Universel (\forall)

- Noté `forall x : T, P x` en Rocq.

Le Quantificateur Universel (\forall)

- Noté `forall x : T, P x` en Rocq.
- Informatiquement : un **type dépendant**.

Le Quantificateur Universel (\forall)

- Noté $\text{forall } x : T, P \ x$ en Rocq.
- Informatiquement : un **type dépendant**.
- C'est une fonction qui à tout x associe une *preuve* de $P(x)$.

Le Quantificateur Universel (\forall)

- Noté `forall x : T, P x` en Rocq.
- Informatiquement : un **type dépendant**.
- C'est une fonction qui à tout x associe une *preuve* de $P(x)$.
- L'implication $A \rightarrow B$ est juste un cas particulier !

Le Quantificateur Existentiel (\exists)

- Noté `exists x : T, P x` en Rocq.

Le Quantificateur Existentiel (\exists)

- Noté `exists x : T, P x` en Rocq.
- Constructivisme exige : il faut trouver *le témoin*.

Le Quantificateur Existentiel (\exists)

- Noté `exists x : T, P x` en Rocq.
- Constructivisme exige : il faut trouver *le témoin*.
- Preuve = Paire (t, π)
 - ▶ $t : T$ (le témoin)
 - ▶ $\pi : P(t)$ (la justification)

Le Quantificateur Existentiel (\exists)

- Noté `exists x : T, P x` en Rocq.
- Constructivisme exige : il faut trouver *le témoin*.
- Preuve = Paire (t, π)
 - ▶ $t : T$ (le témoin)
 - ▶ $\pi : P(t)$ (la justification)
- Mêmes mécanismes que la conjonction \wedge .

L'Égalité Propositionnelle (=)

Leibniz & Substitution

- $x = y$ est une proposition.

L'Égalité Propositionnelle (=)

Leibniz & Substitution

- $x = y$ est une proposition.
- Indiscernabilité conceptuelle.

L'Égalité Propositionnelle (=)

Leibniz & Substitution

- $x = y$ est une proposition.
- Indiscernabilité conceptuelle.
- Tactique clé : rewrite



Égalité Définitionnelle (\equiv) vs Propositionnelle ($=$)

- Propositionnelle : Exige une *preuve* manuelle (rewrite, lemmes).

Égalité Définitionnelle (\equiv) vs Propositionnelle ($=$)

- Propositionnelle : Exige une *preuve* manuelle (rewrite, lemmes).
- Définitionnelle : La *machine* évalue le code pour vous.
- Exemples :
 - ▶ $2 + 2 \equiv 4$ (calcul)
 - ▶ `doubler 2` \equiv 4 (expansion/ β -réduction)

Égalité Définitionnelle (\equiv) vs Propositionnelle ($=$)

- Propositionnelle : Exige une *preuve* manuelle (rewrite, lemmes).
- Définitionnelle : La *machine* évalue le code pour vous.
- Exemples :
 - ▶ $2 + 2 \equiv 4$ (calcul)
 - ▶ doubler $2 \equiv 4$ (expansion/ β -réduction)
- Tactique : reflexivity triomphe sans effort sur \equiv .

Synthèse

- `forall` \sim fonctions (\rightarrow) / tactique `intro`
- `exists` \sim paires (\wedge) / tactique `destruct`
- `rewrite` pour la substitution équationnelle
- `reflexivity` pour laisser l'ordinateur calculer