

Preuves de programmes – TP n° 1

Premiers pas en Coq

Pour résoudre les exercices, on s'aidera de la correspondance suivante

Tactique	Règle(s) de déduction
assumption	Axiome
intro, intros	\Rightarrow -intro, \forall -intro, \neg -intro
apply	\Rightarrow -élim, \forall -élim, \neg -élim
split	\wedge -intro, \top -intro
left, right	\vee -intro ₁ , \vee -intro ₂
exists	\exists -intro
destruct	\wedge -gauche, \vee -gauche, \perp -gauche, \top -gauche, \exists -gauche

et de la documentation en ligne :

- le petit guide : <http://www.pps.jussieu.fr/~miquel/enseignement/mpri/guide.html>
- la documentation complète : <http://coq.inria.fr/doc>
- un tutoriel et une F.A.Q. sont aussi disponible sur <http://coq.inria.fr>

Exercice 1 (Calcul propositionnel) Établir en Coq les formules suivantes :

1. forall A : Prop, A -> A
2. forall A B C : Prop, (A -> B) -> (B -> C) -> A -> C
3. forall A B : Prop, A /\ B <-> B /\ A
4. forall A B : Prop, A \/ B <-> B \/ A
5. forall A : Prop, A -> ~~A
6. forall A B : Prop, (A -> B) -> ~B -> ~A
7. forall A : Prop, ~~(A \/ ~A)
8. forall A B C : Prop, (A \/ B) /\ C <-> (A /\ C) \/ (B /\ C)
9. forall A B C : Prop, (A /\ B) \/ C <-> (A \/ C) /\ (B \/ C)

Exercice 2 (Calcul des prédicats) Après avoir effectué les déclarations suivantes

```
Parameter X Y : Set.
Parameter A B : X -> Prop.
Parameter R : X -> Y -> Prop.
```

établir en Coq les formules suivantes :

1. (forall x : X, A x /\ B x) <-> (forall x : X, A x) /\ (forall x : X, B x)
2. (exists x : X, A x \/ B x) <-> (exists x : X, A x) \/ (exists x : X, B x)
3. (exists y : Y, forall x : X, R x y) -> (forall x : X, exists y : Y, R x y)

Que peut-on dire de la réciproque de la dernière formule ?

Exercice 3 (Relations d'ordre) En Coq, on considère un type $E : \text{Set}$ muni d'une relation binaire R dont on suppose qu'elle satisfait aux axiomes des relations d'ordre :

Parameter E : Set.

Parameter R : E -> E -> Prop.

Axiom refl : forall (x : E), R x x

Axiom trans : forall (x y z : E), R x y -> R y z -> R x z.

Axiom antisym : forall (x y : E), R x y -> R y x -> x = y.

On définit les notions de plus petit élément et d'élément minimal de la façon suivante :

Definition smallest (x0 : E) := forall (x : E), R x0 x.

Definition minimal (x0 : E) := forall (x : E), R x x0 -> x = x0.

Quels sont les types des objets `smallest` et `minimal` ?

Énoncer en Coq puis démontrer les lemmes suivants :

1. Si R admet un plus petit élément, alors celui-ci est unique.
2. Le plus petit élément, s'il existe, est un élément minimal.
3. Si R admet un plus petit élément, alors il n'y a pas d'autre élément minimal que celui-ci.

Indications : En Coq, une définition s'utilise en remplaçant le *definiendum* par son *definiens* à l'aide de la tactique `unfold <definiendum>` (`unfold` = déplier). L'égalité se traite à l'aide des tactiques `reflexivity`, `symmetry`, `transitivity <terme>` et `rewrite <hypothèse>`.

Exercice 4 (Le paradoxe des buveurs) Dans cet exercice, on suppose la règle de raisonnement par l'absurde, que l'on déclare en Coq de la manière suivante :

Axiom not_not_elim : forall A : Prop, ~~A -> A.

1. Montrer en Coq que cet axiome entraîne le tiers-exclus : `forall A : Prop, A \/ ~ A`.

On se propose maintenant de formaliser le paradoxe des buveurs, dû à Smullyan :

*Dans toute pièce non vide on peut trouver une personne ayant la propriété suivante :
Si cette personne boit, alors tout le monde dans la pièce boit.*

2. Déclarer en Coq les divers éléments du problème (en s'inspirant de l'exercice 3).
3. Énoncer le paradoxe et en effectuer la preuve (laquelle repose sur le tiers-exclus).