TP 1 - Logique dans Coq

1 Utilisation de Coq

Sur `turing`, `Coq` peut être utilisé directement en ligne de commande (`coqtop`). L'exécutable `/usr/bin/coqtop` permet de lancer la boucle d'interprétation, `/usr/bin/coqc` de compiler des théories. Néanmoins, on privilégiera l'utilisation de l'interface graphique `coqide` plus conviviale.

La documentation complète du système est disponible en ligne `http://coq.inria.fr`, notamment le manuel de référence (en anglais).

2 Démonstrations en logique intuitionniste

On reprend les exemples vus en cours. Le tableau ci-dessous rappelle la correspondance entre les règles d'introduction et d'élimination et les tactiques de Coq. On rappelle que $\bot \equiv$ `False` et $\neg A \equiv A \rightarrow \bot$.

	règle	tactique	règle	tactique
$\rightarrow$	$\dfrac{\Gamma, H : A \rightarrow B \vdash A}{\Gamma, H : A \rightarrow B \vdash B}$	`apply` H	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$	`intros`
$\rightarrow$	$\dfrac{\Gamma \vdash A \quad \Gamma, A \vdash P}{\Gamma \vdash P}$	`assert` A		
$\wedge$	$\dfrac{\Gamma, A, B \vdash P}{\Gamma, A \wedge B \vdash P}$	`destruct` H	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	`split`
$\vee$	$\dfrac{\Gamma, A \vdash P \quad \Gamma, B \vdash P}{\Gamma, H : A \vee B \vdash P}$	`destruct` H	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$	`left` *ou* `right`
$\bot$	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$	`exfalso`		
$\neg$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}$	`absurd` A	$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$	`intro`
$\exists$	$\dfrac{\Gamma, x : A, P\ x \vdash Q}{\Gamma, H : (\exists x : A, P\ x) \vdash Q}$	`destruct` H	$\dfrac{\Gamma, v : A \vdash P\ v}{\Gamma, v : A \vdash \exists x : A, P\ x}$	`exists` v
$\forall$	$\dfrac{\Gamma \vdash P\ t}{\Gamma, H : (\forall x : A, P\ x \rightarrow Q\ x) \vdash Q\ t}$	`apply` H	$\dfrac{\Gamma, x : A \vdash P\ x}{\Gamma \vdash \forall x : A, P\ x}$	`intros`
	$\dfrac{}{\Gamma \vdash A} A \in \Gamma$	`assumption`		

1- Démontrer en Coq les énoncés suivants :
- `l0:` $\forall A\ B\ C :$ `Prop`$, (A \rightarrow B \wedge B \rightarrow C) \rightarrow (A \rightarrow C)$
- `l1:` $\forall A\ B :$ `Prop`$, A \vee B \rightarrow B \vee A$
- `l2:` $\forall A\ B\ C :$ `Prop`$, ((A \wedge B) \rightarrow C) \rightarrow A \rightarrow B \rightarrow C$
- `l3:` $\forall A\ B\ C :$ `Prop`$, (A \rightarrow B \rightarrow C) \rightarrow (A \wedge B) \rightarrow C$
- `l4:` $\forall A\ B\ C :$ `Prop`$, (A \wedge (B \vee C)) \rightarrow ((A \wedge B) \vee (A \wedge C))$
- `l5:` $\forall A\ B\ C :$ `Prop`$, ((A \wedge B) \vee (A \wedge C)) \rightarrow (A \wedge (B \vee C))$

Ne pas oublier de faire *valider* la démonstration par le système avec la commande `Qed`.

2- Observer les termes de preuve construits par le système avec la commande `Print` suivi du nom d'un théorème, par exemple `Print l1`. On peut également observer les étapes de construction du terme de preuve grâce à la commande `Show Proof.` à n'importe quel moment de la démonstration.

3- Ajouter l'axiome du tiers-exclu de la logique classique par la commande suivante :

```
Axiom tiers_exclu : forall A:Prop, A \/ ~A.
```

Démontrer les propriétés suivantes en logique classique :
— Contraposée : $\forall A\ B : \texttt{Prop}, (A \to B) \longleftrightarrow (\neg B \to \neg A)$
— Double-négation : $\forall A : \texttt{Prop}, \neg(\neg A) \longleftrightarrow A$
— Loi de Pierce : $\forall A\ B : \texttt{Prop}, ((A \to B) \to A) \to A$

4- Démontrer dans Coq l'énoncé suivant :

```
forall (A : Set) (B : A -> A -> Prop),
(exists y : A, (forall x : A, B x y)) -> (forall x : A, exists y : A, B x y).
```

Essayer de démontrer sa réciproque et expliquer pourquoi cela est impossible.