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 $\neg A = A \to \bot$.

	règle d'élimination	tactique à appliquer	règle d'introduction	tactique à appliquer
$\to \forall$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}$	`apply` H avec $H : A \to B$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	`intros`
$\wedge$	$\dfrac{\Gamma, A, B \vdash P \quad \Gamma \vdash A \wedge B}{\Gamma \vdash P}$	`elim` H avec $H : A \wedge B$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	`split`
$\vee$	$\dfrac{\Gamma, A \vdash P \quad \Gamma, B \vdash P \quad \Gamma \vdash A \vee B}{\Gamma \vdash P}$	`elim` H avec $H : A \vee B$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$ $\dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$	`left` *ou* `right`
$\bot$	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$	`elim` H avec $H : \bot$		
$\neg$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}$	`elim` H avec $H : \neg A$	$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$	`intro`
$\exists$	$\dfrac{\Gamma, x : A, P\ x \vdash Q \quad \Gamma \vdash \exists x : A, P\ x}{\Gamma \vdash Q}$	`elim` H avec $H : \exists x : A, P\ x$	$\dfrac{\Gamma, v : A \vdash P\ v}{\Gamma \vdash \exists x : A, P\ x}$	`exists` v

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

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 : $\forall A\ B$: `Prop`, $(A \to B) \longleftrightarrow (\neg B \to \neg A)$
- Double-négation : $\forall A$: `Prop`, $\neg(\neg A) \longleftrightarrow A$
- Loi de Pierce : $\forall A\ B$: `Prop`, $((A \to B) \to A) \to A$

N.B. Pour la loi de Pierce, on pourra utiliser la tactique `generalize` pour faire apparaître une instance de l'axiome du tiers-exclu dans le contexte et appliquer la règle d'élimination du $\vee$ sur cette hypothèse.

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.