Contrôle continu - Ingénierie de la preuve

Durée : 1h30

Les notes de cours, de travaux dirigés et de travaux pratiques sont autorisées. Le sujet comporte 2 pages et les trois parties sont complètement indépendantes. Les règles logiques utiles pour ce contrôle sont redonnées à la fin du sujet. Le barème est donné à titre indicatif.

On s'attachera à soigner la présentation, en particulier lors des démonstrations par récurrence et on sera vigilant quant à la syntaxe lors de l'écriture de fragment de code devant être accepté par `Coq`*.*

1 Facile (3 pts)

Question 1 Expliquer en quelques phrases la différence entre les sortes `Set` et `Prop`. En particulier expliquer pourquoi le type `nat` est défini dans `Set` et le type `le` (l'ordre sur les entiers naturels) dans `Prop`.
Question 2 A quoi sert la commande `Qed` utilisée à la fin des démonstrations ?

2 Logique et Isomorphisme de Curry-Howard (5 pts)

Question 3 Proposer une démonstration sous forme d'arbre en déduction naturelle pour la formule suivante :

$$\forall A\ B : \texttt{Prop}, (A \vee B) \rightarrow \neg B \rightarrow A.$$

On précisera bien à chaque étape quelle règle a été utilisée et on s'assurera que la démonstration proposée n'utilise pas les règles de la négation. On rappelle à cet effet que $\neg A$ est simplement un raccourci pour $A \rightarrow$ `False`.
Question 4 En déduire un terme de preuve pouvant utiliser les constructions suivantes et démontrant cette propriété.

```
or_introl : forall A B : Prop, A -> A \/ B
or_intror : forall A B : Prop, B -> A \/ B
or_ind : forall A B P : Prop, (A -> P) -> (B -> P) -> A \/ B -> P
False_ind : forall P : Prop, False -> P
```

Question 5 Soit p le terme de preuve suivant :

```
fun (A B C : Prop) (H : A -> B -> C) (H0 : A -> B) (H1 : A) => H H1 (H0 H1)
```

Indiquez de quelle formule c'est une démonstration.

3 Définitions inductives (12 pts)

3.1 Entiers naturels

On reprend la définition des entiers naturels étudiés en cours :

```
Inductive nat : Set := O : nat | S : nat -> nat.
```

Question 6 Programmer en `Coq` une fonction `puissance` de type `nat -> nat -> nat` qui, pour tout x : `nat` et n : `nat` calcule x^n.
Question 7 D'après votre définition de `puissance`, est-il possible de démontrer les lemmes suivants par réflexivité de l'égalité ? Si oui, expliquer pourquoi. Si non, proposer un autre moyen de les démontrer.

```
Lemma puissance_1 : forall x : nat, puissance x 0 = 1.
Lemma puissance_2 : forall n : nat, puissance 1 n = 1.
```

3.2 Piles d'entiers

On considère maintenant les piles d'entiers naturels :

```
Inductive pile : Set := vide : pile | empiler : nat -> pile -> pile.
```

Question 8 Programmer une fonction `vide ?` de type `pile -> Prop` pour tester si une pile est vide ou non.
Question 9 Programmer les fonctions `depiler` et `echanger`. `depiler` retire l'élément au sommet de la pile s'il existe, sinon ne fait rien. `echanger` échange le sommet et l'antésommet de la pile s'ils existent, sinon cette fonction retourne la pile fournie en entrée.
Question 10 Programmer une fonction `hauteur` de type `pile -> nat` permettant de calculer la hauteur d'une pile.

3.3 Raisonnement par récurrence

On considère maintenant une fonction `mod2` de calcul de la parité.

```
Fixpoint mod2 (n:nat) : nat :=
match n with
| O         => O
| (S O)     => 1
| (S (S p)) => mod2 p
end.
```

Question 11 Rappeler l'énoncé du principe d'induction habituel `nat_ind` sur les entiers naturels.
Question 12 On suppose que l'on dispose d'un principe d'induction de la forme suivante :

$$\forall P : \texttt{nat} \to \texttt{Prop}, P(0) \to P(1) \to (\forall n : \texttt{nat}, P(n) \to P(n+2)) \to \forall n : \texttt{nat}, P(n)$$

Expliquer (sans le faire en `Coq`) comment démontrer que $\forall n : \texttt{nat}, 0 \leq \texttt{mod2}\ n \leq 1$ en utilisant ce principe d'induction. On prendra soin de préciser la propriété P mise en jeu ainsi que les éventuelles hypothèses de récurrence.
Question 13 Comme on ne dispose malheureusement pas de ce principe d'induction dans `Coq`, on va chercher à montrer un énoncé plus général $\forall n : \texttt{nat}, (0 \leq \texttt{mod2}\ n \leq 1) \wedge (0 \leq \texttt{mod2}\ (S\ n) \leq 1)$. Expliquer (vous pourrez, dans un deuxième temps, essayer de l'écrire en `Coq`) comment démontrer cette propriété en utilisant le principe d'induction habituel `nat_ind` sur les entiers naturels.
Question 14 Comment déduire de cette démonstration que $\forall n : \texttt{nat}, 0 \leq \texttt{mod2}\ n \leq 1$?

A Rappel des règles logiques

On rappelle que $\bot \equiv \texttt{False}$ et $\neg A = A \to \bot$.

	règle d'élimination	règle(s) d'introduction
$\to \forall$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$
$\wedge$	$\dfrac{\Gamma, A, B \vdash P \quad \Gamma \vdash A \wedge B}{\Gamma \vdash P}$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$
$\vee$	$\dfrac{\Gamma, A \vdash P \quad \Gamma, B \vdash P \quad \Gamma \vdash A \vee B}{\Gamma \vdash P}$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$
$\bot$	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$	
$\exists$	$\dfrac{\Gamma, x : A, P\ x \vdash Q \quad \Gamma \vdash \exists x : A, P\ x}{\Gamma \vdash Q}$	$\dfrac{\Gamma, v : A \vdash P\ v}{\Gamma \vdash \exists x : A, P\ x}$