# 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 induction et on sera vigilant quant à la syntaxe lors de l'écriture de fragment de code devant être accepté par* `Coq`*.*

## 1 Questions de cours (2 pts)

**Question 1** On considère la définition inductive suivante :

```
Inductive X : nat -> Prop :=
| X0 : (X 0)
| X1 : forall n:nat, (X n) -> X (S (S n)).
```

Expliquer quel est le prédicat décrit par X. Expliquer le rôle de chacun des constructeurs X0 et X1.
**Question 2** On considère la définition de la fonction puissance $x, n \mapsto x^n$ à savoir :

```
Fixpoint puissance (x:nat) (n:nat) {struct n} : nat :=
match n with
| 0 => S 0
| S p => x * (puissance x p)
end.
```

Ecrire les règles de calcul associées à cette définition. On rappelle que les règles demandées sont celles qui s'appliquent lors de l'appel à la tactique `simpl`.

## 2 Logique et Isomorphisme de Curry-Howard (6 pts)

**Question 3** Rappeler la différence entre logique intuitioniste et logique classique.
**Question 4** Pour chacune des formules suivantes, proposer une démonstration sous forme d'arbre en déduction naturelle :

$$(A \to B) \to (\neg A \vee B) \qquad (\neg A \vee B) \to (A \to B).$$

On travaillera *a priori* en logique intuitioniste. On pourra toutefois passer en logique classique si cela s'avère nécessaire, sous réserve de le justifier clairement.
**Question 5** En déduire une suite de tactiques `Coq` prouvant $\forall A\ B : \texttt{Prop}, (A \to B) \to (\neg A \vee B)$.
**Question 6** Déduire, toujours de la question 2, un terme de preuve démontrant l'énoncé suivant :

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

Si nécessaire, on pourra utiliser les constructions suivantes ainsi que le fait que $\neg A$ est simplement un raccourci pour $A \to \texttt{False}$.

```
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
```

# 3 Définitions inductives - entiers binaires positifs (12 pts)

On souhaite construire un type inductif représentant les entiers représentés en binaire. On commence par définir un type inductif, nommé `positive`, qui aura 3 constructeurs permettant de construire les entiers 1, $2x$ à partir de $x$ et $2x+1$ à partir de $x$. On notera que ce type ne permet de décrire que les entiers strictement positifs. Nous verrons plus loin comment l'étendre pour y intégrer 0.

**Question 7** Ecrire la définition de `positive` en `Coq`. Définir 4 constantes `un`, `trois`, `quatre` et `sept`, représentant respectivement les nombres positifs 1, 3, 4 et 7.

**Question 8** Programmer une fonction récursive $Sp$ : `positive` → `positive` de calcul du successeur (au sens des entiers de Peano) d'un nombre de type `positive`.

**Question 9** Programmer la fonction `f` : $x \mapsto 2 * x - 1$ sur le type `positive`. En déduire une manière de programmer la fonction prédécesseur $Pp$ sur le type `positive`. On prendra la convention que le prédecesseur de 1 est 1.

**Question 10** Ecrire le principe d'induction associé à la définition du type inductif `positive`.

**Question 11** Démontrer en `Coq` le théorème suivant : $\forall p$ : `positive`, $(Sp\ (f\ x)) =$ `times2` $x$, où `times2` est le constructeur correspondant à la fonction $x \mapsto 2 * x$. On précisera bien toutes les étapes de la démonstration, notamment les réductions intervenant lors de l'application de la tactique `simpl`.

**Question 12 (facile)** Définer un nouveau type inductif `bin` permettant de décrire tous les entiers binaires positifs y compris 0. On pourra réutiliser le type inductif `positive` dans cette définition. Ecrire le principe d'induction correspondant à cette définition inductive.

**Question 13** Déduire des questions précédentes une fonction $Sb$ : `bin` → `bin` de calcul du successeur.

**Question 14 (bonus)** Programmer une fonction d'addition sur le type `positive`. On déclarera une fonction `pplus` de type `positive` → `positive` → `bool` → `positive` où les deux premiers arguments de type `positive` sont les nombres à additionner et le troisième argument est un booléen indiquant à chaque appel récursif la présence ou non d'une retenue. Initialement la fonction est appelée avec l'argument booléen à `false`, i.e. pas de retenue.

# A Rappel des règles logiques

On rappelle que $\bot \equiv$ `False` et $\neg A = A \to \bot$.

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