Preuves assistées par ordinateur

Examen de la 1[e] session
7 juin 2006 – durée : 3h

Documents autorisés : notes de Cours/TD/TP

Le barème ci-dessous est indicatif et est susceptible d'être modifié.
Il est recommandé de ***lire*** *le sujet.*

Exercice 1 ($\approx$ 4 points) – Logique classique

Dériver les formules suivantes en calcul des séquents (LK) puis en déduction naturelle classique (NK) :

1. $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ (loi de Peirce)
2. $\exists x_0\,(B(x_0) \Rightarrow \forall x\; B(x))$ (paradoxe des buveurs)

Exercice 2 ($\approx$ 4 points) – Théories du premier ordre

On considère la théorie $\mathscr{T}$ du premier ordre dont la signature est formée par un symbole de fonction f d'arité 1 et un symbole de prédicat binaire $=$ d'arité 2, et dont les axiomes sont les suivants :

$$\begin{array}{ll}
(A_1) & \forall x\ (x = x) \\
(A_2) & \forall x\ \forall y\ (x = y \Rightarrow y = x) \\
(A_3) & \forall x\ \forall y\ \forall z\ (x = y \Rightarrow y = z \Rightarrow x = z) \\
(A_4) & \forall x\ \forall y\ (x = y \Rightarrow f(x) = f(y)) \\
(A_5) & \forall x\ \big(f(f(x)) = x\big)
\end{array}$$

Dériver dans la théorie $\mathscr{T}$ les deux théorèmes suivants :

1. $\forall x\ \forall y\ (f(x) = f(y) \Rightarrow x = y)$ (Injectivité)
2. $\forall y\ \exists x\ (y = f(x))$ (Surjectivité)

Dans cet exercice, on construira les dérivations en déduction naturelle intuitionniste (NJ).

Exercice 3 ($\approx$ 4 points) – Codages dans le système T

On rappelle que les coefficients du binôme C_n^p sont donnés pour tous $n, p \in \mathbb{N}$ par

$$C_n^p \;=\; \begin{cases} \dfrac{n!}{p!(n-p)!} & \text{si } p \leq n \\ 0 & \text{si } p > n \end{cases}$$

Une façon simple de calculer ces coefficients est d'utiliser les relations de récurrence :

$$C_0^0 \;=\; 1, \qquad C_0^{p+1} \;=\; 0, \qquad C_{n+1}^0 \;=\; 1 \qquad \text{et} \qquad C_{n+1}^{p+1} \;=\; C_n^p + C_n^{p+1} \qquad\qquad (n, p \in \mathbb{N})\,.$$

1. Définir dans le système T une fonction $\mathsf{F} : \mathsf{Nat} \to \mathsf{Nat}$ telle que $\mathsf{F}\ \overline{0} = \overline{1}$ et $\mathsf{F}\ \overline{n} = \overline{0}$ pour tout $n \geq 1$ (en notant $\overline{n}$ la représentation de l'entier n dans le système T).

2. Définir dans le système T une fonction $\mathsf{G} : (\mathsf{Nat} \to \mathsf{Nat}) \to (\mathsf{Nat} \to \mathsf{Nat})$ telle que pour toute fonction $f : \mathsf{Nat} \to \mathsf{Nat}$ on ait

$$\mathsf{G}\ f\ \overline{0} \ = \ \overline{1} \qquad \text{et} \qquad \mathsf{G}\ f\ (\overline{p+1}) \ = \ \mathsf{plus}\ (f\ p)\ (f\ (\overline{p+1})) \qquad\qquad (n \in \mathbb{N})\,,$$

   où $\mathsf{plus} : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$ designe la fonction calculant la somme de deux entiers dans le système T.

3. En déduire une fonction $\mathsf{C} : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$ telle que $\mathsf{C}\ \overline{n}\ \overline{p} \ = \ \overline{C_n^p}$ pour tous $n, p \in \mathbb{N}$.

Exercice 4 ($\approx$ 8 points) – Diamètre d'un arbre binaire en Coq

A – Graphes et chemins En Coq on représente un graphe (orienté) par un type A : `Set` dont les éléments représentent les sommets du graphe et une relation binaire $R : A \to A \to$ `Prop` donnant les arêtes du graphe, en convenant que $R\ x\ y$ exprime que « x est connecté à y par une arête ».

On suppose maintenant les paramètres A : `Set` et $R : A \to A \to$ `Prop` fixés dans cette partie.

1. Définir inductivement une relation `path` : $A \to A \to$ `Prop` telle que pour tous $x, y : A$, la proposition `path` $x\ y$ exprime qu'il existe un chemin (éventuellement vide) reliant x à y. On convient qu'un sommet $x : A$ est toujours relié à lui-même par le chemin vide.
2. Énoncer en Coq un lemme qui exprime que lorsque la relation R est symétrique, la relation `path` est symétrique également. (On ne demande pas la preuve.)

On se place à présent dans le cas particulier où la relation R est symétrique, c'est-à-dire dans le cas où A et R representent un graphe non orienté. Étant donnés deux nœuds $x, y : A$ reliés par au moins un chemin, on appelle *distance* de x à y la longueur du plus petit chemin qui relie x à y.

3. Définir un prédicat `dist` : $A \to A \to$ `nat` $\to$ `Prop` où pour tous $x, y : A$ et n : `nat` la proposition `dist` $x\ y\ n$ exprime que « la distance de x à y vaut n »

On appelle *diamètre* du graphe la distance maximale entre deux sommets du graphe.

4. Définir un prédicat `diam` : `nat` $\to$ `Prop` où pour tout entier n : `nat` la proposition `diam` n exprime que « le graphe (A, R) a pour diamètre n »

B – Positions dans un arbre binaire On considère à présent le type `tree` : `Set` des arbres binaires défini en Coq par :

```
Inductive tree : Set := Empty : tree | Node : tree → tree → tree.
```

Étant donné un arbre t : `tree`, n'importe quel sous-arbre de t peut être représenté par un chemin partant de la racine de t. Un tel chemin peut être représenté comme une liste de directions (`Left` ou `Right`). On introduit ainsi les types `dir` (le type des *directions*) et `pos` (le type des *positions*) en posant :

```
Inductive dir : Set := Left : dir | Right : dir.
Definition pos : Set := list dir.
```

5. Définir un prédicat `valid` : `tree` $\to$ `pos` $\to$ `Prop` exprimant qu'une position est *valide* pour un arbre t, en ce sens qu'elle pointe sur l'un des sous-arbres de t (éventuellement vide).
6. Définir un prédicat d'*adjacence* `adj` : `pos` $\to$ `pos` $\to$ `Prop` où pour tous p_1, p_2 : `pos`, la proposition `adj` $p_1\ p_2$ exprime que « le sous-arbre pointé par p_1 est un sous-arbre immédiat du sous-arbre pointé par p_2, ou inversement. » (On ne cherche pas à vérifier si ces positions sont valides pour un arbre donné.)
7. Définir un prédicat `gt` : `tree` $\to$ `pos` $\to$ `pos` $\to$ `Prop` où pour tous t : `tree` et p_1, p_2 : `pos`, la proposition `gt` $p_1\ p_2$ exprime que « les positions p_1 et p_2 sont toutes les deux valides pour t, et l'une représente un sous-arbre immédiat de l'autre. » En déduire (informellement) la notion de diamètre d'un arbre binaire.

Dans ce qui suit, on note *diam*(t) le diamètre d'un arbre binaire t.

C – Calcul du diamètre d'un arbre binaire

8. Soient t_1 et t_2 deux arbres. Montrer que

$$diam(\texttt{Node}\ t_1\ t_2) = \max\bigl(diam(t_1),\ diam(t_2),\ 2 + h(t_1) + h(t_2)\bigr)\,,$$

   où $h(t)$ désigne la hauteur de l'arbre t (en considérant que $h(\texttt{Empty}) = 0$).
   On demande une preuve ***informelle***, pas une preuve en Coq.
9. En déduire un algorithme calculant le diamètre d'un arbre binaire en temps linéaire par rapport au nombre de nœuds. (Indication : calculer la hauteur en même temps que le diamètre.)
10. Écrire en Coq la fonction qui calcule le diamètre d'un arbre binaire.