Université de Marne-la-Vallée. Septembre 2000
Examen de Logique. Licence et IUP d'informatique
Durée 2 heures. Tous les documents sont autorisés.

Tous les exercices sont indépendants. N'oubliez pas de tourner la page.

Exercice 1.— On définit en Caml le type suivant `'a arbre` qui permet de manipuler des arbres binaires dont les étiquettes des noeuds sont formés d'éléments de type `'a`.

```
type 'a arbre =
    Vide
  | Noeud of 'a * 'a arbre * 'a arbre ;;
```

On appelle arbre binaire de recherche un arbre binaire dont les étiquettes des noeuds sont des entiers tels que, pour tout noeud, les étiquettes des noeuds du sous-arbre gauche sont inférieures à l'étiquette du noeud, elle-même inférieure aux étiquettes des noeuds du sous-arbre droit.

a) (2 points)
Écrire une fonction Caml `split` qui prend en argument un entier `x` et un arbre binaire de recherche `t` et renvoie un couple d'arbres binaires de recherche `(g,d)` tels que les étiquettes de `g` sont inférieures ou égales à `x`, les étiquettes de `d` sont supérieures à `x`, et la réunion des étiquettes de `g` et `d` est l'ensemble des étiquettes de `t`. Donner le type Caml de cette fonction.

b) (2 point)
Écrire une fonction Caml `merge` qui prend en argument deux arbres binaires de recherche et renvoie un arbre binaire de recherche dont les étiquettes sont la réunion des étiquettes des arbres donnés en argument. Donner le type Caml de cette fonction.

c) (1 point)
Écrire une fonction Caml `concat` qui prend en argument un arbre binaire de recherche `t1` et un arbre binaire de recherche `t2` tels que la plus grande étiquette de `t1` est inférieure à la plus petite étiquette de `t2`, et renvoie un arbre binaire de recherche dont les étiquettes sont la réunion des étiquettes des arbres donnés en argument. Donner le type Caml de cette fonction. La complexité de `concat` devra être inférieure à la complexité de `merge`.

d) (1 point)
Ecrire une fonction Caml `list_merge` qui prend en argument une liste d'arbres binaires de recherche et renvoie un arbre binaire de recherche dont les étiquettes sont la réunion des étiquettes des arbres donnés en argument. Il est recommandé d'utiliser une fonctionnelle d'itération sur les listes dont on précisera le type. Donner le type Caml de cette fonction.

Exercice 2.—

a) (2 points)
Montrer que le système de connecteurs $\{\neg, \rightarrow\}$ du calcul propositionnel est sémantiquement complet. Montrer que par contre le système de connecteurs $\{\rightarrow\}$ n'est pas sémantiquement complet.

b) (2 points)
On définit en Caml le type suivant pour représenter des formules propositionnelles bâties sur les connecteurs $\neg, \vee, \wedge, \rightarrow$.

```
type formule  =
    Atome of char
  | Non of formule
  | Ou of formule * formule
  | Et of formule * formule
  | Imp of formule * formule
;;
```

Le constructeur d'arité 2 `Imp` représente le connecteur $\rightarrow$. Écrire une fonction Caml `transformation` qui prend en argument une formule f et renvoie une formule sémantiquement équivalente à f qui n'utilise que les connecteurs $\{\neg, \rightarrow\}$. Donner le type Caml de cette fonction.

Exercice 3.— Si a, b, c désignent des atomes du calcul des propositions, indiquer si les formules suivantes sont des tautologies, des antilogies ou ni l'un ni l'autre.

a) (2 points)
$((\neg a \rightarrow b) \wedge (a \rightarrow c)) \rightarrow (b \vee c)$

b) (2 points)
$((a \vee b) \rightarrow (a \wedge c)) \rightarrow (b \rightarrow c)$

Exercice 4.— (2 points)
On définit par récurrence structurelle une transformation sur les formules du calcul propositionnel, notée com, par

- $\mathrm{com}(a) = a$ si a est une formule atomique.
- $\mathrm{com}(\neg F) = \neg\, \mathrm{com}(F)$.
- $\mathrm{com}(F\ c\ G) = \mathrm{com}(G)\ c\ \mathrm{com}(F)$, où F, G sont des formules et c est un connecteur binaire dans $\{\wedge, \vee, \rightarrow\}$ (Noter l'inversion entre F et G).

Donner une tautologie F telle que $\mathrm{com}(F)$ n'est pas tautologie. Montrer, par récurrence structurelle sur les formules, que si F est une tautologie qui n'utilise comme seuls connecteurs binaires que $\wedge$ et $\vee$, alors $\mathrm{com}(F)$ est une tautologie.

Exercice 5.— Si a, b, c, d désignent des atomes (ou variables) du calcul des propositions, montrer que :

a) (2 points)
$(a \to b \to c \to d) \to (a \to b) \to (a \to c \to d)$;

Le système de déduction à utiliser est le système de Hilbert présenté en cours. On pourra utiliser tous les résultats du cours (notamment le théorème de déduction), à l'exclusion toutefois du théorème de complétude.

b) (1 point)
Définir une fonction Caml de type

```
('a->'b->'c->'d)->('a->'b)->'a->'c->'d
```

c) (1 point)
Est-il possible de définir une fonction Caml de type

```
('a->'b->'c->'d)->('a->'b->'c)->'a->'d
```

Pourquoi ?

Exercice 6.— (points en plus)
QUESTION SUBSIDIAIRE (ne traitez cette question que s'il vous reste du temps)
Soit $\mathcal{A}$ un ensemble infini de formules et F une formule (du calcul propositionnel) tels que $\mathcal{A} \models F$ (ce qui signifie que F est conséquence sémantique de $\mathcal{A}$. Montrer qu'il existe un sous-ensemble fini $\mathcal{B}$ de $\mathcal{A}$ tel que $\mathcal{B} \models F$.