Université de Marne-la-Vallée. Septembre 1999
Examen de Logique. Licence et IUP d'informatique
Durée 2 heures. Tous les documents sont autorisés.

Tous les exercices sont indépendants.

Exercice 1.— Une contrée est habitée par deux types d'individus : les Good et les Bad. Les Good disent toujours la vérité. Les Bad mentent toujours. Un étudiant logicien est confronté à trois habitants de cette contrée : A,B,C. Un dialogue s'engage à l'issue duquel l'étudiant sait immédiatement qui est Good et qui est Bad.
A : Nous sommes tous des Bad
B : Un seul d'entre nous est un Good
Lesquels d'entre A,B,C sont Good, lesquels sont Bad ?

Exercice 2.— On note $q\#_p r$ la formule $(q \wedge r) \vee (\neg p \wedge q) \vee (\neg p \wedge r)$. Déterminer lesquelles des formules suivantes sont des tautologies. On explicitera pour chaque formule soit une preuve (syntaxique ou sémantique), soit un contre-exemple.

- $(q\#_p r) \leftrightarrow (r\#_p q)$
- $((q\#_p r)\#_p s) \leftrightarrow (q\#_p(r\#_p s))$
- $((q\#_p r)\#_p(s\#_p q)) \leftrightarrow (s\#_p r)$

Exercice 3.— Si a,b,c désignent des propositions, montrer que :

1) $\vdash (a \to b \to a) \to a \to b \to b \to a$.
2) $\vdash (a \wedge b) \to ((a \wedge c) \vee (a \wedge \neg c))$.

Le système de déduction à utiliser est le système de Hilbert présenté en cours. On pourra utiliser le fait que $a \vee b$ est une abréviation de $\neg a \to b$. On pourra utiliser tous les résultats du cours (notamment le théorème de déduction), à l'exclusion toutefois du théorème de complétude.
Définir une fonction Caml du type :

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

Exercice 4.— Effectuer une résolution restreinte et ordonnée de la famille de clauses suivantes :
$C = \{p \vee r, q \vee r, q \vee s, \neg p \vee \neg q, \neg r \vee \neg s, \neg q \vee \neg r\}$.
Que peut-on en déduire sur la famille C ?

Exercice 5.— 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 ;;
```

Écrire une fonction Caml qui soit du type :

```
('a * 'b) arbre ->('a arbre) * ('b arbre)
```

Exercice 6.— On note $\mathcal{F}(\mathcal{C},\mathcal{P})$ les formules construites à partir de l'ensemble de connecteurs $\mathcal{C} = \{\wedge, \vee, \neg\}$ et de l'ensemble des formules atomiques $\mathcal{P}$. On note $\mathcal{F}'(\mathcal{C},\mathcal{P})$ les formules construites à partir des seuls connecteurs $\wedge$ et $\vee$ et des formules élémentaires. On dit qu'une formule est *élémentaire* si elle est une formule atomique ou la négation d'une formule atomique.

- Montrer que toute formule de $\mathcal{F}(\mathcal{C},\mathcal{P})$ est sémantiquement équivalente à une formule de $\mathcal{F}'(\mathcal{C},\mathcal{P})$. On pourra utiliser les lois de De Morgan.
- Écrire une fonction Caml `simplifie` qui transforme une formule de $\mathcal{F}(\mathcal{C},\mathcal{P})$ en une formule de $\mathcal{F}'(\mathcal{C},\mathcal{P})$. Le type des formules sera le suivant :

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

- Simplifier la formule $\neg((p \vee q) \wedge r)$