L3 - Programmation logique et fonctionnelle
Examen - juin 2009

Modalités

Les documents distribués en cours, TD et TP, ainsi que les notes personnelles, sont autorisés.

1 λ-calcul

Soit le terme suivant :

$$(\lambda t.\lambda x.tx)\lambda y.y$$

1. Combien y a t-il d'abstractions dans ce terme ?
2. Combien y a t-il d'occurrences libres de variables dans ce terme ?
3. Détaillez l'évaluation de ce terme et donnez son résultat.

2 Logique propositionnelle

Soit la formule suivante :

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

1. Quelle est la valeur de vérité de cette formule pour l'interprétation $\{a = \texttt{faux}, b = \texttt{faux}, c = \texttt{vrai}\}$?
2. Cette formule est elle satisfaisable ? Est elle valide ? Justifiez votre réponse.

3 Logique du premier ordre

3.1

La formule suivante est elle valide ? Est elle satisfaisable ? Justifiez votre réponse.

$$\forall X \exists Y (p(X) \rightarrow p(Y))$$

3.2

Modélisez la phrase « *Tous les hommes qui n'ont ni voiture ni chat possèdent un chien* » sous la forme d'une formule de logique du premier ordre en utilisant les prédicats `homme/1`, `possède/2`, `voiture/1`, `chat/1` et `chien/1`.

4 Listes en CAML et PROLOG

4.1

Complétez la définition suivante pour que la fonction `maxi` appliquée à une liste d'entiers retourne la plus grande valeur de cette liste :

```
let rec maxi w = match w with
  |[]->failwith("erreur␣:␣liste␣vide")
  |[x]-> ...
  |t::q-> ... ;;
```

4.2

Complétez la définition suivante pour que le prédicat `maxi(L,M)`, où `L` est supposé être une liste d'entiers, soit satisfait si et seulement si `M` est instanciée avec la plus grande valeur de `L` :

```
maxi([X],X).
maxi([A|Q],M) :- ...
maxi([A|Q],A) :- ...
```

5 Arbres en CAML et PROLOG

On appelle (dans la cadre de cet exercice) *arbre mixte* un arbre constitué d'une ou plusieurs feuilles, de zéro, un, ou plusieurs noeuds binaires et de zéro, un ou plusieurs noeuds unaires (i.e., ayant un seul fils).
Voici par exemple une représentation d'un tel arbre en CAML :

```
type mixte = F | U of mixte | B of mixte * mixte;;
let t = B(F,U(B(F,F)));;
```

En PROLOG, cet arbre sera représenté par le terme fonctionnel :

```
b(f,u(b(f,f)));;
```

5.1

La fonction CAML suivante calcule le nombre de noeuds unaires d'un arbre mixte :

```
let rec nu t = match t with
  |F -> 0
  |U(w) -> 1 + (nu w);
  |B(g,d) -> (nu d) + (nu g);;
```

1. Modifiez cette fonction de manière à définir une fonction `nf` qui retourne le nombre de feuilles d'un arbre mixte.
2. En vous inspirant du principe de la fonction CAML `nu`, définissez un prédicat PROLOG `nu/2` tel que `nu(T,N)` soit satisfait si et seulement si `T` est un arbre mixte comportant `N` noeuds unaires.

5.2

Un *chemin* est une suite de noeuds commençant par la racine d'un arbre, se terminant par une feuille, et telle que chaque noeud est un fils de celui qui le précède dans la suite. On appelle *chemin binaire* tout chemin ne comportant aucun noeud unaire.

1. Complétez la définition suivante de manière à ce que la fonction `cb` appliquée à un arbre mixte `t` retourne `true` si et seulement si `t` comporte au moins un chemin binaire.

   ```
   let rec cb t = match t with
     |F-> ...
     |U(_)-> ...
     |B(g,d)-> (...) or (...);;
   ```

2. Définissez un prédicat `cb/1` tel que pour tout arbre mixte `T`, `cb(T)` soit satisfait si et seulement si `T` comporte au moins un chemin binaire.