L3 - Programmation fonctionnelle

Examen - session de juin 2008

Modalités

Tous les documents sur papier sont autorisés. Les calculatrices sont autorisées. Durée 2h00.

1 Lambda-calcul (3 points)

Réduisez les termes de lambda-calcul suivants, en identifiant à chaque étape la tête, le corps et l'argument du redex.

1. $(\lambda x.\lambda y.yx)(\lambda x.xx)$
2. $(\lambda x.(\lambda y.y)x)(\lambda x.xx)$
3. $(\lambda x.\lambda y.yx)((\lambda x.x)x)$

2 Logique

Justifiez de manière concise chacune de vos réponses.

2.1 Logique propositionnelle (3 points)

1. Quelle est la valeur de vérité de la formule $a \rightarrow (b \wedge (a \rightarrow c))$ lorsque $a =$ `vrai`, $b =$ `vrai` et $c =$ `faux` ?
2. Modélisez la phrase « Le raccordement à l'eau et à l'électricité sont deux conditions nécessaires pour que la maison soit habitable. » en logique propositionnelle avec les variables `eau`, `électricité` et `habitable`.
3. La formule propositionnelle $a \rightarrow ((a \rightarrow b) \rightarrow (b \rightarrow a))$ est elle valide ? est elle satisfaisable ?

2.2 Logique du premier ordre (4 points)

1. On considère la formule $G = \forall X \exists Y((p(X) \wedge p(Y)) \rightarrow (q(X) \vee q(Y)))$ et l'interprétation I sur le domaine $\{1, 2, 3, 4\}$ telle que $p(1) =$ `vrai`, $p(2) =$ `vrai`, $p(3) =$ `faux`, $p(4) =$ `faux`, $q(1) =$ `vrai`, $q(2) =$ `faux`, $q(3) =$ `vrai`, $q(4) =$ `faux`.
   - Quelle est la valeur de vérité de G pour cette interprétation I ?
   - G est elle valide ?
2. Modélisez en logique du premier ordre la phrase « Tout homme ayant une voiture ne possède ni chien, ni chat. » en utilisant les prédicats `homme/1` `possede/2` `chien/1` `chat/1`.

3 Prolog et Caml

Il sera tenu compte de la *simplicité* et de la *concision* de vos programmes. Les programmes en langage Caml devront être écrits dans un style purement fonctionnel.

3.1 Listes (4 points)

1. Définissez une fonction Caml `verif` qui, appliquée à une liste d'entiers q, retourne `true` si les éléments de q sont triés par ordre croissant, et `false` dans le cas contraire.
2. Spécifiez en Prolog un prédicat `verif/1` tel que pour toute liste d'entiers `L`, `verif(L)` soit satisfait si et seulement si les éléments de `L` sont triés par ordre croissant.

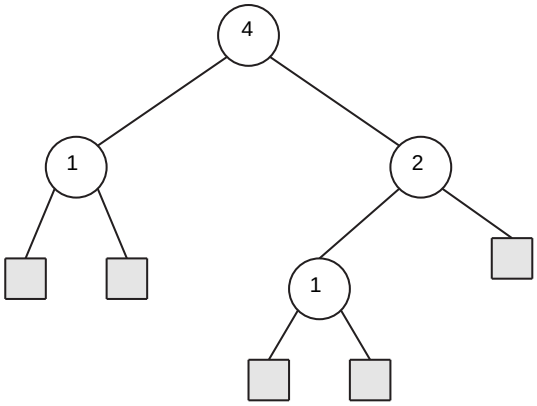

FIG. 1 – Un exemple d'arbre informé.

3.2 Arbres (6 points)

Introduction et définitions

On considère le type `arbre` défini de la manière suivante : un `arbre` est
- soit une feuille, n'ayant aucun attribut,
- soit un noeud ayant pour attributs :
  - un entier appelé *étiquette*,
  - deux arbres appelés respectivement fils gauche et fils droit.

La *taille* d'un `arbre` est le nombre de noeuds qu'il comporte : la taille d'une feuille est 0, celle d'un noeud vaut 1 plus la taille de son fils droit plus celle de son fils gauche.

On dit qu'un `arbre` t est *informé* lorsque la valeur de l'étiquette de tout noeud de t est la taille de l'arbre ayant ce noeud pour racine (un exemple est donné figure 1).

On définit la notion de *sous-arbre* d'un arbre t de la manière suivante :
- si t est une feuille alors il n'a aucun sous-arbre,
- si t est un noeud, il a pour sous-arbres son fils droit, son fils gauche, tous les sous-arbres de son fils droit, et tous les sous-arbres de son fils gauche.

Partie Caml

Écrivez une fonction `testinf` qui, appliquée à un `arbre`, retourne la taille de cet arbre s'il est informé, et `-1` dans la cas contraire.

Partie Prolog

1. Spécifiez un prédicat `testinf/2` tel que `testinf(T,N)` soit satisfait si et seulement si `T` est un arbre informé de taille `N`.
2. Spécifiez un prédicat `sa/2` tel que pour tout arbre `T`, `sa(T,I)` soit satisfait si et seulement si `I` est un sous-arbre de `T`.