Programmation logique et fonctionnelle - contrôle final - session 2

Documents autorisés : Tous documents manuscrits et imprimés.

Lambda-calcul

Exercice 1 (1 point)

Décomposez le terme suivant en abstractions (à encadrer en pointillés) et applications (à encadrer en trait plein). Entourez les variables libres.

$\lambda x.yx\lambda y.ax$

Exercice 2 (1 point)

Pour chacune des paires de termes suivantes, indiquez si les deux termes sont équivalents.

1. $\lambda x.x\lambda x.x$ et $(\lambda x.x)\lambda x.x$
2. $\lambda x.x\lambda x.x$ et $\lambda x.x(\lambda x.x)$
3. $\lambda x.x\lambda x.x$ et $\lambda x.x\lambda y.y$
4. $a\lambda x.x$ et $b\lambda x.x$

Exercice 3 (2 points)

Évaluez chacun des termes suivants. Encadrez le redex utilisé à chaque étape.

1. $(\lambda x.\lambda y.xyyx)ab$
2. $(\lambda x.\lambda y.xyyx)(ab)$y

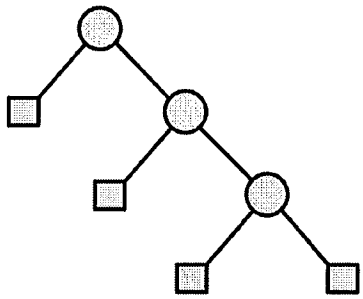
CAML

Exercice 4 (2 points)

On appelle somme de deux listes $[x_1,\ldots,x_k]$ et $[y_1,\ldots,y_k]$ la liste $[x_1+y_1,\ldots,x_k+y_k]$. Si l'une des listes est plus longue que l'autre, le résultat est complété avec les éléments restants de la plus longue liste. Par exemple, la somme de [1,2] et [3,4,5] a pour résultat [4,6,5]. Réalisez une fonction `sum` telle que si `w1` et `w2` sont deux listes d'entiers alors `sum w1 w2` retourne la somme des listes `w1` et `w2` telle que définie préalablement.

Exercice 5 (2 points)

On appelle peigne droit tout arbre dont tous les fils gauches sont des feuilles. Voici un exemple de peigne droit. Une feuille est un cas particulier de peigne droit.



Vous devez définir un type **arbre** permettant de représenter des arbres binaires non étiquetés. Un tel arbre est soit une feuille, soit constitué d'un nœud racine ayant un fils gauche et un fils droit qui sont des arbres binaires non étiquetés. Vous devez réaliser une fonction pgd telle que si `t` est un arbre alors pgd `t` retourne le booléen `true` si `t` est un peigne droit et `false` dans le cas contraire.

Logique propositionnelle

Exercice 6 (2 points)

Voici des formules. Donnez la liste de celles qui sont cohérentes mais non valides et la liste de celles qui sont valides (i.e., qui sont des tautologies).

1. $(a \wedge b) \rightarrow (a \vee b)$
2. $(a \vee b) \rightarrow (a \wedge b)$
3. $(a \vee b) \rightarrow (a \vee b \vee c)$
4. $(a \wedge b) \rightarrow (a \wedge b \wedge c)$

Exercice 7 (1 point)

Donnez tous les modèles de la formule suivante :

$(a \wedge \neg b \wedge \neg c \wedge d \wedge e) \vee (\neg a \wedge b \wedge c \wedge \neg d \wedge f)$

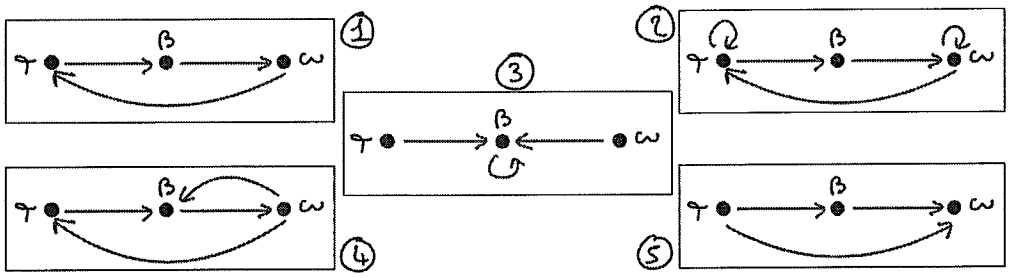
Exercice 8 (1 point)

Les variables a_2, a_1, a_0 représentent un entier A en base 2. Par exemple, $a_2 = 0, a_1 = 1, a_0 = 1$ signifie que $A = 3$. Donnez une formule propositionnelle qui modélise la propriété $A < 6$.

Logique du 1er ordre

Exercice 9 (2 points)

Soit la formule $\Sigma = \forall X \exists Y (p(X,Y) \wedge \neg p(Y,X))$. Parmi les interprétations suivantes, donnez la liste de celles qui satisfont Σ.



Exercice 10 (2 points)

Modélisez la phrase "Quiconque fait confiance à quelqu'un qui a trompé tout le monde commet une erreur" en utilisant les symboles de prédicats suivants :

- `confiance/2` : `confiance(X,Y)` est vrai si et seulement si X fait confiance à Y.
- `trompe/2` : `trompe(X,Y)` est vrai si et seulement si X a trompé Y.
- `erreur/1` : `erreur(X)` est vrai si et seulement si X commet une erreur.

Prolog

Exercice 11 (2 points)

Compléter la définition du prédicat `lcut` de manière à ce que le but...

```
lcut(L,N,G,D).
```

...où L est une liste d'entrée, N est un entier également donné en entrée, G et D sont deux variables utilisées en sortie, fait remonter dans G la liste des N premiers éléments de L et dans D la liste des éléments suivants. Si N est supérieur à la longueur de L, tous les éléments de de L vont dans G et D est vide. Voici quelques exemples de buts et leur résultats :

- `lcut([1,2,3,4,5],7,G,D).` → `G = [1,2,3,4,5], D = [].`
- `lcut([1,2,3,4,5],2,G,D).` → `G = [1,2], D = [3,4,5].`
- `lcut([1,2,3,4,5],0,G,D).` → `G = [], D = [1,2,3,4,5].`

```
lcut([],_,   ,   ).
lcut(G,0,   ,   ).
lcut([T|Q],N,[T|R],D) :-
    N>0,
```

Exercice 12 (2 points)

Il est possible de représenter des listes par des termes fonctionnels avec la convention suivante : la liste vide est représentée par `v` et toute liste non vide est représentée par `tq(T,R)`, où T est le premier élément de la liste et R la représentation du reste de la liste. Par exemple, la liste `[1,2,3]` sera représentée par `tq(1,tq(2,tq(2,v)))`. Donnez la définition d'un prédicat `trad/2` permettant de traduire une liste représentée de manière standard dans la représentation fonctionnelle et vice-versa. Voici deux exemples d'utilisation de ce prédicat :

- `trad(tq(1,tq(2,tq(3,v))),L).` → `L = [1,2,3].`
- `trad(R,[1,2,3]).` → `R = tq(1,tq(2,tq(3,v))).`
