Examen de programmation logique et fonctionnelle

Licence d'informatique, Université de Bourgogne, UFR Sciences et Techniques.

Sujet de la session 2, juin 2023. Durée 2h.

Documents autorisés : 5 pages A4 recto-verso manuscrites ou imprimées, avec le contenu de votre choix.

Calculatrices et appareils électroniques communicants interdits. Écrivez vos réponses directement sur le sujet, dans les espaces laissés après les questions.

Lambda-calcul (4 points)

Décomposez le terme de λ-calcul suivant en abstractions et applications (vous pouvez par exemple encadrer les applications et souligner les abstractions) et entourez ses variables libres.

$$\lambda x.\ z\ (\lambda z.yx)\ a\ \lambda x.x$$

Évaluez complètement le terme $(\lambda d.\lambda t.\lambda x.tdx)ab$ en soulignant le redex utilisé à chaque étape.

Évaluez complètement le terme $(\lambda d.(\lambda w.w)(\lambda w.w)d)b$ en soulignant le redex utilisé à chaque étape.

CAML (4 points)

On définit le produit scalaire de deux listes comme la somme des produits des valeurs de même rangs. Si une des listes est plus longue que l'autre, les valeurs de la plus grande liste ayant un rang supérieur à celui de la dernière valeur de l'autre liste ne sont pas prises en compte.

Par exemple, le produit scalaire de [1;2;3] et de [10;20;30;45] vaut 1x10 + 2x20 + 3x30 = 140. La valeur 45 de la plus longue liste est ignorée car il n'y a pas de valeur de même rang dans la plus petite.

Définissez la fonction `ps` telle que si `a` et `b` sont des listes, alors `ps a b` retourne le produit scalaire de `a` et `b`.

On représente des séquences de valeurs Booléennes avec le le type suivant :

```
type seq = End | S of bool * seq;;
```

Définissez la fonction `count` telle que si `w` est une séquence Booléenne de type `seq` et `x` une valeur Booléenne `true` ou `false`, alors `count w x` retourne le nombre d'occurrences de la valeur `x` dans la séquence `w`.

Logique propositionnelle (4 points)

Soit la formule Σ suivante :

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

Donnez 4 modèles de Σ.

Combien y a t-il de modèles de Σ ?

Indiquez, pour chacune des formules suivantes, si elle est cohérente et si c'est une tautologie. Répondez en écrivant oui ou non dans chacune des cases blanches du tableau.

	Cohérente	Tautologie
$a \rightarrow \neg a$		
$(a \wedge b) \vee (\neg a \wedge \neg b)$		
$(a \vee b) \vee (\neg a \vee \neg b)$		
$(a \vee b) \wedge (\neg a \vee \neg b)$		
$(a \wedge b) \wedge (\neg a \wedge \neg b)$		

Logique du premier ordre (4 points)

Soit la formule Σ suivante :

$\forall X[(\exists Y(P(X,Y) \wedge P(Y,X))) \rightarrow P(X,X)]$

Donnez, en représentation sagittale, trois contre-modèles de Σ avec le domaine d'interprétation $\{1,2\}$.

Existe-t-il plus de trois contre-modèles avec ce domaine d'interprétation ? Justifiez brièvement votre réponse.

En utilisant les symboles relationnels `patron/2` et `salarié/1`, modélisez l'énoncé suivant sous la forme d'une formule de la logique du premier ordre :

> *Toute personne salariée a au moins un patron, et toute personne ayant au moins un patron est salariée.*

PROLOG (4 points)

Le prédicat `concat` est défini de la manière suivante :

```
concat([],B,B).
concat([T|R],B,[T|Q]) :-  concat(R,B,Q).
```

Il définit une relation entre trois listes qui est vérifiée si et seulement si la troisième liste résulte de la concaténation des deux premières. Il est réversible, c'est à dire que les deux premiers slots peuvent être utilisés en entrées avec le troisième en sortie, mais il est aussi possible, par exemple, d'utiliser le troisième slot en entrée, avec les deux premiers en sortie.

Donnez tous les résultats du but suivant :

```
concat(P,[E|S],[1,2,3]).
```

En utilisant le prédicat `concat`, définissez le prédicat `retire/3` tel que si `L` est une liste connue (entrée) et les variables E et R ne sont pas instanciées (sorties), alors le but...

```
retire(L,E,R).
```

...fait remonter dans la variable `E` un élément de `L` et dans `R` la liste obtenue en retirant cet élément de `L`.

Par exemple, le but...

```
retire([1,2,3],E,R).
```

...doit avoir les résultats suivants :

```
E = 1, R = [2, 3]
E = 2, R = [1, 3]
E = 3, R = [1, 2]
```
