Examen de programmation logique et fonctionnelle

L3 informatique - Université de Bourgogne - année 2023-2024

Documents autorisés : trois feuilles A4 recto verso avec contenu libre (imprimé ou manuscrit).

Modalités : calculatrices, ordinateurs, objets communicants interdits.

Logique propositionnelle

C1 (2 points)

Soit la formule suivante :

$$\Sigma = \\ \neg(a \wedge b) \wedge \neg(a \wedge c) \wedge \neg(a \wedge d) \\ \wedge \neg(b \wedge c) \wedge \neg(b \wedge d) \\ \wedge \neg(c \wedge d)$$

1. Donnez trois modèles de Σ.
2. Σ est-elle une tautologie ? Justifiez très brièvement votre réponse.

C2 (2 points)

Soient les deux formules suivantes :

- $\Sigma_1 = \neg(a \vee b \vee c)$
- $\Sigma_2 = \neg a \vee \neg b \vee \neg c$

Certaines de ces propositions sont-elles correctes ? Lesquelles ? Justifiez votre réponse.

1. $\Sigma_1 \models \Sigma_2$
2. $\Sigma_2 \models \Sigma_1$
3. $\Sigma_2 \equiv \Sigma_1$

C3 (2 points)

Donnez une formule comportant trois variables nommées a, b, et e, qui modélise la propriété suivante :

> e est vrai si et seulement si a et b ont la même valeur de vérité (c'est à dire ont toutes les deux la valeur vrai ou toute les deux la valeur faux).

Logique du premier ordre

D1 (1 point)

Donnez les représentations sagittales de trois contre-modèles de la formule suivante avec le domaine d'interprétation $\{1, 2\}$.

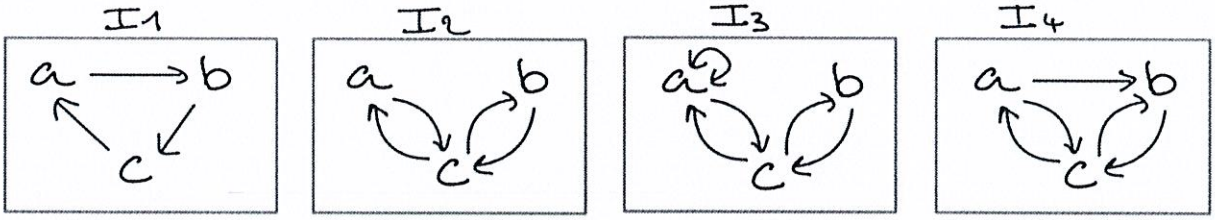
$$\Sigma_1 = \forall X[\exists Y\, q(X, Y) \vee \exists Y\, q(Y, X)]$$

D2 (1 point)

Soit la formule suivante :

$$\Sigma_2 = \forall X[\exists Y\, q(X,Y) \wedge \exists Y\, q(Y,X) \wedge \neg q(X,X)]$$

Parmi les interprétations suivantes, lesquelles sont des modèles de Σ_2 ?



D3 (2 points)

Soient les deux formules suivantes :

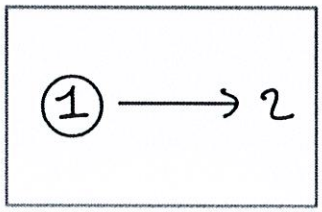
- $\Omega_1 = \forall X[p(X) \rightarrow \exists Y\, q(X,Y)]$
- $\Omega_2 = \forall X[\exists Y\, q(X,Y) \rightarrow p(X)]$

Donnez une interprétation I_1 sur le domaine $\{1,2\}$ qui montre que Ω_2 n'est pas conséquence logique de Ω_1, et une interprétation I_2 sur le domaine $\{1,2\}$ qui montre que Ω_1 n'est pas conséquence logique de Ω_2.

Ces deux interprétations doivent être représentées sous forme sagittale :

- $q(X,Y) = \text{vrai}$ est représenté par une flèche allant de X à Y, et $q(X,Y) = \text{faux}$ par l'absence de flèche reliant X à Y.
- $p(X) = \text{vrai}$ est représenté par un cercle autour de X, et $p(X) = \text{faux}$ est représenté par l'absence de cercle autour de X.

Par exemple, l'interprétation suivante indique que $p(1)$ est vrai, $p(2)$ est faux, $q(1,2)$ est vrai, $q(1,1)$, $q(2,2)$ et $q(2,1)$ sont faux. (Elle satisfait les deux formules.)



D4 (2 points)

Dans un univers de jeux de rôle, le prédicat $\text{mentor}/2$ représente le fait qu'une personne est mentor d'une autre. Par exemple, $\text{mentor}(\text{Anne}, \text{Bob})$ indique que Anne est un mentor de Bob.

Vous devez modéliser l'énoncé suivant par une formule de la logique du premier ordre :

> Toute personne n'ayant pas de mentor est mentor d'au moins une personne.

PROLOG

E1 (1 point)

Définissez le prédicat `max/3` tel que si `A` et `B` sont deux entiers connus, alors le but `max(A,B,R)` fait remonter dans `R` la plus grande des deux valeurs `A` et `B`. Par exemple le but...

```
max(5,7,R).
```

...doit avoir pour résultat `R = 7`.

E2 (1 point)

En supposant que le prédicat `max` de la question précédente soit correctement définit, utilisez ce prédicat pour définir le prédicat `maxi/4` tel que si `A`, `B` et `C` sont des entiers connus, alors le but `maxi(A,B,C,R)` fait remonter dans `R` la plus grande des trois valeurs `A`, `B` et `C`. Par exemple le but...

```
maxi(5,7,6,R).
```

... doit avoir pour résultat `R = 7`.

E3 (1 point)

Rappel

Le prédicat `ajouteFin` a été défini de la manière suivante dans les ressources d'apprentissage :

```
ajouteFin([],D,[D]).
ajouteFin([T|R],D,[T|R1]) :- ajouteFin(R,D,R1).
```

La notation `ajouteFin(L,D,R)` établit une relation entre une liste `L`, un élément `D` et une liste `R`. Cette relation exprime le fait que `R` est la liste résultant de l'ajout de `D` à la fin de `L`. Comme c'est une relation, et non une fonction, les trois slots peuvent être utilisés comme entrée ou sortie, dès lors que l'interpréteur Prolog a suffisamment d'information pour inférer les valeurs possibles des variables non instanciées utilisée en sorties.

Par exemple, le but...

```
ajouteFin(L,D,[1,2,3]).
```

... a pour résultat :

```
D = 3, L = [1,2]
```

Vous devez définir le prédicat `sep/4` tel que si `L` est une liste ayant au moins deux éléments (utilisée comme entrée) et les variables `P`, `M` et `D` ne sont pas instanciées (utilisées comme sorties), alors le but...

```
sep(L,P,M,D).
```

...doit faire remonter dans `P` le premier élément de `L`, dans `D` son dernier élément, et dans `M` la liste de tous les éléments restants, c'est à dire la liste `L` privée de son premier et de son dernier élément.

Par exemple, le but `sep([1,2,3,4],P,M,D)` doit produire le résultat `P=1`, `M=[2,3]`, `D=4`.

Pour définir ce prédicat `sep/4`, vous **devez** faire appel au prédicat `ajouteFin`, et ne **pas** utiliser de récursivité. Une seule clause est suffisante.

E4 (1 point)

On suppose que le prédicat `sep/4` de l'exercice précédant est parfaitement défini. En utilisant ce prédicat, définissez le prédicat `reverse/2` tel que si `L` est une liste et `R` une variable non instanciées, alors le but...

```
reverse(L,R).
```

...fait remonter dans `R` la liste comportant les mêmes éléments que `L`, mais dans l'ordre inverse. Par exemple, le but `reverse([1,2,3],R)` doit avoir pour résultat `R=[3,2,1]`.

Le seul prédicat autorisé, en plus de l'appel récursif de `reverse/2`, est le prédicat `sep/4`.

E5 (2 points)

Définissez le prédicat `insert/3` tel que `insert(E,L,R)` exprime la relations suivante :

> La liste `R` résulte de l'insertion de l'élément `E` quelque part dans la liste `L`.

Par exemple, le but `insert(9,[1,2,3],R)` doit avoir pour résultat :

```
R = [9, 1, 2, 3]
R = [1, 9, 2, 3]
R = [1, 2, 9, 3]
R = [1, 2, 3, 9]
```

Autre exemple, le but `insert(E,L,[1,2,3])` doit avoir pour résultat :

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

E6 (2 points)

Un multi-ensemble est une structure de données qui contient des éléments et dont chaque élément à un certain nombre d'occurrences. Par exemple, le multi-ensemble $\{1,1,3,3,3,4\}$, qui pourrait aussi être noté $\{(1,2),(3,3),(4,1)\}$, contient 2 occurrences de l'élément 1, 3 occurrences de l'élément 3 et 1 occurrence de l'élément 4.

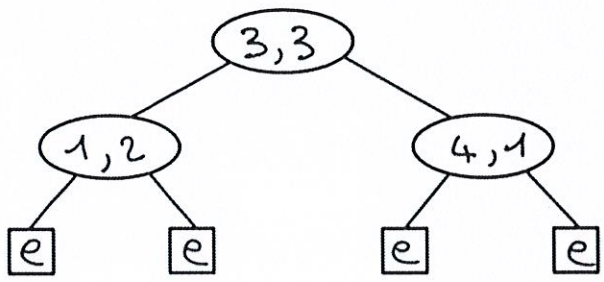
Dans cet exercice, on représente un multi-ensemble d'entiers par un arbre binaire ordonné avec la convention suivante :

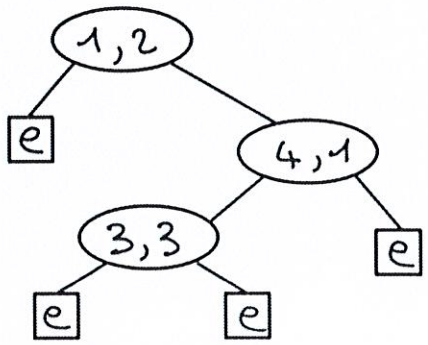
- Un multi-ensemble vide est représente par le terme fonctionnel `e`.
- Un multi-ensemble non vide est représenté par le terme fonctionnel `mu(V,N,G,D)` où :
  - V est élément.
  - N est le nombre d'occurrences de l'élément V.
  - G est un multi-ensemble dont les éléments sont inférieurs à V.
  - D est un multi-ensemble dont les éléments sont supérieur à V.

Par exemple, le multi-ensemble $\{(1,2),(3,3),(4,1)\}$ pourrait être représenté par les termes fonctionnels suivants, parmi d'autres possibilités :

```
mu(3,3,mu(1,2,e,e),mu(4,1,e,e))
mu(1,2,e,mu(4,1,mu(3,3,e,e),e))
```

Voici les représentations graphiques des arbres correspondants :





Chaque élément apparait dans un seul noeud de l'arbre.

Définissez le prédicat `occ/3` tel que si `M` est un terme fonctionnel représentant un multi-ensemble et `V` un entier, alors le but...

```
occ(M,V,N).
```

...fait remonter dans la variable `N` le nombre d'occurrences de `V` dans `M`. Si `V` n'est pas élément de `M`, la valeur remontée doit être 0.
