Université Polytechnique HAUTS-DE-FRANCE

Licence 3 Informatique

INSTITUT DES SCIENCES ET TECHNIQUES DE VALENCIENNES

Année 2019-2020 / session 1
Logique et Programmation logique

Responsable : S. PIECHOWIAK

Matériel : Tous documents interdits et calculatrices, ordinateurs et téléphones interdits

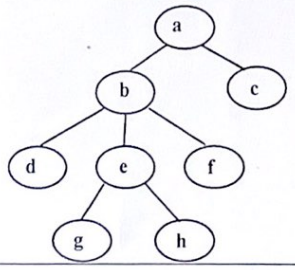
1. Exercice de logique (7 pts)

1) Rappeler ce que sont une forme normale conjonctive (FNC) et une forme normale disjonctive (FND) d'une formule logique propositionnelle.
2) Parmi les formules suivantes, quelles sont celles qui sont des FNC ? des FND ?
   1. $\varphi_1 = \neg(a \vee b \vee c) \wedge \neg(a \vee c) \wedge (a \vee c)$
   2. $\varphi_2 = \neg(a \wedge b \vee c) \vee \neg(a \vee c) \wedge (a \wedge c)$
   3. $\varphi_3 = \neg(a \vee \neg b \vee c) \wedge \neg(a \wedge c) \wedge (\neg a \vee c)$
   4. $\varphi_4 = (\neg a \vee b \vee c) \wedge (a \vee \neg c) \wedge (a \vee c)$
   5. $\varphi_5 = (\neg a \vee b \vee \neg c)$
   6. $\varphi_6 = (\neg a \wedge (b \vee \neg c))$
3) Mettre sous FNC la formule :
   $\varphi = \big(a \rightarrow \neg\big(b \rightarrow (c \vee \neg d)\big)\big) \rightarrow \neg\big(b \rightarrow (\neg a \vee c)\big)$
4) Simplifier la formule :
   $\gamma = \Big[\big(\neg(a \wedge b \wedge c) \vee \neg\big(b \rightarrow (c \vee \neg a)\big)\big) \rightarrow \neg\big((\neg b \wedge \neg c) \rightarrow (\neg a \vee c)\big)\Big] \wedge \big[d \rightarrow (e \rightarrow d)\big]$
5) Est-elle satisfiable ? Est-une tautologie ? Est-une antilogie ? Donner un modèle de γ

2. Les arbres en PROLOG (4 pts)

On peut représenter des arbres en Prolog en utilisant un fait pour chaque nœud de l'arbre. Chaque fait est composé du nœud lui-même (c'est à dire la valeur associée au nœud), et de la liste de ses descendants directs.

exemple :



Arbre

```
noeud(a, [b, c]).
noeud(b, [d, e, f]).
noeud(c, []).
noeud(d, []).
noeud(e, [g, h]).
noeud(f, []).
noeud(g, []).
noeud(h, []).
```

Représentation de l'arbre en Prolog

1) Ecrire les clauses (programme PROLOG) définissant le prédicat `fils(A, B)` qui réussit si **B** est le fils direct de **A**.
2) Ecrire les clauses définissant le prédicat `descendant(A, B)` qui réussit si **B** est un descendant de **A**. Avec l'exemple précédent, on aurait :

```
? descendant(a,e).
yes
? descendant (f,d).
```

```
no
```

3) Ecrire les clauses (programme PROLOG) définissant le prédicat **ancetre_commun(X, A, B)** qui réussit si **X** est un ancêtre commun de **A** et **B**. Avec l'exemple précédent, on aurait :

```
? ancetre_commun(b, e, f).
yes
? ancetre_commun(a, e, f).
yes
? ancetre_commun(f, g, h).
no
```

4) Ecrire les clauses (programme PROLOG) définissant le prédicat **hauteur(A, H)** qui unifie **H** avec la hauteur de l'arbre enraciné en **A**. La hauteur d'un arbre est la longueur du plus long chemin qui relie la racine avec ses feuilles (une feuille est un nœud sans descendant). Un arbre constitué d'un seul nœud a une hauteur égale à 1. L'arbre vide a une hauteur nulle.

Avec l'exemple précédent, on aurait :

```
?hauteur(a,H).
H=4
```

3. Exercice (4 pts)

Pour mémoire, en cours et en TD nous avons vu le prédicat suivant :

```
selectionner(X,[X|L],L).
selectionner(X,[Y|L1],[Y|L2]) :- selectionner(X,L1,L2).
```

1) Dessiner très soigneusement la trace de résolution lorsqu'on pose la requête :
   `? selectionner (3,[1,3,5,3],L).`

On complète le programme avec :

```
mystere([],0).
mystere([_|Q],N) :- mystere(Q,NQ), N is 1 + NQ.

question(Element, Maliste, Reponse) :-
                findall(L,selectionner(Element, Maliste, L), R),
                mystere(R,Reponse).
```

2) Que donne la requête :

```
? question([1,3,5,3],3,R).
```

Expliquer.

4. Exercice (5 pts)

On souhaite définir les prédicats qui vont permettre de trier dans l'ordre croissant une liste de nombres par la méthode dite de fusion dont on rappelle le principe. On commence par scinder la liste L en 2 sous listes SL1 et SL2. Les 2 sous listes sont triées par fusion la méthode du tri par fusion puis elles sont fusionnées. La liste obtenue est la liste initiale triée.

Définir le prédicat **fusionner** qui fusionne 2 listes triées dans le même ordre en une seule (triée dans le même ordre que les 2 précédentes).

Définir le prédicat **scinder** le prédicat qui scinde une liste en 2 sous listes. Comment scindez-vous votre liste ?

Définir le prédicat **trierFusion** le prédicat qui trie une liste de nombre par ordre croissant avec la méthode du tri par fusion.

Université Polytechnique HAUTS-DE-FRANCE

Licence 3 Informatique

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Année 2020-2021 / session 1

Logique et Programmation logique

Responsable : S. PIECHOWIAK

Matériel : Tous documents interdits, calculatrices, ordinateurs et téléphones interdits

1. Exercice de logique (1+2+3+3 pts)

1) Rappeler très précisément ce qu'est une forme normale conjonctive (FNC) et une forme normale disjonctive (FND) d'une formule logique propositionnelle.
2) Parmi les formules suivantes, quelles sont celles qui sont des FNC ? des FND ?
   1. $\varphi_1 = \neg(a \vee b \vee c) \wedge \neg(a \vee c) \wedge (a \vee c)$
   2. $\varphi_2 = \neg(a \wedge b \vee c) \vee \neg(a \vee c) \wedge (a \wedge c)$
   3. $\varphi_3 = \neg(a \vee \neg b \vee c) \wedge \neg(a \wedge c) \wedge (\neg a \vee c)$
   4. $\varphi_4 = (\neg a \vee b \vee c) \rightarrow (a \vee c)$
   5. $\varphi_5 = (\neg a \vee b \vee \neg c)$
   6. $\varphi_6 = (\neg a \wedge (b \vee \neg c))$
   7. $\varphi_7 = (\neg a \wedge a)$
   8. $\varphi_8 = (\neg a \vee a)$
3) Soit la formule $\delta = \big(a \rightarrow (b \rightarrow (c \vee \neg d))\big) \rightarrow \neg(b \rightarrow (\neg a \vee c))$
   a) Mettre δ sous FNC
   b) Mettre δ sous FND
4) La formule :
   $\varphi = \Big[\big(\neg(a \wedge b \wedge \neg c) \vee \neg(\neg b \rightarrow (c \vee \neg a))\big) \rightarrow \big((\neg b \wedge \neg c) \rightarrow (\neg a \vee c)\big)\Big] \wedge [d \rightarrow (\neg e \rightarrow d)]$
   est-elle satisfiable ? Est-ce une tautologie ? Est-ce une antilogie ? Donner un modèle de γ

2. Exercice (2+1+1 pts)

Pour mémoire, en cours et en TD nous avons vu le prédicat suivant :

```
selectionner(X,[X|L],L).
selectionner(X,[Y|L1],[Y|L2]) :- selectionner(X,L1,L2).
```

1) Dessiner très soigneusement la trace de résolution lorsqu'on pose la requête :

```
selectionner (3,[1,3,4,3],L).
```

2) Donner un prédicat **permuter** qui calcule la liste de toutes les permutations d'une liste donnée.
3) Donner un prédicat **nbPermutations** qui calcule le nombre de permutations d'une liste donnée, après les avoir calculer.

3. Tri fusion (1+2+1 pts)

On souhaite définir les prédicats qui vont permettre de trier dans l'ordre croissant une liste de nombres par la méthode dite de fusion vue en cours et dont on rappelle le principe. On commence par scinder la liste L en 2 sous listes SL1 et SL2. Les 2 sous listes sont triées par la méthode du tri par fusion puis elles sont fusionnées. La liste obtenue est la liste initiale triée.

1) Définir le prédicat **fusionner** qui fusionne 2 listes triées dans le même ordre en une seule (triée dans le même ordre que les 2 précédentes).
2) Définir le prédicat **scinder** le prédicat qui scinde une liste en 2 sous listes, c'est-à-dire en 2 sous listes dont les tailles diffèrent d'au plus 1.
3) Définir le prédicat **trierFusion** le prédicat qui trie une liste de nombre par ordre croissant avec la méthode du tri par fusion.

4. Evaluation d'un programmes Prolog (1+1+1+1 pts)

On définit le programme Prolog suivant :

```
R1) mystere(X) :- mystere2(X,Y), mystere3(Y).
R2) mystere (X) :- idiot3(X).
R3) mystere2 (a,b).
R4) mystere2 (a,c).
R5) mystere3 (b).
R6) mystere3 (a).
```

1) Que donne l'évaluation de la requête : `findall(X,mystere(X),L).`
2) Que donne cette requête si on remplace

   a) R1) par `mystere(X) :- mystere2(X,Y), !, mystere3(Y).`
   b) R3) par `mystere2(a,b) :- !.`
   c) R5) par `mystere3(b) :- !.`