Examen de Logique et de Prolog — ESSI2

(Documents autorisés: *tous*)

Hélène Collavizza et Michel Rueher

Mai 2001

1 Logique

1. **Formalisation (2.5 points)**

   Soit le langage du premier ordre composé de l'ensemble de variables $V = \{x, y\}$, les symboles fonctionnels 0-aires l et s, les symboles de prédicats binaires a et f.

   Dans ce langage, exprimer les énoncés suivants :

   (a) Tout le monde aime Laurent

   (b) Quelqu'un aime Laurent et sa femme

   (c) Tous ceux qui aiment Laurent n'aiment pas Stéphane

   (d) Laurent ne peut aimer la femme d'un autre

2. **Validité (3.5 points)**

   (a) Soit la formule $F = (\forall\ x\ \ p(x,x)) \wedge (\forall\ x \forall\ y \forall\ z\ \ (p(x,y) \wedge p(y,z) \Rightarrow p(x,z))) \wedge (\exists y\ \forall x\ p(x,y))$.
   Donner une interprétation I_1 qui est un modèle de F et une interprétaion I_2 qui n'est pas un modèle de F.

   (b) Montrer par résolution que la formule $F = \forall\ x\ (p(x) \vee \neg p(x))$ est universellement valide.

3. **Preuve dans tous les modèles (5 points)**

   Soit les axiomes :

   A1 : $\forall\ x\ ((A(x) \wedge\ \neg\ B(x)) \Rightarrow\ \exists\ y\ (C(x,y) \wedge\ D(y)))$

   A2 : $\exists\ x\ (E(x) \wedge\ A(x) \wedge\ \forall\ y\ (C(x,y) \Rightarrow E(y)))$

   A3 : $\forall\ x\ (\neg\ E(x) \vee\ \neg\ B(x))$

   Montrer que $\Phi = \exists\ x\ (E(x) \wedge D(x))$ est valide dans tous les modèles de {A1,A2,A3}.

2 Prolog

(a) Manipulations élémentaires (2 points)

- Construire (et dessiner !) l'arbre de recherche pour la requête `retrait_elt0(X,[a,b,Z])` avec la définition du prédicat `retrait_elt0/2` ci-dessous:

```
retrait_elt0(J,[J|L],L).
retrait_elt0(J,[K|L],[K|M]):- retrait_elt0(J,L,M).
```

- Comparer les performances des prédicats `retrait_elt0/2` et `retrait_elt1/2`

```
retrait_elt1(E,Ldepart,Lresultat):-
      append(L1,[E|L2],Ldepart),
      append(L1,L2,Lresultat).
append([],L,L).
append([T|L1],L2,[T|L3]):- append(L1,L2,L3).
```

(b) Canmiss(2 points)

Dans le programme ci-dessous (extrait de `canmiss/1` expliquer le rôle du prédicat `enum/3`. Pourquoi est-il incorrect de remplacer `enum/3` par `enum1/3`.

```
genere_etat_valide([Mc,Cc,Rc],[Mn,Cn,Rn]):-
      select(M,C,Mc,Cc),
      Mo is Mc-M,   Co is Cc -C,  ok(Mo,Co),
      Mn is 3 - Mo, Cn is 3 - Co, ok(Mn,Cn),
      succ(Rc,Rn).
select(M,C,Mc,Cc):-
      enum(M,0,Mc),     enum(C,0,Cc),
      MC is M+C,MC >= 1,MC =< 2.
enum(X,_,X).
enum(X,Y,Z):- Z > Y,Z1 is Z-1, enum(X,Y,Z1).
enum1(X,Y,Z):- X =< Z , Y =< Z.
```

(c) Petit démonstrateur (4 points)

- En vous inspirant du méta-interprète Prolog vu en cours, écrire un prédicat `solve/1` qui a pour argument un ensemble de buts à démontrer. la démonstration sera basée sur la résolution et utilisera un ensemble de clauses de Horn. On supposera que les clauses et les buts à résoudre sont représentés par des listes.
  Par exemple, les clauses :
  $p(X) \Leftarrow q1(X,1) \wedge$
  $p(X) \Leftarrow q2(X,2) \wedge$
  $p(X) \Leftarrow (q3(X,3) \wedge v3(X,3)) \wedge$
  $q1(a1,1) \wedge$
  ... pourraient être rreprésentées par un fait `list_of_clauses/1` :

```
list_of_clauses([[p(X),[q1(X,1)]],[p(X),[q2(X,2)]],[p(X),[q3(X,3),v3(X,3)]],
[q1(a1,1),vide],...]).
```

- Est ce que la stratégie de recherche standard de Prolog garantit la correction et la complétude des résultats de ce démonstrateur ?
  Pourquoi ?

4. DGC (3 points)
   Ecrire la DGC qui analyse et évalue des expressions qui respectent la grammaire $a^n b^n$ où a est un nombre entier et n un chiffre. Pour calculer a^n on multipliera n fois a par lui-même. Exemple:

```
expr(Z,"2*3",[ ]) -> Z=6
expr(Z,"2*2*2*3*3*3",[ ]) -> Z=216
expr(Z,"2*2*2*3*3*3*3",[ ]) -> no
expr(Z,"2*2*2*3*3*3*3",X) ->
     X = [42,50,42,51,42,51,42,51,42,51],
     Z = 4 ? ;
     X = [42,51],
     Z = 216 ? ;
```