EISTI - DÉPARTEMENT INFORMATIQUE

EXAMEN DE LOGIQUE COMPUTATIONNELLE & PROLOG

29 juin 2011 – **DURÉE 2h00**

La consultation des documents et l'échange des documents et des calculatrices est interdit.
L'utilisation des 3 feuilles manuscrites recto-verso, format A4 est autorisée

- **Ne pas détacher les feuilles.**
- **Utiliser l'espace blanc pour vos réponses et le verso pour brouillon.**
- **Pensez à indiquer votre nom sur chaque feuille**

NOM : .....

NOTE

DÉTAIL		**NOTE LOG. CO.**	**NOTE PROLOG**
Exercice 1.	**1** **2**		
Exercice 2.	**1** **2** **3**		
Exercice 3.			
Exercice 4.	**1** **2** **3** **4**		
Exercice 5.			
Exercice 6.			
Exercice 7.			
Exercice 8.			
Exercice 9.			
Exercice 10.			
Exercice 11.			

Le corrigé se trouve sur `http://sifoci.eisti.fr`***, rubrique Logique computationnelle – Prolog.***

Exercice 1 – Calcul propositionnel

1. Compléter la table de vérité suivante :

p	p	pc	$(p \to (q \to r)) \to ((p \to q) \to (p \to r))$	$(p \wedge (q \vee \neg r)) \wedge ((\neg q \to p) \vee r)$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

2. Pour chacune de deux formules précédentes, préciser leur type (tautologie, formule satisfiable ou insatisfiable).

SOL.- La première est une tautologie et la seconde est satisfiable.

Exercice 2 – Calcul des prédicats

Soient les formules :

$$A = \forall X(p(X) \to p(a)), \quad B = (\forall X p(X) \to p(a)), \quad C = (p(a) \to \forall X p(X)),$$
$$E = \exists X(p(X) \to p(a)), \quad F = (\exists X p(X) \to p(a)), \quad G = (p(a) \to \exists X p(X)),$$

et l'interprétation I dont le domaine D est l'ensemble des salles de cours de l'EISTI et telle que $I(a)$ est la salle 101 et $I(p)$ est l'application de D dans $\{0, 1\}$ telle que pour tout élément d de D, $I(p)(d) = 1$ ssi la salle d est disponible.

1. Calculer $\phi_I(A), \phi_I(B), \cdots, \phi_I(G)$, sachant que la salle 101 est disponible et que toutes les autres sont occupées.
2. Calculer $\phi_I(A), \phi_I(B), \cdots, \phi_I(G)$, sachant que la salle 101 est occupée et que toutes les autres sont disponibles.
3. Lesquelles des formules A, B, C, D, E, F et G sont elles valides ?

Exercice 3 – Forme conjonctive normale
Donner la forme conjonctive normale équivalente de la formule

$$\exists X\left(\left(q\left(X\right)\vee\forall Y p\left(X,Y\right)\right)\rightarrow\exists X p\left(X,X\right)\right)$$

SOL.- On a

$$\begin{aligned}
& \exists X\left(\left(q\left(X\right)\vee\forall Y p\left(X,Y\right)\right)\rightarrow\exists X p\left(X,X\right)\right)\\
= \;& \exists X\left(\left(q\left(X\right)\vee\forall Y p\left(X,Y\right)\right)\rightarrow\exists X_1 p\left(X_1,X_1\right)\right)\\
= \;& \forall X\left(\neg\left(q\left(X\right)\vee\forall Y p\left(X,Y\right)\right)\vee\exists X_1 p\left(X_1,X_1\right)\right)\\
= \;& \forall X\left(\left(\neg q\left(X\right)\wedge\neg\forall Y p\left(X,Y\right)\right)\vee\exists X_1 p\left(X_1,X_1\right)\right)\\
= \;& \forall X\left(\left(\neg q\left(X\right)\wedge\exists Y\neg p\left(X,Y\right)\right)\vee\exists X_1 p\left(X_1,X_1\right)\right)\\
= \;& \forall X\left(\left(\neg q\left(X\right)\wedge\neg p\left(X,s_Y\left(X\right)\right)\right)\vee p\left(s_{X_1}\left(X\right),s_{X_1}\left(X\right)\right)\right)\\
= \;& \left(\neg q\left(X\right)\vee p\left(s_{X_1}\left(X\right),s_{X_1}\left(X\right)\right)\right)\wedge\left(\neg p\left(X,s_Y\left(X\right)\right)\vee p\left(s_{X_1}\left(X\right),s_{X_1}\left(X\right)\right)\right)
\end{aligned}$$

Exercice 4 – Modèle de Herbrand
Soit le programme

$$E=\left\{\begin{array}{l} p\left(X\right)\leftarrow q\left(X\right)\\ q\left(X\right)\leftarrow r\left(X\right)\\ s\left(toto,X\right)\leftarrow p\left(X\right)\\ r\left(koko\right)\leftarrow\end{array}\right.$$

1. Donner, en le justifiant l'univers de Herbrand.
2. Donner, en la justifiant la base de Herbrand.
3. Calculer le modèle minimal de Herbrand.
4. Quelles sont les informations que nous fournit le modèle minimal de Herbrand ? Justifier votre réponse.

SOL.- Le programme est défini, donc on peut appliquer la méthode de Herbrand.

1. $\mathcal{U}_E=\{toto,koko\}$
2. $\mathcal{B}_E=\{p\left(toto\right),p\left(koko\right),q\left(toto\right),q\left(koko\right),r\left(toto\right),r\left(koko\right),s\left(toto,toto\right),\ldots,s\left(koko,koko\right)\}$
3. $\mathcal{T}\uparrow 0=\emptyset$
   $\mathcal{T}\uparrow 1=\{r\left(koko\right)\}$
   $\mathcal{T}\uparrow 2=\{r\left(koko\right),q\left(koko\right)\}$
   $\mathcal{T}\uparrow 3=\{r\left(koko\right),q\left(koko\right),p\left(koko\right)\}$
   $\mathcal{T}\uparrow 3=\{r\left(koko\right),q\left(koko\right),p\left(koko\right),s\left(toto,koko\right)\}$
   $\mathcal{T}\uparrow 4=\mathcal{T}\uparrow 3$
   Donc modèle minimal de Herbrand $\widetilde{I_E}=\{r\left(koko\right),q\left(koko\right),p\left(koko\right),s\left(toto,koko\right)\}$.
4. Le modèle minimal de Herbrand contient toutes les réponses du programme.

Exercice 5 – Listes numériques
Écrire en Prolog un prédicat `ordonnee(L)` qui est vrai si `L` est une liste croissante d'entiers.
SOL.-

```
ordonnee([]).
ordonnee([X]).
ordonnee([X,Y|R]) :- X =< Y,
                     ordonnee([Y | R]).
```

Exercice 6 – Accumulateur et listes numériques

Écrire 2 versions (l'une sans accumulateur et l'autre avec) du prédicat `prodlist(L,N)` qui est vrai si `N` est le produit des éléments de la liste d'entiers `L`.

SOL.-

```
prodList([],1).
prodList([T|R], P) :- prodList(R,P1),
                      P is T*P1.

prodListA(L,P) :- prodList(L,1,P).
prodList([],Accu,Accu).
prodList([T|R], Accu,P) :- Accu1 is Accu*T,
                           prodList(R,Accu1,P).
```

Voir aussi page 130 du poly.

Exercice 7 – Listes

Écrire un prédicat `permutation(L1,L2)` qui est vrai si la liste `L2` est l'une des permutations de la liste `L1`.

SOL.-

```
suppres(X, [X|T], T).
suppres(X, [H|T], [H|S]):- suppres(X, T, S).
permut([], []).
permut([H|T], R):- permut(T, X), suppres(H, R, X).
permutation(L1,L2) :- L2 \==L1, permut(L1,L), L == L2.
```

Exercice 8 – Représentation des opérations numériques par Prolog

1. On rappelle que le prédicat `display(Terme)` donne la représentation interne de `Terme`.
   Donner et justifier la réponse de Prolog à la question
   `?- display(1+2*3).`.
2. Donner et justifier la réponse de Prolog à la question
   `?- 2+3=3+2.`.

SOL.-

```
1.-   +(1, *(2, 3))
2.-  false.
```

car les deux représentations `+(2, 3)` et `+(3, 2)` ne sont pas identqiues.

Exercice 9 – Programmation Prolog

Donner et justifier la réponse de Prolog à la question
`not(member(X,[1,2,3])),member(X,[4,5,6]).` .

Exercice 10 – Unification

Donner et justifier la réponse de Prolog à la question `?-p.` lorsque le progamme est constitué de l'unique règle `p :-p.`.

SOL.- Il boucle car il ne fait pas la vérification des occurrences avant d'unifier `p` à `p`.

Exercice 11 – Techniques de programmation

La conjecture de Goldbach stipule que tout entier pair supérieur ou égal à 4 peut s'écrire comme somme de 2 nombres premiers. Écrire un prédicat `goldbach(Borne)` qui affiche une telle somme pour tous les nombres pairs entre 4 et `Borne`.

```
premier(2).
premier(3).
premier(P) :- integer(P), P > 3, P mod 2 =\= 0, not(facteur(P,3)).


premierSuivant(P,P1) :- P1 is P + 2, premier(P1), !.
premierSuivant(P,P1) :- P2 is P + 2, premierSuivant(P2,P1).

facteur(N,L) :- N mod L =:= 0.
facteur(N,L) :- L * L < N, L2 is L + 2, facteur(N,L2).

goldb(4,[2,2]) :- !.
goldb(N,L) :- N mod 2 =:= 0, N > 4, goldb(N,L,3).

goldb(N,[P,Q],P) :- Q is N - P, premier(Q), !.
goldb(N,L,P) :- P < N, premierSuivant(P,P1), goldb(N,L,P1).

goldbach(Borne) :- Borne >= 4, goldbach(4,Borne).
goldbach(N, Borne) :- N =< Borne,
                        goldb(N,L),
                        write(L),nl,
                        N1 is N + 2,
                        goldbach(N1, Borne).
```

Les prédicats `premier`, `premierSuivant` et `facteur` seront considérés comme connus.
Autre solution :

```
premier(N) :- findall(Dir, (between(1,N,Dir), N mod Dir =:= 0),[1,N]).

gb(Borne) :- findall((N1,N2,N),
            (between(1,Borne,N), N mod 2 =:= 0,
             between(1,N,N1), premier(N1), N2 is N-N1, premier(N2)), L),
             affiche(L).

affiche([]).
affiche([T|R]) :- write(T),nl, affiche(R).
```