TP3 - Liste

Paul Chaignon - ClÃ©ment Gautrais

September 30, 2013

1 Questions

Listing 1: tp_listes.pro

```
1  /**
2   * TP Listes Prolog
3   * @author Paul CHAIGNON
4   * @author Clement GAUTRAIS
5   * @version Annee scolaire 2013/2014
6   */
7
8  /**
9   * membre(?A, +X)
10  */
11 membre(A, [A|R]).
12 membre(A, [X|R]):-
13   membre(A, R).
14
15 /**
16  * compte(+A, +X, ?N)
17  */
18 compte(A, [], 0).
19 compte(A, [A|R], N):-
20   compte(A, R, M),
21   N is M + 1.
22 compte(A, [X|R], N):-
23   A \== X,
24   compte(A, R, N).
25
26 /**
27  * renverser(+X, ?Y)
28  */
29 renverser(X, Y):-
30   renv(X, [], Y).
31 renv([X|R], A, Y):-
32   renv(R, [X|A], Y).
33 renv([], Y, Y).
34
35 /**
36  * palind(+X)
37  */
38 palind(X):-
39   renverser(X, X).
40
```

```
41 /**
42  * nieme1(+N, +X, -A)
43  */
44 nieme1(0, [X|R], X).
45 nieme1(N, [Y|R], X):-
46   N \== 0,
47   M is N - 1,
48   nieme1(M, R, X).
49
50 /**
51  * nieme2(-N, +X, +A)
52  */
53 nieme2(0, [X|R], X).
54 nieme2(N, [Y|R], X):-
55   X \== Y,
56   nieme2(M, R, X),
57   N is M + 1.
58 /**
59  * Pas possible d'avoir un algo commun car ils ne font pas la mÃªme
      chose.
60  * De plus, il y aurait des problÃ¨mes si une liste contient 2 valeurs
      identique.
61  */
62
63 /**
64  * hors_de(+A, +X)
65  */
66 hors_de(A, X):-
67   compte(A, X, 0).
68
69 /**
70  * tous_diff(+X)
71  */
72 tous_diff([]).
73 tous_diff([X|R]):-
74   hors_de(X, R),
75   tous_diff(R).
76
77 /**
78  * conc3(+X, +Y, +Z, ?T)
79  */
80 conc3([], [], Z, Z).
81 conc3([], [P|R], Z, [P|T]):-
82   conc3([], R, Z, T).
83 conc3([P|R], Y, Z, [P|T]):-
84   conc3(R, Y, Z, T).
85 /**
86  * Oui c'est possible et c'est le cas de notre algorithme.
87  */
88
89 /**
90  * debute_par(+X, ?Y)
91  */
92 debute_par(X, []).
93 debute_par([X|R], [X|Q]):-
94   debute_par(R, Q).
95
96 /**
```

```
 * sous_liste(+X, ?Y)
 */
sous_liste(X, Y):-
  debute_par(X, Y).
sous_liste([X|R], Y):-
  sous_liste(R, Y).

/**
 * elim(+X, -Y)
 */
elim(X, Y):-
  elimin(X, Y, []).
elimin([], Z, Z).
elimin([X|R], Y, Z):-
  compte(X, Z, 1),
  elimin(R, Y, Z).
elimin([X|R], Y, Z):-
  compte(X, Z, 0),
  elimin(R, Y, [X|Z]).

/**
 * inserer(+E, +L1, -L2)
 */
inserer(E, [], [E]).
inserer(E, [P|R], [E,P|R]):-
  E =< P.
inserer(E, [P|R], [P|Z]):-
  E > P,
  inserer(E, R, Z).

/**
 * tri(+X, -Y)
 */
tri(X, Y):-
  trier(X, Y, []).
trier([], Acc, Acc).
trier([X|R], Y, Acc):-
  inserer(X, Acc, NewAcc),
  trier(R, Y, NewAcc).

/**
 * inclus(X, Y)
 */
inclus([], Y).
inclus([X|R], Y):-
  membre(X, Y),
  inclus(R, Y).

/**
 * non_inclus(X, Y)
 */
non_inclus([X|R], Y):-
  hors_de(X, Y).
non_inclus([X|R], Y):-
  membre(X, Y),
  non_inclus(R, Y).

/**
```

```
155  * Le cut dans les 3 fonctions suivates servent Ã  gÃ©nÃ©rer des
         ensembles sans doublons.
156  */
157 /**
158  * union_ens(X, Y, Z)
159  */
160 union_ens(X, Y, Z):-
161   union_ensem(X, Y, Z, []).
162 union_ensem([], [], Z, Acc):-
163   inclus(Acc, Z),
164   inclus(Z, Acc),
165   !.
166 union_ensem([X|R], Y, Z, Acc):-
167   hors_de(X, Acc),
168   union_ensem(R, Y, Z, [X|Acc]).
169 union_ensem([X|R], Y, Z, Acc):-
170   membre(X, Acc),
171   union_ensem(R, Y, Z, Acc).
172 union_ensem([], [Y|R], Z, Acc):-
173   hors_de(Y, Acc),
174   union_ensem([], R, Z, [Y|Acc]).
175 union_ensem([], [Y|R], Z, Acc):-
176   membre(Y, Acc),
177   union_ensem([], R, Z, Acc).
178
179 /**
180  * inter_ens(X, Y, Z)
181  */
182 inter_ens(X, Y, Z):-
183   inter_ensem(X, Y, Z, []).
184 inter_ensem([], Y, Z, Acc):-
185   inclus(Acc, Z),
186   inclus(Z, Acc),
187   !.
188 inter_ensem([X|R], Y, Z, Acc):-
189   membre(X, Y),
190   inter_ensem(R, Y, Z, [X|Acc]).
191 inter_ensem([X|R], Y, Z, Acc):-
192   hors_de(X, Y),
193   inter_ensem(R, Y, Z, Acc).
194
195 /**
196  * diff_ens(X, Y, Z)
197  */
198 diff_ens(X, Y, Z):-
199   diff_ensem(X, Y, Z, []).
200 diff_ensem([], Y, Z, Acc):-
201   inclus(Acc, Z),
202   inclus(Z, Acc),
203   !.
204 diff_ensem([X|R], Y, Z, Acc):-
205   hors_de(X, Y),
206   diff_ensem(R, Y, Z, [X|Acc]).
207 diff_ensem([X|R], Y, Z, Acc):-
208   membre(X, Y),
209   diff_ensem(R, Y, Z, Acc).
```

2 Tests

Listing 2: tp_listes_tests.pro

```
1  /**
2   * TP Listes Prolog
3   * @author Paul CHAIGNON
4   * @author Clement GAUTRAIS
5   * @version Annee scolaire 2013/2014
6   */
7
8  /**
9   * membre(?A, +X)
10  */
11 membre(A, [A|R]).
12 membre(A, [X|R]):-
13   membre(A, R).
14
15 /**
16  * compte(+A, +X, ?N)
17  */
18 compte(A, [], 0).
19 compte(A, [A|R], N):-
20   compte(A, R, M),
21   N is M + 1.
22 compte(A, [X|R], N):-
23   A \== X,
24   compte(A, R, N).
25
26 /**
27  * renverser(+X, ?Y)
28  */
29 renverser(X, Y):-
30   renv(X, [], Y).
31 renv([X|R], A, Y):-
32   renv(R, [X|A], Y).
33 renv([], Y, Y).
34
35 /**
36  * palind(+X)
37  */
38 palind(X):-
39   renverser(X, X).
40
41 /**
42  * nieme1(+N, +X, -A)
43  */
44 nieme1(0, [X|R], X).
45 nieme1(N, [Y|R], X):-
46   N \== 0,
47   M is N - 1,
48   nieme1(M, R, X).
49
50 /**
51  * nieme2(-N, +X, +A)
52  */
53 nieme2(0, [X|R], X).
54 nieme2(N, [Y|R], X):-
```

```
55    X \== Y,
56    nieme2(M, R, X),
57    N is M + 1.
58  /**
59   * Pas possible d'avoir un algo commun car ils ne font pas la mÃªme
        chose.
60   * De plus, il y aurait des problÃ¨mes si une liste contient 2 valeurs
        identique.
61   */
62
63  /**
64   * hors_de(+A, +X)
65   */
66  hors_de(A, X):-
67    compte(A, X, 0).
68
69  /**
70   * tous_diff(+X)
71   */
72  tous_diff([]).
73  tous_diff([X|R]):-
74    hors_de(X, R),
75    tous_diff(R).
76
77  /**
78   * conc3(+X, +Y, +Z, ?T)
79   */
80  conc3([], [], Z, Z).
81  conc3([], [P|R], Z, [P|T]):-
82    conc3([], R, Z, T).
83  conc3([P|R], Y, Z, [P|T]):-
84    conc3(R, Y, Z, T).
85  /**
86   * Oui c'est possible et c'est le cas de notre algorithme.
87   */
88
89  /**
90   * debute_par(+X, ?Y)
91   */
92  debute_par(X, []).
93  debute_par([X|R], [X|Q]):-
94    debute_par(R, Q).
95
96  /**
97   * sous_liste(+X, ?Y)
98   */
99  sous_liste(X, Y):-
100   debute_par(X, Y).
101 sous_liste([X|R], Y):-
102   sous_liste(R, Y).
103
104 /**
105  * elim(+X, -Y)
106  */
107 elim(X, Y):-
108   elimin(X, Y, []).
109 elimin([], Z, Z).
110 elimin([X|R], Y, Z):-
```

```
111   compte(X, Z, 1),
112   elimin(R, Y, Z).
113 elimin([X|R], Y, Z):-
114   compte(X, Z, 0),
115   elimin(R, Y, [X|Z]).
116
117 /**
118  * inserer(+E, +L1, -L2)
119  */
120 inserer(E, [], [E]).
121 inserer(E, [P|R], [E,P|R]):-
122   E =< P.
123 inserer(E, [P|R], [P|Z]):-
124   E > P,
125   inserer(E, R, Z).
126
127 /**
128  * tri(+X, -Y)
129  */
130 tri(X, Y):-
131   trier(X, Y, []).
132 trier([], Acc, Acc).
133 trier([X|R], Y, Acc):-
134   inserer(X, Acc, NewAcc),
135   trier(R, Y, NewAcc).
136
137 /**
138  * inclus(X, Y)
139  */
140 inclus([], Y).
141 inclus([X|R], Y):-
142   membre(X, Y),
143   inclus(R, Y).
144
145 /**
146  * non_inclus(X, Y)
147  */
148 non_inclus([X|R], Y):-
149   hors_de(X, Y).
150 non_inclus([X|R], Y):-
151   membre(X, Y),
152   non_inclus(R, Y).
153
154 /**
155  * Le cut dans les 3 fonctions suivates servent Ã  gÃ©nÃ©rer des
      ensembles sans doublons.
156  */
157 /**
158  * union_ens(X, Y, Z)
159  */
160 union_ens(X, Y, Z):-
161   union_ensem(X, Y, Z, []).
162 union_ensem([], [], Z, Acc):-
163   inclus(Acc, Z),
164   inclus(Z, Acc),
165   !.
166 union_ensem([X|R], Y, Z, Acc):-
167   hors_de(X, Acc),
```

```
  union_ensem(R, Y, Z, [X|Acc]).
union_ensem([X|R], Y, Z, Acc):-
  membre(X, Acc),
  union_ensem(R, Y, Z, Acc).
union_ensem([], [Y|R], Z, Acc):-
  hors_de(Y, Acc),
  union_ensem([], R, Z, [Y|Acc]).
union_ensem([], [Y|R], Z, Acc):-
  membre(Y, Acc),
  union_ensem([], R, Z, Acc).

/**
 * inter_ens(X, Y, Z)
 */
inter_ens(X, Y, Z):-
  inter_ensem(X, Y, Z, []).
inter_ensem([], Y, Z, Acc):-
  inclus(Acc, Z),
  inclus(Z, Acc),
  !.
inter_ensem([X|R], Y, Z, Acc):-
  membre(X, Y),
  inter_ensem(R, Y, Z, [X|Acc]).
inter_ensem([X|R], Y, Z, Acc):-
  hors_de(X, Y),
  inter_ensem(R, Y, Z, Acc).

/**
 * diff_ens(X, Y, Z)
 */
diff_ens(X, Y, Z):-
  diff_ensem(X, Y, Z, []).
diff_ensem([], Y, Z, Acc):-
  inclus(Acc, Z),
  inclus(Z, Acc),
  !.
diff_ensem([X|R], Y, Z, Acc):-
  hors_de(X, Y),
  diff_ensem(R, Y, Z, [X|Acc]).
diff_ensem([X|R], Y, Z, Acc):-
  membre(X, Y),
  diff_ensem(R, Y, Z, Acc).
```