TP3 - Liste

Paul Chaignon - Clément Gautrais

October 3, 2013

1 Questions

Listing 1: tp_listes.pro

```
/**
 * membre(?A, +X)
 */
membre(A, [A|R]).
membre(A, [X|R]):-
  membre(A, R).

/**
 * compte(+A, +X, ?N)
 */
compte(A, [], 0).
compte(A, [A|R], N):-
  compte(A, R, M),
  N is M + 1.
compte(A, [X|R], N):-
  A \== X,
  compte(A, R, N).

/**
 * renverser(+X, ?Y)
 */
renverser(X, Y):-
  renv(X, [], Y).
renv([X|R], A, Y):-
  renv(R, [X|A], Y).
renv([], Y, Y).

/**
 * palind(+X)
 */
palind(X):-
  renverser(X, X).

/**
 * nieme1(+N, +X, -A)
 */
nieme1(0, [X|R], X).
nieme1(N, [Y|R], X):-
  N \== 0,
  M is N - 1,
  nieme1(M, R, X).

/**
 * nieme2(-N, +X, +A)
 */
nieme2(0, [X|R], X).
nieme2(N, [Y|R], X):-
  X \== Y,
  nieme2(M, R, X),
  N is M + 1.
/**
 * Pas possible d'avoir un algo commun car ils ne font pas la mÃªme chose.
 * De plus, il y aurait des problÃ¨mes si une liste contient 2 valeurs identique.
 */

/**
 * hors_de(+A, +X)
 */
hors_de(A, X):-
  compte(A, X, 0).

/**
 * tous_diff(+X)
 */
tous_diff([]).
tous_diff([X|R]):-
  hors_de(X, R),
  tous_diff(R).

/**
 * conc3(+X, +Y, +Z, ?T)
 */
conc3([], [], Z, Z).
conc3([], [P|R], Z, [P|T]):-
  conc3([], R, Z, T).
conc3([P|R], Y, Z, [P|T]):-
  conc3(R, Y, Z, T).
/**
 * Oui c'est possible et c'est le cas de notre algorithme.
 */

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

/**
 * 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).

/**
 * Le cut dans les 3 fonctions suivates servent Ã  gÃ©nÃ©rer des ensembles sans doublons.
 */
/**
 * union_ens(+X, +Y, ?Z)
 */
union_ens(X, Y, Z):-
```

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

2 Tests

Listing 2: tp_listes_tests.pro

```
membre(2, [1,2,3,4]). % Yes
membre(5, [1,2,3,4]). % No
```

```

compte(1, [1,2,3,4], X). % X = 1
compte(5, [1,2,3,4], X). % X = 0
compte(1, [1,2,1,4], X). % X = 2
compte(1, [1,2,1,4], 2). % Yes

renverser([1,2,3,4], Y). % Y = [4,3,2,1]
renverser([1,2,3,4], [4,3,2,1]). % Yes

palind([1,2,3,4]). % No
palind([4,3,3,4]). % Yes
palind([4]). % Yes

nieme(0, [1,2,3,4], A). % A = -1
nieme(3, [1,2,3,4], A). % A = 2

hors_de(0, [1,2,3,4]). % Yes
hors_de(3, [1,2,3,4]). % No

tous_diff([1,2,3,4]). % Yes
tous_diff([1,2,4,4]). % No
tous_diff([]). % Yes

conc3([1,2], [3,4], [5,6], T). % T = [1,2,3,4,5,6]
conc3([1,2], [3,4], [5,6], [1,2,3,4,5,6]). % Yes
conc3([1,2], [3,4], [5,6], [1,2,3,4,5]). % No
conc3([], [], [], []). % Yes

debute_par([1,2,3], []). % Yes
debute_par([1,2,3], [1]). % Yes
debute_par([1,2,3], [1,2,3]). % Yes
debute_par([1,2,3], [2,3]). % No

sous_liste([1,2,3], []). % Yes
sous_liste([1,2,3], [1]). % Yes
sous_liste([1,2,3], [2,3]). % Yes
sous_liste([1,2,3], [1,3]). % No

elim([1,2,3], Y). % Y = [1,2,3]
elim([3,1,2,3,3], Y). % Y = [1,2,3]
elim([], Y). % Y = []

tri([1,2,3,4], Y). % Y = [1,2,3,4]
tri([4,3,2,1], Y). % Y = [1,2,3,4]
tri([3,1,2,4], Y). % Y = [1,2,3,4]

inclus([], [1,2,3,4]). % Yes
inclus([1,2,3,4], [1,2,3,4]). % Yes
inclus([1,2,4], [1,2,3,4]). % Yes
inclus([1,2,5], [1,2,3,4]). % No

non_inclus([1,2,3,4], []). % Yes
non_inclus([5], [1,2,3,4]). % Yes
non_inclus([], [1,2,3,4]). % No
non_inclus([2,3], [1,2,3,4]). % No

union_ens([1,2,3], [2,3,4], [1,2,3,4]). % Yes
union_ens([2,3,1], [4,2,3], [3,2,1,4]). % Yes
union_ens([2,3,1], [4,2,3], [3,2,1]). % No
union_ens([1,2,3], [2,3,4], Z). % Z = [1,2,3,4]
union_ens([2,3,1], [4,2,3], Z). % Z = [1,2,3,4]

inter_ens([1,2,3], [2,3,4], [2,3]). % Yes
inter_ens([2,3,1], [4,2,3], [3,2]). % Yes
inter_ens([2,3,1], [4,2,3], [3,2,1]). % No
inter_ens([1,2,3], [2,3,4], Z). % Z = [2,3]
inter_ens([2,3,1], [4,2,3], Z). % Z = [2,3]

diff_ens([1,5,3], [5,4,7], Z). % Z = [1,3]
diff_ens([1,5,3], [], Z). % Z = [5,1,3]
diff_ens([1,5,3], [5,4,7], [3,1]). % Yes
diff_ens([1,5,3], [5,4,7], [1,3]). % Yes
diff_ens([1,5,3], [5,4,7], [1,3,5]). % No
diff_ens([1,5,3], [5,4,7], [1]). % No
```