TP9 - Binômes

Paul Chaignon - Clément Gautrais

November 28, 2013

1 Résolution par force brute

1.1 Questions

Listing 1: tp9.pro

```
1  /**
2   * Question 1.1
3   * combiner(+Buddies, -Pairs)
4   */
5  combiner([], []).
6  combiner([First|Buddies], Pairs):-
7    make_pairs(First, Buddies, Pairs1),
8    combiner(Buddies, Pairs2),
9    concat(Pairs1, Pairs2, Pairs).
10
11 /**
12  * make_pairs(+Buddy, +Buddies, -Pairs)
13  */
14 make_pairs(Buddy, [], []).
15 make_pairs(Buddy, [First|Buddies], [(Buddy, First)|Pairs]):-
16   make_pairs(Buddy, Buddies, Pairs).
17
18 /**
19  * concat(+X, +Y, ?T)
20  */
21 concat([], Y, Y).
22 concat([P|R], Y, [P|T]):-
23   concat(R, Y, T).
24
25
26 /**
27  * Question 1.2
28  * extraire(+AllPossiblePairs, +NbPairs, -Tp, -RemainingPairs)
29  */
30 extraire(AllPossiblePairs, 0, [], AllPossiblePairs).
31 extraire([PossiblePair|AllPossiblePairs], NbPairs, [PossiblePair|Tp],
      NewRemainingPairs):-
32   NbPairs > 0,
33   NewNbPairs is NbPairs - 1,
34   extraire(AllPossiblePairs, NewNbPairs, Tp, RemainingPairs),
35   not(pair_in_array(PossiblePair, Tp)),
36   delete_pair(RemainingPairs, PossiblePair, NewRemainingPairs).
37 extraire([PossiblePair|AllPossiblePairs], NbPairs, Tp,
      [PossiblePair|RemainingPairs]):-
```

```
38    NbPairs > 0,
39    extraire(AllPossiblePairs, NbPairs, Tp, RemainingPairs),
40    pair_in_array(PossiblePair, Tp).
41
42  /**
43   * delete_pair(+Pairs, +Pair, -PairsWithoutPair)
44   */
45  delete_pair([], _, []).
46  delete_pair([Pair|Pairs], Pair, Pairs):-!.
47  delete_pair([FirstPair|Pairs], Pair, [FirstPair|PairsWithoutPair]):-
48    delete_pair(Pairs, Pair, PairsWithoutPair).
49
50  /**
51   * pair_in_array(+Pair, +Pairs)
52   */
53  pair_in_array((A, B), [(C, D)|Pairs]):-
54    (A == C ; B == D ; A == D ; B == C),
55    !.
56  pair_in_array(Pair, [FirstPair|Pairs]):-
57    pair_in_array(Pair, Pairs).
58
59
60  /**
61   * Question 1.3
62   * les_tps(+Buddies, -Tps)
63   */
64  les_tps(Buddies, Tps):-
65    combiner(Buddies, PossiblePairs),
66    length(Buddies, NbBuddies),
67    NbPairs is integer(NbBuddies / 2),
68    findall(Tp, extraire(PossiblePairs, NbPairs, Tp, _), Tps).
```

1.2 Tests

Listing 2: tp9_tests.pro

```
1  combiner([pluto, riri, fifi, loulou], Pairs).
2    Pairs = [(pluto, riri), (pluto, fifi), (pluto, loulou),
3          (riri, fifi), (riri, loulou), (fifi, loulou)]
4
5  combiner([pluto, riri, fifi, loulou], Pairs), extraire(Pairs, 2, Tp, R).
6    Pairs = [(pluto, riri), (pluto, fifi), (pluto, loulou), (riri, fifi),
       (riri, loulou), (fifi, loulou)]
7    Tp = [(pluto, riri), (fifi, loulou)]
8    R = [(pluto, fifi), (pluto, loulou), (riri, fifi), (riri, loulou)]
9
10   Pairs = [(pluto, riri), (pluto, fifi), (pluto, loulou), (riri, fifi),
       (riri, loulou), (fifi, loulou)]
11   Tp = [(pluto, fifi), (riri, loulou)]
12   R = [(pluto, riri), (pluto, loulou), (riri, fifi), (fifi, loulou)]
13
14   Pairs = [(pluto, riri), (pluto, fifi), (pluto, loulou), (riri, fifi),
       (riri, loulou), (fifi, loulou)]
15   Tp = [(pluto, loulou), (riri, fifi)]
```

```
16   R = [(pluto, riri), (pluto, fifi), (riri, loulou), (fifi, loulou)]
17
18 les_tps([pluto, riri, fifi, loulou], Tps).
19   Tps = [[(pluto, riri), (fifi, loulou)],
20       [(pluto, fifi), (riri, loulou)],
21       [(pluto, loulou), (riri, fifi)]]
22
23 les_tps([a,b,c,d,e,f], Tps).
24   Tps = [[(a, b), (c, d), (e, f)], [(a, b), (c, e), (d, f)],
25       [(a, b), (c, f), (d, e)], [(a, d), (b, e), (c, f)],
26       [(a, d), (b, f), (c, e)], [(a, e), (b, f), (c, d)]]
```

2 Résolution optimale

2.1 Questions

Listing 3: tp9_optional.pro

```
1  /**
2   * Part 2.1
3   * make_lists(+Buddies, -First, -FirstList, -SecondList)
4   */
5  make_lists(Buddies, FirstBuddy, FirstList, SecondList):-
6    length(Buddies, NbBuddies),
7    LengthFirstList is integer(NbBuddies / 2),
8    split_buddies(Buddies, LengthFirstList, [FirstBuddy|FirstList],
       SecondList).
9
10 /**
11  * split_buddies(+Buddies, +LengthFirstList, -FirstList, -SecondList)
12  */
13 split_buddies(Buddies, 0, [], Buddies).
14 split_buddies([FirstBuddy|Buddies], LengthFirstList,
     [FirstBuddy|FirstList], SecondList):-
15   NewLengthFirstList is LengthFirstList - 1,
16   split_buddies(Buddies, NewLengthFirstList, FirstList, SecondList).
17
18
19 /**
20  * Part 2.2
21  * rotate(+FirstList, +SecondList, -NewFirstList, -NewSecondList)
22  */
23 rotate(FirstList, [FirstSecondList|SecondList],
     [FirstSecondList|NewFirstList], NewSecondList):-
24   popLast(FirstList, LastFirstList, NewFirstList),
25   putLast(SecondList, LastFirstList, NewSecondList).
26
27 /**
28  * popLast(+List, -Last, -NewList)
29  */
30 popLast([Last], Last, []):-!.
```

```
31 popLast([First|List], Last, [First|NewList]):-
32   popLast(List, Last, NewList).
33
34 /**
35  * putLast(+List, +Last, -NewList)
36  */
37 putLast([], Last, [Last]):-!.
38 putLast([First|List], Last, [First|NewList]):-
39   putLast(List, Last, NewList).
40
41
42 /**
43  * Part 2.3
44  * get_pairs(+FirstList, +SecondList, -Pairs)
45  */
46 get_pairs([], [], []).
47 get_pairs([FirstOfFirstList|FirstList], [FirstOfSecondList|SecondList],
      [(FirstOfFirstList, FirstOfSecondList)|Pairs]):-
48   get_pairs(FirstList, SecondList, Pairs).
49
50
51 /**
52  * Part 2.4
53  * les_tps(+Buddies, -Tps)
54  */
55 les_tps(Buddies, Tps):-
56   NbRotations is length(Buddies) - 1,
57   make_lists(Buddies, FirstBuddy, FirstList, SecondList),
58   les_tps(FirstBuddy, FirstList, SecondList, NbRotations, Tps).
59
60 /**
61  * les_tps(+FirstBuddy, +FirstList, +SecondList, +NbRotations, -Tps)
62  */
63 les_tps(_, _, _, 0, []):-!.
64 les_tps(FirstBuddy, FirstList, SecondList, NbRotations, [Pairs|Tps]):-
65   NewNbRotations is NbRotations - 1,
66   get_pairs([FirstBuddy|FirstList], SecondList, Pairs),
67   rotate(FirstList, SecondList, NewFirstList, NewSecondList),
68   les_tps(FirstBuddy, NewFirstList, NewSecondList, NewNbRotations, Tps).
69
70 /**
71  * range(+I, +J, -List)
72  * range(+Nb, -List)
73  */
74 range(Nb, List):-
75   range(1, Nb, List).
76 range(I, J, []):-
77   I > J.
78 range(I, J, [I|Tail]):-
79   I =< J,
80   I1 is I + 1,
81   range(I1, J, Tail).
```

2.2 Tests

Listing 4: tp9_optional_tests.pro

```
make_lists([1,2,3,4,5,6,7,8], First, FirstList, SecondList).
  First = 1
  FirstList = [2, 3, 4]
  SecondFirst = [5, 6, 7, 8]
rotate([2, 3, 4], [5, 6, 7, 8], FirstList, SecondList).
  FirstList = [5, 2, 3]
  SecondList = [6, 7, 8, 4]
get_pairs([1,2,3,4], [5,6,7,8]).
  Pairs = [(1, 5), (2, 6), (3, 7), (4, 8)]
range(8, Buddies), les_tps(Buddies, Tps).
  Tps = [[(1, 5), (2, 6), (3, 7), (4, 8)], [(1, 6), (5, 7), (2, 8), (3,
     4)],
      [(1, 7), (6, 8), (5, 4), (2, 3)], [(1, 8), (7, 4), (6, 3), (5,
         2)],
      [(1, 4), (8, 3), (7, 2), (6, 5)], [(1, 3), (4, 2), (8, 5), (7,
         6)],
      [(1, 2), (3, 5), (4, 6), (8, 7)]]
range(1000, Buddies), les_tps(Buddies, Tps).
  Tps = ... on vous passe les resultats... mais ca semble correct.
```