TP5 - Arithmétique

Paul Chaignon - Clément Gautrais

October 11, 2013

1 Arithmétique de Peano

1.1 Questions

Listing 1: tp_peano.pro

```
1  /**
2   * Question 1
3   * add(?, ?, ?)
4   */
5  add(zero, X, X).
6  add(s(X), Y, s(Sum)):-
7    add(X, Y, Sum).
8
9  /**
10  * Question 2
11  * sub(?, ?, ?)
12  */
13 sub(X, zero, X).
14 sub(s(X), s(Y), Sub):-
15   sub(X, Y, Sub).
16
17 /**
18  * Question 3
19  * prod(+, +, -)
20  */
21 prod(zero, Y, zero).
22 prod(s(X), Y, Prod):-
23   prod(X, Y, Prod1),
24   add(Y, Prod1, Prod).
25
26 /**
27  * Question 4
28  * factorial(+, -)
29  */
30 factorial(zero, s(zero)).
31 factorial(s(X), Y):-
32   factorial(X, Y1),
33   prod(s(X), Y1, Y).
```

1.2 Tests

Listing 2: tp_peano_tests.pro

```
add(s(zero), s(s(zero)), Sum).
% Sum = s(s(s(zero)))
add(X, Y, s(s(zero))).
% X = zero
% Y = s(s(zero)) ?
% X = s(zero)
% Y = s(zero) ?
% X = s(s(zero))
% Y = zero ?

sub(s(s(zero)), s(zero), Sub).
% Sub = s(zero)
sub(s(s(zero)), s(s(zero)), Sub).
% Sub = zero
sub(s(s(zero)), s(s(s(zero))), Sub).
% No

prod(s(s(zero)), s(s(s(zero))), Prod).
% Prod = s(s(s(s(s(zero)))))

factorial(s(s(s(zero))), F).
% F = s(s(s(s(s(zero)))))
```

2 Représentation binaire

2.1 Questions

Listing 3: tp_binaire.pro

```
sub_bit(0, 0, 0, 0, 0).
sub_bit(0, 0, 1, 1, 1).
sub_bit(0, 1, 0, 1, 1).
sub_bit(0, 1, 1, 0, 1).
sub_bit(1, 0, 0, 1, 0).
sub_bit(1, 0, 1, 0, 0).
sub_bit(1, 1, 0, 0, 0).
sub_bit(1, 1, 1, 1, 1).

/**
 * bin2int(+, -)
 */
bin2int(Bin, Int):-
  bin2int(Bin, Int, 0, 0).
bin2int([], Res, _, Res).
bin2int([Bit|X], Res, Rank, Acc):-
  Rank2 is Rank + 1,
  Acc2 is Acc + Bit*(2^Rank),
  bin2int(X, Res, Rank2, Acc2).

/**
 * int2bin(+, -)
 */
int2bin(1, [1]).
```

```
25 int2bin(X, [Rest|Y]):-
26   X =\= 1,
27   Next is div(X, 2),
28     Rest is mod(X, 2),
29     int2bin(Next, Y).
30
31 /**
32  * Question 5
33  * add(?, ?, ?)
34  * La surcharge est-elle possible en Prolog ?
35  * Cad: mettre add a la place de add_bis.
36  */
37 add_bis([], [], [], 0).
38 add_bis([], [], [1], 1).
39 add_bis([], [Bit2|R2], [Res | Result], CarryIn):-
40   add_bit(0, Bit2, CarryIn, Res, CarryOut),
41   add_bis([], R2, Result, CarryOut).
42 add_bis([Bit1|R1], [], [Res | Result], CarryIn):-
43   add_bit(Bit1, 0, CarryIn, Res, CarryOut),
44   add_bis(R1, [], Result, CarryOut).
45 add_bis([Bit1|R1], [Bit2|R2], [Res | Result], CarryIn):-
46   add_bit(Bit1, Bit2, CarryIn, Res, CarryOut),
47   add_bis(R1, R2, Result, CarryOut).
48 add(R1, R2, Res):-
49   add_bis(R1, R2, Res, 0).
50
51 /**
52  * Question 6
53  * sub(?, ?, ?)
54  */
55 sub_bis([], [], [], 0).
56 sub_bis([], [Bit2|R2], [Res|Result], CarryIn):-
57   sub_bit(0, Bit2, CarryIn, Res, CarryOut),
58   sub_bis([], R2, Result, CarryOut).
59 sub_bis([Bit1|R1], [], [Res|Result], CarryIn):-
60   sub_bit(Bit1, 0, CarryIn, Res, CarryOut),
61   sub_bis(R1, [], Result, CarryOut).
62 sub_bis([Bit1|R1], [Bit2|R2], [Res | Result], CarryIn):-
63   sub_bit(Bit1, Bit2, CarryIn, Res, CarryOut),
64   sub_bis(R1, R2, Result, CarryOut).
65 sub(R1, R2, Res):-
66   sub_bis(R1, R2, Res, 0).
67
68 /**
69  * Question 7
70  * prod(+, +, -)
71  */
72 prod_bis([], _, [], _).
73 prod_bis([1|X], Y, Res, Offset):-
74   prod_bis(X, Y, ResSuite, [0|Offset]),
75   add(ResSuite, Offset, Res).
76 prod_bis([0|X], Y, Res, Offset):-
77   prod_bis(X, Y, Res, [0|Offset]).
78 prod(X, Y, Res):-
79   prod_bis(X, Y, Res, Y).
80
81 /**
82  * Question 8
```

```
 * factorial(+, -)
 */
egal([], []).
egal([0|R1], []):-
  egal(R1, []).
egal([X|R1], [X|R2]):-
  egal(R1, R2).

factorial(X, [1]):-
  egal(X, []).
factorial(X, Y):-
  sub(X, [1], X1),
  factorial(X1, Y1),
  prod(X, Y1, Y).

/**
 * Question 9
 * factorial(+, -)
 */
factorial_is(0, 1).
factorial_is(X, Res):-
  X1 is X - 1,
  factorial_is(X1, Res1),
  Res is X * Res1,
  !.
```

2.2 Tests

Listing 4: tp_binaire_tests.pro

```
add([1], [0, 0, 1, 1], Sum).
% Sum = [0, 0, 1, 1]
add([1, 0, 0, 0], [0, 0, 1, 1], [1, 0, 1, 1]).
% Yes
add([1, 0, 0, 0], [0, 0, 1, 1], Sum).
% Sum = [1, 0, 1, 1]

sub([1, 1], [1, 0], Sub).
% Sub = [0, 1]
sub([1, 1, 0, 1], [1, 0, 0, 1], Sub).
% Sub = [0, 1, 0, 0]
sub([1, 1, 0, 1], [1, 0, 0, 1], [0, 1, 0, 0]). % Yes

prod([1, 1, 0, 1], [1, 0, 0, 1], Mul).
% Mul = [1, 1, 0, 0, 0, 1, 1]
prod([], [1, 1], Mul).
% Mul = []
prod([0, 1, 0, 1], [1, 1], Mul).
% Mul = [0, 1, 1, 1, 1]

int2bin(5, IntBin), factorial(IntBin, OutBin), bin2int(OutBin, Fact).
% Fact = 120
% IntBin = [1, 0, 1]
% OutBin = [0, 0, 0, 1, 1, 1, 1]
int2bin(10, IntBin), factorial(IntBin, OutBin), bin2int(OutBin, Fact).
% Fact = 3628800
```

```
% IntBin = [0, 1, 0, 1]
% OutBin = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
    1, 1]
```

3 Evaluation d'expressions arithmétiques

3.1 Questions

Listing 5: tp_expression.pro

```
/**
 * Question 1
 * evaluate(+, -)
 */
/**
 * Verifie que X est booleen.
 */
boolean(X):-
  X = t; X = f.

/**
 * Le resultat de l'evaluation doit etre booleen.
 */
evaluate_boolean(B1, B2):-
  evaluate(B1, B2),
  boolean(B2).

/**
 * Le resultat doit etre un booleen ou un nombre.
 */
evaluate(N, N):-
  number(N);
  boolean(N).

/**
 * Evaluation de conditionnelles.
 */
evaluate(if(Cond1, Then1, Else1), Res):-
  evaluate_boolean(Cond1, Cond2),
  (
    Cond2 = t, evaluate(Then1, Res)
    ;
    Cond2 \= t, evaluate(Else1, Res)
  ).


/**
 * Question 2
 * assoc(+, +, -)
 */
fresh_variables(Number, _, Number):-
  number(Number),
  boolean(Number).

assoc(X, [(X, Y)], Y):-!.
assoc(X, [(In, Out)|Assoc], Res):-
  X == In,
```

```
48   Res = Out.
49 assoc(X, [(In, Out)|Assoc], Res):-
50   X \== In,
51   assoc(X, Assoc, Res).
52
53
54 /**
55  * Question 3
56  * Evaluation de l'application d'une fonction a une expression.
57  */
58 evaluate(apply(Expr, Param), Res):-
59   fresh_variables(Expr, ExprFreshed),
60   evaluate(ExprFreshed, fun(X, Body)),
61   X = Param,
62   evaluate(Body, Res).
```

3.2 Tests

Listing 6: tp_expression_tests.pro

```
1 fresh_variables(fun(X, fun(Y, add(Y, prod(X, X)))), Fresh).
2 % Fresh = fun(A,fun(B,add(B,prod(A,A))))
3 F = fun(X, prod(X, X)), evaluate(apply(F, 1), Res1), evaluate(apply(F,
    2), Res2).
4 % Res1 = 1
5 % Res2 = 4
6 Fun = fun(N, fun(F, if(eq(N, 0), 1, prod(N, apply(apply(F, sub(N, 1)),
    F))))), Factorial = fun(N, apply(apply(Fun, N), Fun)),
    evaluate(apply(Factorial, 19), Res).
7 % Res = 121645100408832000
```