非手続き型言語1回目課題 解答例

April 23, 2020

1　演習問題3.1.1 e) – f)

```
(* c  -- e で使う*)
fun thirdelement (L) = hd(tl(tl(L)));

(* e *)
fun thirdchar (S) = thirdelement(explode S);

(* f *)
fun cyclelist (L) =
  if L = nil then nil
  else   tl(L)@[hd(L)];
```

2　演習問題3.1.2 a) – d)

```
(* a *)
fun minmaxpair (a, b, c) =
    if a > b then
       if a > c then
          if b > c then (c, a)
          else          (b, a)
       else             (b, c)

    else
       if b > c then
          if a > c then (c, b)
          else          (a, b)
       else             (a, c);


(* 3.1.2 a) 別解 *)
fun min3 (a, b, c) =
  if a < b then
```

```
     if a < c then a
     else  c
  else if b < c then b
     else c;

fun max3 (a, b, c) =
  if a > b then
     if a > c then a
     else  c
  else if b > c then b
     else c;

fun minmaxpair2(a,b,c) = (min3(a,b,c), max3(a,b,c));


(* b *)
fun sorted3tuple (a, b, c) =
    if a > b then
       if a > c then
          if b > c then [c, b, a]
          else          [b, c, a]
       else             [b, a, c]

    else
       if b > c then
          if a > c then [c, a, b]
          else          [a, c, b]
       else             [a, b, c];


(* 別解 *)
fun mid3(a,b,c) =
  if(a < b) then if (b < c) then b
                 else if a > c then a
                      else c
  else if (a < c) then a
       else if (b < c) then c
       else b;

fun sorted3tuple2(a,b,c)=
    [min3(a,b,c), mid3(a,b,c), max3(a,b,c)];

(* c *)
fun round10 (x) = round (x / 10.0) * 10;

(* d *)
```

```
fun deletesecond (L) = hd(L)::tl(tl(L));
```

3　実行結果

```
- use "ML1answer.ml";
[opening ML1answer.ml]
val thirdelement = fn : 'a list -> 'a
val thirdchar = fn : string -> char
ML1answer.ml:11.8 Warning: calling polyEqual
val cyclelist = fn : ''a list -> ''a list
val minmaxpair = fn : int * int * int -> int * int
val min3 = fn : int * int * int -> int
val max3 = fn : int * int * int -> int
val minmaxpair2 = fn : int * int * int -> int * int
val sorted3tuple = fn : int * int * int -> int list
val mid3 = fn : int * int * int -> int
val sorted3tuple2 = fn : int * int * int -> int list
val round10 = fn : real -> int
val deletesecond = fn : 'a list -> 'a list
val it = () : unit
- thirdchar ("abcdef");
val it = #"c" : char
- cyclelist ([1,2,3,4,5]);
val it = [2,3,4,5,1] : int list
- minmaxpair (10, 5, ~3);
val it = (~3,10) : int * int
- sorted3tuple(33, ~6, ~19);
val it = [~19,~6,33] : int list
- round10(113.7);
val it = 110 : int
- deletesecond ([1,2,3,4,5]);
val it = [1,3,4,5] : int list
-
```

非手続き型言語2回目課題 解答例

April 27, 2020

1　関数 comb (第2回のスライドの問題)

```
fun comb (n, m) =
  if m = 0 orelse n = m then 1
  else comb(n-1, m) + comb(n-1, m-1);
```

2　演習問題 3.2.1 d) – f)

```
(* 3.2.1 d)*)

fun ex321d (L) = if L = [] then 0
                    else 1 + ex321d(tl(L));

(* 3.2.1 e)*)

fun ex321e (x, i) = if i = 0 then 1.0
                         else x * ex321e(x, i-1);

(* 3.2.1 f) 素直な考え方*)
fun ex321f (L) = if tl(L) = [] then hd(L)
else if hd(L) > ex321f(tl(L)) then hd(L)
     else ex321f(tl(L));

(* ex321f の再帰呼び出しを減らす工夫をした解答 *)
fun maxList(L) =
    if tl(L) = nil then hd(L)
    else (* 少なくとも二つの要素がある場合 *)
       if hd(L)>hd(tl(L)) then maxList(hd(L)::tl(tl(L)))
       else maxList(tl(L));
```

3　実行結果

```
- use "ML2answer.ml";
[opening ML2answer.ml]
val comb = fn : int * int -> int
ML2answer.ml:7.23 Warning: calling polyEqual
val ex321d = fn : ''a list -> int
val ex321e = fn : real * int -> real
val ex321f = fn : int list -> int
val maxList = fn : int list -> int
val it = () : unit
- comb (5,2);
val it = 10 : int
- ex321d [];
val it = 0 : int
- ex321d (["abc", "c", "dd"]);
val it = 3 : int
- ex321e (2.0, 4);
val it = 16.0 : real
- ex321f ([4,1,5,~2]);
val it = 5 : int
-
```

非手続き型言語3回目課題 解答例

April 30, 2020

1　関数 mygcd (第3回のスライドの問題)

```
fun mygcd(a, 0) = a
| mygcd(a, b) = mygcd(b, a mod b);
```

2　演習問題 3.3.2

```
fun swapElement ([]) = []
|   swapElement ([x]) = [x]
|   swapElement  (x::y::zs) = y::x::swapElement(zs);
```

3　演習問題 3.3.3

```
(* 先頭が 1 番目 *)
fun deleteIElement ([], i) = []
|   deleteIElement (x::xs, 1) = xs
|   deleteIElement (x::xs, i) = x::deleteIElement(xs, i-1);
```

4　演習問題 3.3.11

```
(* a *)
fun member(x,[]) = false
 |  member(x, y::ys) = if x = y then true
                       else member(x, ys);
(*
Warning: calling polyEqual が出るが、これは無視してよい
*)

(* orelse を使ったこういうのもできる これも  Warning: calling polyEqual がでる*)
fun member2(x,[]) = false
|   member2(x,y::ys) =
        x=y orelse member(x,ys);
```

```
(* b *)
fun delete(x,[]) = nil
 |  delete(x,y::ys) = if x = y then ys
                      else y::delete(x,ys);
(*
Warning: calling polyEqual が出るが、これは無視してよい
*)
(* c *)
Fun insert(x,S) = if member(x, S) then S
                  else x::S;

(* member を使わない 以下のものでもよい*)
(* この解答例で as は別名をつけるというもの. S as y::ys は, S としても y::ys としても同じリストを意味するという意味 *)

fun insert2(x,[]) = [x]
|   insert2(x,S as y::ys) =
        if x=y then S else y::insert2(x,ys);
```

5 実行結果

```
- use "ML3answer.ml";
[opening ML3answer.ml]
val mygcd = fn : int * int -> int
val swapElement = fn : 'a list -> 'a list
val deleteIElement = fn : 'a list * int -> 'a list
ML3answer.ml:33.29 Warning: calling polyEqual
val member = fn : ''a * ''a list -> bool
ML3answer.ml:42.10 Warning: calling polyEqual
val member2 = fn : ''a * ''a list -> bool
ML3answer.ml:47.28 Warning: calling polyEqual
val delete = fn : ''a * ''a list -> ''a list
val insert = fn : ''a * ''a list -> ''a list
ML3answer.ml:62.13 Warning: calling polyEqual
val insert2 = fn : ''a * ''a list -> ''a list
val it = () : unit
- mygcd (100, 55);
val it = 5 : int
- swapElement([2,3,4,5,6]);
val it = [3,2,5,4,6] : int list
- deleteIElement ([1,2,3,4,5], 3);
val it = [1,2,4,5] : int list
- member (3, [1,2,3,4,5]);
```

```
val it = true : bool
- delete (3, [1,2,3,4,5]);
val it = [1,2,4,5] : int list
- insert (7, [1,2,3,4,5]);
val it = [7,1,2,3,4,5] : int list
- insert2 (7, [1,2,3,4,5]);
val it = [1,2,3,4,5,7] : int list
```

非手続き型言語4回目課題 解答例

May 7, 2020

1　演習問題 3.3.13

```
fun inserteach(_,nil) = nil  (* これは演習問題 3.3.12 の解答 *)
 |  inserteach(a,x::xs) = (a::x)::inserteach(a,xs);

fun powerset(nil) = [nil]
 |  powerset(x::xs) = powerset(xs)@inserteach(x,powerset(xs));
```

2　演習問題 3.4.3

```
(* Ex. 3.3.13 の局所環境版 *)
fun PowerSet(nil) = [nil]
 |  PowerSet(x::xs) =
    let
        val s = PowerSet(xs)
    in
        s@inserteach(x,s)
    end;
```

3　演習問題 3.4.4

```
(* 演習問題 3.2.1 f のときはこうしたはず
fun maxnumber2(x::nil) = x:real
 |  maxnumber2(x::xs) = if x > maxnumber2(xs) then x
                        else maxnumber2(xs);
*)

fun maxnumber2(x::nil) = x:real
 |  maxnumber2(x::xs) =
     let
       val m = maxnumber2(xs)
     in
       if x > m then x
```

```
      else m
    end;

(* こういう解答でもいい *)
fun maxnumber3(x::nil) = x:real
|   maxnumber3(x::xs) =
   let
     fun max(a,b) = if a > b then a else b
   in
     max(x, maxnumber3(xs))
   end;

(* 上の解答例では Warning : match nonexhaustive が出る.
   例外を用いてそれが出ないようにした改良版*)

exception NullList;
fun maxnumber4(nil) = raise NullList
 | maxnumber4(x::nil) = x:real
 |  maxnumber4(x::xs) =
     let
       val m = maxnumber4(xs)
     in
       if x > m then x
       else m
       end;
```

4 演習問題 5.2.1

```
exception TooShort;
fun thirdelement(nil) = raise TooShort
|   thirdelement(x::nil) = raise TooShort
|   thirdelement(x::y::nil) = raise TooShort
|   thirdelement(x::y::z::zs) = z;

(*スマートな解答*)
exception TooShort;
fun   thirdelement(x::y::z::zs) = z
|     thirdelement(_) = raise TooShort;
```

5 演習問題 5.2.2

```
exception Negative;
fun fact(0) = 1
|   fact(n) =
```

```
    if n > 0 then n * fact(n-1)
    else raise Negative;
```

6 実行結果

```
- use "ML4answer.ml";
[opening ML4answer.ml]
val inserteach = fn : 'a * 'a list list -> 'a list list
val powerset = fn : 'a list -> 'a list list
val PowerSet = fn : 'a list -> 'a list list
ML4answer.ml:26.5-33.9 Warning: match nonexhaustive
          x :: nil => ...
          x :: xs => ...

val maxnumber2 = fn : real list -> real
ML4answer.ml:36.5-42.7 Warning: match nonexhaustive
          x :: nil => ...
          x :: xs => ...

val maxnumber3 = fn : real list -> real
exception NullList
val maxnumber4 = fn : real list -> real
exception TooShort
val thirdelement = fn : 'a list -> 'a
exception TooShort
val thirdelement = fn : 'a list -> 'a
exception Negative
val fact = fn : int -> int
val it = () : unit
- powerset ([1,2,3]);
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
- PowerSet ([1,2,3]);
val it = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]] : int list list
- maxnumber2 ([3.0, 1.2, ~1.1, 4.0]);
val it = 4.0 : real
- maxnumber4 ([]);

uncaught exception NullList
  raised at: ML4answer.ml:48.29-48.37
- thirdelement ([1,2,3,4,5]);
val it = 3 : int
- thirdelement ([1,2]);

uncaught exception TooShort
  raised at: ML4answer.ml:79.31-79.39
```

```
- fact (5);
val it = 120 : int
- fact (~3);

uncaught exception Negative
  raised at: ML4answer.ml:88.17-88.25
-
```

非手続き型言語5回目課題 解答例

May 11, 2020

1　局所環境を使った関数 rev

```
fun rev(L) =
    let
        fun rev1(M, []) = M
        | rev1(M, x::xs) = rev1(x::M, xs)
    in
        rev1([],L)
    end;
```

2　演習問題 3.5.2

ここでは局所環境を使わない解答例を示すが，局所環境を用いた方がさらによい.

```
fun cat(nil,M) = M
|   cat(x::xs,M) = x::cat(xs,M);

fun cycle1(L, M, 0) = cat(L, rev(M))
|   cycle1(x::xs, M, i) = cycle1(xs, x::M, i-1);

fun cycle(L,i) = cycle1(L, nil, i);
```

関数 rev を使わない別解 (問題文のヒントはこちらのことを言っていると思われる).

```
fun split (nil, ys, i) = (nil, ys)
| split (x::xs, ys, 0) = (x::xs, ys)
|  split (x::xs, ys, i) =
     let
       val (M, N) = split(xs, ys, i-1)
     in
       (M, x::N)
     end;

fun cat(nil,M) = M
```

```
|   cat(x::xs,M) = x::cat(xs,M);

fun cycle2(L, i) =
     let
        val (M, N) = split(L, nil, i)
     in
      cat(M, N)
     end;
```

3　1 から n までの総和を求める末尾再帰の関数

```
fun sum(n) =
    let
        fun f1(0, s) = s
        | f1(n, s) = f1(n-1,s+n)
    in
        f1(n,0)
    end;
```

4　実行結果

```
- use "ML5answer.ml";
[opening ML5answer.ml]
val rev = fn : 'a list -> 'a list
val cat = fn : 'a list * 'a list -> 'a list
ML5answer.ml:15.5-16.48 Warning: match nonexhaustive
          (L,M,0) => ...
          (x :: xs,M,i) => ...

val cycle1 = fn : 'a list * 'a list * int -> 'a list
val cycle = fn : 'a list * int -> 'a list
val split = fn : 'a list * 'a list * int -> 'a list * 'a list
val cat = fn : 'a list * 'a list -> 'a list
val cycle2 = fn : 'a list * int -> 'a list
val sum = fn : int -> int
val it = () : unit
- rev ([1,2,3,4,5]);
val it = [5,4,3,2,1] : int list
- cycle ([1,2,3,4,5], 4);
val it = [5,1,2,3,4] : int list
- cycle2 ([1,2,3,4,5], 4);
val it = [5,1,2,3,4] : int list
- sum (10);
val it = 55 : int
```

非手続き型言語6回目課題 解答例

May 14, 2020

1 演習問題 3.6.3

```
fun eval(nil, _) = 0.0
|  eval(x::xs, a) = x + a * eval(xs, a);
```

2 演習問題 3.6.4

```
fun padd(P, nil) = P
|   padd(nil, Q) = Q
|   padd((p:real)::ps, q::qs) = (p+q)::padd(ps, qs);

fun smult(nil,(q:real)) = nil
|   smult(p::ps, q) = (p*q)::smult(ps,q);

fun pmult(P, nil) = nil
|   pmult(P, q::qs) = padd(smult(P,q), 0.0::pmult(P, qs));

fun getPoly(nil) = [1.0]
|   getPoly(x::xs) = pmult([~x, 1.0], getPoly(xs));

(* これは間違い
fun getPoly2(nil) = nil  <- ここが違う．掛け算していくので基底は1.0でないとダメ
|   getPoly2(x::xs) = pmult([~x, 1.0], getPoly2(xs));
*)
```

3 演習問題 3.6.5

```
(* 加算 *)
fun mpadd(PP, nil) = PP
|   mpadd(nil, QQ) = QQ
|   mpadd(P::PS, Q::QS) = padd(P,Q)::mpadd(PS, QS);
```

```
(*スカラー倍 *)
fun mpsmult(nil, q) = nil
| mpsmult(P::PS, q) = smult(P, q)::mpsmult(PS,q);

(* 積 *)
(* 補助関数としてリストのリストとリストをかける関数 mppmult(PP, Q)*)
(*
PP = P + PSx, Q
PP * Q = P* Q + PS * Q x
PP はリストのリスト。x と y がまざっている
P, Q はリスト. ここでいうなら y のみ.
*)

fun mppmult(nil, Q) = nil
|   mppmult(P::PS, Q) = mpadd([pmult(P,Q)], [0.0]::mppmult(PS,Q));


(* 二つの変数を含む多項式の乗算 *)
(* 二つの多項式を  PP と QQ = Q+QSx とすると *)
(* PP * QQ = PP * Q + PP * QS x *)
(* つまり *)

fun mpmult(PP, []) = []
|   mpmult(PP, Q::QS) = mpadd(mppmult(PP, Q), [0.0]::mpmult(PP, QS));
```

4 実行結果

```
- use "ML6answer.ml";
[opening ML6answer.ml]
val eval = fn : real list * real -> real
val padd = fn : real list * real list -> real list
val smult = fn : real list * real -> real list
val pmult = fn : real list * real list -> real list
val getPoly = fn : real list -> real list
val mpadd = fn : real list list * real list list -> real list list
val mpsmult = fn : real list list * real -> real list list
val mppmult = fn : real list list * real list -> real list list
val mpmult = fn : real list list * real list list -> real list list
val it = () : unit
- val P = [[1.0,1.0]];
val P = [[1.0,1.0]] : real list list
- val Q = [[1.0,1.0]];
val Q = [[1.0],[1.0]] : real list list
```

```
- eval ([1.0, 1.0], 5.0);
val it = 6.0 : real
- getPoly ([1.0, 2.0, 3.0]);
val it = [~6.0,11.0,~6.0,1.0] : real list
- mpadd (P, Q);
val it = [[2.0,1.0],[1.0]] : real list list
- mpsmult (P, 2.0);
val it = [[2.0,2.0]] : real list list
- mpmult (P, Q);
val it = [[1.0,1.0],[1.0,1.0]] : real list list
```

非手続き型言語7回目課題 解答例

May 18, 2020

(今回の課題は第 7 回講義のスライドにある通りに解答すればよい．独自に調べたことを付け加えればさらによいことはいうまでもない．しかし，調べ始めるときりがないほどラムダ計算は奥が深いのでこの課題の解答としてはほどほどのところで止めるのがよいだろう．)

1　ラムダ計算とは何か

(スライド 2, 3, 4 枚目の内容を適宜まとめる)

ラムダ計算 (lambda calculus) とは計算の実行を関数への引数の評価 (evaluvation) と適用 (application) としてモデル化・抽象化した計算体系である．関数と関数の計算結果とを明確に区別するラムダ記法を用いて記述されたラムダ式を用いる．関数型言語の理論的基礎である．

2　ラムダ式の定義を述べよ

ラムダ式 (λ 式) の定義は以下である．(スライド 4 ページ目)

1. 変数 $x_0, x_1, \ldots$ は λ 式.
2. M が λ 式で x が変数のとき $(\lambda x.M)$ は λ 式.
3. M と N が λ 式のとき (MN) は λ 式.

非手続き型言語8回目課題 解答例

May 22, 2020

1　問題1

関数 trap を使って x*x*x と x*x*x*x の 0 から 1 までの積分値がそれぞれ 0.25, 0.2 となる最小の n を求める問題．(本当に「最小」のものを求めるのが大変なときは自分が計算したものの中で一番それに近い n とその時の積分値を答える．)

スライドにある以下の関数 trap を自分の処理系で定義する．

```
fun trap (a, b, n, F) =
   if n <= 0 orelse b - a <= 0.0 then 0.0
   else
       let
            val delta = (b - a) / real(n)
       in
             delta * (F(a) + F(a + delta)) /2.0
                 + trap(a+delta, b, n-1, F)
       end;
```

そして例えば以下のようにして結果を見る．$f(x) = x^3$ の場合．理論値は 0.25.

```
trap(0.0, 1.0, 1000, fn x => x * x * (x:real));
trap(0.0, 1.0, 10000, fn x => x * x * (x:real));
trap(0.0, 1.0, 100000, fn x => x * x * (x:real));
trap(0.0, 1.0, 1000000, fn x => x * x * (x:real));
```

$f(x) = x^4$ の場合．理論値は 0.2.

```
trap(0.0, 1.0, 1000, fn x => x * x * x * (x:real));
trap(0.0, 1.0, 10000, fn x => x * x * x * (x:real));
trap(0.0, 1.0, 100000, fn x => x * x * x * (x:real));
trap(0.0, 1.0, 1000000, fn x => x * x * x * (x:real));
```

(結果はマシンによって違うのでここでは省略．自分のマシンでやってみてください．)

2　資料演習問題 5.4.2

いろいろな解答例がありうるが，例えばこれ.

```
fun simpson(a,b,n,F) =
    let
       val delta = (b-a)/(2.0*real(n));
       val keisu = delta/3.0;
       fun simpsonsub(ai) = 4.0*F(ai) + 2.0*F(ai+delta);
       fun simpsonsub2(_,0)  = 0.0
        | simpsonsub2(ai,m) =
           simpsonsub(ai) + simpsonsub2(ai+delta+delta,m-1);
    in
       if n <= 0 orelse b-a <= 0.0 then 0.0
       else
           keisu*(F(a)+simpsonsub2(a+delta,n)-F(b))
    end;
```

これを使って

```
simpson(0.0,1.0,10,fn (x:real)=>x*x);
simpson(0.0,1.0,10,fn (x:real)=>x*x*x);
```

のようにして積分値が正しくなくなる最小の n を探す.

3　資料演習問題 5.4.3 b)

すでに演習問題 5.4.2 の解がこの条件を満たしていました．代わりに a) の方の解答を示しますので、問題の意図を汲み取ってください.

4　資料演習問題 5.4.3 a)

以下の関数 trap を

```
fun trap (a, b, n, F) =
   if n <= 0 orelse b - a <= 0.0 then 0.0
   else
       let
            val delta = (b - a) / real(n)
       in
             delta * (F(a) + F(a + delta)) /2.0
                 + trap(a+delta, b, n-1, F)
       end;
```

delta を再帰呼び出しのたびに計算しないようにしたもの.

```
fun trap(a,b,n,F) =
    if n <= 0 orelse b-a <= 0.0 then 0.0
    else
        let
            val delta = (b-a)/real(n);
            fun trap1(_,0) = 0.0
             |  trap1(x,i) = delta*(F(x)+F(x+delta))/2.0
                            + trap1(x+delta,i-1)
        in
            trap1(a,n)
        end;
```

5　資料演習問題 5.4.5

```
fun map(F,nil) = nil
 |  map(F,x::xs) = F(x)::map(F,xs);


(* a *)
val L=[1.0, 6.5, ~3.3, 1.2];
map(fn x=> if x > 0.0 then x else 0.0, L);

(* b *)
val L2=[1, 6, ~3];
map(fn x=> x+1, L2);


(* c *)
val S=[#"a", #"N", #"f", #"r"];
map(fn c=> if c>= #"a" andalso c<= #"z" then chr(ord(c)-32) else c, S);
(* 32 がわからなかったら*)
map(fn c=> if c>= #"a" andalso c<= #"z" then chr(ord(c)-ord(#"a")+ord(#"A")) else c, S);

(* d *)
val S2=["abcde", "Nonmonoton", "fa", "rest"];
map(fn s=> if size(s) > 5 then substring(s,0,5) else s, S2);
```

非手続き型言語9回目課題 解答例

May 25, 2020

1 問題1

資料演習問題 5.4.7 c) で空の文字列があっても動くようにする．例えば [“abc”, “def”, “ba”, “aa”, ””] を入力しても動くようにするにはどうしたらいいかを解答せよ．

解答例:関数 filter が以下のように定義されているとする．

```
fun filter (P, []) = []
|   filter (P, x::xs) = if P (x) then x::filter(P, xs) else filter(P, xs);
```

ここで例えば以下のようにすればよい．

```
filter( fn s => if s = "" then false
else hd(explode(s)) = #"a",["abc", "def", "ba", "aa",""]);
```

2 資料演習問題 5.4.11

解答例:

```
fun reduceB(g,F,[]) = g
 |  reduceB(g,F,x::xs) = F(x, reduceB(g,F,xs));
```

注釈:関数 reduce と同じように例外を定義してリストが空リストの時に例外を発生させている解答もあるが，この関数は関数 foldr の導入を見据えたもので，定数 g というのは初期値 (デフォルト値) の役割をしている．だからリストが空リストの時は g を返すようにする関数を書いて欲しいというのが問題の意図．

3 資料演習問題 5.4.12

a) リストの長さ

```
reduceB(0, fn (x,y)=>y+1, [1,2,3]);
```

b) リストの接尾辞のリスト

```
reduceB([nil], fn (x,y)=>(x::hd(y))::y, [1,2,3]);
```

解説:接尾辞のリスト．後ろから追加していく．初期値は [nil] それに [3] を加えて [2,3] を加えて [1,2,3] を加えるということをするように考える．

解答例の関数は以下のように動く．[1,2,3] に対し初期値を [nil] とする

- 最初は y=[nil], x = 3 でくるので (x::hd(y)) は [3] で (x::hd(y))::y は [[3], nil]
- 次は y = [[3], nil], x = 2 でくるので (x::hd(y)) は [2,3] で (x::hd(y))::y は [[2,3],[3],nil]
- 次は y = [[2,3], [3], nil], x = 1 でくるので (x::hd(y)) は [1,2,3] で (x::hd(y))::y は [[1,2,3],[2,3],[3],nil]

非手続き型言語10回目課題 解答例

May 29, 2020

1　演習問題5.6.1を関数mapを使って行う

c)

```
val S=[#"a", #"N", #"f", #"r"];
map (fn c=> if c>= #"a" andalso c<= #"z" then chr(ord(c)-32) else c) S;
```

d)

```
val S2=["abcde", "Nonmonoton", "fa", "rest"];
map (fn s=> if size(s) > 5 then substring(s,0,5) else s) S2;
```

2　演習問題5.6.1

d) foldr もしくは foldr で

```
val D = foldr (fn (x,y) => x^y) "";
```

foldl でするなら

```
val D2 = foldl (fn (x,y) => y^x) "";
```

e) foldr でなければならない

```
val E = foldr (op -) 0;
```

以下の解答でも OK.

```
val E = foldr (fn (x,y) => x-y) 0;
```

f) foldr もしくは foldl で

```
val F = foldr (fn (x,y) => x andalso y) true;
```

g) foldr もしくは foldl で

```
val G = foldr (fn (x,y) => x orelse y) false;
```

h) foldr もしくは foldl で

```
val H = foldr (fn (x,y) => not(x=y)) false;
```

以下でもいい (意味は一緒).

```
val H2 = foldr (fn (x,y) => x<>y) false;
```

以下のようにしてもできる．これはまずリストの true/false を 1/0 に関数 map で置き換え，関数 foldr でリスト内の 1 の数を数え，最後にその数を 2 で割ったあまりが 1 かどうかを判定している．ここで o は関数の合成 (授業ではやっていない).

```
val H3 = (fn x => if (x mod 2) = 1 then true else false ) o
(foldr (fn (x,y) => x+y) 0 ) o map (fn x=> if x then 1 else 0);
```

3 実行例

```
- use "ML10answer.ml";
[opening ML10answer.ml]
val S = [#"a",#"N",#"f",#"r"] : char list
val it = [#"A",#"N",#"F",#"R"] : char list
val S2 = ["abcde","Nonmonoton","fa","rest"] : string list
val it = ["abcde","Nonmo","fa","rest"] : string list
val D = fn : string list -> string
val D2 = fn : string list -> string
val E = fn : int list -> int
val F = fn : bool list -> bool
val G = fn : bool list -> bool
val H = fn : bool list -> bool
val H2 = fn : bool list -> bool
val H3 = fn : bool list -> bool
val it = () : unit
- D ["abc", "def", "ghi"];
val it = "abcdefghi" : string
- D2 ["abc", "def", ""];
val it = "abcdef" : string
- E [5,4,3,2,1];
val it = 3 : int
- F [true, false, true];
val it = false : bool
- G [true, false, true];
val it = true : bool
- H [true, false, true];
```

```
val it = false : bool
- H [true, true, true];
val it = true : bool
```

非手続き型言語11回目課題 解答例

June 1, 2020

1 演習問題 5.5.6

まず，部分並びの関数 subsequence を作成する．作成する方針は以下である．

1. まず二つの文字列を関数 explode を使って文字のリストに変換する．
2. 一つ目の文字のリストを X, 二つ目の文字のリストを Y とすると X の一つ目の文字が Y の中にでてくるまで調べる．
   (a) 一つ目と一致する文字が Y の中に現れたら X の 2 文字目について繰り返す
   (b) もし Y の方が先に nil になったら false
   (c) もし X の方が先に nil になったら X の文字がすべて Y に出てきたということだから true

解答例 (他にもいろいろ書き方がある．結果としてきちんと部分並びを判定できていれば OK)

```
fun sq1(nil, _) = true
|   sq1(_, nil) = false
|   sq1(x::xs, y::ys) = if x=y then sq1(xs, ys)
                        else sq1 (x::xs, ys);
fun subsequence x y = sq1(explode x, explode y);
```

次に文字列のリスト [“ear”, “part”, “trap”, “seat”] がある文字列の部分並びであるかどうかを判定するための関数リストを作る．演習 5.5.2 で作成した関数 maleFnList を使う

```
fun makeFnList F nil = nil
|   makeFnList F (x::xs) = F(x)::(makeFnList F xs);

val g = makeFnList(subsequence);
val G = g ["ear", "part", "trap", "seat"];
```

(参考)

```
val [ear, part, trap, seat] = g ["ear", "part", "trap", "seat"];
```

ということもできる．

最後にここで作成した関数リストを”separate” に対して適用する．演習問題 5.5.1 で作成した関数 applyList を使う．

```
fun applyList nil _ = nil
 |  applyList (F::Fs) a = F(a)::(applyList Fs a);

applyList G "separate";
```

2 実行例

```
- use "ML11answer.ml";
[opening ML11answer.ml]
ML11answer.ml:4.29 Warning: calling polyEqual
val sq1 = fn : ''a list * ''a list -> bool
val subsequence = fn : string -> string -> bool
val makeFnList = fn : ('a -> 'b) -> 'a list -> 'b list
val g = fn : string list -> (string -> bool) list
val G = [fn,fn,fn,fn] : (string -> bool) list
val applyList = fn : ('a -> 'b) list -> 'a -> 'b list
val it = () : unit
- applyList G "separate";
val it = [true,true,false,true] : bool list
```

非手続き型言語12回目課題 解答例

June 5, 2020

1　演習問題 3.2.3

問題:
fun f(a:int, b, c, d, e) = ...
の本体が以下のそれぞれであったときに b,c,d,e の型についてどんなことが推論できるかを述べよ.

a) if a<b+c then d else e.
　int 型である a と比較しているので b,c は int 型.
　d, e はそれぞれ then, else の後の式なので同じ型.

b) if a<b then c else d.
　b は a と比較しているので int 型.
　c,d は同じ型.

c) if a<b then b+c else d+e.
　b は a と比較しているので int 型.
　c は b と演算しているので int 型.
　d+e は b+c と同じ方でなければならないので d, e は int 型.

d) if a<b then b<c else d.
　b は a と比較しているので int 型.
　c は b と比較しているので int 型.
　d は b<c と同じ型なので bool 型.

e) if b<c then a else c+d.
　c+d は a と同じ型なので c, d は int 型.
　b は int 型である c と比較しているので int 型.

f) if b<c then d else e.
　b と c は比較しているので同じ型．ML の処理系ではこのようなときはデフォルトで int 型と推論される.
　d と e は同じ型.

g) if b<c then d+e else d*e.
　b と c は比較しているので同じ型．ML の処理系ではこのようなときはデフォルトで int 型と推論される.
　d と e は加算・乗算を行なっているので int 型もしくは real 型．ML の処理系ではこのようなときはデフォルトで int 型と推論される.

2　MLOO 問 3.3 の 2

問題: fn x => twice id x ただし，fun twice f x = f (f x); fun id x = x;
解答例:関数 twice の型が ('a → 'a) → 'a → 'a であることは問 3.4 で示したのでそれを使う．また関数 id の型が 'a → 'a であることも容易にわかるのでそれを用いる．

twice id x は twice の第一引数 ('a → 'a) に関数 id 'a → 'a を，第二引数'a に x を入れたものであるので x の型は'a.

twice id x の結果の型は'a.

よって fn x => twice id x の型は 'a → 'a.

3　MLOO 問 3.3 の 3

問題: fun thrice f x = f (f (f x)).
解答例:
thrice : τ_1, f : τ_2, x : τ_3, f x: τ_4, (f (f x)) : τ_5, f (f (f x)): τ_6 とする．すると

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_6$$

関数 f の引数がすべて同じ型でなければならないことから

$$\tau_3 = \tau_4 = \tau_5$$

関数 f の結果がすべて同じ型でなければならないことから

$$\tau_4 = \tau_5 = \tau_6$$

関数 f の引数と結果の関係から

$$\tau_2 = \tau_3 \to \tau_4$$

これを最初の式にいれると

$$\tau_1 = (\tau_3 \to \tau_4) \to \tau_3 \to \tau_6$$

$\tau_3, \tau_4, \tau_5, \tau_6$ はすべての同じ型であるからこれを τ_3 で表して

$$\tau_1 = (\tau_3 \to \tau_3) \to \tau_3 \to \tau_3$$

つまり ('a → 'a) → 'a → 'a.

4　演習問題 5.1 の 1

問題:S x y z = (x z) (y z) の型、
解答例:
S: τ_1, x : τ_2, y : τ_3, z: τ_4, (x z) : τ_5, (y z): τ_6, (x z) (y z) : τ_7 とする．すると

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_7$$

関数 x の形から

$$\tau_2 = \tau_4 \to \tau_5$$

関数 y の形から

$$\tau_3 = \tau_4 \to \tau_6$$

関数 (x z) の形から

$$\tau_5 = \tau_6 \to \tau_7$$

τ_2, τ_3 の式を最初の式に入れる.

$$\tau_1 = (\tau_4 \to \tau_5) \to (\tau_4 \to \tau_6) \to \tau_4 \to \tau_7$$

τ_5 の式をこれに入れる.

$$\tau_1 = (\tau_4 \to \tau_6 \to \tau_7) \to (\tau_4 \to \tau_6) \to \tau_4 \to \tau_7$$

これ以上簡単にならないので τ_4 : 'a, τ_6 : 'b, τ_7 : 'c とすると

`('a -> 'b -> 'c) -> ('a -> 'b) -> 'a -> 'c`.

非手続き型言語13回目課題 解答例

June 8, 2020

1　MLOO 問 3.5 の 3

問題: fun A x y z = z y x の型
　A: τ_1, x: τ_2, y : τ_3, z : τ_4, z y x : τ_5 とおく．求めたいのは

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_5$$

関数 z について

$$\tau_4 = \tau_3 \to \tau_2 \to \tau_5$$

τ_4 に関する式を τ_1 の中にいれて

$$\tau_1 = \tau_2 \to \tau_3 \to (\tau_3 \to \tau_2 \to \tau_5) \to \tau_5$$

これ以上簡単にならないので τ_2 を'a, τ_3 を'b, τ_5 を'c としてこの関数の型は

```
'a -> 'b -> ('b -> 'a -> 'c) -> 'c
```

2　MLOO 問 3.5 の 4

問題: fun B f g = f g g の型
　B: τ_1, f: τ_2, g : τ_3, f g g : τ_4 とおく．求めたいのは

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4$$

関数 f について

$$\tau_2 = \tau_3 \to \tau_3 \to \tau_4$$

τ_2 に関する式を τ_1 の式に代入して

$$\tau_1 = (\tau_3 \to \tau_3 \to \tau_4) \to \tau_3 \to \tau_4$$

これ以上簡単にならないので τ_3 を'a, τ_4 を'b としてこの関数の型は

```
('a -> 'a -> 'b) -> 'a -> 'b
```

3 MLOO 問 3.7 の 2,3,4,5

準備:

(i) fun f x y z = x y z
スライドで解説したように
f: τ_1, x : τ_2, y: τ_3, z: τ_4,x y z : τ_5,

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_5$$

関数 x より

$$\tau_2 = \tau_3 \to \tau_4 \to \tau_5$$

これらから

$$\tau_1 = (\tau_3 \to \tau_4 \to \tau_5) \to \tau_3 \to \tau_4 \to \tau_5$$

これの型は

```
('a -> 'b -> 'c) -> 'a -> 'b -> 'c
```

(ii) fun f x y z = x (y z) の型は
f: τ_1, x : τ_2, y: τ_3, z: τ_4, y z : τ_5, x (y z): τ_6,

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_6$$

関数 x より

$$\tau_2 = \tau_5 \to \tau_6$$

関数 y より

$$\tau_3 = \tau_4 \to \tau_5$$

これらから

$$\tau_1 = (\tau_5 \to \tau_6) \to (\tau_4 \to \tau_5) \to \tau_4 \to \tau_6$$

すなわち

```
('a -> 'b) -> ('c -> 'a) -> 'c -> 'b
```

2. fun f x y z = x (y z) : int
これは (ii) 型で x (y z) つまり τ_6 が int. よって

```
('a -> int) -> ('b -> 'a) -> 'b -> int
```

3. fun f x y z = (x y z) : int
これは (i) 型で x y z つまり τ_5 が int. よって

```
('a -> 'b -> int) -> 'a -> 'b -> int
```

4. fun f x y z = x y (z: int)
   これは (i) 型で z つまりが τ_4 が int. よって

```
('a -> int -> 'b) -> 'a -> int -> 'b
```

5. fun f x y z = x (y z : int)
   これは (ii) 型で (y z) つまり τ_5 が int. よって

```
(int -> 'a) -> ('b -> int) -> 'b -> 'a
```

4 MLOO 問 3.11 の 1,2

1. fn x => x > 1
   x : τ_1, x > 1 : τ_2 とすると求めたいのは $\tau_1 \to \tau_2$. x は 1 と比較されているので τ_1 は int 型. また x > 1 は bool 型なので τ_2 は bool 型. よって
   int → bool.

2. fn x => fn y => fn z => (x y, x "Ada", y > z)
   x: τ_1, y: τ_2, z: τ_3, x y: τ_4, (x y, x "Ada", y > z) : τ_5 とすると求めたいのは $\tau_1 \to \tau_2 \to \tau_3 \to \tau_5$.
   まず τ_5 は組で τ_4 * τ_4 * bool. 関数 x の引数の型が同じということから τ_2 = string. また y と z で比較しているということから τ_3 = string. 関数 x について $\tau_1 = string \to \tau_4$.
   これらをすべてもとの式にいれると求める型は
   $(string \to \tau_4) \to string \to string \to \tau_4 * \tau_4 * bool$. よって

```
(string -> 'a) -> string -> string -> 'a * 'a * bool.
```

5 演習問題 5.6.4

```
fun comp F G =
  let
    fun C x =G(F(x))
  in
    C
  end;

fun add1 x = x + 1;
```

ここで関数 comp の型は

```
val comp = fn : ('a -> 'b) -> ('b -> 'c) -> 'a -> 'c
```

関数 add1 の型は

```
val add1 = fn : int -> int
```

a) val compA1 = comp add1;

関数. comp の第一引数が int → int なので, 'a と 'b が int. よって
(int→'a) → int → 'a
　注) この状態では処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

b) val compCompA1 = comp compA1;

compA1 が (int → 'a) → int → 'a
ということは、comp における
'a が int → 'a
'b も int → 'a
だから、答えは
((int → 'a) → 'b) → (int → 'a) → 'b
　注) この状態も処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

c) val f = compA1 add1;

compA1 が (int → 'a) → int → 'a add1 が int → int だから int → int
　内容としては f(x) は add1(x+1) つまり x + 2
　注)

```
- val compA1 : (int -> int) -> int -> int = comp add1;
- val f = compA1 add1;
```

とすれば処理系で計算できる.

5.1 d) f(2);

答えは 4.

e) val g = compCompA1 compA1;

compCompA1 の型: ((int→'a)→'b) → (int → 'a) → 'b
compA1 の型: (int→'a) → int → 'a
つまり compCompA1 の 'b は int → 'a.
よって関数 g の型は (int → 'a) → int → 'a
　注) この状態では処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

f) val h = g add1;

型は int→int. この関数は引数に対し x+3 を行う.
　注)

```
- val compA1 : (int -> int) -> int -> int = comp add1;
- val compCompA1 : ((int -> int) -> int -> int) -> (int -> int) -> int -> int
 = comp compA1;
- val g : (int -> int) -> int -> int = compCompA1 compA1;
- val h = g add1;
```

とすれば処理系で計算できる.

g) h(2);

答えは 5.