プログラミング言語 No.2-3

2025.4.25 中川博之

[ミニレポート解答例]　問題 2-2-3)
n個中m個を選ぶ組み合わせの数を返す関数comb

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$

基底は　m = 0 のときと　n = m のとき．どちらも組み合わせの個数は1

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

[ミニレポート解答例]　問題 2-2-4)
二つの自然数 a, b の最大公約数を求める関数mygcd

(ユークリッドの互除法）

1. 基底
   b = 0 ならば a .

2. 帰納段階
   a を b で割った余りを b とし，元のbをa として再帰する.

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

[ミニレポート解答例] 問題 2-2-5)

リストの奇数番目と偶数番目を入れ替える関数

- 基底は2つ.

解答例：

```
fun swapodev [] = []
| swapodev [x] = [x]
| swapodev (x::y::xs) = y::x:: swapodev xs;
```

[ミニレポート解答例] 問題 2-2-6)

a) xがSに含まれているとき true を返す関数member (x, S)

```
fun member (_, []) = false
| member (x, y::ys) = if x = y then true
                      else member(x, ys);
```

定義時に Warning: calling polyEqual
という警告がでるが，とりあえずこれは無視してよい

[ミニレポート解答例] 問題 2-2-6)

b) xをSから削除する関数 delete (x, S)

```
fun delete (_, []) = []
| delete (x, y::ys) = if x = y then ys
                      else y::delete(x, ys);
```

定義時に Warning: calling polyEqual
という警告がでるが，とりあえずこれは無視してよい

[ミニレポート解答例] 問題 2-2-6)

c) xをSに追加する関数 insert (x, S)

```
fun insert (x, []) = [x]
| insert (x, S) = if member (x, S) then S
                  else x::S;
```

リストのリスト

問題 2-3-1)

- 任意の要素 a と，a と同じ型の要素からなるリストのリストLを取り，L中の各リストの最初に a を挿入する関数inserteachを書け.
- たとえば，a = 1 で L = [[2,3], [4,5,6], []] であれば，結果は [[1,2,3], [1,4,5,6], [1]]となる.

問題 2-3-1)

- 任意の要素 a と，a と同じ型の要素からなるリストのリストLを取り，L中の各リストの最初に a を挿入する関数inserteachを書け.
- たとえば，a = 1 で L = [[2,3], [4,5,6], []] であれば，結果は [[1,2,3], [1,4,5,6], [1]]となる.

解答例：

```
fun inserteach (_, []) = []
| inserteach (a, x::xs) = (a::x)::inserteach(a, xs);
```

x: 現在扱っているリスト，xs: 残り (未対応) のリスト

問題 2-3-2)

- 集合Sのべき集合を計算する関数 powerset を書け.
- S = [1,2] の場合，結果は [[], [1], [2], [1,2]] となる.
- 問題2-3-1で作成した関数を利用する.

問題 2-3-2)

powerset([1,2])
→powerset([2])@inserteach(1, powerset([2])
→(powerset([])@inserteach(2,[])@inserteach(1, (powerset[]@inserteach(2,[]))
→([[]]@[[2]])@inserteach(1, ([[]]@[[2]]))
→[[],[2]]@inserteach(1, [[],[2]])
→[[],[2]]@[[1],[1,2]]
→[[],[2],[1],[1,2]]

- 集合Sのべき集合を計算する関数 powerset を書け.
- S = [1,2] の場合，結果は [[], [1], [2], [1,2]] となる.
- 問題2-3-1で作成した関数を利用する.

解答例：

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

局所環境

局所環境

- 一時的な変数や関数を作りたいときに使う
  - そこでしか使わない補助関数を定義する場合
  - 一度呼び出した関数の結果を何度も使いたい場合
- let と in の間に定義した変数や関数を式の中で使うことができる

```
let
    val または fun の宣言の列
 in
    式
end
```

局所環境の例

```
- fun test () =
  let
    val a = 1;
    val b = 2
  in
    a + b
  end;

- test ();
- val it = 3: int
```

- let と in の間は宣言であり, 代入ではない
- 局所環境を使わないと書けないものはないが, 局所環境を使うと簡潔だったり, 効率の良いプログラムが書けたりすることがある

局所環境　例題1 (問題 2-3-3)

- 実数xの100乗を計算する関数 hundredthPower
- 実数xのi乗を計算する関数のiを100にして再帰的に実行しても良いが…
- 一般的に再帰呼び出しが多くなると遅くなる
- 局所環境を用いて再帰呼び出しを使わないで計算してみよう

```
fun hundredthPower (x:real) =
   let
      val  four = x * x* x* x;
      val  twenty = four * four * four * four * four
   in
       twenty * twenty * twenty * twenty * twenty
   end;
```

局所環境　例題2 (問題 2-3-4)

- リストを奇数番目に現れる要素だけからなるリストと偶数番目に現れる要素だけからなるリストに2分割してその対(pair)を値とする関数 split.
- 引数が [1,2,3,4,5] ならば，結果は ([1,3,5], [2,4]) .

対 (pair) とは
任意の型の式を二つ以上並べたものを組(tuple)という.
( ) でくくり，区切りは ,
例：(1, 2)
　(1.0, ”abc”, ”def”)
　(1, (#”a”, 4, 5.0), [true, false])
組のうち，要素が2個だけのものを対(pair)とも呼ぶ

局所環境　例題2 (問題 2-3-4)

- リストを奇数番目に現れる要素だけからなるリストと偶数番目に現れる要素だけからなるリストに2分割してその対(pair)を値とする関数 split.
- 引数が [1,2,3,4,5] ならば，結果は ([1,3,5], [2,4]) .

```
fun split ([]) = ([], [])
|    split ([a]) = ([a], [])
|    split (a::b::cs) =
       let
              val (M, N) = split(cs)
       in
              (a::M, b::N)
       end;
```

局所環境　例題3 (問題 2-3-5)

- 実数のリストの最大値を求める.
- 局所環境を使わない解答例

```
fun maxnumber ([x:real]) = x
|     maxnumber (x::xs) =
      if x >  maxnumber(xs) then x
      else maxnumber(xs);
```

maxnumber(xs) を2回も呼び出していて無駄.
ここは再帰呼び出しになるのでとてもコストが高い.
一度で済ませたい.

局所環境　例題3 (問題 2-3-5)

- 実数のリストの最大値を求める.
- 局所環境を使った解答例

```
fun maxnumber2 ([x:real]) = x
|    maxnumber2 (x::xs) =
    let
        val m = maxnumber2(xs)
    in
        if x >  m then x else m
    end;
```

一度で済む.

局所環境まとめ

- 局所環境は一時的な変数や関数を作りたい時に使う
- let と in の間で定義した変数や関数は in と end の間でだけ有効
- 局所環境を使わなければ書けない関数というものはないが，次のような利点があるので非常によく使われる．
  - 局所環境を使うことである変数や関数がその環境内でのみ使用するということが明確になりプログラムがわかりやすくなる
  - 局所環境を使うことで効率の良いプログラムが書ける

問題 2-3-5)

「ピッグ・ラテン(Pig Latin)」へ翻訳する簡単な規則は，母音で始まる単語には”yay”を付加し，１つ以上の子音で始まる単語にはその子音を後ろに回してから“ay” を追加する．例えば，“able” は“ableyay” になり，“stripe” は“ipestray” となる．文字列をピッグ・ラテンに変換する関数を書け．

- 母音の無い単語は考慮しなくて良い

各関数をMLで定義せよ（ミニレポート）

問題 2-3-5) を行うために

- 文字のリストを取り，最初の要素が母音であったらtrueを，そうでなければfalseを返す関数 vowel
  - SML/NJ では文字は　#”a” のように表す
- 以下の関数も役に立つかもしれない
  - - explode("abc");
    val it = [#"a",#"b",#"c"] : char list
  - - implode([#"a",#"b",#"c"]);
    val it = "abc" : string

```
fun  vowel(#"a"::_) = true
 |   vowel(#"e"::_) = true
 |   vowel(#"i"::_) = true
 |   vowel(#"o"::_) = true
 |   vowel(#"u"::_) = true
 |   vowel(#"A"::_) = true
 |   vowel(#"E"::_) = true
 |   vowel(#"I"::_) = true
 |   vowel(#"O"::_) = true
 |   vowel(#"U"::_) = true
 |   vowel(_) = false;
```

問題 2-3-6)

べき集合を計算する関数 powerset を, let を使い尾部(tail)のべき集合の計算を一度に済ませるように修正せよ.

各関数をMLで定義せよ（ミニレポート）

問題 2-3-7)

整数のリストを取り，その偶数番目の数の和と奇数番目の数の和の対(pair) を返す関数pairofSumsを書け．ただし，補助関数を使ってはいけない.

各関数をMLで定義せよ（ミニレポート）

問題 2-3-8)

リストL とM を連結する関数cat(L,M) を書け．ただし，@演算子を使ってはいけない．cons演算子 (::)を使うこと．この関数はリストL の長さに比例した時間で動作し，リストM の長さには依存しないこと．

各関数をMLで定義せよ（ミニレポート）

本日の演習およびミニレポート

1. 問題 2-3-5)
2. 問題 2-3-6)
3. 問題 2-3-7)
4. 問題 2-3-8)

- 局所関数を使えるものは利用を検討すると良い
- じっくり考えてもわからない場合は，正しく動かなくても良いので，考えたプログラムとなぜ上手く動かないのか，どうすればうまく動きそうかを考察して提出すること
- 締切：次回講義前日の23:59