多重順列とその応用について

清水洋輔

目的

この研究は，リストを用いて様々な多重順列生成のプログラムを作成し、更にそのプログラムを基に、ローマ字多重順列生成のプログラムを作成することを目的とする。

リストとは，数や文字列をコンマで区切って並べ、その両端を大括弧[]ではさんだものである。Prologではリストを使って、任意の長さのデータ列を表すことができる。

・[jan,31,1957]
・[X,11,[R,5]]
・[]

最初の例は、３つの要素jan,31,1957からなるリスト。２番目の例も３つの要素からなるリスト。３番目の要素がリストになっている。最後の例は、要素のない空リストである。

リストの一般形は[要素１、要素２、，，，要素ｎ]である。要素としては、任意の項が使える。要素の数ｎは、リストの長さであり、長さ０のリストは空リストと呼ばれている。

プログラム

```
:-dynamic sh/1.
shi(Y,_):-
        ctr_set(0),permutation(X,Y),
        not0(sh(X)),asserta(sh(X)),ctr_inc(CC),
        write(cc=CC),
        write(X),nl,
        fail.
        shi(Y,C):- ctr_is(C).
```

出力結果例

```
?- exectime(shi([d,f,g,f,f,y],P),T).
cc=0[d, f, g, f, f, y] cc=1[f, d, g, f, f, y]
cc=2[g, d, f, f, f, y] cc=3[f, d, f, g, f, y]
cc=4[y, d, f, g, f, f] cc=5[d, g, f, f, f, y]…
……
cc=117[g, f, f, y, f, d] cc=118[f, f, g, y, f, d]
cc=119[f, f, f, y, g, d].

P = 120 C= 1.422
```

```
fuu(L,N):- len0(L,A),sum1(A,A1),fuu_aux(L,N,A1,1).
fuu_aux([],N,A,B):- N is A//B.
fuu_aux(L,N,A,B):- L=[X|L1],ge_list(L,X,M,K),len0(M,B1),
sum1(B1,S),B2 is B*S,fuu_aux(K,N,A,B2).
```

出力結果例

```
?- fuu([a,a,b,b,c,c,d,d,e,e,f,f,g,g,h,h,…,z,z],P).
P = 90641061072687441240586662787392046583824281600000000
    00000
T = 0.471
```

多重順列の公式

n 個の物のうち、p 個、q 個、…、r 個がそれぞれ同じであるとき、これらを１列に並べる順列の数は

$$\mu \binom{n}{p, q, \cdots, r} \P = \frac{n!}{p!, q!, \cdots, r!} \quad , p + q + \cdots + r = n$$

ローマ字プログラム

```
:- dynamic ro/1.

roma(L,_):-
abolish(ro/1),asserta(ro(0)),ctr_set(0),part(L=A+B),
shi(A,A1),shi(B,B1),part(Y=A1+B1),not0(ro(Y)),
asserta(ro(Y)),write(Y),ctr_inc(_),nl,fail.
roma(L,C):- ctr_is(C).
```

ローマ字多重順列

リストで入力した文字からできる、ローマ字の文字列の組合せを全て出力するプログラム。組合せ数を求めることも出来る。ただし、yi やwe,kye などの文字は考えないことにする。また、プログラムの構成上、リストの長さを偶数にしなければ正しいローマ字ではない為、母音のみの要素の前にはX を付けることを前に述べておく。実際に例を出して説明する。

EX

[x,i,x,i,t,a,k,a]

まずリストの要素を奇数番目の要素(子音)のリストと、偶数番目の要素(母音)のリストに分ける。

奇数番目＝[x,x,t,k]

偶数番目＝[i,i,a,a]

(この作業はpartのプログラムで行った。) 次に、それぞれのリストに多重順列のプログラムを使い全ての組合せを出す。

[x, x, t, k]
[t, x, x, k]
[k, x, x, t]
[x, t, x, k]
[x, k, x, t]
[t, k, x, x]
[k, t, x, x]
[x, x, k, t]
[t, x, k, x]
[k, x, t, x]
[x, t, k, x]
[x, k, t, x]

×

[i, i, a, a]
[a, i, i, a]
[i, a, i, a]
[a, a, i, i]
[a, i, a, i]
[i, a, a, i]

最後に、それぞれの組合せを結合してローマ字列の組合せを出力する。多重順列のプログラムを使っても、重複してしまう部分については、プログラム全体にassertaを用いて重複を取り除いた。

Partのプログラム

```
part([]=[]+[]).
part([A,B|L]=[A|P]+[B|E]):-
part(L=P+E).
?- part([s,i,m,i,z,u]=A+B).
A = [s, m, z]
B = [i, i, u]

?- part(L=[1,3,5]+[2,4,6]).
L = [1, 2, 3, 4, 5, 6]
```

一つのプログラムで分解、結合の二つのことが出来る。

Prologならではの素晴らしいプログラム。

考察

この研究では、リストを使用する機会が非常に多かった。プログラムの必要性に応じて、リストの形を変化させたり、必要な要素を取り除いたりした。リストを上手く活用することでプログラムの効率の向上や、自由に応用が効くようになることが研究の過程で解ってきた。Prologという言語の中で、リストは非常に大切な要素であることが理解できた。