Perm: Permutation Manipulator の使用法

1. 対象機種と起動法

本プログラムは Linux OS で動作する。(gcc, g++, bison, flex を使用)
64 ビット計算機を想定しているが、コンパイルオプション設定により
32 ビット機にも対応可能。

プログラムの起動方法は、
Perm [-最大πDD 節点数] [ファイル名]
である。

起動時にファイル名を指定するとファイルに書かれた命令を実行する。ファイル名を省くと標準入力から読み込む。計算結果は標準出力に出力される。端末入出力によるインタプリタ形式の利用法と、ファイル入出力によるフィルタ形式の利用法の両方が可能である。

本プログラムでは、内部で生成するπDD データ(約 30 byte/node)が主記憶をあふれると処理効率が急激に低下するため、マシンの主記憶サイズに応じて最大 ZDD 節点数を指定し、その範囲内で記憶管理を行う。(省略値は 400000。)
記憶領域は、起動時に最小限確保され、必要に応じて徐々に限界まで拡張される。計算中に指定した最大ノード数に達した場合は計算を打ちきり、警告メッセージを出力する (計算結果は 0 となる)。

(例)
perm　　インタプリタモードで起動
perm <script>　　ファイル名 <script> に書かれた命令を実行
perm -100000　　最大 ZDD 節点数 100000 で起動

2. 変数名

本プログラムでは、順列の位置を表す「位置記号」と、計算結果を一時的に保持する記憶場所を表す「プログラム変数」を使用する。位置記号は英小文字または数字で始まる英数字列、プログラム変数は英大文字で始まる英数字列で表す。
英数字列は最大 255 文字までで、２文字目以降にアンダーバーを含んでもよい。
位置記号は最大 254 個まで使用できる。プログラム変数の個数に制約はない。

空集合は 0, 無置換順列のみからなる集合を 1 と表す。
0 と 1 は位置記号として用いても良い。集合表現とは文脈で区別される。

コマンドの中でファイル名を指定する場合、および入力した文字列をエコーバックさせるときには、両端を引用符「″」で囲む。

（例）

位置記号	a b 1 25 bdd a5A f_t6
プログラム変数	A B X1 X2 BDD A5a F_t6
ファイル名	″a″ ″b″ ″script1.bem″

3. 命令の構成

本プログラムは、基本的に行単位（１行１命令）で動作する。１行に複数個の命令を書く場合は、セミコロン ; で区切る。複数行にわたって１命令を書く場合は、改行の直前にバックスラッシュ「」を置く。文中にシャープ「#」を書くと、次の改行までコメントとして読み飛ばされる。

プログラムの制御に関する命令としては、次の３つがある。

source ファイル名	ファイルに書かれた命令を呼び出して実行する。
help または ?	使用法を表示する。
quit または exit	プログラムを終了する。ファイルの終り(EOF)でも終了。

Perm の計算を実行する命令は、次の３種からなる。

・宣言文 --- 使用する位置記号の名前と順序を宣言する。
・代入文 --- 計算結果をプログラム変数に代入する。
・出力文 --- 計算結果を種々の形式で表示する。

4. 宣言文

宣言文は、使用する位置記号の名前と順序をあらかじめ宣言するものである。

symbol [位置記号名 [, 位置記号名, ...]]

区切りのコンマは空白でもよい。変数の順序は、後から宣言した記号が上位（右側）に配置され、πDD で先に展開される。位置記号名を１つも書かなかった場合は、現在使用中の位置記号が一覧表示される。

宣言文は複数回に分けて実行してもよい。同じ変数を２度宣言した場合は、警告が出て２度目の宣言は無視される。

（例）
symbol a, b, c
symbol b, d, e
とすると、ａｂｃｄｅの順になる。

宣言していない位置記号が順列集合代数式の中で使われた場合は、その場で新たに宣言したものとして、最上位に追加される（警告メッセージが出る）。
（例）
symbol a, b, c
print [b a d c]
とすると、ａｂｃｄの順になる。

5. 代入文
代入文は、右辺の順列集合代数式を計算し、得られた順列集合を左辺のプログラム変数に代入するものである。

プログラム変数名 = 順列集合代数式

プログラム変数名は、あらかじめ宣言する必要はない。代入文の左辺に初めて現れた時点で、記憶領域が確保される。プログラム変数の使用個数に制限はない。同じプログラム変数に重ねて代入すると、以前の内容が消去された後に、新しい値が代入される。位置記号を左辺に置くことはできない。また、代入されたことのないプログラム変数を右辺で参照することはできない。

右辺の順列集合代数式の文法は、C 言語に準拠している。使用できる演算子を、実行優先順に挙げる。

(式)
(位置記号:位置記号)
[位置記号 位置記号 … 位置記号]
式.Cofact(位置記号,位置記号)
式 式

式 * 式
式 & 式
式 + 式
式 - 式

(位置記号:位置記号) は、2 つの位置記号だけを交換しそれ以外は移動しないという順列 1 個からなる順列集合を表す。

[位置記号 位置記号 位置記号 …] は、位置記号の移動先を列挙して表現した順列 1 個からなる順列集合を表す。

式.Cofact(位置記号 1, 位置記号 2) は、式が表す集合に含まれる順列のうち、位置記号 1 が位置記号 2 に移動するような順列のみを部分集合として取り出し、さらに位置記号 1 と位置記号 2 を交換した順列集合を返す。

式 式と、式*式は同じ意味で、２つの順列集合の直積集合を返す。(２つの集合に属する順列同士すべての組合せの合成演算を行った順列からなる集合)

式&式は積集合 (intersection)、式+式は和集合(union)、式-式は差集合(difference set)を計算する。

6. 出力文
出力文の形式は次の通りである。

print [/スイッチ] 算術論理式
print "文字列"

print は 「?」で代用できる。スイッチは出力形式を指定するもので、
省略した場合は、関数に応じて適当に見やすい形式が選択される。
引用符「"」で囲んだ文字列はそのままエコーバックされる。

以下に、現在使用できるスイッチとその出力形式を説明する。

(スイッチなし)
　　位置記号の行き先を並べる形式で集合に含まれる全順列を列挙する。
/enum2　　2 変数交換の積の形式で集合に含まれる全順列を列挙する。

/size　　計算結果のπDD 節点数（および処理系全体の節点数）を表示。
/count　　集合に含まれる順列の個数を表示。

7. 実行例

【インタプリタモード】

```
%
***** Permutation Manipulator (v0.1) *****
perm> symbol a b c d e
perm> F = [b a d e c]
perm> print F
 [b a d e c]
perm> F = F + F (b:e)
perm> print F
 [b a d e c] + [e a d b c]
perm> F = F (1 + F)
perm> print F
 [- - e c d] + [- e b c d] + [c e b a d] + [c - e a d] + [b a d e c]
  + [e a d b c]
perm> print /enum2 F
 (d:c)(e:d) + (c:b)(d:c)(e:d) + (b:a)(c:b)(d:a)(e:d) + (c:a)(d:a)(e:d)
  + (b:a)(d:c)(e:c) + (b:a)(c:b)(d:b)(e:c)
perm> print /size F
 10 (10)
perm> print /count F
 6
perm> exit
%
```

以上。