整数値組合せ集合計算プログラム VSOP 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である。

プログラムの起動方法は、
vsop [-最大 BDD 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。

VSOP では、内部で生成する BDD(約 30 バイト/節点)が主記憶をあふれて２次記憶が使われ出すと処理速度が急激に(100 倍以上) 低下するため、マシンの主記憶サイズに応じて最大 BDD 節点数を指定し、その範囲内で実行する。(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節点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)

vsop	インタプリタモードで起動
vsop script	ファイル名 script に書かれた命令を実行
vsop -100000	最大 BDD 節点数 100000 で起動

2. 変数名・数値

VSOP では、多項式のアイテム変数を表す「シンボル変数」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。シンボル変数は英小文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。シンボル変数は最大 65510 個まで使用できる。プログラム変数の個数に制約はない。

数値は 10 進数で表現する。数値の範囲は正負 100 万ビットまで原理的には扱えるが、100 ビットを越えるような大きな数を扱うと計算時間が増大する。数式中の係数・定数は整数値のみに限られ、小数は使えない。なお symbol 文の中でシンボル変数の値(value)を定義するときに限り小数が使える。

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¥verb1″1 で囲む。

（例）

シンボル変数	a b x1 x2 bdd a5A f_t6
プログラム変数	A B X1 X2 BDD A5a F_t6
数値	0 29 32767 -213
ファイル名・文字列	″a″ ″b″ ″script1.bem″

3. 命令の構成

VSOP 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。

VSOP の計算を実行する命令は、次の３種からなる。

・宣言文 --- 使用するシンボル変数の名前とその数値および展開順序を宣言する。
・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、使用するシンボル変数の名前と順序をあらかじめ宣言するものである。

symbol [シンボル変数名 [, シンボル変数名, ...]]

区切りのコンマは空白でもよい。変数の順序は、左から順に上位（先に展開される）に配置される。シンボル変数名を１つも書かなかった場合は、現在使用中のシンボル変数が一覧表示される。

また、VSOP では、シンボル変数に値(value)を指定することができる。

symbol [シンボル変数名(代入値) [, シンボル変数名(代入値), ...]]

上記の値(value)とは、print 文の /value, /mincover, /mincost スイッチのときに使用するもので、各変数に指定した数値を代入したときの式全体の数値を計算したり、コスト最小となる組合せをみつけたりできる。VSOP では扱う数値は整数に限られているが、シンボル変数の値(value)に限り小数を使用できる。ただし実装の都合上、固定小数点(整数部 16bit+小数部 16bit)型数値として計算され、その範囲を超えると桁あふれ・桁落ちが発生する。値の指定を省略した場合は、0.5 が設定される。

宣言文は複数回に分けて実行してもよいが、その場合、あとに宣言されたシンボル変数が下位に配置される。同じシンボル変数を２度宣言した場合、変数順序は変わらないが、値の指定は後の宣言の方が有効となる。
（例）
```
symbol a, b, c
symbol b, d, e
```
とすると、ａｂｃｄｅの順になる。

宣言していないシンボル変数が算術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 a b + c d
```
とすると、ｄａｂｃ の順になる。

5. 代入文

代入文は、右辺の算術式を計算し、得られた整数値組合せ集合の式を

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 = 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。一旦代入されたプログラム変数は、以後、右辺の式の中で参照することができる。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。シンボル変数は右辺で参照することはできるが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。

右辺の式で使用できる演算子は以下の通りである。演算子の優先順位の高い順に記述している。

（ 式 ）	括弧内を最優先で計算。
式.MaxVal	式に含まれる整数値の最大値
式.MinVal	式に含まれる整数値の最小値
式.Items	式に表れるアイテム変数を列挙する。
式.Restrict(式)	第１式に含まれる組合せの中から、第２式中の少なくとも１つの組合せを包含する要素だけを抽出する。
式.Permit(式)	第１式に含まれる組合せの中から、第２式中の少なくとも１つの組合せに包含される要素だけを抽出する。
式 式	乗算
式 * 式	乗算
式 / 式	除算（整数除算の商）。
式 % 式	剰余（整数除算の剰余）。
式 + 式	加算。
式 - 式	減算。
+ 式	正の符号（何もしない）。
- 式	補数計算。
式 == 式	比較演算（少なくとも一方の式に含まれる組合せで、その整数値が等しいものだけを取り出す）。
式 != 式	比較演算（少なくとも一方の式に含まれる組合せで、その整数値が等しくないものだけを取り出す）。
式 > 式	比較演算（少なくとも一方の式に含まれる組合せで、

その整数値が条件を満たすものだけを取り出す)。

式 >= 式　　比較演算（少なくとも一方の式に含まれる組合せで、
その整数値が条件を満たすものだけを取り出す)。

式 < 式　　比較演算（少なくとも一方の式に含まれる組合せで、
その整数値が条件を満たすものだけを取り出す)。

式 <= 式　　比較演算（少なくとも一方の式に含まれる組合せで、
その整数値が条件を満たすものだけを取り出す)。

式 ? 式 : 式　　If-Then-Else 演算。第１式に含まれる組合せに対しては
第２式の値をそのまま返し、第１式に含まれない組合せ
に対しては第３式の値をそのまま返す演算。

6. 出力文

出力文の形式は次の通りである。

print [/スイッチ] 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、括弧を含まない積和形に展開した式を表示する。
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、使用できるスイッチとその使用例を示す。

（スイッチ無し）式のカッコを展開した積和形で表示する。
/bit　　内部の２進数の各桁の組合せを順に表示。
/hex　　整数値を 16 進数で表現する積和形表示。
/map　　カルノー図で表示。アイテム変数６個まで表示できる。
/rmap　　カルノー図で表示。冗長なアイテム変数は省いて表示。
/case　　整数値ごとに場合分けして積和形表示。
/size　　計算結果のＢＤＤ節点数（および処理系全体の節点数）を表示。
/count　　式に現れる(0 以外の値を持つ）組合せの個数を表示。
/density　　集合の濃度（0 以外の値を持つ組合せの比率）を表示。
/value　　シンボル変数にすべて数値を代入したときの式の値を表示
/maxcover　　式に含まれる（0 以外の値を持つ）コスト最大の組合せを１つ表示。
/maxcost　　コスト最大組合せのコスト値を表示。

/mincover　式に含まれる（0 以外の値を持つ）コスト最小の組合せを１つ表示。
/mincost　コスト最小組合せのコスト値を表示。
/plot　ＢＤＤの形を図示する。
/plot0　ＢＤＤの形を図示する（否定枝不使用）

7. 実行例

【インタプリタモード】

```
***** VSOP (Valued Sum-Of-Products) calculator - ver 0.7 *****
 vsop> symbol a b c d e
 vsop> F = (a + 2 b)(c + d)
 vsop> print F
  a c + a d + 2 b c + 2 b d
 vsop> print /rmap F
  a b : c d
      |    00    01    11    10
   00 |     0     0     0     0
   01 |     0     2     0     2
   11 |     0     0     0     0
   10 |     0     1     0     1
 vsop> G = (2 a - d)(c - e)
 vsop> print G
  2 a c - 2 a e - c d + d e
 vsop> print /rmap G
  a c : d e
      |    00    01    11    10
   00 |     0     0     1     0
   01 |     0     0     0    -1
   11 |     2     0     0     0
   10 |     0    -2     0     0
 vsop> H = F * G
 vsop> print H
  4 a b c d - 4 a b c e + 4 a b c - 4 a b d e + a c d e - 2 a c e + 2 a
   c - a d e + 2 b c d e - 4 b c d + 2 b d e
 vsop> print /rmap H
  a b : c d e
```

```
     |   000   001   011   010 |   110   111   101   100
  00 |     0     0     0     0 |     0     0     0     0
  01 |     0     0     2     0 |    -4     2     0     0
  11 |     0     0    -4     0 |     4     0    -4     4
  10 |     0     0    -1     0 |     0     1    -2     2
vsop> print /count H
   11
vsop> print /size H
   24 (35)
vsop> print H > 0
 a b c d + a b c + a c d e + a c + b c d e + b d e
vsop> print (H > 0)? H: 0
 4 a b c d + 4 a b c + a c d e + 2 a c + 2 b c d e + 2 b d e
vsop> print (H > 0)? H: -H
 4 a b c d + 4 a b c e + 4 a b c + 4 a b d e + a c d e + 2 a c e + 2 a
  c + a d e + 2 b c d e + 4 b c d + 2 b d e
vsop> print H / (a + b)
 c d e - d e
vsop> print H / (a + b) * (a + b)
 a c d e - a d e + b c d e - b d e
vsop> print H % (a + b)
 4 a b c d - 4 a b c e + 4 a b c - 4 a b d e - 2 a c e + 2 a c + b c d
  e - 4 b c d + 3 b d e
vsop> print H / (c e)
 - 4 a b + a d - 2 a + 2 b d
vsop> quit
```