

## SSQ – manipulator for Sets of SeQuences 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)

32 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

64 ビット機にも対応可能。

プログラムの起動方法は、

SSQ [-最大 SeqBDD 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 SeqBDD 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 SeqBDD 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）

ssq                    インタプリタモードで起動

ssq <script>        ファイル名 <script> に書かれた命令を実行

ssq -100000        最大 ZDD 節点数 100000 で起動

## 2. 変数名

本プログラムでは、アイテム名を表す「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 65500 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, 空列のみからなる集合を 1 と表す。

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）

シンボル変数     a   b   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。

SSQ 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 [ 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。シンボル変数の順序は、先に宣言したシンボル変数が辞書順の上位になり、先に展開される。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。

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、a b c d eの順になる。

宣言していないシンボル変数が計算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、最下位に追加される（警告メッセージが出る）。

(例)

```
symbol a, b, c
```

```
print a d + b c
```

とすると、a b c d の順になる。

## 5. 代入文

代入文は、右辺の計算式を計算し、得られた系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 = 計算式

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。シンボル変数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。

右辺の計算式の文法は、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)

式 式

式 / 式

式 % 式

式 & 式

式 + 式

式 - 式

式 式は、2つの系列集合の直積集合を返す。(2つの式から任意の系列を1つずつ取りだし、1つ目の式から取り出した系列の末尾に2つ目の式から取り出した系列を接続することで得られる系列をすべて集めた集合を返す。)

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) 集合に含まれる系列を辞書順に列挙する。

/size            計算結果の SeqBDD 節点数 (および処理系全体の節点数) を表示。

/card            集合に含まれる系列の個数を表示。

/lit             集合に含まれる系列の文字数の総和を表示。

/depth          計算結果の SeqBDD の深さ (最長経路長) を表示。

/export          計算結果の SeqBDD の構造情報を書き出す。

/export "<filename>"

計算結果の SeqBDD の構造情報をファイルに書き出す。

## 7. 実行例

### 【インタプリタモード】

```
% ssq
```

```
***** SSQ - manipulator for Sets of Sequences (v0.1) *****
```

```
ssq> symbol a b c d
```

```
ssq> F = (a b + c) (b d + 1)
```

```
ssq> print F
```

```
a b + a b b d + c + c b d
```

```
ssq> print /card F
```

```
4
```

```
ssq> print /lit F
```

10

```
ssq> print /size F
```

5 (5)

```
ssq> G = (a b + d) (b d + a c)
```

```
ssq> print G
```

$a b a c + a b b d + d a c + d b d$

```
ssq> print F + G
```

$a b + a b a c + a b b d + c + c b d + d a c + d b d$

```
ssq> print F & G
```

$a b b d$

```
ssq> print F - G
```

$a b + c + c b d$

```
ssq> print G / a b
```

$a c + b d$

```
ssq> print G % a b
```

$d a c + d b d$

```
ssq> print G / (a b + d)
```

$a c + b d$

```
ssq> print G % (a b + d)
```

0

```
ssq> quit
```



%

以上