ブール関数の数式処理システム*

吉田雄二** 福村晃夫**

Abstract

We have developed the Formula Manipulation System for Boolean functions (BALOC-2). This system is different from existing formula manipulation systems which are intended for the use of polynomials, rational functions or analytical functions. This system uses many features of computers in binary operations and so Boolean functions are processed with high efficiency and speed. The Boolean functions are expressed in the canonical forms of ring sum, which are uniquely defined for the functions, and stored in memory partitioned into groups of several words (we call this group the 'page').

BALOC-2 system consists of BALOC-2 language, compiler and utility routines for execution. All of those are based upon FORTRAN and a few bitwise operations for words are implemented by assembler language.

1. まえがき

ブール関数の数式処理は，整数計画法の解法，論理回路の設計，命題論理の問題などにおいて重要な部分を占めていてその内容が通常の数式処理におとらず煩雑であるため，電子計算機を用いた自動化が強く望まれる．しかしながら，従来の数式処理システムは，対象となる数式が多項式，有理式，あるいはそれらを含む一般の数式（ブール関数は除く）がほとんどで，上記の目的のためには不十分である．また，ブール変数およびブール関数は2値しかとらないこと，演算の種類が限られていて，比較的単純であるうえに，それらの多くが，電子計算機の語内ビット処理と対応づけられることなどから，ブール関数の数式処理が，電子計算機によってかなり能率的にシステム化されることが期待される．

本論文では，このような観点から構成されたブール関数の数式処理システム（**BALOC-2**, **B**oolean **AL**gorithm **O**riented **C**ompiler-2）について論ずる．

このシステムは，（1）ブール関数の数式処理のための，FORTRAN を拡張した言語，（2）この言語で書かれたプログラムを FORTRAN のプログラムに翻訳するための翻訳プログラム，（3）翻訳されてできた FORTRAN プログラムの実行に必要な基本的な数式処理の演算を行なうための種々のユーティリティルーチンの3つから構成される．

以下，**2.** において BALOC-2 の言語の構成について，**3.** においてその翻訳プログラムについて述べる．**4.** では，ブール関数の内部表現の方法，数式の記憶のための，記憶装置の動的な割り付けの方法およびユーティリティルーチンについて述べ，最後に **5.** で，BALOC-2 を使用した簡単な例について述べる．

2. 言語の構成

新しい言語を構成する場合，すでにできている言語の拡張として構成する場合と，全く新しい言語を構成する場合が考えられる．ここでは前者の方法を選んだ．この理由は，1つには，システム全体の構成が容易になること，他の1つにはブール関数の数式処理は，応用例（たとえば整数計画法の解法）からもわかるように，それ自身で1つの問題となるよりは，より大きな問題の一部分として要求される場合が多いため，作成されるシステムは，数式処理のみならず，通常の数値処理も含めて，高い能率が要求されることがあげられる．

BALOC-2 の記述言語は，このような理由から，FORTRAN に，ブール関数の数式処理と，それに付随する種々の処理のためのステートメントを追加するという形で構成されている．以下では，FORTRAN に新たに追加された機能，ステートメントなどを中心

* Formula Manipulation System for Boolean Functions (BALOC-2),
by Yuzi YOSHIDA and Teruo FUKUMURA (Faculty of Engineering, Nagoya University)

** 名古屋大学工学部

に述べる.

2.1 名前の型の追加

一般に，数式を表現する場合には，ある値をとる変数の名前，ある式を表わす名前，および定数が用いられる．BALOC-2では，第1の名前，すなわち，ブール変数を表わす名前を，BASIC 型，第2の名前，すなわちブール関数を表わす名前を FORMAL 型と名付ける．また定数は，真を1，偽を0で表わす．ブール関数が内部的に処理される場合には，必ず BASIC 型の名前だけを用いて表わされる形になっている．名前の型の宣言のために，次の2つのステートメントが設けられている．

(i) BASIC **list**

(ii) FORMAL **list**

list は，名前または，寸法つき名前のならびで，次元は1次元のみ許される．後に述べるように BASIC 型の名前は，代入文の左辺に現われてはいけない．

2.2 ブール関数の記述

ブール関数，ブール変数，定数を用いて別のブール関数を記述することは，数式処理の基本的な操作である．BALOC-2では，ブール代数において基本的な数種の演算については，対応する演算記号を定めることによって，ブール関数を数式の形で記述することができる．これらは，.EQV.(対等)，+(環和)，.IMP.(含意)，/(論理和)，*(論理積)，'(否定，オペランドの前に付す)である．また，これらの間の優先順位は，上に列記した順の逆に従い，' が最優先される．

プログラム上で記述されたブール関数に，FORMAL 型の名前が用いられている場合には，その名前によって表わされるブール関数を用いて演算を行なうことを表わしている．このようにして表わされるブール関数は演算終了後は，定数または，BASIC 型の名前のみを用いて表現される形になっいる．

たとえば，いま仮に，2つの3変数ブール関数

$$f(x,y,z)=xy\bar{z},$$

$$g(x,y,z)=xy\vee\bar{z}$$

から

$$h(x,y,z)=f(x,y,z)\vee\overline{g(x,y,z)}\vee x\bar{y}\bar{z}$$
$$=y\bar{z}\vee\bar{x}z\vee\bar{y}z\vee x\bar{z}$$

を求める手順は，**Fig. 1** のように記述される．

上で述べた以外の特殊な演算は，後に述べる各演算ごとに定められた専用のステートメントによって行なう．

2.3 ステートメントの追加

```
FORMAL F,G H
BASIC X,Y,Z
F=X*Y*'Z
G=X*Y/'Z
H=F/'G/X*'Y*'Z
```

Fig. 1 Example of description of Boolean function.

BALOC-2 の言語は，名前の型の宣言，ブール関数の代入，特殊な演算，ある種のチェック，ブール関数の入出力，などのためのいくつかのステートメントが FORTRAN に追加されている．以下に，それらについて順に述べる．

2.3.1 型宣言文

これについてはすでに 2.1 で述べたので省略する．

2.3.2 代入文

数式の形で記述されたブール関数を計算して，結果を FORMAL 型の変数に代入するためのステートメントで，形式的には FORTRAN の代入文と全く変わらない．

注意すべきことは，代入文の左辺は FORMAL 型の名前（配列のときは添字つき）でなければならないことである．FORTRAN の代入文との区別はこの性質を用いることで容易にできる．代入文の例は Fig. 1 に示されている．

2.3.3 特殊演算文

次にかかげる4種のステートメントがある．

(1) FSUBST $f_1,(b_1,g_1),\cdots,(b_n,g_n),f_2$

f_1, f_2: FORMAL 型変数名，

b_i: BASIC 型変数名，

g_i: FORMAL 型変数名または BASIC 型変数名，または定数のいずれか．

(以下特にことわらないかぎり，この記法に従う)．

(意味) ブール関数 f_1 のブール変数 b_1 に，g_1 が代入され，その結果得られたブール関数のブール変数 b_2 に g_2 が代入され，以下同様に逐次代入が行なわれて，最後にブール変数 b_n に g_n が代入された結果が f_2 に代入される．

(2) VSUBST f_1, V, n, f_2

V: 整数型1次元配列名．各要素は0，1，2のいずれかの値でなければならない．

n: 整数値．ただし，V の寸法以下の値．

(意味) ブール関数 f_1 のブール変数 $b_1, b_2, \cdots\cdots, b_n$ に $V(1)$, $V(2)$, $\cdots\cdots$, $V(n)$ を代入する．ただし，$V(i)$ の値が2の場合には，変数 b_i には代入しないことを表わす．ブール変数の順序は，BASIC 型宣言文で list に現われた順と一致する．たとえば

```
FORMAL  F,G
BASIC   X1,X2,Y1,Y2,Z1,Z2
F=X1*Y2/Y1*'Z2/Z1*'X2
        ⋮
BSUBST  F,V,G
```

Fig. 2 Example of VSUBST statement.

Fig. 2 の例では，仮に，$V=(1, 2, 0, 2, 1, 2)$ とすれば，G は

$G=Y2/'X2$

となる．

（3） NEGATE $f_1,(b_1, b_2, \cdots, b_n), f_2$

（意味） ブール関数 f_1 にブール変数，$b_1, b_2, \cdots, b_n$ が含まれるとき，それらをすべて否定して得られる関数を f_2 に代入する．否定は，$b_1, b_2, \cdots, b_n$ の順に，逐次的に行なわれる．

（4） PERMUTE f_1, b_1, b_2, f_2

（意味） ブール関数 f_1 に含まれるブール変数 b_1 と b_2 を置換する．

2.3.4 特殊チェック文

与えられたブール関数がある性質を満たすか否かを調べて，その結果によって後の処理内容を変えることを可能にするために，次の3種のステートメントが用意されている．

（1） TEST f_1, v_1

v_1：整数型変数

（意味） ブール関数 f_1 が恒等的に真のとき，$v_1=1$，f_1 が恒等的に偽のとき，$v_1=0$，それ以外の場合は $v_1=2$ となる．

（2） EQUAL f_1, f_2, v_1

（意味） ブール関数 f_1 とブール関数 f_2 とが対等のとき，$v_1=1$，そうでないとき，$v_1=0$ となる．

（3） DEPEND f_1, b_1, v_1

（意味） ブール関数 f_1 がブール変数 b_1 に真に依存するとき $v_1=1$，そうでないとき，$v_1=0$ となる．

ここにあげた3つのステートメントの内容が正しく実行されるためには，ブール関数の表現の一意性が問題となるが，**4.** で述べるように，BALOC-2 では，環和標準形を内部表現に用いることにより，この問題を解決している．

2.3.5 作業領域の消去のための文

BALOC-2 では，**4.** で述べるように，ブール関数を記憶するのに，ページ化された大領域を用いている．この領域が能率的に用いられるためには，不要のブール関数を消去するためのステートメントとして次のステートメントがある．

ERASE $f_1, f_2, \cdots, f_n$

（説明） ブール関数 $f_1, f_2, \cdots, f_n$ を消去する．ただし，$f_1, f_2, \cdots, f_n$ は消去後も名前としては有効であるので，新しい関数をそれらに代入することは可能である．

3. 翻訳プログラム

BALOC-2 翻訳プログラムは，BALOC-2 の言語で記述されたプログラムを FORTRAN プログラムに翻訳する機能をもつ．全体が FORTRAN で作成されていて，約 1300 ステートメントより成っている．

この翻訳プログラムが処理する内容を列記すると，次のようになる．

（1） FORMAL 型の名前に対応する式番号を割り当てる．

（2） BASIC 型の名前に対応する語内ビット位置を割り当てる．

（3） 種々のステートメントの翻訳．

（4） ソースプログラム，オブジェクトプログラム，および FORMAL 型，BASIC 型の名前について，名前と型，対応する式番号，または，語内ビット位置の印刷．

以下では，各項目について詳しく述べる．

3.1 式番号の割りつけ

BALOC-2 のプログラム上である名前で参照されるブール関数は，実行時には，すべて，ある番号（これを式番号と呼ぶ）で参照され，その名前は残らない．したがって，翻訳プログラムは，ソースプログラムの FORMAL 型の宣言文が現われた段階で，そのリスト部分にあげられた名前すべてに通し番号をつけて，その名前に対応する式番号とする．以後，この名前が，実行ステートメントに現われたときには，常に，その名前の部分を式番号でおきかえる処理をする．なお，配列の場合（1次元に限られている）には，その寸法の大きさだけの通し番号を用いる．

たとえば，次のようなステートメントを考える．

FORMAL $F, G(10), H$

この場合，それぞれの式番号は，F が 1，$G(1)$ が 2，$G(2)$ が $3, \cdots, G(10)$ が 11，H が 12 となる．式番号は，オブジェクトプログラムのあとに印刷される名前のリストにおいてプログラマに知らされる．ただし，配列については，その先頭の番号のみが与えら

れる.

BALOC-2の出力ステートメントによってブール関数を出力すると，その関数の名前は印刷されず，対応する式番号が印刷されるので，プログラマは，名前のリストと照らしあわせることによって，実際に出力された式と，その名前とを対応づけなければならない.

3.2 語内ビット位置の割りつけ

BALOC-2プログラムにおいてブール関数を構成するためのブール変数を表わすBASIC型変数は実行時においては，計算機の1語を構成するビットに対応づけられる．翻訳プログラムは，このための処理として，BASIC型宣言文にもとづいて，**3.1**の式番号の割りつけと同様の処理を行なう．ビットの割りつけの意味については**4.**で詳しく述べる.

3.3 種々のステートメントの翻訳

BALOC-2のステートメントのうち，宣言文は上に述べた**3.1**，**3.2**の処理が行なわれるが，他はすべて，実行文でしかもその多くは対応する処理ルーチンを呼ぶCALLステートメントあるいはその並びでおきかえられる．代入文のみがやや複雑な処理を必要とする．ブール関数を記述するために定められた6つの演算子，およびブール関数を代入するための演算子(=)に対しては，それぞれ対応する処理ルーチンが設けられている．代入文は，FORTRANの代入文を機械命令の系列に変換するのと全く同じ方法で，これらの処理ルーチンを呼ぶCALLステートメントの系列に変換される．また，演算の途中で，中間結果を保持する必要が生じた場合には，これらに対しては，特別な式番号が割りつけられる．この番号は，プログラマが用いるFORMAL型の変数名に対応する式番号とは全く混同が生じないようになっている．また，この式番号で与えられるブール関数は，一度演算対象として使われると自動的に消去されるようになっているので，そのための消去ステートメントは，オブジェクトプログラムでは作成されない.

なお，今回作成した翻訳プログラムでは，その作成労力と規模を縮小するために，BALOC-2ステートメントには，その第1コラムにBを付すことになっている．しかし，このことは本質的には不要である.

具体的な翻訳例は，**5.**においてFig. 6にあげられている.

4. ブール関数の内部表現とユーティリティルーチン

数式処理システムを構成する場合，最も重要な問題は，1つは数式を計算機内部においてどのように表現するかという問題，他の1つは，長さが動的に変化する数式を記憶するための記憶装置の動的な使用法の問題である．これらの問題に対しどのような解決策をとるかにより数式処理システムの能力や，能率が大きく左右される．この節では，これらの問題を中心にしてBALOC-2の実行時における処理について述べる.

4.1 ブール関数の内部表現

ブール関数を表現するための標準形として，加法標準形，乗法標準形，環和標準形などがあげられるが，電子計算機内部における内部表現として用いるためには，次のような条件が満たされていることが望ましい.

(1) 内部表現への変換が容易であること.
(2) 1つのブール関数を記憶するのに必要な記憶領域が小さいこと.
(3) 基本的な演算が容易に，速く行なえること.
(4) 外部表現への変換が容易であること.

BALOC-2では，内部表現として環和標準形が採用されている．この標準形によれば，ブール関数は，ブール変数の否定を含まない論理積の環和として表わされる．すなわち，任意のn変数ブール関数$f(x_1, \cdots x_n)$は次のように表わされる.

$$f(x_1, \cdots, x_n)=\sum_{i=1}^{N} x_1^{v_{i1}} \cdot x_2^{v_{i2}} \cdot \cdots \cdot x_n^{v_{in}} \qquad (1)$$

ここに

Σ: 環和による和

N: 論理積の項数

v_{ij}: 第i番目の論理積が変数x_jを含むとき1，そうでないとき0.

$(i=1, \cdots, N;\ j=1, \cdots, n)$

$$x_j^{v_{ij}}=\begin{cases}1 & (v_{ij}=0)\\ x_j & (v_{ij}=1)\end{cases} \qquad (2)$$

このことから，任意のブール関数$f(x_1, \cdots, x_n)$は式(1)の添字のならび$\{v_{ij}\}$によって一意に表現されることがわかる．ブール関数の$\{v_{ij}\}$による表現の例を**Fig. 3**に示す．いま，仮にベクトル$(v_{i1}, v_{i2}, \cdots, v_{in})$を計算機の1語の相続く$n$ビットを用いて1語で表現するものとすれば，式(1)のブール関数はN語で表現される

$X_1 \vee X_2 X_3$ (積和形)

↓

$X_1 \oplus X_2 \oplus X_2 X_3 \oplus X_1 X_2 \oplus X_1 X_2 X_3$ (環和標準形)

↓

i \ j	1	2	3
1	1	0	0
2	0	1	0
3	0	1	1
4	1	1	0
5	1	1	1

$\{v_{ij}\}$ の表，内部表現)

Fig. 3 Conversion of Boolean function into internal representation.

BALOC-2では，ブール関数をこのような方法を用いて内部的に記憶している．この方法による場合，初めに述べた条件のうち，(4) を除いては十分満たされていることが容易にわかる．とくに，語内のビットを用いたためほとんどの計算機の機械命令に含まれているビットごとの論理積，論理和，否定などを巧みに利用することにより，ブール関数同志の演算がきわめて高速，かつ，能率的に行なえることが期待できる．たとえば，$\{v_{ij}^{(1)}\}$ と $\{v_{ij}^{(2)}\}$ によって表わされる2つのブール関数の論理積は次のようにして得られる．

$$f^{(1)}(x_1, \cdots, x_n) \cdot f^{(2)}(x_1, \cdots, x_n)$$
$$= \left(\sum_{i=1}^{N_1} x_1^{v_{i1}^{(1)}} \cdots x_n^{v_{in}^{(1)}}\right) \cdot \left(\sum_{i=1}^{N_2} x_1^{v_{i1}^{(2)}} \cdots x_n^{v_{in}^{(2)}}\right)$$
$$= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} x_1^{(v_{i1}^{(1)} \vee v_{j1}^{(2)})} \cdots x_n^{(v_{in}^{(1)} \vee v_{jn}^{(2)})} \quad (3)$$

式 (3) の添字のならび $((v_{i1}^{(1)} \vee v_{j1}^{(2)}), \cdots, (v_{in}^{(1)} \vee v_{jn}^{(2)}))$ は計算機内部では，1個の論理和命令で値が決まる．また，$\sum$ の操作は，論理和で得られた結果を単にならべるだけでよい．環和の演算では，公式

$$x \oplus x = 0 \quad (4)$$

が適用できるが，このことは，内部表現では，同じ値をもつ語があればその双方を削除するという操作で実行される．この他に，あるブール関数を否定する操作は，単に $(0, 0, \cdots, 0)$ というならびを1つ加えるという操作で実行される．これらのことは，加法標準形の場合などと比べるとはるかに簡単であるといえる．

初めに述べた条件のうち (4) については，どの標準形にも難点があるが，環和標準形は，とくに，人間にとってあまり親しみやすいものではない．

計算機内部においてブール関数を上に述べたようにビット表現する場合，どのビットをどのブール変数に割り当てるかが問題となるが，BALOC-2の場合は，**3.2** で述べたように，ソースプログラムの BASIC 文のリストに現われた順に，各ビットと各 BASIC 型変数とを対応づけている．1つのビットがブール関数を表わすことはありえない．このことから BASIC 型変数の名前が代入文の左辺に現われてはいけない理由の1つがわかるであろう．

また，**2.3.4** で述べた種々のチェック文が常に正しく動作することは，ブール関数が環和標準形という一意な形で内部表現されていることから保証されている．

4.2 記憶の動的な割りつけ

4.1 において述べたようにブール関数は，計算機内部においては FORTRAN における1次元配列の形で表わされている．この配列の長さはブール関数が処理されるに従ってたえず変化する．たとえば，n 変数ブール関数の場合であれば環和標準形で表現されたときの論理積の項数は最大 2^n になる*．

したがって，もしおのおののブール関数に固定した大きさの配列を用意すると，全体で必要となる記憶領域はきわめて大きくなり，記憶装置が効率よく利用されなくなるばかりでなく，実用的な規模のシステムができなくなる．

一方，数式処理システムを構成するときにしばしば利用されるリスト処理の技法は，記憶装置の利用効率がよくなる反面，数式処理のための処理能率が，リスト処理のための能率によってかなりおさえられてしまうという欠点がある．とくに，数式処理の場合には，1つのデータ (数式) の内容を1回の演算において，幾度も反復して参照することが多いので，データの内容を参照するための時間があまり多くなることは許されない．

BALOC-2においては，記憶の動的割りつけの方法として，上に述べた2つの方法の中間的な方法を採用している．すなわち，あらかじめ確保された大きな領域を，比較的小さい領域に等分し，この領域 (以下ではページと呼ぶ) を記憶使用の1単位とする，いわゆるページ方式である．ブール関数は，一般には複数個のページを使用して記憶され，1ページの範囲内では通常の1次元配列と同じように参照できる．ページの大きさは，記憶領域の使用効率と記憶内容の参照に要する時間とに影響し，両者のかね合いのもとに決定されなければならない．今回作成されたシステムでは1ページ 100 語であるので，ブール関数の内容を参照す

* $\bar{x}_1 \cdot \bar{x}_2 \cdot \cdots \cdot \bar{x}_n$ の形のとき項数が最大となる．なお，加法標準形または乗法標準形の場合には，最大項数は $3^n - 1$ となる．さらにこの場合には，内部表現においては1つの項を表現するのに (項が変数の肯定と否定の双方を含みうるため) 2語を必要とするので所要の記憶はこの2倍になる．

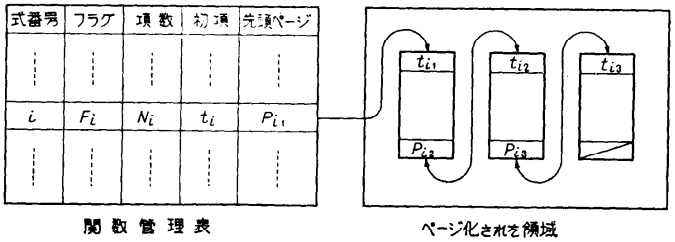

Fig. 4 Form of Boolean function in storage.

る場合，約100項に1回の割り合いでページ処理が必要となる．

ブール関数は実際には **Fig. 4** に示すような形式で記憶されている．図において，F_i は，式番号 i のブール関数が定義されているか否かを示すフラグで，$F_i=1$ のとき，既定義とする．残りの N_i, t_i, P_i はこのときにのみ意味をもつ．N_i はこのブール関数を構成する論理積の項数である．$N_i=0$ は，ブール関数が恒等的に偽であることを表わす．t_i は，$N_i=1$ のときにのみ意味をもつ値で，ブール関数がただ1項のみからなるとき，この1項の内部表現を与える．$N_i \leqq 1$ のときには，したがって，ページ化された領域は全く参照されない．P_i は $N_i \geqq 2$ の場合について，ブール関数が記憶されている先頭のページの番号を与える．また，各ページには，それぞれのページ内に記憶されている論理積の項数 t_i と，そのページに続くページの番号が付されている．

システムは，あきページの表をもっていて，必要なときにはこれに基づいてページの割りつけ，開放を行なうようになっている．

4.3 実行用ユーティリティルーチン

BALOC-2のオブジェクトプログラムを実行するためには，**4.2** で述べた，動的な記憶の割りつけの処理を行なうルーチンと，実際に，種々の基本的な数式処理のための演算を行なうルーチンとが必要である．実際には，前者はほとんどが後者のなかにうめこまれていて，それ専用のルーチンはほとんどない．

実行用のルーチンは，**2.** において述べた各種のBALOC-2のステートメントに対応するルーチンと，6つのブール演算（.EQV.，+，.IMP.，／，＊，'）と代入（=）の処理を行なうルーチンを合わせて17個と，特殊な処理を行なう6個のルーチンからなる．このうち1つをのぞいてはすべてFORTRANで作成されているが，残りの1個は，語内のビットごとの各種論理演算を行なうルーチンで，アセンブラで作成されている．

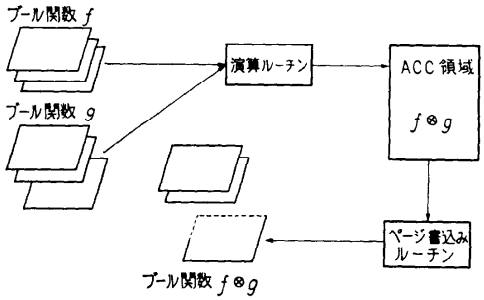

Fig. 5 Process of operations for Boolean functions

4.2 において述べたように，ブール関数は，一般にはページ化されて記憶されているが，ある演算によって得られるブール関数を演算途中で直接ページ化して書き込むことは，能率的でない．なぜならば，演算によって得られるブール関数は，演算が終了するまでは，その項数が定まらないからである．なぜなら，積和標準形の場合には，式（4）の公式の適用によって項数が減少することがあるためである．

BALOC-2では，このことを考慮して演算結果を直接ページ領域に書き込むことはせず，あらかじめ確保された連続した領域（これを ACC 領域と呼ぶ）に書き込む．演算が終了して，ブール関数の形が定まるとACC 領域の内容がページ領域に書き込まれる．したがって，ブール関数同志の演算は，図式的には**Fig. 5** のようになる．

今回作成したシステムは実験的なシステムであるため，比較的小規模でページ領域は全体で 200 ページ（1ページ 100 語），ACC 領域は 2000 語である．また，ユーティリティルーチンは，全体で，FORTRAN ステートメントにして約 800 ステートメント程度である．

```
7       4      (項数と変数の数)
1 0 0 0 ┐
1 0 1 0 │
1 0 0 1 │
0 1 1 0 ├ 各 積 項
1 1 0 1 │
1 0 1 1 │
1 1 1 1 ┘
```

Fig. 6a Input format

```
OBJECT PROGRAM LIST

      CALL EINP( 1)
      CALL ENEG( 1, 1,100)
      CALL ENEG(100, 2,100)
      CALL ENEG(100, 3,100)
      CALL ENEG(100, 4, 2)
      CALL ENOT( 2,3,100)
      CALL EIMF(100,3, 2)
      CALL EEQU( 1, 2,INDIC )
      CALL EOUT( 1)
      CALL EOUT( 2)
      IF(INDIC.EQ.1) GO TO 1000
      WRITE(6,100)
  100 FORMAT(/1H ,19HF IS NOT SELF-DUAL./)
      STOP
 1000 WRITE(6,200)
  200 FORMAT(/1H ,15HF IS SELF-DUAL./)
      STOP
      END
```

Fig. 6c BACOL-2 object program list.

```
BALOC-2 COMPILED LIST

SOURCE PROGRAM LIST

CARD NO.   LABEL      SOURCE STATEMENT

    1      B      FORMAL F,FDUAL
    2      B      BASIC X1,X2,X3,X4
    3      B      INPUT(F)
    4      B      NEGATE F,(X1,X2,X3,X4),FDUAL
    5      B      FDUAL='FDUAL
    6      B      EQUAL F,FDUAL,INDIC
    7      B      OUTPUT(F,FDUAL)
    8             IF(INDIC.EQ.1) GO TO 1000
    9             WRITE(6,100)
   10       100 FORMAT(/1H ,19HF IS NOT SELF-DUAL./)
   11             STOP
   12      1000 WRITE(6,200)
   13       200 FORMAT(/1H ,15HF IS SELF-DUAL./)
   14             STOP
   15             END
```

Fig. 6b BALOC-2 source program list.

```
VARIABLE NAME LIST

   NAME      TYPE      EXP.NO. OR BIT POSITION

F            F                    1
FDUAL        F                    2
X1           B                    1
X2           B                    2
X3           B                    3
X4           B                    4
```

Fig. 6d Variable name list.

```
EXPRESSION NO.      1      NUMBER OF TERMS =      7

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

EXPRESSION NO.      2      NUMBER OF TERMS =      5

1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

F IS NOT SELF-DUAL.
```

Fig. 6e Computed results.

5. 応 用 例

この節では，BALOC-2 の簡単な例題をあげ，入出力の形式，翻訳などの例とする．用いた例題は，入力されたブール関数と双対なブール関数を求め，それら2つの関数を印刷するとともに，入力された関数が自己双対か否かを調べて，その結果をプリントするという内容である．入力データとして用いたブール関数は次式で与えられる．

$$x_1x_2 \vee x_1\bar{x}_3\bar{x}_4 \vee x_2x_3$$
$$= x_1 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_2x_3x_4$$

Fig. 6 に，入力データの形式，ソースプログラム，オブジェクトプログラム，および出力データを示す．

出力された結果より，双対な関数は

$$x_2 \oplus x_1x_3 \oplus x_2x_3x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_4$$
$$= x_1x_2 \vee x_1x_3 \vee x_2\bar{x}_3 \vee x_2\bar{x}_4$$

であることが知られる．また，もとのブール関数が自己双対か否かが EQUAL 文によって，チェックされ，そのプリント結果が同じく Fig. 6 に示されている．

例に示されているように，ブール関数の出力形式は，内部表現ほぼそのままであり，しかも，許される BASIC 型変数の数の最大値 (30) を見込んだ形でプリントされるので，あまり見やすいものとはいいがたい．この点は改善を要する．

6. むすび

本論文では，ブール関数を環和標準形で表わして単純な添字のならびに変換することにより，ブール関数の数式処理を，数値処理と同じ形で行なうシステムについて述べた．このシステムは，数式の内部表現および実行ルーチンなどの点では，従来の数式処理システムのうちでは ALPAK システムに類似しているが，言語構成の点からは，FORMAC システムに類似している．

BALOC-2 システムは，内部表現の形式から，種々の数式処理のための演算の能率，速度はかなり高いが，外部表現との互換性の点で改善すべき点が多い．今後残されたおもな問題をあげれば，入力，出力の形式の改善，**4.2** で述べたページの大きさについての検討，各実行ルーチンの処理能率の向上，システム全体の拡張整備がある．

筆者らは，BALOC-2 とは別に，現在，ブール関数を記号列として処理するシステム (BALOC-3) を開発検討中である．このシステムは，筆者らが開発した FORTRAN を拡張した記号処理システムに基づいていて，BALOC-2 では実現されなかった，公式の定義，任意の演算子の定義，およびこれらの利用などについても考慮されている．

なお，本システムの開発作成にあたっては，東京大学大型計算機センターの HITAC-5020E システムを利用したのでここに明記する(課題番号 4001DB0049)．

終わりに，熱心なご指導をいただいた東北大学本多波雄教授に深謝する．また，日ごろ熱心なご討論をいただく福村研究室の皆様に謝意を表する．

参考文献

1) 稲垣，福村：制約条件をもつ擬似ブール計画法について，電子通信学会誌，**50**，6，p. 47 (昭 42)．
2) 吉田，稲垣，福村：整数形 Min Max 問題の擬似ブール計画法による解法，電子通信学会オートマトン研究会資料 (昭 43-10)．
3) 吉田，福村：擬似ブール関数の数式処理用言語のマクロアセンブラ，昭 44，電四連大，3298
4) 吉田，福村：擬似ブール関数の数式処理用言語 (BALOC-1)，昭 44，信学全大，886
5) 吉田，福村：ブール関数の数式処理言語 (BALOC-2)，昭 44 情報処理学会大会，13
6) 吉田，福村：FORTRAN を拡張した記号処理システム (COSMOS-2)，昭 45 情報処理学会大会，101
7) W. G. Brown: The ALPAK system for non-numerical algebra on a digital computer-I: polynomials in several variables and truncated power series with polynomial coefficients, BSTJ, **42**, 5 (1963)
8) J. E. Sammet: An Overall view of FORMAC, IBM Systems Development Div., TR 00. 1367 (1965)
9) 吉田，福村：記号処理にもとづくブール関数の数式処理言語 (BALOC-3)，昭 46，信学全大，1034．

(昭和 46 年 1 月 27 日受付)