Formula Manipulation System for Boolean Functions (BALOC−3) based upon Symbol Manipulations

Yuuzi Yoshida* and Teruo Fukumura*

1. Introduction

We have developed a Formula Manipulation System for Boolean Functions (BALOC-3). This system is different from the existing formula manipulation systems which are usually intended to process polynomials, rational functions or analytical functions. This system is based upon the Symbol Manipulation System (COSMOS-2) developed by us, and makes use of the facility of virtual symbols in COSMOS-2 for the representation of boolean functions. This is the essential difference between this system and BALOC-2 system developed by us for similar purposes. Owing to this fact, this system trades efficiency for important advantages such as applicability of the boolean laws and easy internal-to-external transformation of boolean representations.

2. Specification of BALOC-3 Language

BALOC-3 language is embedded in FORTRAN. In the following sections, essential parts of the language will be described.

2.1 Descriptions of Boolean Functions

Boolean functions in BALOC-3 are expressed in almost the same manner as the logical expressions of FORTRAN. The FORTRAN logical expressions can be used as constants in Boolean functions, but they must be quoted. Boolean operators which are declared by OPERATOR statements can also be used in the form of functions.

For example, expression,

$$x\cdot(p^2 > q)\vee\overline{x\cdot y\cdot z}$$

is described in BALOC-3 as Fig.1, where $n(x,y,z)=\overline{x\cdot y\cdot z}$.

```
X.AND.'P**2.GT.Q'.OR.N(X,Y,Z)
```

Fig.1 Boolean function in BALOC-3

2.2 Identifiers for Boolean Operations

There are four new types of identifiers in BALOC-3. They are;

(1) FORMAL type for variables whose values are boolean functions.

(2) OPERATOR type for boolean operators which are declared in programms.

(3) FORMULA type for boolean laws which are defined in programms.

(4) Boolean variables. All identifiers which are used in boolean functions and not declared as any types shown above belongs to this category.

2.3 Statements Added to FORTRAN

(1) FORMAL $v_1, v_2, \ldots, v_n$

$v_1, v_2, \ldots, v_n$ are declared as FORMAL type variables.

(2) OPERATOR p(arg-list)= expr

This paper first appeared in Japanese in Joho − Shori (Journal of the Information Processing Society of Japan), Vol. 14, No. 1 (1973), pp. 8 ∼ 14.

* Faculty of Engineering, Nagoya University

p is declared as a boolean operator. arg-list has the form of '*1,*2,...,*n' (number of variables is n) or '**' (arbitrary). expr is the definition of p.

(3) FORMULA(name) expr.1 = expr.2

name is declared as a name representing a boolean law of the form:

expr.1 = expr.2

(4) v = expr (Assignment statement)

v is a FORMAL variable. This statement has just the same form as the logical assignment statement in FORTRAN.

(5) APPLY(f) $p_1, p_2, \ldots, p_n$

Boolean laws $p_1, p_2, \ldots, p_n$ are applied to f in this order. p_i is 'name' or '-name' where name is FORMULA type. In the former case, p_i is applied as string transformation rule from left side part to right side part of law. In the latter case, p_i is applied reversely.

(6) Other statements

There are eight more statements in BALOC-3. They are the statements for special boolean operations and several tests with respect to boolean functions. All of them are similar to those of BALOC-2.

3. BALOC-3 Compiler

BALOC-3 compiler is the FORTRAN program (about 1200 statements) and translates BALOC-3 source programs to COSMOS-2 programs. BALOC-3 compiler and runtime routines mentioned in the following section make use of many features of COSMOS-2, and because of this fact BALOC-3 system has become concise. Some important points in the compilation process are described below.

(1) Processing of BALOC-3 variables

Only FORMAL variables are translated into COSMOS-2 variables and other variables are translated into references to the corresponding name tables which are COSMOS-2 STRING arrays.

(2) Processing of boolean functions

Boolean functions are translated into COSMOS-2 STRING expressions. They are partitioned into three parts. These are FORMAL variables, constants, and others. Each parts are translated separately and the results are concatenated.

(3) Processing of definitions

Definitions of boolean operators and laws are stored in the corresponding STRING arrays at run time. The corresponding statements are therefore translated into STRING assignment.

(4) Processing of other statements

Most of other statements are translated into CALL statements which call the corresponding service routines.

An example of translation is shown in Fig.4.

4. Internal Representation of Boolean Functions and Their Processing

There are two essential problems in the implementation of a formula manipulation system. One is how to represent formulas in storage and the other is how to allocate storage to formulas dynamically. The solutions of these problems have significant influences on the performance of the system. BALOC-3 system is based upon COSMOS-2 system, and so the latter problem is solved by COSMOS-2 system. Thus only the former problem is dealt with in the implementation of BALOC-3 system. In the following the solution is described.

4.1 Internal Representation of Boolean Functions

Boolean function is transformed into the reverse polish notation and represented internally by COSMOS-2 string of virtual symbols. Each virtual symbol corresponds to constant, boolean variable, operator or delimiter. The symbol number of a virtual symbol consists of two parts (k_i and n_i). k_i represents the class of the symbol and n_i represents auxiliary information for each class. The meaning of (k_i, n_i) is shown in Table 1.

For example, the following boolean function is represented as in Fig.2.

X.AND.Y.OR.NAND(X,Y,.TRUE.)

When an operator takes several operands in a boolean function, they are lexicographically ordered in the corresponding internal representation. This makes it easy to execute some statements. The internal representation described here gives some new features to the system, Which were not attained in BALOC-2. They are as follows,

(1) External to internal (and vice versa) translation of boolean functions is easily done.

(2) Internal representation is very compact. This fact is mainly due to the facility of virtual symbols in COSMOS-2.

4.2 Operations on Boolean Functions

Among various operations on boolean functions, some of them are particularly sophisticated and depend on the internal representation of boolean functions. In this section, we show how to process such operations in BALOC-3.

(1) Automatic applications of simple identities

In BALOC-3 system, several laws are automatically applied to boolean functions. These are:

$$\bar{0} = 1,\ \bar{1} = 0,\ x\cdot 1 = x,\ x\cdot 0 = 0,\ x\vee 1 = 1,\ x\vee 0 = x.$$

Table 1. Classification of virtual symbols

k	class	n
0	ordinary characters	$1 \leq n \leq 48$

1	constants	n =0:true
		n =1:false

2	variables	$n \leq 10$:arbitrary terms $n > 10$:boolean variables
3	operators	n =1;OR;n =2:AND;n =3:NOT; otw.:users operators
4	delimiters	n =0:initial/terminal symbol n =1:delimiter of list of operands

list of (k_i, n_i)

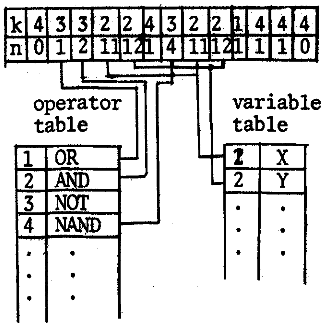

Fig.2 Internal representation of boolean function

Applications of these rules are not so difficult. Since BALOC-3 is based upon the string manipulation system, these operations are executed by pattern matching facilities for strings in COSMOS-2. For example, the application of the law $x \cdot 1 = x$ is done by the application of the rewriting rule,

$$\text{and}(1,'\ldots') \longrightarrow '\ldots' ,$$

where '...' is an arbitrary expression.

(2) Execution of the APPLY statement

One of the central problems in BALOC-3 is the execution of APPLY statements. The applications of laws are possible in various levels of expressions but doing them throughout all levels is virtually impossible. For example, in the case where the operator of the law can have arbitrary number of operands, finding the appropriate operands in the given boolean functions becomes extremely tedious. Therefore, in BALOC-3 system, only rather simple applications are performed. In the example above, an operand is restrictively interpreted to correspond to an expression but not to a list of expressions.

5. Illustrative Example

In this section, one simple example of the BALOC-3 program is given to show various processings performed by BALOC-3 system. In this example, one operator,

$$\text{nand}(x,y,z) = \overline{x \cdot y \cdot z},$$

and two laws,

$$x \vee \bar{x} = 1,$$
$$\bar{x} \vee x \cdot y = \bar{x} \vee y,$$

are defined. Then, they are applied to

$$x \cdot y \vee \overline{x \cdot y \cdot z}$$

and, the reduced expression is printed.

Fig.3 is the BALOC-3 source program for this problem, and Fig.4 is its object program (COSMOS-2 program). Fig.5 is the printed results.

6. Concluding Remarks

BALOC-3 system is superior to BALOC-2 in some aspects such as the facilities of operator and law, the compatibility to FORTRAN and the flexible representation of boolean functions. But these advantages are obtained at the expense of efficiency. It is not very difficult to improve efficiency, but introducing more processing power concerning, for example, application of boolean law will not be so easy.

In general any formula manipulation systems have problems similar to the above, and these problems will not be solved without developments of heuristic tree search.

```
SOURCE PROGRAM LIST

/FORMAL F
/OPERATOR NAND(*1,*2,*3)=.NOT.(*1.AND.*2.AND.*3)
/FORMULA(LAW1) .NOT.*1.OR.*1.AND.*2=.NOT.*1.OR.*2
/FORMULA(LAW2) *1.OR..NOT.*1=.TRUE.
/F=X.AND.Y.OR.NAND(X,Y,Z)
/EXPAND(2),F
/APPLY(F) LAW1
/OUTPUT(F)
/APPLY(F) LAW2
/OUTPUT(F)
STOP
END
```

Fig. 3 BALOC-3 source program

```
OBJECT PROGRAM LIST

 COMMON OPNAM,OPDEF,FRMNAM,FMDEFL,FMDEFR,DEXPR,NOPRND(10)
 /STRING OPNAM(20),OPDEF(20),FRMNAM(20),FMDEFL(20),FMDEFR(20),
1    DEXPR(10)
 /INITIAL
 CALL B3INIT
 /STRSET OPNAM,OPDEF,FRMNAM,FMDEFL,FMDEFR,DEXPR
 /STRING F
 /STRSET F
 /OPNAM(4)='NAND'
 NOPRND(4)=3
 /OPDEF(4)=INCONV('.NOT.(*1.AND.*2.AND.*3)')
 /FRMNAM(1)='LAW1'
 /FMDEFL(1)=INCONV('.NOT.*1.OR.*1.AND.*2')
 /FMDEFR(1)=INCONV('.NOT.*1.OR.*2')
 /FRMNAM(2)='LAW2'
 /FMDEFL(2)=INCONV('*1.OR..NOT.*1')
 /FMDEFR(2)=INCONV('.TRUE.')
 /F=INCONV('X.AND.Y.OR.NAND(X,Y,Z)')
 CALL EXPAND(2,F)
 CALL APPLY(F, 1)
 CALL BOUT(F)
 CALL APPLY(F, 2)
 CALL BOUT(F)
 STOP
 END
```

Fig. 4 BALOC-3 object program

```
Y.OR..NOT.X.OR..NOT.Y.OR..NOT.Z
.TRUE.
```

Fig. 5 Computed results

References

[1] Y. Yoshida and T. Fukumura:Formula Manipulation System for Boolean Functions (BALOC-2), J. IPSJ, 12, 7 (1971).

[2] Y. Yoshida and T. Fukumura:Symbol Manipulation System based upon FORTRAN (COSMOS-2), J. IPSJ, 13, 4 (1972).