EFFICIENT IMPLEMENTATION FOR SOME OPERATORS IN TABULAR ALGEBRA

Layla Abou-Hadeed Abd-Allah
Department of Computer Science, Faculty of Engineering
Alexandria University, Alexandria, Egypt.

ABSTRACT

A tabular method to solve switching equations was presented in [1]. Some enhancements of the method are presented in [2] and [3]. Specifically a method to complement a table and to perform Exclusive-OR(XOR) operation between two tables were presented. These two operators allow to solve equations of a general form. This paper presents an efficient implementation for these operators. Two implementation for finding the complement of a table are introduced. They have the advantage that no table-multiplication is required. Also an efficient approach for finding the XOR between two tables is introduced. Again no table-multiplication is required. The proposed approaches are simpler than the previous ones.

Keywords: Tabular algebra, Switching equations.

I. INTRODUCTION

Trabado et al.[1] have proposed a method to solve switching equations based on a tabular algebra. They proposed a tabular representation for a Boolean sum of product expressions. Each product term is represented by a row (of 0s,1s, and dashes) in a table. Column i contains 0,1,or - for complemented, uncomplemented or absent literal respectively. For example: $f(A,B,C)=AB+\bar{B}C$ can be represented as follows:

A	B	C
1	1	-
-	0	1

They proposed some operators to manipulate these tables in order to solve equations of the form f(x)=1. Unger[2] proposed a method to complement the tables. Jung [3] presented a method to perform XOR operation between two tables. This allows to solve the equations in a rather direct manner. This paper presents two approaches for complementing a table and an approach for performing XOR between two tables. The rest of the paper is organized as follows: section 2 introduces the definition of some operations and procedures. Section 3 presents two approaches for complementing a table. Section 4 presents an approach for performing XOR between two tables and section 5 is for conclusion.

II. BASIC OPERATIONS AND PROCEDURES

Unger's approach for complementing one-row table [2]:

Let q denote the number of columns that are equal to 0 or 1. Generate q rows such that the i^{th} row is built as follows:
all dashes are in the same positions. The first (i-1) columns of the remaining columns are the same. Column i has an opposite value and the remaining columns are all dashes.

Separation operation;#[1]:
Separation defined by Trabado et al. [1] means that if the a table consists of more than one row, overlapping is eliminated.
Example: Let A=--10, and B=001- then A#B={1-10,0110}.

In this paper, Separation is also defined between tables as follows: $T_1\#T_2$ -for two tables T_1 and T_2- means eliminating overlapping between the two tables from T_1, or simply separating T_2 from T_1.

III. COMPLEMENTING A TABLE

The first approach:

The theorem proposed by Hong et al. [4] for complementing a Boolean function is adopted. The theorem states that the complement of the Boolean function $F=\sum_i E_i f_i$ is given by $\overline{F}=\sum_i E_i \overline{f_i}$ if and only if $\sum_i E_i=1$ and $E_i F=E_i f_i$ $\forall i$. This theorem can be applied to a function represented in a tabular form as follows:

Let n denote the number of variables. Assume that the initial value of n≥2.

```
Procedure CMP(n,T)  ; n denotes the current number of variables.
                    ; T denotes the current table.
if there exists one row
then begin
      apply Unger's approach [2] ;for complementing one-row table;
      nullify T;
      save the resulting row(s) in T;
      exit;
    end;
if there exist two rows and n=1
then begin
      nullify T;
      exit ; the two rows may be either (0,1),(0,-) or (1,-) which represents the unity function. Its
             complement is the zero function.
    end;
If the first column is all-dash column
then begin
      eliminate the first column;
      CALL CMP(n-1,Ti);
      if Ti is nonempty
      then begin
            append an all-dash column to Ti;
            append Ti to T;
          end;
    end;
else begin
      split the rows having a dash in column 1 into two rows;
      assign the values 0(1) to the first column of one (the other) row, all other columns remain the same;
      partition the table into at most two tables T0 and T1 such that rows having value 0(1) in column one
      are in the table T0(T1);
      eliminate the first column in each table;
      nullify table T;
      if there exists one nonempty table Ti (i=0 or 1)
      then begin
            Call CMP(n-1,Ti);
            If Ti is nonempty
            then begin
                  append all-i column to Ti;
                  append Ti to T;
                end;
```

```
        append to T a row as follows:
        column 1=ī  and the remaining columns are all dashes;
      end;
  else for i=0 to 1 do
      begin
        Call CMP(n-1,T_i);
        If T_i is not empty
        then begin
             append all-i column to T_i;
             append T_i to T;
           end;
      end;
end.
```

This approach has the advantage that only one-row table complementation is required. The table also can have small number of variables which simplifies the solution as we get rid of the table multiplication proposed by Unger.

We will consider the same examples proposed by Unger[2].

Example 1:

$$T=\begin{pmatrix}0 & 1 & - & 0\\ 1 & - & - & 0\\ - & - & 1 & 1\end{pmatrix}.$$

Splitting the dashed row results in the following table $\begin{pmatrix}0 & 1 & - & 0\\ 0 & - & 1 & 1\\ 1 & - & - & 0\\ 1 & - & 1 & 1\end{pmatrix}$ from which T_0 and T_1 will be as follows:

$T_0=\begin{pmatrix}1 & - & 0\\ - & 1 & 1\end{pmatrix}^0$ and $T_1=\begin{pmatrix}- & - & 0\\ - & 1 & 1\end{pmatrix}^1$ where the superscript denotes the column value appended from left to the complement of the associated table to get the final answer. T_0 is subdivided into $(1\ \ 1)^{00}$ and $\begin{pmatrix}- & 0\\ 1 & 1\end{pmatrix}^{01}$. Applying Unger's approach to the first one results in $\begin{pmatrix}0 & -\\ 1 & 0\end{pmatrix}^{00}$. Appending 00 to the resulting rows gives $\begin{pmatrix}0 & 0 & 0 & -\\ 0 & 0 & 1 & 0\end{pmatrix}$ which represent some rows of the final answer. The second table generates both $(0)^{010}$ and $\begin{pmatrix}0\\ 1\end{pmatrix}^{011}$. The first of them generates $(0\ \ 1\ \ 0\ \ 1)$ and the last generates NULL (nothing).

T_1 generates the following table: $\begin{pmatrix}- & 0\\ 1 & 1\end{pmatrix}^{1-}$ which is subdivided into $(0)^{1-0}$ and $\begin{pmatrix}0\\ 1\end{pmatrix}^{1-1}$. The first of them generates $(1\ \ -\ \ 0\ \ 1)$ and the last generates NULL.

Appending all resulting rows gives $\bar{T}=\begin{pmatrix}0 & 0 & 0 & -\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & - & 0 & 1\end{pmatrix}.$

This approach can be enhanced by rearranging the columns according to the number of dashes per column such that column with the least number of dashes becomes the first column. Applying this to the example above yield $T'=\begin{pmatrix}0 & 0 & 1 & -\\ 0 & 1 & - & -\\ 1 & - & - & 1\end{pmatrix}$ which gives two tables $T_0=\begin{pmatrix}0 & 1 & -\\ 1 & - & -\end{pmatrix}^0$ and $T_1=(-\ \ -\ \ 1)^1$. T_0 is subdivided into $(1\ \ -)^{00}$ and $(-\ \ -)^{01}$. The first one generates $(0\ \ 0\ \ 0\ \ -)$ and the second generates NULL. T_1 generates row $(1\ \ -\ \ -\ \ 0)$. Thus the final result $\overline{T'}=\begin{pmatrix} 0 & 0 & 0 & - \\ 1 & - & - & 0 \end{pmatrix}$. Rearranging the columns again results in the same solution.

The implementation of this approach can be simplified as follows:

Instead of rearranging the columns, each column is assigned a number representing its order. Starting from the least-order column, apply the approach above. To preserve the deleted columns (its value and location) assign to each generated table a tag (a string of length equals to the number of variables). Assign the value of the deleted column to the corresponding location of the tag. When the complement of some table is found (one-row table or table with one column), The rows of the required result con be formed by adding the deleted columns to the table's complement in their places (known from the tag).

Example 2:

$$T=\begin{pmatrix} 0 & 1 & - & 0 \\ 1 & - & - & 0 \\ - & - & 1 & 1 \end{pmatrix}^{****}.$$

The columns are ordered as follows: begin with column 4 then 1 then 2 and finally 3. Splitting the table with respect to column 4 gives two tables $T_0=\begin{pmatrix} 0 & 1 & - \\ 1 & - & - \end{pmatrix}^{***0}$ and $T_1=(- \ - \ 1)^{***1}$. T_0 is subdivided into $(1 \ -)^{0**0}$ and $(- \ -)^{0**1}$. The complement of the first one is $(0 \ 1)^{0**0}$. Combining it with the tag gives $(0 \ 0 \ - \ 0)$ and the second generates NULL. The complement of T_1 is $(- \ - \ 0)^{***1}$. Combining it with the tag gives $(- \ - \ 0 \ 1)$. Thus the final result $\overline{T}=\begin{pmatrix} 0 & 0 & - & 0 \\ - & - & 0 & 1 \end{pmatrix}$.

The other approach:

Another approach for complementing a table without multiplying tables is introduced. It is based on the separation operation proposed by Trabado et al. [1]. The procedure are as follows:

1. Construct another table having an all-dash row.
2. Separate the given table from the constructed one.

The complement is found in the constructed table.

Verifying the procedure is simple. Let T_g and T_c denote the given and constructed tables respectively. All-dash row means that the table represents a unity function. Applying separation results in $\overline{f}$.

Example 3:

$$T_g=\begin{pmatrix} 0 & 1 & - & 0 \\ 1 & - & - & 0 \\ - & - & 1 & 1 \end{pmatrix} \text{ and } T_c=(- \ - \ - \ -).$$

Applying separation results in the following tables:

1. with respect to row (0 1 - 0):

$$T=\begin{pmatrix} 1 & - & - & - \\ 0 & 0 & - & - \\ 0 & 1 & - & 1 \end{pmatrix}$$

2. with respect to row (1 - - 0):

$$T=\begin{pmatrix} 1 & - & - & 1 \\ 0 & 0 & - & - \\ 0 & 1 & - & 1 \end{pmatrix}$$

3. with respect to row (- - 1 1):

$$T=\begin{pmatrix} 1 & - & 0 & 1 \\ 0 & 0 & 0 & - \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

IV. APPLYING EXCLUSIVE-OR BETWEEN TWO TABLES

It is required to find T_1 XOR T_2 for two tables T_1 and T_2.

Theorem

Let $T_1'(T_2')$ be the table T_1 (T_2) after separating each row of T_2 (T_1) from it. Then T_1 XOR T_2 = T_1'OR T_2'.

Proof:

If the table is viewed as a set of rows, and from the definition of separation, $T_1'=T_1-(T_1 \cap T_2)$ and $T_2'=T_2-(T_1 \cap T_2)$ then

T_1' OR $T_2'=T_1$ OR $T_2-(T_1 \cap T_2)=T_1$ XOR T_2

i.e. by appending the rows of both T_1' and T_2', the required table is obtained.

This solution is simpler than that of Jung [3] as multiplication between tables is eliminated. The simplicity can be shown by solving the same example of Jung.

Example 4:

$$f(X)=\begin{bmatrix} 0 & 1 & 1 & - \\ 1 & 0 & - & 1 \end{bmatrix},\ g(X)=\begin{bmatrix} - & 0 & 0 & - \\ - & - & 1 & 1 \end{bmatrix}.$$

Separating first row of g from f results in $\begin{bmatrix} 0 & 1 & 1 & - \\ 1 & 0 & 1 & 1 \end{bmatrix}$.

Separating second row of g from the resulting table results in $T_1'=[0\ \ 1\ \ 1\ \ 0]$. Separating first row of f from g results in $\begin{bmatrix} - & 0 & 0 & - \\ 1 & - & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$. Separating second row of f from the resulting table results in

$$T_2'=\begin{bmatrix} 0 & 0 & 0 & - \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

$$f(X) \text{ XOR } g(X)=T_1' \text{ OR } T_2'=\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & - \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Unlike that of Jung, the proposed approach results in a separated table.

CONCLUSION

The paper presents efficient approaches for complementing a table and applying XOR between two tables. The approaches are simple. They eliminate the need for table-multiplication required in previous solutions.

REFERENCES

[1] P.P. Trabado, and A. Lloris-Ruis, "Solution of Switching Equations Based on a Tabular Algebra", *IEEE Trans. on Comp.* vol. 42, no. 5, pp. 591-596, May 1993.

[2] S.H. Unger, "Some Additions to "Solution of Switching Equations Based on a Tabular Algebra", *IEEE Trans. on Comp.* vol. 43, no. 3, pp. 365-367, Mar. 1994.

[3] G. Jung, "Comments on "Some Additions to Solution of Switching Equations Based on a Tabular Algebra", *IEEE Trans. on Comp.* vol. 44, no. 11, pp. 1357-1358, Nov. 1995.

[4] S.J. Hong and D.L. Ostapko, "On complementation of Boolean Functions", *IEEE Trans. on Comp.*, vol. C-21, no. 9, pp. 1022, Sep. 1972.