# Semi-Tensor Product Approach to Boolean Functions*

Yin Zhao†, Xu Gao‡, Daizhan Cheng†

† Key Lab. of Systems and Control, AMSS, Chinese Academy of Sciences, Beijing 100190, China
Email: zhaoyin@amss.ac.cn, dcheng@iss.ac.cn

‡ Department of Mathematics, University of Science and Technology of China, Hefei 230026, China
Email: gaoxu@mail.ustc.edu.cn

**Abstract.** Boolean function can be expressed as matrix form using semi-tensor product of matrices. Using this approach, we give a neat proof of the conversion between the truth table and polynomial form of a Boolean function. The linear structure of Boolean functions is also investigated.

**Key Words.** Boolean function, semi-tensor product, truth table, polynomial form, linear structure.

## 1 Introduction

Boolean function (or logical function) is a basic concept in Boolean algebra, which plays a fundamental role in computer sciences, circuit design, cryptography, etc.[1, 2, 3, 4]. In investigation of cellular networks, Kauffman proposed the Boolean network, which is a dynamic system consisting of Boolean functions[5]. It has then become a very useful tool in modeling of cell regulation[6, 7]. Boolean function can also be used in game theory[8, 9, 10], it is called a simple game or voting game, by which the social choice (e.g. the election) can be analyzed.

Denoting $\mathcal{D} = \{0, 1\}$, an $n$-ary Boolean function is a function $f : \mathcal{D}^n \to \mathcal{D}$. There are many ways to express a Boolean function, among them the truth table is the most natural one. The polynomial expression and Walsh spectral expression are two of the most useful tools in the analysis of Boolean function. There are also some graphic expressions which are more efficient in computing, such as binary decision diagrams[11] and propositional directed acyclic graphs[12]. The conversion among different expressions is a fundamental and challenging topic.

Different applications invoke different properties of Boolean functions. Linearity is a basic property. In the design of circuits, linear Boolean functions are widely used, since they are easily realizable. But the linearity is a critical weakness in cryptography for the functions with linear structure are easily breakable [13], the ones with high nonlinearity are preferred. Thus, it is important to check whether a Boolean function has a linear structure.

Recently, using the semi-tensor product of matrices, a Boolean function can be expressed in a matrix form[14, 15]. In this paper, using the matrix form of Boolean function, we give a formula for the conversion between the truth table and the polynomial expression of Boolean functions and propose a way to find out the linear structure of a Boolean function.

The rest of the paper is organized as follows: Section 2 reviews the polynomial expression and the matrix form of Boolean function. Using the matrix form, a formula converting the truth table to the

*This work is supported partly by National Nature Science Foundation 61074114, 60736022 and 60821091 of China.

polynomial expression is obtained in Section 3, which is essentially the same as the known result [3], but neat proof is provided by using semi-tensor product. Section 4 investigates the linear structure of Boolean networks. Section 5 is a brief conclusion.

## 2 Expressions of Boolean Functions

To introduce the matrix expression of Boolean functions, we first briefly review the semi-tensor product (STP) of matrices.

**Definition 2.1** *[14] Let $A \in \mathcal{M}_{m\times n}$, $B \in \mathcal{M}_{p\times q}$, and $c = lcm(n,p)$ (the least common multiple of $n$ and $p$). Then the semi-tensor product (STP) of matrix $A$ and $B$, denoted by $A \ltimes B$, is defined as*

$$A \ltimes B = (A \otimes I_{\frac{c}{n}})(B \otimes I_{\frac{c}{p}}), \tag{1}$$

*where "$\otimes$" is Kronecher product.*

When $n = p$, it is easy to see that $A \ltimes B = AB$, and hence "$\ltimes$" will be omitted hereafter. For statement ease, we introduce some notations.

**Notations:**

(i) Let $\delta_n^i$ be the $i$-th column of the identity matrix $I_n$, and $\Delta_n := \{\delta_n^1, \delta_n^2, \cdots, \delta_n^n\}$. When $n = 2$ we simply use $\Delta := \Delta_2$.

(ii) Denote by Col($A$) (Row($A$)) the set of columns (rows) of $A$, and Col$_i$($A$) (Row$_i$($A$)) the $i$-th column (row) of $A$.

(iii) Assume a matrix $L = [\delta_n^{i_1}\ \delta_n^{i_2}\ \cdots\ \delta_n^{i_s}] \in \mathcal{M}_{n\times s}$ , i.e., its columns, Col($L$) $\subset \Delta_n$. We call $L$ a logical matrix, and simply denote it as

$$L = \delta_n[i_1\ i_2\ \cdots\ i_s].$$

The set of $n \times s$ logical matrices is denoted by $\mathcal{L}_{n\times s}$.

(iv) $W_{[n,m]}$ is a swap matrix which can swap two vectors in their "product". We refer to [14] for its definition and properties, and (2) for its basic functions.

One advantage of the STP is pseudo-commutativity. This property is important for obtaining the matrix form of Boolean functions. The following proposition about pseudo-commutativity and reducing matrix will be frequently used in this paper.

**Proposition 2.2** *[15]*

*1. Let $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ be two column vectors. Then*

$$W_{[m,n]}xy = yx. \tag{2}$$

*2. Let $x \in \mathbb{R}^t$ and $A$ is a given matrix. Then*

$$xA = (I_t \otimes A)x. \tag{3}$$

*3. Let $x \in \Delta_n$. Then $x^2 = M_r^n x$, where*

$$M_r^n := \operatorname{diag}[\delta_n^1 \; \delta_n^2 \; \cdots \; \delta_n^n].$$

*is call the base-n order reducing matrix.*

Then, we consider the expression of elements in $\mathcal{D}^n$. We propose the following three ways to express it.

(i) Component-wise (C-W) Form:

$$X = (x_1, x_2, \cdots, x_n), \quad x_i \in \mathcal{D}, \; i = 1, \cdots, n. \tag{4}$$

(ii) Scalar Form: Consider $x_1 \; x_2 \; \cdots \; x_n$ as a binary number. Then in decimal form we have a number as

$$\chi = x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n, \tag{5}$$

where $0 \le \chi \le 2^n - 1$.

(iii) Vector form: Identify $1 \sim \delta_2^1$ and $0 \sim \delta_2^2$, then $x_i \in \Delta_2$ and we set

$$x := \ltimes_{i=1}^n x_i \in \Delta_{2^n}. \tag{6}$$

The component-wise form can be considered as the binary representation of the scalar form, thus they are equivalent. The vector form is obtained from the component-wise form. Then, the following conversion between scalar form and vector form ensures that the three expressions of elements in $\mathcal{D}^s$ are equivalent. Thus, in the sequel we will not distinct them if there is no possible confusion.

**Proposition 2.3** *Let $\chi$ be a scalar form of $x \in \Delta_{2^n}$. Then*

$$x = \delta_{2^n}^{2^n - \chi}. \tag{7}$$

*Equivalently, let $x = \delta_{2^n}^t$. Then*

$$\chi = 2^n - t. \tag{8}$$

Using the definitions and Proposition 2.3, it is easy to convert an element in $\mathcal{D}^n$ from one form to another. We give an example for this.

**Example 2.4** *Let $n = 8$. Then*

$$\begin{array}{lclcl}
\chi = 51 & \Leftrightarrow & X = (0,0,1,1,0,0,1,1) & \Leftrightarrow & x = \delta_{2^8}^{205}; \\
X = (1,1,0,0,1,0,1,0) & \Leftrightarrow & \chi = 2^7 + 2^6 + 2^3 + 2^1 = 202 & \Leftrightarrow & x = \delta_{2^8}^{54}; \\
x = \delta_{2^8}^{120} & \Leftrightarrow & \chi = 2^8 - 120 = 136 & \Leftrightarrow & X = (1,0,0,0,1,0,0,0).
\end{array}$$

A natural way to express a Boolean function $f$ is to range the values of all the elements in $\mathcal{D}^n$ in a vector:

$$[f(\delta_{2^n}^1), f(\delta_{2^n}^2), \cdots, f(\delta_{2^n}^{2^n})]^T. \tag{9}$$

That is the truth table of $f$.

The addition $\oplus$ and the multiplication $\odot$ over $\mathcal{D}$ is defined as

$$\begin{cases} a \oplus b := a + b \pmod 2 \\ a \odot b := ab \pmod 2. \end{cases} \tag{10}$$

Then $GF(2) := \{\mathcal{D}, \oplus, \odot\}$ forms a field, call the Galois Field. In fact, $\oplus$ and $\odot$ are logical operators $\bar{\vee}$ and $\wedge$ respectively. For the sake of compactness, we simply denote $a \oplus b = a + b$, and $a \odot b = ab$.

For $x \in \mathcal{D}$, denote $x^1 := x, \quad x^0 = \neg x$. Then, it is obvious that for $c \in \mathcal{D}$,

$$x^c = \begin{cases} 1, & x = c \\ 0, & x \neq c. \end{cases} \tag{11}$$

For $X = (x_1, x_2, \cdots, x_n) \in \mathcal{D}^n$ and $C = (c_1, c_2, \cdots, c_n) \in \mathcal{D}^n$, define

$$X^C := \prod_{i=1}^{n} x_i^{c_i} = \begin{cases} 1, & X = C \\ 0, & X \neq C. \end{cases} \tag{12}$$

Hence, it is obvious that

$$f(X) = \sum_{C=0}^{2^n-1} f(C) X^C. \tag{13}$$

Replacing $x_i^1$ by $x_i$, and $x_i^0$ by $x_i + 1$ in (13), the Boolean function $f$ can be expressed as a polynomial in $GF(2)$ as

$$\begin{aligned} f(x) =& a_0 + a_1 x_1 + \cdots a_n x_n + a_{12} x_1 x_2 + \cdots + a_{n-1\, n} x_{n-1} x_n + \cdots + a_{12\cdots n} x_1 x_2 \cdots x_n \\ =& a_0 + \sum_{k=1}^{n} \sum_{1 \le j_1 < \cdots < j_k \le n} a_{j_1 \cdots j_k} x_{j_1} \cdots x_{j_k}. \end{aligned} \tag{14}$$

The following theorem about the matrix form of Boolean functions was proved in [14].

**Theorem 2.5** *Let $f : \mathcal{D}^n \to \mathcal{D}$ be a Boolean function. Then there exists a unique logical matrix $M_f \in \mathcal{L}_{2 \times 2^n}$, called the structure matrix of $f$, such that in vector form $f$ can be expressed as*

$$f(x_1, \cdots, x_n) = M_f \ltimes_{i=1}^{n} x_i, \quad x_i \in \Delta. \tag{15}$$

The following table is the structure matrices of some logical operators.

Table 1: Structure Matrices of Operators

| Operator | Structure Matrix |
|---|---|
| $\neg$ | $M_\neg = \delta_2[2\ 1]$ |
| $\wedge$ | $M_\wedge = \delta_2[1\ 2\ 2\ 2]$ |
| $\vee$ | $M_\vee = \delta_2[1\ 1\ 1\ 2]$ |
| $\rightarrow$ | $M_\rightarrow = \delta_2[1\ 2\ 1\ 1]$ |
| $\leftrightarrow$ | $M_\leftrightarrow = \delta_2[1\ 2\ 2\ 1]$ |
| $\bar{\vee}$(or $\oplus$) | $M_\oplus = \delta_2[2\ 1\ 1\ 2]$ |

**Remark 2.6** *Since the structure matrix is a logical matrix, it can be totally determined by its first row:*

$$m_f = \mathrm{Row}_1(M_f).$$

*Actually, $m_f$ is the truth table of Boolean function $f$.*

We give an example to depict these expressions.

**Example 2.7** *Consider a Boolean function*

$$f(x_1, x_2, x_3) = x_1 \wedge \neg(x_2 \to x_3).$$

*In vector form, we have*

$$\begin{aligned} f(x_1, x_2, x_3) =& M_\wedge x_1 M_\neg M_\to x_2 x_3 \\ =& M_\wedge (I_2 \otimes M_\neg M_\to) x_1 x_2 x_3 \\ :=& M_f x_1 x_2 x_3 = \delta_2[2\ 1\ 2\ 2\ 2\ 2\ 2\ 2] x_1 x_2 x_3. \end{aligned}$$

*It is easy to check that*

$$m_f = \mathrm{Row}_1(M_f) = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$

*is the truth table of $f$.*

*Then, by (13), the polynomial expression of $f$ is*

$$f(x_1, x_2, x_3) = x_1^1 x_2^1 x_3^0 = x_1 x_2 (x_3 + 1) = x_1 x_2 + x_1 x_2 x_3.$$

Walsh spectral expression is also a very useful expression of Boolean functions, and there are also some graph expressions. But they are beyond the scape of this paper, thus we do not discuss them here.

## 3 Conversion Between Truth Table and Polynomial Expression

Note that in vector form,

$$x_i \sim \begin{pmatrix} x_i^1 \\ x_i^0 \end{pmatrix} = \begin{pmatrix} x_i \\ x_i + 1 \end{pmatrix},$$

and for an $n \times m$ matrix $A$, $A(I_m \otimes B) = A \otimes B$. Then by the matrix form (15) of Boolean function $f$, we have

$$\begin{aligned} f(x) &= m_f \begin{pmatrix} x_1 \\ x_1 + 1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_2 + 1 \end{pmatrix} \cdots \begin{pmatrix} x_n \\ x_n + 1 \end{pmatrix} \\ &= m_f \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ x_n \end{pmatrix} \\ &= m_f \left( \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{n} \right) \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \cdots \begin{pmatrix} 1 \\ x_n \end{pmatrix} \\ &:= m_f P_n \xi_n, \end{aligned}$$

where

$$P_n = \left( \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{n} \right), \tag{16}$$

and

$$\xi_n = \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \cdots \begin{pmatrix} 1 \\ x_n \end{pmatrix} = (1, x_n, x_{n-1}, x_{n-1}x_n, x_{n-2}, \cdots, x_1 x_2 \cdots x_n)^T \tag{17}$$

is a basis of the polynomials on $GF^n(2)$. Then $m_f P_n \xi_n$ is already a polynomial.

Alternatively, the standard polynomial expression (14) of a Boolean function $f$ can be written as

$$f(x) = \beta \eta_n \tag{18}$$

where

$$\eta_n = (1, x_1, \cdots, x_n, x_1x_2, \cdots, x_{n-1}x_n, \cdots, x_1 x_2 \cdots x_n)^T$$

arranged as a natural alphabetic and power increasing order. To convert $m_f P_n \xi_n$ into the standard polynomial expression, we just need to reorder the entries in $\xi_n$.

**Lemma 3.1** *Let $\mu_{i_1,i_2,\cdots,i_r}$, $i_1 < i_2 \cdots < i_t$ be the positions where $x_{i_1}x_{i_2}\cdots x_{i_t}$ appear in $\xi_n$. Then*

$$\mu_{i_1,i_2,\cdots,i_r} = \sum_{j=1}^{r} 2^{n-i_j} + 1. \tag{19}$$

*Proof.* By direct computation, we know that for any $j$, the position where $x_{n-j}$ appears for the first time is $\mu_{n-j}$=$2^j + 1$. And then $x_{n-j}x_{n-k}$ appears at $2^k$ after $x_{n-k}$, thus its position is $2^j + 2^k + 1$.

Then for any $x_{n-i_1}x_{n-i_2}\cdots x_{n-i_t}$, we can locate $x_{n-i_1}, x_{n-i_1}x_{n-i_2}, \cdots, x_{n-i_1}x_{n-i_2}\cdots x_{n-i_t}$ one after another in a sequence, and in that way the conclusion follows. □

Using Lemma 3.1, we construct $\Phi_n$ as follows

$$\Phi_n = \delta_{2^n}[1, \phi_1, \phi_2, \cdots, \phi_n], \tag{20}$$

where $\phi_r = (\mu_{1,2,\cdots,r}, \mu_{2,\cdots,r+1}, \cdots, \mu_{n-r+1,n-r+2,\cdots,n})$, $\quad r = 1, 2, \cdots, n$. Then one can check easily that

$$\xi_n = \Phi_n \eta_n.$$

Note that $\Phi_n^{-1} = \Phi_n^T$, the following theorem is obvious.

**Theorem 3.2** *Assume $m_f$ is the truth table of a Boolean function $f$, and the corresponding standard polynomial expression is (18). Then we have*

$$\begin{aligned} \beta &= m_f P_n \Phi_n \\ m_f &= \beta \Phi_n^T P_n^{-1}. \end{aligned} \tag{21}$$

We give an example to depict this.

**Example 3.3** *Recall Example 2.7. The truth table of $f$ is*

$$m_f = [0\,1\,0\,0\,0\,0\,0\,0].$$

*By straightforward computation we have*

$$P_3 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

*Then*

$$\begin{aligned} f(x) &= m_f P_3 \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \begin{pmatrix} 1 \\ x_3 \end{pmatrix} \\ &= [0\,0\,0\,0\,0\,0\,1\,1](1, x_3, x_2, x_2x_3, x_1, x_1x_3, x_1x_2, x_1x_2x_3)^T \\ &= x_1x_2 + x_1x_2x_3. \end{aligned}$$

*It is the same as the polynomial expression obtained in Example 2.7.*

## 4 Linearity of Boolean Functions

In this section, we consider the linearity of Boolean functions. Since Boolean functions can be expressed in polynomial form, the Boolean functions with only linear terms are called the linear Boolean functions. The linear structure is a generalized linearity. It was pointed out in introduction that the existence of linear structures is a weakness in cryptography. We first give the rigorous definition of linear structure.

**Definition 4.1** *Let $f : \mathcal{D}^n \to \mathcal{D}$ ba a Boolean function.*

1. *$a \in \mathcal{D}^n$ is called an invariant linear structure (ILS) of $f$, if $f(x + a) + f(x) = 0$.*
2. *$a \in \mathcal{D}^n$ is called a variant linear structure (VLS) of $f$, if $f(x + a) + f(x) = 1$.*
3. *Denote by*

$$\begin{array}{rcl} E_0 & := & \{a \in \mathcal{D}^n \mid f(x + a) + f(x) = 0\} \\ E_1 & := & \{a \in \mathcal{D}^n \mid f(x + a) + f(x) = 1\} \\ E & := & E_0 \cup E_1. \end{array} \qquad (22)$$

   *Then $E$ is called the linear structure subspace of $f$.*

4. *If $E \neq \{\mathbf{0}\}$, $f$ is called a Boolean function with linear structure (BFLS). For a BFLS, if $E_0 \neq \{\mathbf{0}\}$, it is said to be of type I, otherwise, it is said to be of type II.*

Though there have already been many efficient algorithms to check wether a Boolean function have linear structure[13], we can give a precise formula to calculate $E_0$ and $E_1$. Let $f : \mathcal{D}^n \to \mathcal{D}$ be a Boolean function with its structure matrix $M_f \in \mathcal{L}_{2\times 2^n}$. Denote by $a = \ltimes_{i=1}^{n} a_i$, $x = \ltimes_{i=1}^{n} x_i$. Then it is easy to see that $(a_1, \cdots, a_n) \in E_0$, if and only if

$$M_f M_{\oplus} a_1 x_1 M_{\oplus} a_2 x_2 \cdots M_{\oplus} a_n x_n = M_f x_1 x_2 \cdots x_n. \tag{23}$$

A straightforward computation shows that (23) is equivalent to

$$M_f M_{\oplus} \ltimes_{i=1}^{n-1} (I_{2^{2i}} \otimes M_{\oplus}) \ltimes_{i=1}^{n-1} \left(I_{2^i} \otimes W_{[2,2^i]}\right) ax = M_f x. \tag{24}$$

Define

$$\Psi_f := M_f M_{\oplus} \ltimes_{i=1}^{n-1} (I_{2^{2i}} \otimes M_{\oplus}) \ltimes_{i=1}^{n-1} \left(I_{2^i} \otimes W_{[2,2^i]}\right),$$

where $M_{\oplus} \ltimes_{i=1}^{n-1} (I_{2^{2i}} \otimes M_{\oplus}) \ltimes_{i=1}^{n-1} \left(I_{2^i} \otimes W_{[2,2^i]}\right)$ is a constant matrix depends only on $n$. Split $\Psi_f$ into $2^n$ blocks as

$$\Psi_f = [\psi_1 \ \psi_2 \ \cdots \ \psi_{2^n}],$$

where $\psi_k = \mathrm{Blk}_k(\Psi_f)$, $k = 1, 2, \cdots, 2^n$. Then the following result is straightforward verifiable.

**Proposition 4.2** *Let $\alpha = \ltimes_{i=1}^{n} a_i = \delta_{2^n}^{i}$. Then $(a_1, \cdots, a_n) \in E_0$, iff $\psi_i = M_f$. $(a_1, \cdots, a_n) \in E_1$, if and only if $\psi_i = M_{\neg} M_f$.*

**Example 4.3** *Recall Example 2.7. From the polynomial form*

$$f(x) = x_1x_2 + x_1x_2x_3,$$

*one sees easily that it is not linear Boolean function. Next, we will check whether it has a linear structure.*

*Since*

$$M_f = \delta_2[2\ 1\ 2\ 2\ 2\ 2\ 2\ 2],$$

*and*

$$M_{\neg} M_f = \delta_2[1\ 2\ 1\ 1\ 1\ 1\ 1\ 1],$$

*then*

$$\begin{array}{rllllllllllllllll} \Psi_f = \delta_2[2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2]. \end{array}$$

*Since only $\psi_8 = M_f$, we know that $E_0 = \{\mathbf{0}\}$ and $E_1 = \emptyset$, $f$ has no linear structure.*

## 5 Conclusion

Using semi-tensor product of matrices, Boolean functions can be expressed in matrix form. This paper showed that it is more convenient to convert a Boolean functions truth table into its polynomial expression using the semi-tensor product approach. The formula to find the linear structure is also given in the matrix form. The semi-tensor product approach is a new method to represent Boolean functions, this paper only investigates a few usages of this approach. We believe that more properties of Boolean functions can be revealed using this approach in the future.

## References

[1] F. Macwilliams, N. Sloane. *The Theory of Error-Correcting Codes*[M]. Amsterdam: North-Holland, 1977.

[2] Q. Wen, X. Niu, Y. Yang. *Boolean Functions in Morden Cryptography*[M]. Beijing: Science Press, 2010. (In Chinese)

[3] C. Carlet. Boolean functions for cryptography and error-correcting codes[C]// Y. Crama and P. Hammer eds. *Boolean Methods and Models in Mathematics, Computer Science, and Engineering*. Cambridge: Cambridge Univ. Press, 2010.

[4] Y. Zhang. *Cryptographic Properties of Permutation Symmetric Boolean Functions*[D]. Beijing: Graduate University of Chinese Academy of Sciences, 2010. (In Chinese)

[5] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets[J]. *J Theoretical Biology*, 1969, 22(3): 437-467

[6] T. Ideker, T. Galitski, L. Hood. A new approach to decoding life: systems biology[J]. *Annu Rev Genomics Hum Genet*, 2001, 2: 343-372.

[7] C. Farrow, J. Heidel, H. Maloney, and J. Rogers. Scalar equations for synchronous Boolean networks with biological applications[J]. *IEEE Trans Neural Networks*, 2004, 15(2): 348C354.

[8] M. Ben-Or, N. Linial. *Collective coin flipping*[M]. Israel: Leibniz Center for Research in Computer Science, Dept of Computer Science, Hebrew University of Jerusalem, 1987.

[9] P. Hammer, R. Holzman. Approximations of pseudo-Boolean functions: applications to game theory[J]. *Mathematical Methods of Operations Research*, 1992, 36(1): 3-21.

[10] R. O'Donnell. Some topics in analysis of Boolean functions[C]. *Proceedings of the 40th annual ACM symposium on Theory of computing*. Victoria, Canada: 2008. 569-578.

[11] S. Akers. Binary decision diagrams[J]. *IEEE Trans Computer*, 1978, C-27(6): 509-516.

[12] M. Wachter, R. Haenni. Propositional DAGs: a New Graph-Based Language for Representing Boolean Functions[C]. *Proc KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning*. Lake District, UK: 2006. 6: 275-285.

[13] S. Dubuc. Characterization of Linear Structures[J]. *Designs, Codes and Cryptography*, 2001, 22(1): 33-45.

[14] D. Cheng. Semi-tensor product of matrices and its applications – A survey[C]. *ICCM 2007*. Zhejiang, China: 2007. 3: 641-668.

[15] D. Cheng, H. Qi, Z. Li. *Analysis and Control of Boolean Networks: A Semi-tensor Product Approach*[M]. London: Springer, 2011.

## 布尔函数的半张量积方法

赵寅[†], 高旭[‡], 程代展[†]

[†] 中国科学院数学与系统科学研究院系统控制重点实验室, 北京100190

Email: zhaoyin@amss.ac.cn, dcheng@iss.ac.cn

[‡] 中国科学技术大学数学科学学院, 合肥230026

Email: gaoxu@mail.ustc.edu.cn

**摘要：** 利用矩阵的半张量积, 布尔函数可以被表示为矩阵形式. 通过这个方法, 我们给出了布尔函数从真值表到多项式形式转换的一个简洁的证明. 并且研究了布尔函数的线性结构.

**关键词：**布尔函数, 半张量积, 真值表, 多项式表示, 线性结构.