Conversion between Logic and Algebraic Expressions of Boolean Control Networks

Cailu Wang 1* and Yuegang Tao 2*

1 School of Automation, Beijing Institute of Technology, Beijing 100081, China; cailu_wang@163.com
2 School of Artificial Intelligence, Hebei University of Technology, Tianjin 300130, China
* Correspondence: yuegangtao@hebut.edu.cn

Received: 12 September 2020; Accepted: 5 October 2020; Published: 15 October 2020

Abstract: The conversion between logic and algebraic expressions of Boolean control networks plays a worthy role in the analysis and design of digital circuits. In this paper, for a single Boolean function, a direct conversion between the minterm canonical form and the structure matrix is provided. For a Boolean control network consisting of systems of Boolean functions, two algorithms are developed to achieve the mutual conversion between the logic and algebraic expressions. The presented algorithms decrease exponentially the complexity of the semi-tensor product based method. Some numerical examples are given to demonstrate the algorithms and to compare our method with the existing ones.

Keywords: Boolean control network (BCN); logic expression; algebraic expression; canonical form; conversion algorithm; computational complexity

1. Introduction

Boolean control network (BCN) is a logical system pioneered by Kauffman in [1] for initially describing genetic regulatory networks, which provides useful modeling tools for dynamical systems whose state-variables take only two values (‘True’ or ‘False’). With the development of information technologies, BCNs have become an effective way to analyze and design circuits by algebraic means in terms of logic gates, and have served as a fundamental approach to digital circuit design, computer programming, and so on (see, e.g., [2–7]). With the development of digital circuit, the research on logic synthesis becomes particularly important. Recently, Rushdi and Ahmad [5] used a novel method for obtaining the parametric general solutions of the ‘big’ Boolean equation, which cover the most prominent type among basic problems of digital circuit design. Bandyopadhyay et al. [4] analyzed the effects of different graph-based intermediate representations, including binary decision diagram, and-inverter graph and majority inverter graph, and their usability in making efficient circuit realisations. Wang and Tao [3] introduced the Karnaugh maps of logical systems and used in the analysis and design of clocked sequential circuits and in the simplification of multivalue gate circuits.

In addition to the traditional logical expression, there are some representations to describe dynamics and estimate performances of BCNs. For example, logic differential calculus [8] represents a mathematical methodology developed for analysis of Boolean functions, which enables us to consider the existing estimates of system importance, as well as some additional properties, of BCNs from the unique methodological standpoint (see, e.g., [9,10]).

The semi-tensor product (STP) of matrices is developed in [11] to convert a Boolean function into an algebraic function, and to model a BCN by a standard discrete-time linear system. A comparison of different representations of Boolean functions can be found in [12]. Describing a logical dynamic system as a linear system is useful for studying BCNs in a control-theoretic framework. Examples include the controllability and observability [13–16], periodicity and stability [17–22], optimal control [23,24],
identification [25], etc. The computation complexity is a series problem in directly calculating the network transition matrix based on STP, which frames a limitation on the application of algebraic state-space approach. It has been calculated in [3] that the complexity of STP based method is $O(2^{(m+n)n})$. Recently, Sarda et al. [26,27] proposed a Khatri-Rao product based method to transform the logical form of a BCN into a discrete time bilinear system, which has a complexity of $O(2^{m+2n})$. In other words, the complexity of the STP based method is $2^{(m+n)n}$ times that of the Khatri-Rao product based method. This paper will provide a constructive and straightforward method with a complexity of $O(2^{m+n})$ to convert the logic expression to the algebraic expression of a BCN. More concretely, the complexity of the STP based method presented in [11] is $2^{5(m+n)}$ times that of the method given in this paper, and the Khatri-Rao product based method presented in [26,27] is $2^{n}/n$ times that of ours.

For a BCN, converting the algebraic expression into the logic expression is an integral part for learning the logical meaning of the algebraic result from the algebraic expression. In [16], a mechanical procedure is provided to retrieve the logical network and logical dynamic equations from the network matrix of a BCN. The conversion formula given in [16] is an implicit equation, which converts the algebraic expression for a Boolean function into an irregular form of the logic expression. This paper will provide an explicit equation to directly convert the algebraic expression to a canonical form of the logic expression.

The remainder of this paper is organized as follows. Section 2 introduces some basic concepts about the logic and algebraic expressions of Boolean functions. Section 3 gives a conversion between the minterm canonical form and the structure matrix of a Boolean function. The algorithms for conversion between the logic and algebraic expressions of a BCN are presented in Section 4. The conclusions and future works are drawn in Section 5.

2. Preliminaries

This section reviews some basic concepts from Boolean functions and Boolean control networks (see, e.g., [6,7]).

A Boolean function is a function of the form

$$f : \Delta^n \rightarrow \Delta,$$

where $\Delta = \{\text{True, False}\}$ is the logical domain and $n$ is the arity of the function. Any logical function with $n$ variables can be expressed as a propositional formula in variables $x_1, x_2, \ldots, x_n$. For simplicity, the logical disjunction $x \lor y$, conjunction $x \land y$ and negation $\neg x$ are denoted by $x + y$, $xy$ and $\bar{x}$, respectively.

A truth table specifies the values of a Boolean function for every possible combination of values of the variables in the function. Label the rows of the truth table of Boolean function (1) by $0, 1, \ldots, 2^n - 1$, respectively, and denote by $s_i$ the function value in the row $i$ of the truth table, where $i \in \{0, 1, \ldots, 2^n - 1\}$.

A minterm of $n$ variables is a product of $n$ literals in which each variable appears exactly once in either true or complemented form, but not both. The minterm which corresponds to the row $i$ of truth table is designated as $m_i$. Boolean function (1) can uniquely be written as a sum of minterms, i.e.,

$$f(x_1, x_2, \ldots, x_n) = m_{i_1} + m_{i_2} + \cdots + m_{i_k} := \sum m(i_1, i_2, \ldots, i_k),$$

which is called the minterm canonical form of $f$.

**Definition 1 ([11])**. For $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, the semi-tensor product (STP) of $A$ and $B$ is defined as

$$(A \leftthreetimes B) = (A \otimes I_{n/p}) \times (B \otimes I_{n/p}),$$

where $\alpha$ is the least common multiple of $n$ and $p$, $I$ is the identity matrix, $\otimes$ and $\times$ are the Kronecker product and conventional product of matrices, respectively.
The STP has left-right dual forms. The STP mentioned in this paper is the left one. Using STP, a Boolean function can be converted into an algebraic form. Denote by \( \delta^n_j \) the j-th column of identity matrix \( I_n \), where \( j \in \{1, 2, \ldots, n\} \). Represent the logical values “True” and “False” by

\[
\delta^1_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \delta^2_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\]

respectively. Then, Boolean function (1) can be equivalently represented as

\[
f : \{\delta^1_2, \delta^2_2\}^n \to \{\delta^1_2, \delta^2_2\}.
\]

For the sake of brevity, a matrix of the form \([\delta^1_n \delta^2_n \ldots \delta^n_n]\) is briefly denoted by \( \delta_n[j_1, j_2, \ldots, j_k] \).

**Lemma 1** ([11]). For Boolean function (1), there exists a unique binary matrix \( M_f \in \{0, 1\}^{2^m \times 2^n} \) such that

\[
f(x_1, x_2, \ldots, x_n) = M_f \times x_1 \times x_2 \times \cdots \times x_n,
\]

in which \( M_f \) is called the structure matrix of \( f \).

A Boolean network (BN) with \( n \) variables is described as

\[
x_1(t+1) = f_1(x_1(t), x_2(t), \ldots, x_n(t)), \\
x_2(t+1) = f_2(x_1(t), x_2(t), \ldots, x_n(t)), \\
\vdots \\
x_n(t+1) = f_n(x_1(t), x_2(t), \ldots, x_n(t)),
\]

(4)

where \( f_j (j \in \{1, 2, \ldots, n\}) \) are Boolean functions, and \( x_j(\cdot) \in \{0, 1\} \ (j \in \{1, 2, \ldots, n\}) \) are states.

**Lemma 2** ([11]). Let \( x(t) = \times_{j=1}^n x_j(t) \in \{0, 1\}^{2^n} \). Then, there exists a unique matrix \( L \in \{0, 1\}^{2^n \times 2^n} \) such that BN (4) can be converted into a standard discrete-time dynamic system

\[
x(t+1) = L \times x(t), \ t = 0, 1, \ldots,
\]

(5)

where \( L \) is called the state transition matrix of BN (4).

A Boolean control network (BCN) with \( m \) inputs and \( p \) outputs is described as

\[
x_1(t+1) = f_1(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t)), \\
\vdots \\
x_n(t+1) = f_n(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t)), \\
y_1(t) = g_1(x_1(t), \ldots, x_n(t)), \\
\vdots \\
y_p(t) = g_p(x_1(t), \ldots, x_n(t)),
\]

(6)

where \( f_j (j \in \{1, 2, \ldots, n\}) \) and \( g_r (r \in \{1, 2, \ldots, p\}) \) are Boolean functions, \( x_j(\cdot) \ (j \in \{1, 2, \ldots, n\}) \) are states, \( u_k(\cdot) \ (k \in \{1, 2, \ldots, m\}) \) are inputs and \( y_r(\cdot) \ (r \in \{1, 2, \ldots, p\}) \) are outputs. Denote by \( N_{\text{in}} \) the network of states and inputs, and by \( N_{\text{out}} \) the network of states and outputs.

Since the dynamics of a BCN is composed of a set of Boolean functions, the STP can be used to establish the algebraic expression of BCN (6) as follows:

\[
\begin{align*}
x(t+1) &= L \times x(t) \times u(t), \\
y(t) &= H \times x(t),
\end{align*}
\]

(7)
where \( x(t) = \kappa_{j=1}^{m} x_j(t) \in \{0,1\}^{2^n} \) is state vector, \( u(t) = \kappa_{k=1}^{p} u_k(t) \in \{0,1\}^{2^p} \) and \( y(t) = \kappa_{r=1}^{P} y_r(t) \in \{0,1\}^{2^p} \) are input and output vectors, respectively, \( L \in \{0,1\}^{2^n \times 2^{n+p}} \) is the control-depending network transition matrix, \( H \in \{0,1\}^{2^p \times 2^n} \) is the transition matrix from \( x(t) \) to \( y(t) \) (see, e.g., [21–23]).

The STP based method has been presented in ([11], Theorem IV.6) for obtaining the algebraic expression of BCNs by introducing the swap matrix, dummy matrix and power reducing matrix, which has a very high computational complexity of \( O(2^{6(m+n)}) \) (see [3], Theorem 1). Recently, Sarda et al. [26,27] proposed a Khatri-Rao product based method to obtain such an algebraic expression.

**Definition 2 ([28]).** For \( A \in \mathbb{R}^{m \times p} \) and \( B \in \mathbb{R}^{n \times p} \), the Khatri-Rao product of \( A \) and \( B \) is defined to be a partitioned matrix of order \( mn \times p \) as follows:

\[
A \ast B = [A_1 \otimes B_1, A_2 \otimes B_2, \ldots, A_p \otimes B_p],
\]

where \( A_r \) (resp. \( B_r \)) \((r \in \{1, 2, \ldots, p\})\) is the \( r \)-th column of \( A \) (resp. \( B \)), and \( \otimes \) is the Kronecker product of matrices.

**Lemma 3 ([26,27]).** For BCN (6), the control-depending network transition matrix is \( L = \sum_{j=1}^{n} M_{j}^{in} \), and the transition matrix from \( x(t) \) to \( y(t) \) is \( H = \sum_{r=1}^{P} M_{r}^{out} \), where \( M_{j}^{in} \) and \( M_{r}^{out} \) are the structure matrices of \( f_j \) and \( g_r \), respectively.

It is known that at most \( mnp \) multiplication operations are required to compute the Khatri-Rao product of two matrices with sizes of \( m \times p \) and \( n \times p \). Then, at most \( 2^n 2^{m+n} = 2^{m+2n} \) (resp. \( 2^p 2^n = 2^{n+p} \)) operations are required to compute the transition matrix \( L \) (resp. \( H \)) by using the lemma above. Hence, the computational complexity of Lemma 3 is \( O(2^{m+2n}) \), which decreases exponentially that of the STP based method. Quantitatively, the complexity of the STP based method is \( 2^{5m+4n} \) times that of the Khatri-Rao product based method. This paper will provide a method with a much lower complexity.

### 3. Conversion of Expressions of Boolean Functions

This section presents a direct conversion between the minterm canonical form and the structure matrix of a Boolean function.

#### 3.1. From Logic Expression to Algebraic Expression

There exists a one-to-one correspondence between the truth table and the structure matrix of a Boolean function.

**Lemma 4 ([29]).** If the truth table of Boolean function (1) is \( (s_0 \ s_1 \ \ldots \ s_{2^n-1})^T \), then the structure matrix of \( f \) is

\[
M_f = \begin{bmatrix}
    s_{2^n-1} & \ldots & s_1 & s_0 \\
    1-s_{2^n-1} & \ldots & 1-s_1 & 1-s_0
\end{bmatrix}.
\]

The structure matrix of a Boolean function can be constructed from the minterm canonical form.

**Theorem 1.** If the minterm canonical form of Boolean function (1) is shown in (2), then the structure matrix of \( f \) is

\[
M_f = \delta_2[c_{2^n-1}, \ldots, c_1, c_0],
\]

where

\[
c_i = \begin{cases}
    1, & i \in \{i_1, i_2, \ldots, i_k\}, \\
    2, & i \in \{0,1, \ldots, 2^n-1\} \setminus \{i_1, i_2, \ldots, i_k\}.
\end{cases}
\]
**Proof.** The minterms presented in (2) are in one-to-one correspondence with the logical value 1 in the truth table. Suppose that the truth table of $f$ is $(s_0, s_1, \ldots, s_{2^n-1})^T$. Then $s_i = 1$ if and only if $m_i$ is a minterm of $f$, i.e., $i \in \{i_1, i_2, \ldots, i_k\}$. It follows from Lemma 4 that $c_i = 1$ if and only if $s_i = 1$. Hence, $c_i = 1$ if and only if $i \in \{i_1, i_2, \ldots, i_k\}$. □

Let us compute the structure matrix of the Boolean function given in ([29], Example 2.5) using Theorem 1.

**Example 1.** Consider the Boolean function

$$f(x_1, x_2, x_3) = (x_1 \leftrightarrow \neg x_2) \land (x_2 \leftrightarrow \neg x_3) \land (x_3 \leftrightarrow \neg x_1 \land \neg x_2).$$

The minterm canonical form of $f$ is $m_2 (= \bar{x}_1 x_2 \bar{x}_3)$. Then, in (8),

$$c_i = \begin{cases} 1, & i = 2, \\ 2, & i \in \{1, 3, 4, 5, 6, 7, 8\}. \end{cases}$$

Hence, the structure matrix of $f$ is $M_f = \delta_2[2, 2, 2, 2, 2, 1, 2, 2]$.

Comparing the above example with ([29], Example 2.5), the method based on the minterm canonical form can avoid the complex calculations of STP.

### 3.2. From Algebraic Expression to Logic Expression

After studying a logical dynamic system in the algebraic state space, it is necessary to convert the algebraic expression back to the logic expression in order to design digital circuits to implement the desired control functions (see, e.g., [3, 5]). For a Boolean function, it is claimed in [16] that ‘converting an algebraic form back to logical form is not an easy job’, and a ‘mechanical procedure’ is provided for converting the algebraic expression back to the logic expression as below.

**Lemma 5 ([16]).** If Boolean function (1) has an algebraic expression as (3), then

$$f(x_1, x_2, \ldots, x_n) = [x_1 \land f_1(x_2, \ldots, x_n)] \lor [\neg x_1 \land f_2(x_2, \ldots, x_n)],$$

where $M_f = (M_{f_1} | M_{f_2})$, i.e., the structure matrices of $f_1$ and $f_2$ are the first and last half of $M_f$, respectively.

It can be seen that Equation (9) is implicit, which converts the algebraic expression of a Boolean function into an irregular form of the logic expression. In fact, the minterm canonical form of a Boolean function can directly be obtained from the structure matrix.

**Theorem 2.** If the structure matrix of Boolean function (1) is shown in (8), then the minterm canonical form of $f$ is

$$f(x_1, x_2, \ldots, x_n) = \sum_{i \in S} m_i,$$

where $S = \{i \in \{0, 1, \ldots, 2^n - 1\} \mid c_i = 1\}$.

**Proof.** Suppose that the truth table of $f$ is $(s_0, s_1, \ldots, s_{2^n-1})^T$. By Lemma 4,

$$S = \{i \mid c_i = 1\} = \{i \mid s_i = 1\} = \{i \mid m_i \text{ is a minterm of } f\}.$$

Then, the minterm canonical form of $f$ is (10). □

In contrast to the implicit Equation (9) given in Lemma 5, Equation (10) presented in Theorem 2 is explicit, by which the algebraic expression is directly converted into a canonical form of the logic function.
expression. Let us compute the minterm canonical form of the Boolean function whose structure matrix is given in ([16], Example 6) using Theorem 2.

**Example 2.** Consider the Boolean function \( f \) with structure matrix
\[
M_f = \delta_2 [1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2].
\]

In (8), \( c_1 = c_2 = c_6 = c_7 = c_8 = c_{10} = c_{12} = c_{15} = 1 \). By (10), the minterm canonical form of \( f \) is
\[
f(x_1, x_2, x_3, x_4) = \sum \{1, 2, 6, 7, 8, 10, 12, 15\}.
\]

Furthermore, the minimization form of \( f \) is
\[
f(x_1, x_2, x_3, x_4) = x_1 \bar{x}_3 \bar{x}_4 + x_2 x_3 x_4 + \bar{x}_1 x_3 \bar{x}_4 + \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4.
\]

Comparing the above example with ([16], Example 6), the explicit equation makes the conversion straightforward. Moreover, given the truth table of a Boolean function, it is possible to write the function as various propositional formulas with irregular forms, while the minterm canonical form is unique up to the order of minterms.

### 4. Conversion of Expressions of Boolean Control Networks

This section extends the presented approach to Boolean control networks to achieve the mutual conversion between logic and algebraic expressions.

#### 4.1. From Logic Expression to Algebraic Expression

A method for computing the next state of BN (4) is provided in [3] based on the algebraic expression as below.

**Lemma 6 ([3]).** For BN (5), if \( x(t) = \delta_2^{2^n-i} \), then \( x(t+1) = L \times \delta_2^{2^n-i} = \delta_2^{2^n-i} c_i \), where
\[
c_i = \sum_{j=1}^{n} s^{(j)}_i \times 2^{n-j}, \quad i = 0, 1, \ldots, 2^n - 1,
\]
and \( s^{(j)}_i \) is the function value in the row \( i \) of the truth table of \( f_j \).

The lemma above can be extend to BCNs for converting the logic expression into the algebraic expression.

**Theorem 3.** For BCN (6), let \( c_i \) be defined by
\[
c_i = \sum_{j=1}^{n} s^{(j)}_i \times 2^{n-j}, \quad i = 0, 1, \ldots, 2^{n+n} - 1,
\]
where \( s^{(j)}_i \) is the function value in the row \( i \) of the truth table of \( f_j \). Then, the transition matrix of \( N_{in} \) is
\[
L = \delta_2^{2^n - c_{2^{n+n-1}, \ldots, 2^n - c_1, 2^n - c_0}}.
\]

In the theorem above, it can be seen that \( c_i \) is the decimal representation of the \( n \)-bit binary number whose \( j \)th bit code is \( s^{(j)}_i \).
Proof. Let \( L \) be the transition matrix of \( N_{in} \) and \( x(t) ∨ u(t) = δ_{2^n}^{2^{i-1}} \). It can be inferred from (7) and Lemma 6 that

\[
x(t + 1) = L ∨ x(t) ∨ u(t) = L ∨ δ_{2^n}^{2^{i-1}} = δ_{2^n}^{2^{i-1}}.
\]

This implies that the \((i + 1)\)th column of \( L \) from right is \( δ_{2^n}^{2^{i-1}} \), i.e., \( L \) is equal to (12).

The following lemma for constructing the transition matrix of \( N_{out} \) can be obtained in almost the same way.

**Theorem 4.** For BCN (6), let \( \tilde{c}_i \) be defined by

\[
\tilde{c}_i = \sum_{r=1}^{p} s_i^{(r)} × 2^{p-r}, \ i = 0, 1, \ldots, 2^n - 1,
\]

where \( s_i^{(r)} \) is the function value in the row \( i \) of the truth table of \( g_r \). Then, the transition matrix of \( N_{out} \) is

\[
H = δ_{2^n}[2^p - \tilde{c}_{2^{n-1}}, \ldots, 2^p - \tilde{c}_1, 2^p - \tilde{c}_0].
\]

The following algorithm for converting the logic expression to the algebraic expression of a BCN can then be developed.

Next, let us consider the computational complexity of Algorithm 1 starting from the minterm canonical form.

**Algorithm 1** Algorithm for converting the logic expression to the algebraic expression of BCN (6).

Input: Logic expression of \( N_{in} \) (resp. \( N_{out} \)).
Output: Transition matrix \( L \) (resp. \( H \)).

Setp 1. For \( N_{in} \), construct the truth table of \( f_j \) for each \( j \in \{1, 2, \ldots, n\} \), which is denoted by

\[
(s_j^{(0)}, s_j^{(1)}, \ldots, s_j^{(2^n-1)})^T.
\]

Setp 2. Compute \( c_j \) by Formula (11).

Setp 3. The control-depending network transition matrix is given in (12).

Setp 4. For \( N_{out} \), construct the truth table of \( g_r \) for each \( r \in \{1, 2, \ldots, p\} \), which is denoted by

\[
(s_r^{(0)}, s_r^{(1)}, \ldots, s_r^{(2^{n-1})})^T.
\]

Setp 5. Compute \( \tilde{c}_j \) by Formula (13).

Setp 6. The transition matrix from \( x(t) \) to \( y(t) \) is given in (14).

**Theorem 5.** Algorithm 1 has a complexity of \( O(2^{m+n}n) \).

Proof. In Step 1 (resp. Step 4), the truth tables of \( f_j \) (resp. \( g_r \)) can be obtained directly from their minterm canonical forms without any calculation. In Step 2, the computation cost of \( c_j \) is one exponent and \( n \) multiplication together with \( n - 1 \) addition. Hence, the number of operations of Step 2 is at most \( 2n × 2^{m+n} = 2^{m+n+1} \). Step 3 requires \( 2^{m+n} \) subtraction operations to compute the transition matrix \( L \). Hence, the computation cost of \( N_{in} \) is \( 2^{m+n+1}n + 2^{m+n} = 2^{m+n}(2n + 1) \). Similarly, the computation cost of \( \tilde{c}_i \) is \( 2p \), and hence the number of operations of Step 5 is at most \( 2p × 2^n = 2^{n+1}p \). Step 6 requires \( 2^n \) subtraction operations to compute the transition matrix \( H \). Hence, the computation cost of \( N_{out} \) is \( 2^{n+1}p + 2^n = 2^n(2p + 1) \). Therefore, the computational complexity of Algorithm 1 is \( O(2^{m+n}n) \).
from Theorems 1 and 2 that the conversion between the minterm canonical form and the structure matrix of a logical function does not need any calculation. This implies that Algorithm 1, Lemma 3 and ([11], Theorem IV .6) can be viewed as starting from the same point. Then, the computational complexity of the STP and Khatri-Rao product based methods are $2^{5(m+n)}$ and $2^n/n$ times that of Algorithm 1, respectively.

Let us illustrate Algorithm 1 with the following example.

**Example 3.** For the BCN depicted in Figure 1, the logical dynamic equations are

$$\begin{align*}
  x_1(t + 1) &= x_2(t) u_1(t), \\
  x_2(t + 1) &= \bar{u}_2(t), \\
  x_3(t + 1) &= x_1(t) + x_2(t), \\
  y_1(t) &= x_3(t).
\end{align*}$$  \hfill (15)

Let us give the algebraic expression of such a BCN using Algorithm 1.

**Step 1.** For $N_{in}$, the truth table of $f_j$ for each $j \in \{1, 2, 3\}$ is listed in Table 1.

**Step 2.** By (11), $c_i$ ($i \in \{0, 1, \ldots, 31\}$) is presented in the last column of Table 1.

**Step 3.** By (12), the control-depending network transition matrix is

$$L = \delta_8[3, 1, 7, 5, 3, 1, 7, 5, 7, 5, 7, 5, 7, 5, 7, 5, 3, 1, 7, 5, 3, 1, 7, 5, 8, 6, 8, 6, 8, 6, 8, 6, 8, 6].$$ \hfill (16)

**Step 4.** For $N_{out}$, the truth table of $g_1$ is given in Table 2.

**Step 5.** By (13), $\tilde{c}_i$ ($i \in \{0, 1, \ldots, 7\}$) is presented in the last column of Table 2.

**Step 6.** By (14), the transition matrix from $x(t)$ to $y(t)$ is

$$H = \delta_2[2, 1, 2, 1, 2, 1, 2, 1].$$ \hfill (17)

Hence, the algebraic expression of BCN (15) is

$$\begin{align*}
  x(t + 1) &= L \times x(t) \times u(t), \\
  y(t) &= H \times x(t),
\end{align*}$$  \hfill (18)

where transition matrices $L$ and $H$ are given in (16) and (17), respectively.
Table 1. Truth table relative to $N_{in}$ (The indexed parameters $a$ and $b$ given together with other parameters are needed only further for Algorithm 2).

| $i$ | $x_1$ | $x_2$ | $x_3$ | $u_1$ | $u_2$ | $f_1$ | $f_2$ | $f_3$ | $c_i$ |
|-----|--------|--------|--------|-------|-------|-------|-------|-------|-------|
| 0   | 0      | 0      | 0      | 0     | 0     | 0     | 0     | 1     | 0     | 2     |
| 1   | 0      | 0      | 0      | 0     | 1     | 0     | 0     | 0     | 0     | 0     |
| 2   | 0      | 0      | 0      | 1     | 0     | 0     | 1     | 0     | 2     | 0     |
| 3   | 0      | 0      | 0      | 1     | 1     | 0     | 0     | 0     | 0     | 0     |
| 4   | 0      | 0      | 1      | 0     | 0     | 0     | 1     | 0     | 2     | 0     |
| 5   | 0      | 0      | 1      | 0     | 1     | 0     | 0     | 0     | 0     | 0     |
| 6   | 0      | 0      | 1      | 1     | 0     | 0     | 1     | 0     | 2     | 0     |
| 7   | 0      | 0      | 1      | 1     | 1     | 0     | 0     | 0     | 0     | 0     |
| 8   | 0      | 1      | 0      | 0     | 0     | 0     | 0     | 1     | 1     | 3     |
| 9   | 0      | 1      | 0      | 0     | 0     | 0     | 1     | 0     | 0     | 1     |
| 10  | 0      | 1      | 0      | 1     | 0     | 0     | 1     | 0     | 1     | 1     |
| 11  | 0      | 1      | 0      | 1     | 1     | 0     | 0     | 1     | 5     | 0     |
| 12  | 0      | 1      | 1      | 0     | 0     | 0     | 1     | 1     | 3     | 0     |
| 13  | 0      | 1      | 1      | 0     | 1     | 0     | 0     | 1     | 5     | 0     |
| 14  | 0      | 1      | 1      | 1     | 0     | 1     | 1     | 1     | 7     | 0     |
| 15  | 0      | 1      | 1      | 1     | 1     | 0     | 1     | 1     | 5     | 0     |

| $i$ | $x_1$ | $x_2$ | $x_3$ | $g_1$ | $\tilde{c}_i$ | $\tilde{b}_{1i}$ | $\tilde{a}_{1i}$ |
|-----|-------|-------|-------|-------|---------------|------------------|------------------|
| 0   | 0     | 0     | 0     | 1     | 1            | 1                | 1                |
| 1   | 0     | 0     | 1     | 0     | 0            | 1                | 0                |
| 2   | 0     | 1     | 0     | 1     | 1            | 0                | 1                |
| 3   | 0     | 1     | 1     | 0     | 0            | 1                | 1                |
| 4   | 1     | 0     | 0     | 1     | 1            | 1                | 1                |
| 5   | 1     | 0     | 1     | 0     | 0            | 1                | 1                |
| 6   | 1     | 1     | 0     | 1     | 1            | 1                | 1                |
| 7   | 1     | 1     | 1     | 0     | 0            | 1                | 1                |

Table 2. Truth table relative to $N_{out}$ (The indexed parameters $\tilde{a}$ and $\tilde{b}$ given together with other parameters are needed only further for Algorithm 2).

| $i$ | $x_1$ | $x_2$ | $x_3$ | $g_1$ | $\tilde{c}_i$ | $\tilde{b}_{1i}$ | $\tilde{a}_{1i}$ |
|-----|-------|-------|-------|-------|---------------|------------------|------------------|
| 0   | 0     | 0     | 0     | 1     | 1            | 1                | 1                |
| 1   | 0     | 0     | 1     | 0     | 0            | 1                | 0                |
| 2   | 0     | 1     | 0     | 1     | 1            | 0                | 1                |
| 3   | 0     | 1     | 1     | 0     | 0            | 1                | 1                |
| 4   | 1     | 0     | 0     | 1     | 1            | 1                | 1                |
| 5   | 1     | 0     | 1     | 0     | 0            | 1                | 1                |
| 6   | 1     | 1     | 0     | 1     | 1            | 1                | 1                |
| 7   | 1     | 1     | 1     | 0     | 0            | 1                | 1                |
Algorithm 2 Algorithm for converting the algebraic expression back to the logic expression of BCN (7). 
Input: Transition matrix \( L \) (resp. \( H \)).
Output: Logic expression of \( N_{in} \) (resp. \( N_{out} \)).

Step 1. For \( N_{in} \), let \( L = \delta_{2^n} [l_1, l_2, \ldots, l_{2^{m+n}}] \), where \( l_v \in \{1, 2, \ldots, 2^n\} \) represents the \( v \)th column of transition matrix \( L \) and \( v \in \{1, 2, \ldots, 2^{m+n}\} \). Let \( a_i \) be defined by

\[
a_i = 2^n - l_{2^{m+n-1} - j}, \quad i = 0, 1, \ldots, 2^{m+n} - 1.
\]

(19)

In fact, \( a_i \) is constructed by conversely applying (12), which has the same value as \( c_i \).

Step 2. Denote by \( b_{ji} \) the \( j \)th bit code of the binary representation of decimal number \( a_i \), where \( i \in \{0, 1, \ldots, 2^{m+n} - 1\} \) and \( j \in \{1, 2, \ldots, n\} \). In fact, \( b_{ji} \) is obtained by conversely applying (11), which has the same value as \( s_i^{(j)} \).

Step 3. Construct the minterm canonical form of \( f_j \) for each \( j \in \{1, 2, \ldots, n\} \) as follows:

\[
f_j(x_1, \ldots, x_n, u_1, \ldots, u_m) = \sum_{i \in S_j} m_{ij}
\]

where \( S_j = \{i \in \{0, 1, \ldots, 2^{m+n} - 1\} \mid b_{ji} = 1\} \).

Step 4. For \( N_{out} \), let \( H = \delta_{2^n} [h_1, h_2, \ldots, h_{2^n}] \), where \( h_v \in \{1, 2, \ldots, 2^n\} \) represents the \( v \)th column of transition matrix \( H \) and \( v \in \{1, 2, \ldots, 2^n\} \). Let \( \bar{a}_i \) be defined by

\[
\bar{a}_i = 2^n - h_{2^{m+n-1} - j}, \quad i = 0, 1, \ldots, 2^n - 1.
\]

(20)

In fact, \( \bar{a}_i \) is constructed by conversely applying (14), which has the same value as \( \bar{c}_i \).

Step 5. Denote by \( \bar{b}_{ri} \) the \( r \)th bit code of the binary representation of decimal number \( \bar{a}_i \), where \( i \in \{0, 1, \ldots, 2^n - 1\} \) and \( r \in \{1, 2, \ldots, p\} \). In fact, \( \bar{b}_{ri} \) is obtained by conversely applying (13), which has the same value as \( \bar{s}_i^{(r)} \).

Step 6. Construct the minterm canonical form of \( g_r \) for each \( r \in \{1, 2, \ldots, p\} \) as follows:

\[
g_r(x_1, x_2, \ldots, x_n) = \sum_{i \in \bar{S}_r} m_{ir}
\]

where \( \bar{S}_r = \{i \in \{0, 1, \ldots, 2^n - 1\} \mid \bar{b}_{ri} = 1\} \).

4.2. From Algebraic Expression to Logic Expression

The logical networks \( N_{in} \) and \( N_{out} \) can be reconstructed from the algebraic expression (7) through conversely applying Algorithm 1.

Let us illustrate the algorithm above with the following example.

Example 4. Reconstruct the logic network of BCN (18).

Step 1. Calculate \( a_i \) for each \( i \in \{0, 1, \ldots, 31\} \) by (19), which is presented in the last column of Table 1.

Step 2. Through the decimal-binary conversion, one can calculate \( b_{ji} \) for each \( i \in \{0, 1, \ldots, 31\} \) and \( j \in \{1, 2, 3\} \), which is presented in Table 1.

Step 3. Let

\[
S_1 = \{i \in \{0, 1, \ldots, 31\} \mid b_{1i} = 1\} = \{10, 11, 14, 15, 26, 27, 30, 31\},
\]

\[
S_2 = \{i \in \{0, 1, \ldots, 31\} \mid b_{2i} = 1\} = \{0, 2, 4, \ldots, 30\},
\]

\[
S_3 = \{i \in \{0, 1, \ldots, 31\} \mid b_{3i} = 1\} = \{8, 9, 10, \ldots, 31\}.
\]
Then, the logic expression of \( N_{in} \) is

\[
\begin{align*}
    x_1(t + 1) &= \sum_{i \in S_1} m_i = x_2(t)u_1(t), \\
    x_2(t + 1) &= \sum_{i \in S_2} m_i = u_2(t), \\
    x_3(t + 1) &= \sum_{i \in S_3} m_i = x_1(t) + x_2(t).
\end{align*}
\]

Step 4. Calculate \( \tilde{a}_i \) for each \( i \in \{0, 1, \ldots, 7\} \) by (20), which is presented in the last column of Table 2.

Step 5. Through the decimal-binary conversion, one can calculate \( \tilde{b}_1 \) for each \( i \in \{0, 1, \ldots, 7\} \), which is presented in Table 2.

Step 6. Let

\[
\tilde{S}_1 = \{i \in \{0, 1, \ldots, 7\} \mid \tilde{b}_1 = 1\} = \{0, 2, 4, 6\}.
\]

Then, the logic expression of \( N_{out} \) is

\[
y_1(t) = \sum_{i \in \tilde{S}_1} m_i = \tilde{x}_3(t).
\]

Hence, the logic expression of BCN (18) is (15), whose logical network is shown in Figure 1.

5. Conclusions

This paper provides an elementary method to achieve the mutual conversion between the logic and algebraic expressions of Boolean functions and Boolean control networks, which are converted by developing two algorithms. The presented method has several advantages comparing with the existing works. Firstly, the computational complexity is reduced. The complexity of the STP based method is quantitatively \( 2^{5(m+n)} \) times that of the method given in this paper. Secondly, the conversion formula is explicit and analytical. Thirdly, the algebraic expression is uniquely converted into the minterm canonical form of logic expression. Finally, the conversion algorithm is reversible, which means that both two sides of the problem for mutual conversion between logic and algebraic expressions can be solved by using a uniform approach. These advantages should be also convenient for investigating some control problems of Boolean control networks, such as the stability, controllability and observability, which are worth studying and exploring in the future work.

Author Contributions: Conceptualization, C.W. and Y.T.; methodology, C.W. and Y.T.; investigation, C.W. and Y.T.; writing—original draft preparation, C.W.; writing—review and editing, Y.T.; supervision, Y.T.; funding acquisition, C.W. and Y.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China under grants 61741307 and 61903037, and China Postdoctoral Science Foundation funded project 2020M670164.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- BN: Boolean network
- BCN: Boolean control network
- STP: semi-tensor product

References

1. Kauffman, S.A. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* **1969**, *22*, 437–467. [CrossRef]

2. Jamshidi, V. A VLSI majority-logic device based on spin transfer torque mechanism for brain-inspired computing architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 1858–1866. [CrossRef]
3. Wang, C.; Tao, Y. Karnaugh maps of logical systems and applications in digital circuit design. *Circuits Syst. Signal Process.* 2020, 39, 2245–2271. [CrossRef]

4. Bandypadhyay, C.; Das, R.; Chattopadhyay, A.; Rahaman, H. Design and synthesis of improved reversible circuits using AIG- and MIG-based graph data structures. *IET Comput. Digit. Tech.* 2019, 13, 38–48. [CrossRef]

5. Rushdi, A.M.; Ahmad, W. Digital circuit design utilizing equation solving over ‘big’ Boolean algebras. *Int. J. Math. Eng. Manag. Sci.* 2018, 3, 404–428. [CrossRef]

6. Roth, C.H.; Kinney, L.L. *Fundamentals of Logic Design*; CENGAGE Learning: Stamford, USA, 2010.

7. Schneeweiss, W.G. *Boolean Functions with Engineering Applications and Computer Programs*; Springer-Verlag: London, UK, 1989.

8. Steinbach, B.; Posthoff, C. Boolean differential calculus. *Synth. Lect. Digital Circuits Syst.* 2017, 12, 1–217. [CrossRef]

9. Zaitseva, E.N.; Levashenko, V.G. Importance analysis by logical differential calculus. *Autom. Remote Control* 2013, 74, 171–182. [CrossRef]

10. Schneeweiss, W.G. A short Boolean derivation of mean failure frequency for any (also non-coherent) system. *Reliab. Eng. Syst. Saf.* 2009, 94, 1363–1367. [CrossRef]

11. Cheng, D.; Qi, H. A linear representation of dynamics of Boolean networks. *IEEE Trans. Autom. Control* 2010, 55, 2251–2258. [CrossRef]

12. Rushdi, A.M.; Ghaaleb, F. A tutorial exposition of semi-tensor products of matrices with a stress on their representation of Boolean functions. *J. King Abdulaziz Univ.* 2016, 5, 3–41. [CrossRef]

13. Fornasini, E.; Valcher, M.E. Observability, reconstructibility and state observers of Boolean control networks. *IEEE Trans. Autom. Control* 2013, 58, 1390–1401. [CrossRef]

14. Laschov, D.; Margaliot, M.; Even, G. Observability of Boolean networks: A graph-theoretic approach. *Automatica* 2013, 49, 2351–2362. [CrossRef]

15. Laschov, D.; Margaliot, M. Controllability of Boolean control networks via the Perron-Frobenius theory. *Automatica* 2012, 48, 1218–1223. [CrossRef]

16. Cheng, D.; Qi, H. Controllability and observability of Boolean control networks. *Automatica* 2009, 45, 1659–1667. [CrossRef]

17. Guo, Y.; Shen, Y.; Gui, W. Asymptotical stability of logic dynamical systems with random impulsive disturbances. *IEEE Trans. Autom.* 2020. [CrossRef]

18. Yerudkar, A.; Del Vecchio, C.; Glielmo, L. Feedback stabilization control design for switched Boolean control networks. *Automatica* 2020, 116, 108934. [CrossRef]

19. Kobayashi, K.; Hiraishi, K. Design of probabilistic Boolean networks based on network structure and steady-state probabilities. *IEEE Trans. Neural Netw. Learn. Syst.* 2017, 28, 1966–1971. [CrossRef]

20. Kadelka, C.; Kuijpers, J.; Laubenbacher, R. The influence of canalization on the robustness of Boolean networks. *Phys. D: Nonlinear Phenom.* 2017, 353–354, 39–47. [CrossRef]

21. Bof, N.; Fornasini, E.; Valcher, M.E. Output feedback stabilization of Boolean control networks. *Automatica* 2015, 57, 21–28. [CrossRef]

22. Fornasini, E.; Valcher, M.E. On the periodic trajectories of Boolean control networks. *Automatica* 2013, 49, 1506–1509. [CrossRef]

23. Fornasini, E.; Valcher, M.E. Optimal control of Boolean control networks. *IEEE Trans. Autom. Control* 2014, 59, 1258–1270. [CrossRef]

24. Laschov, D.; Margaliot, M. Minimum-time control of Boolean networks. *SIAM J. Control Optim.* 2013, 51, 2869–2892. [CrossRef]

25. Melkman, A.A.; Cheng, X.; Ching, W.; Akutsu, T. Identifying a probabilistic Boolean threshold network from samples. *IEEE Trans. Neural Netw. Learn. Syst.* 2018, 29, 869–881. [CrossRef]

26. Sarda, K.; Yerudkar, A.; Del Vecchio, C. Disturbance decoupling control design for Boolean control networks: A Boolean algebra approach. *IET Control. Theory Appl.* 2020. [CrossRef]

27. Sarda, K.; Yerudkar, A.; Del Vecchio, C; Glielmo, L.; Singh, N. Subspace and coordinate transformation for Boolean control networks using binary logic. In Proceedings of the 27th Mediterranean Conference on Control and Automation (MED), Akko, Israel, 1–4 July 2019; pp. 328–333.
28. Khatri, C.G.; Rao, C.R. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhya Indian J. Stat. Ser. A* **1968**, *30*, 167–180.

29. Cheng, D.; Qi, H. Matrix expression of logic and fuzzy control. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 15–15 December 2005; pp. 3273–3278.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.