A Many-Valued Algebra for Switching Systems

ZVONKO G. VRANESIC, MEMBER, IEEE, E. STEWART LEE, AND KENNETH C. SMITH, MEMBER, IEEE

Abstract—Many-valued switching systems have been of considerable academic interest, despite the apparent inability to use them in practical applications. The main drawbacks, as pointed out by a number of authors, are the difficulties associated with the implementation of the functional basic set and the lack of adequate simplification techniques.

This paper describes a many-valued switching algebra based on a basic set which is potentially implementable in the economic sense. An algorithmic simplification technique is developed to facilitate synthesis of nontrivial many-valued switching functions.

Feasibility of physical realization of the basic set is demonstrated by an illustrative circuit.

Index Terms—Combinational circuits, compositional algebra, many-valued logic, simplification, switching algebra.

I. Introduction

THE primary objective of much recent work [1]–[5] in many-valued switching systems is to devise a set of practically implementable basic functions, and to develop an algebra such that functions of arbitrary complexity may be represented in terms of simple algebraic combinations of the basic functions. If, in addition, the choice of basic functions and the algebra permits the development of a technique to simplify in some useful sense the complexity of the functional representations developed, then these representations are of considerable additional interest. Finally, it is potentially advantageous to devise a system which is adaptable to any switching function, regardless of the choice of integral base of the switching variables involved.

The algebra presented in this paper is a generalization of a previously developed ternary algebra [6]. It meets the above requirements, since its basic set contains elementary functions which are potentially implementable in the economic sense, and has a well defined simplification technique.

II. Definitions and Notation

Let T be a switching algebra with the following characteristics.

1) It contains a set of variables $(x, y, z, \cdots)$ which can assume R logic values from the set $Q=\{0, 1, \cdots, R-1\}$, $0<1<\cdots<R-1$.
2) There exists an equivalence $(=)$ operation, that is,

$$x = x$$

$$\text{if } x = y, \text{ then } y = x,$$

$$\text{if } x = y \text{ and } y = z, \text{ then } x = z.$$

3) It has $2R+2$ basic operations (basic set).
   a) Two-Element Operations:

$$\text{sum} \quad x + y = \max(x, y)$$

$$\text{product } x \cdot y = \min(x, y)$$

   where $\max(x, y)$ and $\min(x, y)$ indicate the highest and the lowest values of (x, y), respectively.
   b) R Unary "Inverter" Operations:

$$x^K = K \text{ iff } x = 0$$

$$= 0 \text{ otherwise}$$

   for $K \in Q$.
   c) R Unary "Clockwise Cycling" Operations:

$$x^{\overset{M}{\rightarrow}} = (x + M) \bmod R$$

   where $M \in Q$.
4) The two-element operations obey the idempotent, commutative, associative, distributive, and absorption laws.

Operations of the above basic set are a natural choice because of their potentially simple implementation.

Discrete cycling operations (gates) are used, although an M-order cycling gate $x^{\overset{M}{\rightarrow}}$ can be implemented with M Postian cycling gates [7] in cascade ($M=1$ in a Postian cycling gate). Since the cycling gates differ only in the magnitude of the step by which the truth levels are shifted, it is reasonable to assume that the cost of all discrete gates is approximately of the same order.

The total cost of a functional implementation is the sum of the costs of all individual gates and all inputs. The cost of an individual gate depends upon the physical realization of the basic set.

It is apparent that in electronic implementation, the cycling gates are the most difficult to construct. However, satisfactory circuits have been designed as illustrated by the example of a universal clockwise-cycling gate described in the Appendix.

Post showed [7] that the cycling operation and the product gate are a functionally complete set, and so this expanded collection of gates is also functionally complete.

It is convenient to introduce counter-clockwise cycling operations defined as

$$x^{\overset{M}{\leftarrow}} = (x - M) \bmod R$$

where

$$M \in Q.$$

Manuscript received October 11, 1968; revised August 4, 1969, and January 20, 1970.

The authors are with the Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada.

TABLE I
THE TRUTH TABLE FOR EXAMPLE 1

x	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
y	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$f(x, y)$	2	2	2	3	1	2	3	0	1	0	0	0	2	1	1	0

They have no independent logical significance because

$$x^{\overset{M}{\leftarrow}} = x^{\overset{R-M}{\rightarrow}}.$$

Standard notational liberties are taken. As in Boolean algebra, the product operation need not be shown explicitly. Unary operations are denoted with superscripts as indicated in the definitions. If $M=1$ in the cycling operation, it may be omitted. Note that clearly

$$x^{\overset{R}{\rightarrow}} = x^{\overset{0}{\rightarrow}} = x.$$

When a unary operation is performed on a composite functional expression, the expression is parenthesized and the appropriate superscript is placed outside the parentheses. It is not required to use parentheses for separation of consecutive unary operations. In such cases the leftmost superscript indicates the operation of the highest priority. In this context the following expression can be written concisely as shown, without any ambiguity:

$$\{[(x^{\overset{3}{\rightarrow}} \cdot y^{\leftarrow})^{\rightarrow}]^4\}^{\overset{2}{\leftarrow}} = (x^{\overset{3}{\rightarrow}} y^{\leftarrow})^{\rightarrow 4 \overset{2}{\leftarrow}}.$$

Let Q, the basic space of the R values $0, 1, \cdots, R-1$, and $V=v_1, \cdots, v_n$, $v_i \in Q$, denote any point, "vertex," in the n-dimensional R-valued product space Q^n. An (incompletely specified) n-variable R-valued switching function $f(x_1, \cdots, x_n)$ is a mapping of (a subset of) Q^n into Q. Such a mapping is easily represented by an R-valued truth table. One such truth table for a completely specified two-variable four-valued function is given in Table I.

Any n-variable R-valued switching function $f(x_1, \cdots, x_n)$ has a "sum of products of sums" canonical form:

$$f(x_1, \cdots, x_n) = \sum_{K=1}^{R-1} \left\{ \prod_{V \mid f(V)=K} \left(\sum_{i=1}^{n} x_i^{\overset{v_i}{\leftarrow}} \right) \right\}^K$$

where

$\sum \equiv$ logical sum ($+$ operation)

$\prod \equiv$ logical product ($\cdot$ operation)

$V \mid f(V)=K$ is the set of vertices $V=v_1, \cdots, v_n$ for which $f(V)=K$.

Example 1: The canonical form for the function $f(x, y)$ given in Table I is

$$\begin{aligned} f(x, y) = {} & [(x^{\leftarrow} + y)(x^{\overset{2}{\leftarrow}} + y)(x^{\overset{3}{\leftarrow}} + y^{\leftarrow})(x^{\overset{3}{\leftarrow}} + y^{\overset{2}{\leftarrow}})]^1 \\ & + [(x + y)(x + y^{\leftarrow})(x + y^{\overset{2}{\leftarrow}})(x^{\leftarrow} + y^{\leftarrow})(x^{\overset{3}{\leftarrow}} + y)]^2 \\ & + [(x + y^{\overset{3}{\leftarrow}})(x^{\leftarrow} + y^{\overset{2}{\leftarrow}})]^3. \end{aligned}$$

III. BASIC THEOREMS AND RELATIONSHIPS

A number of important properties exhibited by the selected basic set provide a means for algebraic manipulation.

In the following discussion let

$$1 \le K \le (R-1)$$
$$1 \le M \le (R-1)$$

unless specified otherwise.

The constants $0, 1, \cdots, R-1$ may be generated as

$$(x \cdot x^K) = 0$$
$$(x \cdot x^K)^M = M.$$

Also,

$$(x \cdot x^{\rightarrow} \cdot x^{\overset{2}{\rightarrow}} \cdot \cdots \cdot x^{\overset{R-1}{\rightarrow}}) = 0.$$

From the definitions of the chosen basic set there exist two fundamental relationships.

Theorem 1: $(x_1 + x_2 + \cdots + x_n)^K = x_1^K \cdot x_2^K \cdot \cdots \cdot x_n^K$.

Theorem 2: $(x_1 \cdot x_2 \cdot \cdots \cdot x_n)^K = x_1^K + x_2^K + \cdots + x_n^K$.

These are dual theorems the usage and significance of which resemble that of De Morgan's laws in Boolean algebra. Their proofs, and those for the relationships below can be found in [8].

Purposeful manipulation of general switching functions should result in simplified functional expressions. Tools for such process are provided by the following relationships.

$$f \supseteq f^{\overset{M}{\rightarrow} L} + f^{K1} + f^{(R-1)\rightarrow}, \tag{1}$$

where $1 \le L \le (R-M)$.

$$f^J + f^K = f^K$$
$$f^J \cdot f^K = f^J$$
$$f_1^J \cdot f_2^J = f_1^K \cdot f_2^J = f_1^J \cdot f_2^K, \tag{2}$$

where $1 \le J \le K$.

$$(f + f^{\overset{M}{\rightarrow}})^K = 0. \tag{3}$$

$$f^1 + f^{\leftarrow 2} + f^{\overset{2}{\leftarrow} 3} + \cdots + f^{\overset{R-2}{\leftarrow}(R-1)} = f^{\rightarrow}. \tag{4}$$

The possibility of simplifying functional expressions as in (4) provides the main motivation for the simplification procedure presented in the latter sections. Whenever a "first stage" of an expression in a "sum of products of sums" form can be simplified using (4), then this sum (corresponding to an m-tuple in Section VI) is said to be "well formed."

$$\begin{aligned} & f^1 + f^{\leftarrow 2} + \cdots + f^{\overset{I-2}{\leftarrow}(I-1)} + (f^{\overset{I-1}{\leftarrow} I} + \cdots + f^{\overset{J-1}{\leftarrow} J}(\cdot y \\ & + f^{\overset{J}{\leftarrow}(J+1)} + \cdots + f^{\overset{R-2}{\leftarrow}(R-1)} = f^1 + f^{\leftarrow 2} + \cdots \\ & + f^{\overset{I-2}{\leftarrow}(I-1)} + f^{\rightarrow} \cdot y + f^{\overset{J}{\leftarrow}(J+1)} + \cdots + f^{\overset{R-2}{\leftarrow}(R-1)}, \end{aligned} \tag{5}$$

where

$$1 \le I \le (R-1)$$
$$1 \le J \le (R-1)$$
$$I \le J.$$

$$f + f^{\leftarrow} = f + f^{(R-1)}. \tag{6}$$

$$f^{\rightarrow} \cdot f^{\overset{2}{\rightarrow}} \cdot f^{\overset{3}{\rightarrow}} \cdot \cdots \cdot f^{\overset{R-1}{\rightarrow}} = f^{1}. \tag{7}$$

$$\left[\sum_{I=0}^{R-1} x^{\overset{I}{\rightarrow}(R-1)} \cdot f_I\right]^{\overset{M}{\rightarrow}} = \sum_{I=0}^{R-1} \left[x^{\overset{I}{\rightarrow}(R-1)} \cdot f_I^{\overset{M}{\rightarrow}}\right], \tag{8}$$

where $\sum \equiv$ logical sum (+ operation) and $f_0, f_1, \cdots, f_{R-1}$ are arbitrary switching functions.

Example 2: From the canonical form for the function $f(x, y)$ of Example 1, a simplified expression for f may be derived using the above relationships. Applying Theorem 2 followed by Theorem 1 we get

$$\begin{aligned} f(x, y) = {} & x^{\leftarrow 1}y^{1} + x^{\overset{2}{\leftarrow} 1}y^{1} + x^{\overset{3}{\leftarrow} 1}y^{\leftarrow 1} + x^{\overset{3}{\leftarrow} 1}y^{\overset{2}{\leftarrow} 1} \\ & + x^{2}y^{2} + x^{2}y^{\leftarrow 2} + x^{2}y^{\overset{2}{\leftarrow} 2} + x^{\leftarrow 2}y^{\leftarrow 2} \\ & + x^{\overset{3}{\leftarrow} 2}y^{2} + x^{3}y^{\overset{3}{\leftarrow} 3} + x^{\leftarrow 3}y^{\overset{2}{\leftarrow} 3}. \end{aligned}$$

Using (2) and rearranging the expression in an attempt to achieve simplification by (4) we obtain

$$\begin{aligned} f(x, y) = {} & (x^{\overset{2}{\leftarrow} 1} + x^{\overset{3}{\leftarrow} 2} + x^{3})y^{2} \\ & + (x^{\overset{3}{\leftarrow} 1} + x^{2} + x^{\leftarrow 3})(y^{1} + y^{\leftarrow 2} + y^{\overset{2}{\leftarrow} 3}) \\ & + x^{3}(y^{\leftarrow 1} + y^{\overset{2}{\leftarrow} 2} + y^{\overset{3}{\leftarrow} 3}). \end{aligned}$$

Finally, the simplified functional representation follows directly from (4):

$$f(x, y) = x^{\leftarrow}y^{2} + x^{\overset{2}{\leftarrow}}y^{\rightarrow} + x^{3}y.$$

IV. Mechanization of the Simplification Process

Algebraic manipulation of switching functions can be channelled to lead to simplified forms. However, it is often difficult to direct the manipulation in the proper direction, without excessive trial and error attempts. The difficulties increase rapidly as the number of variables and the radix increase. Thus, it is of upmost importance to have a well defined, automated simplification procedure, even if it does not always yield absolutely minimal forms.

The procedure presented in this paper extends some concepts of Roth's binary covering methods [9], [10] to a general multivalued case and introduces a new concept of transfer covers. It seeks simplification with respect to cost and leads to expressions of the "sum of products of sums" type. These expressions are a particular class of "sum of products of sums" expressions, where 1) first stage sums involve only unary operations on a single variable, and 2) constant factors may appear at all three stages.

In order to facilitate the subsequent discussion it is necessary to define several frequently used terms.

Implicant (W): Let $m_i \subseteq Q$, $i = 1, 2, \cdots, n$ be n nonempty subsets of Q, called m-tuples, and denote by $W = m_1 m_2 \cdots m_n$ the set of vertices in Q^n included in the Cartesian product of subsets m_i. For example $W = m_1 m_2 m_3 = 1(02)2$ represents the set of vertices $V_1 = 102$ and $V_2 = 122$. With reference to an n-variable R-valued switching function $f(x_1, \cdots, x_n)$ we will say that such a set $W = m_1 m_2 \cdots m_n$ is an implicant of f if for some vertex $V \in W$ the value $f(V)$ is specified and greater than zero. In this case, let K be the minimum value of $f(V)$ in the vertices $V \in W$ where f is specified; then we will say that W is an implicant of f of rank K.

Given two implicants $W_r = m_{r1} m_{r2} \cdots m_{rn}$ and $W_s = m_{s1} m_{s2} \cdots m_{sn}$, the inclusion relation is defined so that $W_r \subseteq W_s$ if every vertex $V \in W_r$ is also included in W_s. Thus, $W_r \subseteq W_s$ if $m_{ri} \subseteq m_{si}$ for all i.

Order of implicants: The lowest (zero) order implicant is an n-tuple of 1-tuples. Each additional truth value (in any m-tuple) increases the order of the implicant by 1. Examples of third-order implicants are

(01) 1 (02) (13)

2 (0123) 1 2.

The concept of the order of implicants is a generalization of the concept of dimension of binary cubes. But, unlike the binary case, there exists no direct relation between the order of an implicant and the number of vertices it includes.

Complex (*Com*(K)): A set of implicants $W_r, W_s, \cdots, W_t$ constitutes a "complex" Com(K) having rank K equal to the minimum rank of implicants in the complex.

Cover (*Cov*(K)): If every vertex V for which $f(V) = K$ is included in at least one implicant of Com(K), then such Com(K) is called a "cover" Cov(K).

Transfer implicant (Z): Let an implicant W of rank K include vertices $V_1, V_2, \cdots, V_h$, such that $K < f(V_1) < f(V_2) < \cdots < f(V_h)$. Then W includes implicants $W_1, W_2, \cdots, W_h$ of rank $K_1 = f(V_1), K_2 = f(V_2), \cdots, K_h = f(V_h)$, and $W \supset W_1 \supset W_2 \supset \cdots \supset W_h$. Such a set of implicants, with the information about their respective ranks retained, will be called a "transfer implicant" of f and may be represented by the following shorthand notation. Let

$$\begin{aligned} W_p &= m_{p1} m_{p2} \cdots m_{pn}, && W_p \text{ is of rank A}, \\ W_r &= m_{r1} m_{r2} \cdots m_{rn}, && W_r \text{ is of rank } B, \\ &\vdots && \vdots \\ W_t &= m_{t1} m_{t2} \cdots m_{tn}, && W_t \text{ is of rank } C, \end{aligned}$$

where $A > B > \cdots > C$ and $W_p \subset W_r \subset \cdots \subset W_t$. Then $Z = m_{z1} m_{z2} \cdots m_{zn}$ where m_{zi} is the set of truth values of m_{ti}, subscripted so that for each $E \in m_{ti}$ there corresponds an $E_K \in m_{zi}$, where K is the rank of the highest ranked implicant W_u, $u = p, r, \cdots, t$, for which $E \in m_{ui}$ and $m_{ui} \in W_u$.

As an example consider the following 3-variable implicants:

$$\begin{aligned} W_p &= 33(05), && W_p \text{ is of rank 4}, \\ W_r &= (123)\ 3\ (05), && W_r \text{ is of rank 3}, \\ W_t &= (0123)\ 3(025), && W_t \text{ is of rank 1}. \end{aligned}$$

They can be combined into a transfer implicant

$$Z = (0_1\ 1_3\ 2_3\ 3_4)\ 3_4\ (0_4\ 2_1\ 5_4).$$

Note that the transfer implicant contains all of the information contained in the implicants from which it was formed. Our interest in transfer implicants stems from the fact that they may lead to simplifications based on the

identities of Section III. It is particularly advantageous when the m-tuples m_{zi} of subscripted truth values composing transfer implicant $Z=m_{z1}m_{z2}\cdots m_{zn}$ can be implemented in the simplified form based on (4).

Transfer complex (T_{com}) is any set of transfer implicants.

Transfer cover (T_{cov}) is a transfer complex which specifies the given switching function.

Inclusion of transfer implicants: Let Z_r and Z_s be two transfer implicants. Then $Z_r \subseteq Z_s$ if every vertex V composing Z_r is also included in Z_s and if the logical product of the subscripts associated with V in Z_r is not greater than the logical product of the subscripts associated with V in Z_s.

Cover implicant: Given a particular Cov (K), an implicant in it which cannot be included in, or combined with any other implicant (to yield a higher order implicant) in the same cover is called a cover implicant. The cover implicant corresponds to the binary "prime implicant." However, unlike the binary prime implicants, the cover implicants are not necessarily the components for any minimal normal form.

The simplification procedure that follows starts with a given switching function defined in terms of R covers. It generates the cover implicants using the $*$-product operation similar to Roth's binary [9], [10] method. It then utilizes the concept of transfer implicants to generate a cover of such transfer implicants which are particularly suitable for implementation in terms of the chosen basic functions. Final selection of the required transfer implicants for the simplified representation may be done in several ways, e.g., by means of a McCluskey table [11].

V. Generation of Cover Implicants

Successful derivation of cover implicants requires systematic development of high-order implicants from the low-order ones. To accomplish this a $*$-product is defined.

Let

$$W_r = m_{r1}m_{r2}\cdots m_{rn}$$
$$W_s = m_{s1}m_{s2}\cdots m_{sn}.$$

Then

$$W_r * W_s = \begin{Bmatrix} W_t \\ W_u \\ \vdots \\ W_v \end{Bmatrix}; \qquad \begin{aligned} W_t &= m_{t1}m_{t2}\cdots m_{tn} \\ W_u &= m_{u1}m_{u2}\cdots m_{un} \\ &\vdots \\ W_v &= m_{v1}m_{v2}\cdots m_{vn} \end{aligned}$$

where for $p=t, u, \cdots, v$ have

1) $m_{pi} = \begin{cases} m_{ri} \cup m_{si} \text{ for exactly one } i \\ m_{ri} \cap m_{si} \text{ for all other } i, \end{cases}$
2) $W_p = \phi$ if $m_{pi}=\phi$ for any i.

Some examples of the $*$-product are

$$\{(02)\ 1\ 0\} * \{2\ (01)(23)\} = \{2\ 1\ (023)\}$$
$$\{(24)(025)(23)\} * \{(12)(01)\ 3\} = \begin{Bmatrix} (124)\ 0\ 3 \\ 2(0125)\ 3 \end{Bmatrix}$$
$$\{(01)(34)\ 0\} * \{3(01)\ 0\} = \phi.$$

A $*$-product of two n-variable implicants may yield as many as n new implicants, although some are likely to be directly included in one of the original implicants. The $*$-product is commutative:

$$W_r * W_s = W_s * W_r.$$

The following algorithm generates all cover implicants from any set of initial covers which specify the given function. There are R initial covers, denoted $\text{Cov}_s(K)$, $0 \le K \le R-1$, which may either directly correspond to the truth table, or contain higher order implicants. A cover may be empty, e.g., if the function never attains a truth value greater than or equal to K, then Cov $(K)=\phi$.

1) Let the function be defined by the initial covers $\text{Cov}_s(R-1)$, $\text{Cov}_s(R-2), \cdots, \text{Cov}_s(0)$. Note that $\text{Cov}_s(0)$ will have no effect on the final implementation of the function. Also, let all DON'T CARE vertices be specified in a DON'T CARE complex Com_d. Then, let $K=R-1$ and form
$$\text{Cov}_1(K) = \text{Com}_d \cup \text{Cov}_s(K).$$
2) Generate $\text{Cov}_l(K)$ which consists of cover implicants, formed by successive applications of the $*$-product on $\text{Cov}_1(K)$ as well as the subsequently formed higher order implicants.[1]
3) If $K>1$ then include $\text{Cov}_l(K)$ into the next lower ranked cover as DON'T CARE conditions:
$$\text{Cov}_1(K-1) = \text{Cov}_l(K) \cup \text{Cov}_s(K-1),$$
set $K=K-1$ and go to step 2; else all cover implicants have been found in $\text{Cov}_l(R-1), \cdots, \text{Cov}_l(1)$.

It is evident that this algorithm results in sets of cover implicants which are uniquely dependent on the given switching function.

Example 3: Find the cover implicants for the function defined in Table II. From the truth table, the initial covers are

$$\text{Cov}_s(3) = \begin{Bmatrix} 21 \\ 31 \end{Bmatrix}$$
$$\text{Cov}_s(2) = \begin{Bmatrix} 00 \\ 03 \\ 20 \\ 30 \\ 33 \end{Bmatrix}$$
$$\text{Cov}_s(1) = \begin{Bmatrix} 02 \\ 10 \\ 11 \\ 13 \\ 22 \\ 23 \\ 32 \end{Bmatrix}$$
$$\text{Com}_d = \phi.$$

[1] The detailed procedure is given in [8].

TABLE II
THE TRUTH TABLE FOR EXAMPLE 2

x	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
y	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$f(x, y)$	2	0	1	2	1	1	0	1	2	3	1	1	2	3	1	2

Then

$$\mathrm{Cov}_1(3) = \mathrm{Cov}_s(3)$$
$$\mathrm{Cov}_l(3) = \{(23)1\}$$
$$\mathrm{Cov}_1(2) = \begin{Bmatrix}(23)1\\ \mathrm{Cov}_s(2)\end{Bmatrix}$$
$$\mathrm{Cov}_l(2) = \begin{Bmatrix}(23) & (01)\\ (023) & 0\\ 3 & (013)\\ (03) & (03)\end{Bmatrix}$$
$$\mathrm{Cov}_1(1) = \begin{Bmatrix}\mathrm{Cov}_l(2)\\ \mathrm{Cov}_s(1)\end{Bmatrix}$$
$$\mathrm{Cov}_l(1) = \begin{Bmatrix}(123) & (013)\\ (0123) & (03)\\ (23) & (0123)\\ (023) & (023)\end{Bmatrix}.$$

VI. TRANSFER COVER ALGORITHM

Formation of transfer implicants normally leads to simplified functional forms. Cover implicants can often be merged into "well formed" transfer implicants which produce simplification of the type expressed in the identities of Section III.

Straightforward implementation of an m-tuple may be achieved by one literal appearance (usually necessitating one cycling and one inverting gate) of the corresponding variable for each truth value in the m-tuple. For example, at 5-valued m-tuple $(0_2\ 1_2\ 2_3\ 3_4)$ may be implemented as

$$(x^2 + x^{\overleftarrow{2}3} + x^{\overleftarrow{3}4}).$$

However, often it is possible to implement the mapping indicated by two or more truth values of an m-tuple with a single literal appearance of the corresponding variable, based on the simplifying identities of Section III. This can be done with the above 5-valued m-tuple, which has a simpler implementation as $(x^{\overleftarrow{4}} + x^2)$.

For some m-tuples, we can implement the mapping required by several truth values with a single literal appearance of the variable, only if a limiting constant is introduced to ensure proper mapping. For example, a 5-valued m-tuple $(0_2\ 1_2\ 2_3\ 3_3)$ can be implemented as either

$$(x^{\overleftarrow{4}} \cdot 3 + x^2) \quad \text{or} \quad (x^{\overleftarrow{4}} + x^2) \cdot 3,$$

where both alternatives are simpler than the straightforward implementation.

The m-tuples where such simpler implementations are possible will be called "well formed." A transfer implicant will be called "well formed" if at least one of its m-tuples is "well formed."

"Well formedness" of m-tuples is easily tested using the conditions given below. It is convenient to place a subscript G (as a result of the test) outside the parentheses of the m-tuple, where G indicates the "well formedness" of the m-tuple and also serves as an indication of the limiting constant. For "well formed" m-tuples $G = 1, 2, \cdots, R-1$, while $G = \phi$ otherwise.

Consider an m-tuple m_i with subscripted truth values.

Condition 1: If m_i contains fewer than $R-1$ truth values, it is not "well formed" and $G = \phi$.

Condition 2: If m_i contains exactly $R-1$ truth values, where P is the "missing" truth value, i.e., $P \notin m_i$, then

a) if all subscripts in m_i are the same, say H, then $G = H$; otherwise
b) if for any truth value $D \in m_i$ which has the lowest subscript, say H, we have $D^{\overleftarrow{p}} > H$ then $G = \phi$; otherwise
c) if for some truth value $E \in m_i$ with subscript M, we have $E^{\overleftarrow{p}} > M$ then G is equal to the smallest such M; otherwise $G = R-1$.

Condition 3: If m_i contains R truth values, then if all subscripts in m_i are the same, say H, then $G = H$; otherwise $G = \phi$. Note that if $G = \phi$ as a result of this condition, it may still be possible to have a simpler implementation for m_i in cases where m_i contains a "well formed" m-tuple of exactly $R-1$ truth values. Such cases are dealt with in the algorithm given below.

From the identities in Section III, it is evident that the most significant simplification can be attained for m-tuples containing $R-1$ truth values. The Condition 2c leads to simplification of the type indicated in (4), while 2b shows whether or not such implementation is advantageous. Other conditions indicate simplified implementations for m-tuples where all subscripts are the same, based on Theorem 2 and (7).

From Condition 2c, it follows that $G = 3$ for the above 5-valued m-tuple, which is then written as $(0_2\ 1_2 2_3\ 3_3)_3$.

When $G = \phi$ the subscript G is normally not shown. Note that m-tuples for which $G = \phi$ are, in general, difficult to implement. In contrast, m-tuples for which $G = R-1$ can often be implemented with a single literal appearance of the corresponding variable, e.g., a 4-valued m-tuple $(1_1\ 2_2\ 3_3)_3$ is implemented simply as x.

The following algorithm generates a transfer cover for the given function, with the goal of finding a simplified implementation.

1) Find all transfer implicants from the cover implicants (according to the definition in Section IV) in the covers $\mathrm{Cov}_l(R-1)$, $\mathrm{Cov}_l(R-2), \cdots, \mathrm{Cov}_l(1)$.
2) Apply the above conditions to each Z found in 1) to determine if it is "well formed."
3) Include all "well formed" Z, for which all m-tuples containing R truth values (if any) are "well formed," into a transfer complex T_{com}.

4) For each Z that has at least one m-tuple containing R truth values which is not "well formed," try to generate new "well formed" transfer implicants as follows. Let $m_i \in Z$ be such an m-tuple with R truth values. Exclude one truth value with the lowest subscript, say $D \in m_i$ with a subscript H, from m_i and consider it as being missing. Thus $P = D$ and $m_i' = m_i - D$. Then test m_i' according to the Conditions 2a, b, and c to determine whether or not it is "well formed." If $G \neq \phi$, then replace m_i in Z by m_i', otherwise m_i remains unchanged.

   This process is performed on all $m_i \in Z$ which have R truth values and are not "well formed," and the resultant transfer implicant is included in T_{com}.

   It may be possible to generate more than one m-tuple m_i' where $G \neq \phi$, when several truth values in m_i have the same lowest subscript. All such possibilities must be considered. Thus a string of cover implicants may be combined to form several transfer implicants, although this is rarely the case.

5) Append subscripts to all truth values of cover implicants to indicate their ranks, and designate the ensuing covers as $\acute{C}ov_l(R-1)$, $\acute{C}ov_l(R-2)$, $\cdots$, $\acute{C}ov_l(1)$.

6) Form a transfer cover

$$T_{cov1} = T_{com} \cup \acute{C}ov_l(R-1) \cup \acute{C}ov_l(R-2) \cup \cdots \cup \acute{C}ov_l(1).$$

7) Remove the redundant transfer implicants from T_{cov1}

$$T_{cov2} = T_{cov1} - \{Z_r | Z_r \subseteq Z_s;\ Z_r, Z_s \in T_{cov1}\}.$$

Example 4: Find the transfer cover T_{cov2} for the cover implicants of Example 3. The first transfer implicant is formed by combining

$W_r = (23)\ 1$, W_r is of rank 3,
$W_s = (23)\ (01)$, W_s is of rank 2,
$W_t = (123)\ (013)$, W_t is of rank 1.

Then $Z = (1_1\ 2_3\ 3_3)\ (0_2\ 1_3\ 3_1)$.

From Condition 2c it follows that $G = 3$ for both m-tuples; hence

$$Z = (1_1\ 2_3\ 3_3)_3\ (0_2\ 1_3\ 3_1)_3, \qquad Z \in T_{com}.$$

Next consider

$W_r = (023)\ 0$, W_r is of rank 2,
$W_s = (0123)\ (03)$, W_s is of rank 1,

Then

$$Z = m_1 m_2 = (0_2\ 1_1\ 2_2\ 3_2)\ (0_2\ 3_1).$$

Clearly, $G = \phi$ for both m-tuples. But m_1 has R truth values, thus (according to step 4 of the algorithm) exclude 1_1 from m_1; hence $m_1' = (0_2\ 2_2\ 3_2)$. From Condition 2a it follows that $G = 2$ for m_1'; therefore form $Z' = (0_2\ 2_2\ 3_2)_2\ (0_2\ 3_1)$ and $Z' \in T_{com}$. The third transfer implicant results from

$W_r = (023)\ 0$, W_r is of rank 2,
$W_s = (023)\ (023)$, W_s is of rank 1.

TABLE III
SELECTION OF TRANSFER IMPLICANTS FOR THE FUNCTION OF TABLE II

	x	y	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6
$f(x,y)=3$	2	1	ⓥ					
	3	1	ⓥ					
$f(x,y)=2$	0	0		√		√		
	0	3				ⓥ		
	2	0	√	√				
	3	0	√	√	√	√		
	3	3			√	√		
$f(x,y)=1$	0	2		√		√		
	1	0	√		√		√	
	1	1	√		√			
	1	3	√		√		√	
	2	2		√		√		√
	2	3	√	√	√	√	√	√
	3	2		√		√		√

Then $Z = (0_2\ 2_2\ 3_2)_2\ (0_2\ 2_1\ 3_1)$ and $Z \in T_{com}$. Observe that Z includes the above generated transfer implicant Z'.

Following the same procedure for other transfer implicants and performing steps 5–7, the following transfer cover is obtained:

$$T_{cov2} = \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \end{Bmatrix} = \begin{Bmatrix} (1_1\ 2_3\ 3_3)_3 & (0_2\ 1_3\ 3_1)_3 \\ (0_2\ 2_2\ 3_2)_2 & (0_2\ 2_1\ 3_1) \\ (1_1\ 2_1\ 3_2) & (0_2\ 1_2\ 3_2)_2 \\ (0_2\ 2_1\ 3_2)_2 & (0_2\ 2_1\ 3_2)_2 \\ (0_1\ 1_1\ 2_1\ 3_1) & (0_1\ 3_1) \\ (2_1\ 3_1) & (0_1\ 1_1\ 2_1\ 3_1) \end{Bmatrix}.$$

VII. TRANSFER COVER SELECTION

Some vertices for which $f(x_1, \cdots, x_n) = K$ may be covered by several transfer implicants in T_{cov2}; hence it is important to select a limited number of $Z_j \in T_{cov2}$, sufficient to specify the given function without unnecessary duplication. This can be done in a variety of ways. One method which is easily programmed is given in [8]. For simple functions, such as the one used for illustrative purposes in Examples 3 and 4, a customary McCluskey table [11] is easily constructed.

Table III shows such a table for the transfer cover T_{cov2} found in Example 4. It indicates that transfer implicants Z_1 and Z_4 are essential (because of encircled vertices), and they also constitute a complete cover. Therefore, the optimized transfer cover is

$$T_{covf} = \begin{Bmatrix} (1_1\ 2_3\ 3_3)_3\ (0_2\ 1_3\ 3_1)_3 \\ (0_2\ 2_1\ 3_2)_2\ (0_2\ 2_1\ 3_2)_2 \end{Bmatrix}.$$

One additional point should be emphasized. It can occur that a vertex for which $f(x_1, \cdots, x_n)=K$ is covered by more than one $Z_j \in T_{covf}$, which may result in duplication. For example, consider the transfer implicants

$$Z_r = (1_1\ 3_1)2_1, \qquad Z_r \in T'_{covf},$$
$$Z_s = 1_1\ (0_1\ 2_1), \qquad Z_s \in T'_{covf},$$

Clearly, both Z_r and Z_s specify that $f(1, 2)=1$. Since there is no need for this double coverage, either Z_r or Z_s (but not both) can be modified as follows:

$$Z'_r = 3_1\ 2_1, \qquad Z_s = 1_1\ (0_1\ 2_1),$$
$$Z_r = (1_1\ 3_1)2_1, \qquad Z'_s = 1_1\ 0_1.$$

Elimination of these redundancies may simplify the implementation of the function. In fact, it is best to treat them as DON'T CARE conditions. A special procedure called the β-product method which determines such redundancies and handles them as DON'T CARE cases is given in [8]. However, the resultant additional simplification of the functional expressions often may not be sufficient to justify the extra work involved.

VIII. Implementation of the Transfer Cover

Each m-tuple represents a function of a single variable, which can be realized as a sum of the unary functions of its truth values. A complete transfer implicant is implemented as the product of its m-tuples, with the upper limit that the function may attain (for the particular transfer implicant) restricted by the lowest value of G in any m-tuple. Thus, the transfer implicant is implemented as

$$f_j(x_1, \cdots, x_n) = G_j \cdot \left[\prod_{i=1}^{n} f_i(x_i)\right]$$

where G_j is the lowest value of G in any m-tuple for which $G \neq \phi$. However, G_j need not be included if

1) $G_j = R-1$,
2) $G=\phi$ for all m-tuples,
3) any m-tuple consisting of fewer than $R-1$ truth values ensures that the output levels of its inverter gates do not exceed G_j.

The effects of the identities of Section III are incorporated into the following procedure for implementation.

1) For m-tuples where $G=\phi$, it is necessary to implement the mapping specified by each truth value (and the associated subscript) individually.

If $m_i=(A_I\, B_J \cdots C_L)$, then

$$f_i(x_i) = x_i^{\overleftarrow{A}_I} + x_i^{\overleftarrow{B}_J} + \cdots + x_i^{\overleftarrow{C}_L}.$$

2) For an m-tuple containing $R-1$ truth values, where $G \neq \phi$, let P be the missing truth value.

If $m_1=(A_I B_J \cdots C_L)_G$, then

$$f_i(x_i) = x_i^{\overleftarrow{P}} + \left[\sum_{D} x_i^{\overleftarrow{D}_M}\right];$$

where $D \in m_i$ is a truth value with a subscript $M = I, J, \cdots, L$, such that

$$x_i^{\overleftarrow{D}_K} \notin x_i^{\overleftarrow{P}}$$

where $K=\min(M, G_j)$.

3) An m-tuple containing R truth values, where $G \neq \phi$, represents a constant. Thus if

$$m_i = [0_G 1_G \cdots (R-1)_G]_G, \quad \text{then } f_i(x_i) = G.$$

4) In the implementations arising from 1) and 2), it is possible to have several terms with the same second superscript (indicating the same inverter gate level). Such terms can be grouped using Theorem 2, e.g., if $m_i=(A_J B_J C_J)$, then

$$f_i(x_i) = x_i^{\overleftarrow{A}_J} + x_i^{\overleftarrow{B}_J} + x_i^{\overleftarrow{C}_J} = (x_i^{\overleftarrow{A}} \cdot x_i^{\overleftarrow{B}} \cdot x_i^{\overleftarrow{C}})^J.$$

A very convenient realization occurs for the m-tuples that have $R-1$ truth values with the same subscripts, e.g., if

$$m_i = (A_H B_H \cdots D_H) \quad \text{then } f_i(x_i) = x_i^{\overleftarrow{P}_{KH}}$$

where K is any integer $1 \leq K \leq R-1$.

5) The final functional expression is obtained by combining the transfer implicants in a logical sum. If the optimized transfer cover consists of $Z_r, Z_s, \cdots, Z_t \in T_{covf}$, then

$$f(x_1, \cdots, x_n) = \sum_{j=r}^{t} \left\{G_j \cdot \left[\prod_{i=1}^{n} f_i(x_i)\right]\right\}.$$

6) Further simplification of the resultant expression may be possible in isolated cases by application of the fundamental relationships of Section III.

In view of the above, the simplified transfer cover T_{covf} derived in Section VII may be implemented as

$$f(x, y) = (x + x^{\overleftarrow{2}_3})y^{\overleftarrow{2}} + 2 \cdot x^{\overleftarrow{}} y^{\overleftarrow{}}.$$

IX. Conclusions

The developed algebra allows synthesis of general many-valued switching functions. It has a well defined mechanized simplification procedure, which yields functional expressions of the "sum of products of sums" type.

The described procedure is readily programmed. Such a program was written and used for synthesis of nontrivial many-valued functions, with satisfactory results.

Appendix

Circuit Implementation of a Universal Cycling Gate

From a circuit point of view, cycling of a multi-valued signal may be thought of as equivalent to retransmission of the signal with an added offset corresponding to an integer digit interval. The value of the offset applied is made to depend simply on the range of the input signal. In a base R system an offset of value M is added when the signal level is less than $(R-M)$ and an offset of value $(R-M)$ is subtracted when the signal equals or exceeds the level $(R-M)$, as evident from Table IV.

TABLE IV
CLOCKWISE CYCLING GATE

x	0	1	- - -	$R-M$	- - - -	$R-2$	$R-1$
$x^{\overrightarrow{M}}$	M	$M+1$	- - -	0	- - - -	$M-2$	$M-1$

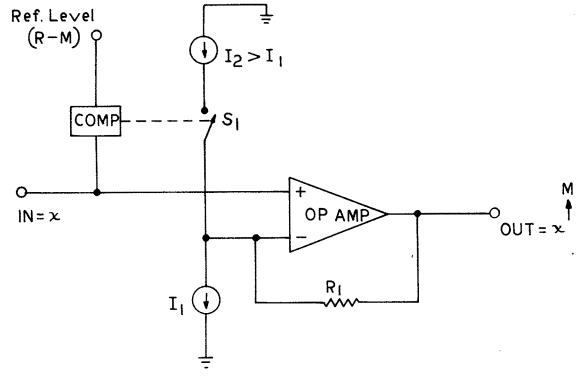

Fig. 1. Conceptual representation of a cycling gate.

A conceptual implementation of the cycling operation for any value of M in a system with base R is shown in Fig. 1. The fundamental idea is that of using a feedback follower, a familiar operational amplifier technique, to achieve the tracking of output and input levels. To provide the required value of offset between input and output, calibrated currents are arranged to flow in the feedback resistor R_1. The values of current available are either I_1 or (I_1-I_2) depending on the state of the switch S_1. When I_1 alone flows, the output follows the input but offset by I_1R_1 volts; when both I_1 and I_2 flow the net current in R_1 is (I_1-I_2) and the offset is $(I_1-I_2)R_1$. Since $I_2>I_1$ the two offsets are of opposite sign.

For the operation corresponding to Table III, the values of R_1, I_1, and I_2 are chosen so that the voltage offset I_1R_1 corresponds to a level change of M and $(I_1-I_2)R_1$ corresponds to $(R-M)$. Simple manipulation indicates that

$$I_1R_1 = M,$$
$$I_2R_1 = R,$$

and

$$I_1/I_2 = M/R.$$

These relations may be applied to the particular design as follows. Once the order of magnitude of currents to be used is established, I_2 can be chosen. Then, R_1 is defined by virtue of the choice of base R. Finally, current I_1 is adjusted to set the required value of M and a reference level corresponding to $(R-M)$ is selected.

Note that the cost of implementation is independent of the value of M used and that adjustment for M can be made as a special final step for an otherwise universal cycling gate.

A more detailed circuit realization is shown in Fig. 2. Here the operational amplifier is formed by transistors T_1, T_2, T_3 [12], T_4, and T_5; the comparator is composed of T_6 and T_7; and the switched constant current source uses T_8 alone. One circuit implemented from this pattern for base 10 uses 0.5-volt level separation for a complete signal range of 0 to 4.5 volts. In all, 11 transistors are used including realization of the current sources.

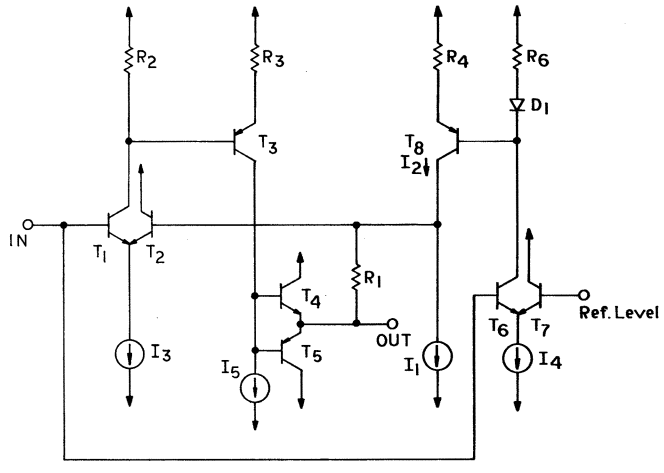

Fig. 2. Circuit realization of a cycling gate.

It is important to note that although the number of transistors required in this straightforward implementation is quite large, this number is totally independent of the base value R over a fairly wide range. In the described circuit, for example, a trivial increase in power supply voltage would allow the range to be extended to accommodate larger bases such as $R=32$.

Another interesting feature of the circuit is that the parameter M is controlled entirely by changes outside the direct path of the signal flow. In fact, changes in M may be made remotely by means of static voltage levels either as a particularizing operation for individual cycling modules, or as a system parameter on all modules at once.

REFERENCES

[1] C. Y. Lee and W. H. Chen, "Several-valued combinational switching circuits," *AIEE Trans. (Commun. Electron.)*, vol. 75, pt. 1, pp. 278–283, July 1956.
[2] O. Lowenschuss, "Non-binary switching theory," *1958 IRE Natl. Conv. Rec.*, pt. 4, pp. 305-317.
[3] R. D. Berlin, "Synthesis of *N*-valued switching circuits," *IRE Trans. Electronic Computers*, vol. EC-7, pp. 52–56, March 1958.
[4] E. Muehldorf, "Multivalued switching algebras and their application to digital systems," *1959 Proc. NEC*, vol. 15, pp. 467–480.
[5] C. M. Allen and D. D. Givone, "A minimization technique for multiple-valued logic systems," *IEEE Trans. Computers*, vol. C-17, pp. 182–184, February 1968.
[6] E. S. Lee and Z. G. Vranesic, "A ternary compositional algebra," to be published.
[7] E. L. Post, "Introduction to a general theory of elementary propositions," *Amer. J. Math.*, vol. 43, pp. 163–185, 1921.
[8] Z. G. Vranesic, "A multi-valued switching theory," Ph.D. dissertation, University of Toronto, Ontario, Canada, April 1968.
[9] J. P. Roth, "Algebraic topological methods for the synthesis of switching systems, I," *Trans. Amer. Math. Soc.*, vol. 88, pp. 301–326, July 1958.
[10] R. E. Miller, *Switching Theory*, vol. 1. New York: Wiley, 1965, pp. 145–184.
[11] E. J. McCluskey, Jr., "Minimization of Boolean functions," *Bell Sys. Tech. J.*, vol. 35, pp. 1417–1444, November 1956.
[12] K. C. Smith and A. Sedra, "The current conveyor—A new circuit building block," *Proc. IEEE* (Letters), vol. 56, pp. 1368–1369, August 1968.