Reprinted from THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 49, No. 8, October, 1970
Printed in U.S.A.

A Mathematical Study of a Model of Magnetic Domain Interactions

By R. L. GRAHAM

(Manuscript received March 18, 1970)

In this paper, we initiate a study into the combinatorial aspects of a model of the interactions between discrete magnetic domains and their potential use in information processing devices. Starting with a simple model suggested by W. Shockley, we demonstrate certain (surprising) capabilities as well as inherent limitations upon the possible applications of the interactions described by this model. It should be noted that this simple model does not take into account all of the possible interactions between magnetic domains.

I. INTRODUCTION

The subject of discrete magnetic domains in certain orthoferrite materials has been under active investigation during the past several years, both from a theoretical physical viewpoint as well as that of the device-oriented physicist (for example, see Refs. 1–6). Considerable progress has resulted from these efforts, although needless to say, the end is certainly not in sight. Particular attention has been directed toward the problem of applying this new technology to the very important area of information processing devices, an area in which it seems to have natural and significant applications.[1,7] It is our intention in this paper to examine certain mathematical aspects of these applications for a simple model of magnetic domain interactions suggested by W. Shockley.

II. DESCRIPTION OF THE MODEL

We shall begin by giving a very brief description of the physical situation and its translation into the mathematical model under consideration. The reader whose interests motivate him to seek a more technical explanation is referred to Refs. 6 or 8.

Roughly speaking, thin platelets of certain orthoferrite materials

possess the property that under suitable (magnetic) conditions, small ($\sim$ 3 mils) discrete cylindrical magnetic domains, hereafter called "bubbles", may be stably supported. Moreover, these bubbles may be manipulated by the application of external magnetic fields as well as by their own mutual interaction (which in general causes two bubbles to repel one another). In a suitable physical environment, the location of a bubble in a piece of orthoferrite can be restricted to a finite set of possible positions within the material; these are ordinarily arranged in a rectangular array. It is possible to apply a local magnetic field to specific locations within the array with the following results:†

(*i*) If a bubble already occupies the position at which the field was applied, then nothing happens.

(*ii*) If no bubble occupies the position at which the field was applied *and* no bubble occupies any "nearby" position as well, then (still) nothing happens.

(*iii*) If no bubble occupies the position at which the field was applied but at least one bubble occupies some "nearby" position, then some bubble at a nearby position will leave its original position and now occupy the position selected by the field.

To eliminate the annoying indeterminancy in item (*iii*) it is possible to apply "holding" fields to all but one of the "nearby" sites which has the effect that only a bubble at the unheld position can move.

The mathematical model which will correspond to the preceding description will be phrased in the terminology of *graph theory*. The discrete positions at which bubbles may lie correspond to the set V of *vertices* of a graph G. Two sites which are "nearby" or "adjacent" to one another (this is assumed to be a symmetric relation) correspond to two vertices of G which are joined by an *edge* of G. Suppose bubbles are located at (the sites corresponding to) the subset X of vertices V. We define a *command* to be a *directed edge* $e = (v_1, v_2)$ with v_1 and v_2 adjacent vertices of G. The command e transforms the locations of the bubbles from X to X^e where

$$X^e = \begin{cases} X - \{v_1\} \cup \{v_2\} & \text{if} \quad v_1 \in X, \quad v_2 \notin X; \\ X, & \text{otherwise.} \end{cases}$$

In other words, if there is a bubble at v_1 but no bubble at v_2 and the

† Of course, "careless" application of a magnetic field to an orthoferrite with bubbles can annihilate bubbles, create bubbles, split bubbles in two, deform bubbles into strips, and so on; but these pathological (though certainly useful) operations will not be considered in our model.

command $e = (v_1, v_2)$ is applied to X, then the bubble at v_1 is moved to v_2. Otherwise, the command e has no effect on X. A *program* is defined to be sequence $P = (e_1, e_2, \cdots, e_r)$ of commands e_i. In general, a program P maps the set 2^V of all subsets of V into itself by $X^P = (\cdots(X^{e_1})^{e_2})\cdots)^{e_r}$. It is the purpose of this paper to investigate the mathematical properties of these maps.

III. SOME BASIC PROPERTIES OF PROGRAMS

We begin by making the assumption that G is the *complete* graph on n vertices, that is, *all* pairs of vertices of G are joined by an edge.† As mentioned in the previous section, a program P is a sequence of directed edges $(e_1, e_2, \cdots, e_r)$ and P acts on a subset X of the vertices V of G by

$$X^P = (\cdots((X^{e_1})^{e_2})\cdots)^{e_r}$$

where for $e = (v, v')$,

$$X^e = \begin{cases} X - \{v\} \cup \{v'\} & \text{if} \quad v \in X, \quad v' \notin X; \\ X, & \text{otherwise.} \end{cases}$$

If $X \subseteq V$ then $|X|$ denotes the cardinality of X. We note

Fact 1: For all $X \subseteq V$, and all programs P, $|X^P| = |X|$.

This follows immediately from the definition of X^P.

The first interesting result we state is due to W. Shockley who called it the

Non-decreasing Overlap Theorem: (Shockley) For all X, $Y \subseteq V$ and all programs P,

$$|X^P \cap Y^P| \geqq |X \cap Y|.$$

Proof: Assume for some $P = (e_1, \cdots, e_r)$ and subsets $X, Y \subseteq V$ we have $|X^P \cap Y^P| < |X \cap Y|$. Since $X^P = (\cdots((X^{e_1})^{e_2})\cdots)^{e_r}$, there must exist a *least* j such that

$$|X^{P_{j+1}} \cap Y^{P_{j+1}}| < |X^{P_j} \cap Y^{P_j}|$$

where P_k denotes the program $(e_1, \cdots, e_k)$. Thus, for $\hat{X} = X^{P_j}$, $\hat{Y} = Y^{P_j}$ and $e = e_{j+1} = (a, b)$ we have

$$|\hat{X}^e \cap \hat{Y}^e| < |\hat{X} \cap \hat{Y}|.$$

† Nothing essential is lost by this simplifying assumption. The vertices and edges of the present model should not be confused with any incidental physical vertices or edges in a particular device. An edge of the model may be generated for example by transferring bubbles from a storage zone to an interaction zone and then returning the resultant to the storage zone.

If $c \neq a$, $c \neq b$ then $c \in \hat{X} \cap \hat{Y}$ implies $c \in \hat{X}^e \cap \hat{Y}^e$. If either $a \in \hat{X} \cap \hat{Y}$ or $b \in \hat{X} \cap \hat{Y}$ but not both then $b \in \hat{X}^e \cap \hat{Y}^e$. If both $a \in \hat{X} \cap \hat{Y}$ and $b \in \hat{X} \cap \hat{Y}$ then $a \in \hat{X}^e \cap \hat{Y}^e$ and $b \in \hat{X}^e \cap \hat{Y}^e$. Hence, in any case

$$| \hat{X}^e \cap \hat{Y}^e | \geqq | \hat{X} \cap \hat{Y} |$$

which is a *contradiction*. This proves the theorem.

Shockley noted that this result shows that there is no *replicating program* P^*. By a replicating program, we mean the following: Starting with two fixed sets of vertices V' and V'' with $V' \cap V'' = \varnothing$ and 1-to-1 map θ: $V'' \to V'$, we require that for each $X \subseteq V$,

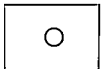
$$X^{P^*} \cap V' = X \cap V' \quad \text{and} \quad \theta(X^{P^*} \cap V'') = X \cap V'.$$

In other words, P^* does not disturb $X \cap V'$ and in V'', P^* creates a "copy" of $X \cap V'$.

To show this, suppose there were such a program P^*. By choosing two subsets X and X' differing in a single element of V, their images X^{P^*} and X'^{P^*} must differ in *two* points, namely, one in V' and the corresponding point (under θ) in V''. This, however, contradicts the non-decreasing overlap (NDO) theorem and therefore P^* cannot exist.

Another consequence of the NDO theorem is the nonexistence of a program P^+ which performs binary addition in the following way.

Suppose V' denotes a set of $m \geqq 1$ pairs of vertices of G, V'' denotes another set of m pairs of vertices disjoint from V', and V''' denotes a set of $m + 1$ pairs of vertices, disjoint from V' and V''. We can imagine these sets arranged as shown in Fig. 1.

We can represent an integer M, $0 \leqq M < 2^m$, in the m pairs of V' by letting the jth pair of V' denote the jth binary digit in the binary expansion of M. This can be done, for example, by assuming that for each pair  U_0, U_1, either $U_0 \in X$, $U_1 \notin X$, which will correspond to a 0, or $U_0 \notin X$, $U_1 \in X$, which will correspond to a 1. Thus, for $m = 5$ the configuration $\{V_1, U_2, U_3, V_4, V_5\}$ (Fig. 2) would denote the integer $10011_{(2)} = 19$.

The addition program P^+ would operate by starting with V''' in some fixed configuration (for example, all zeros) and with arbitrary integers M', M'' loaded into V', V'', respectively, to form the initial state X; after applying P^+ to X we should get the sum $M' + M''$ in V'''.

The reason that P^+ cannot exist as described is precisely that the NDO theorem would be violated. For consider the two additions:

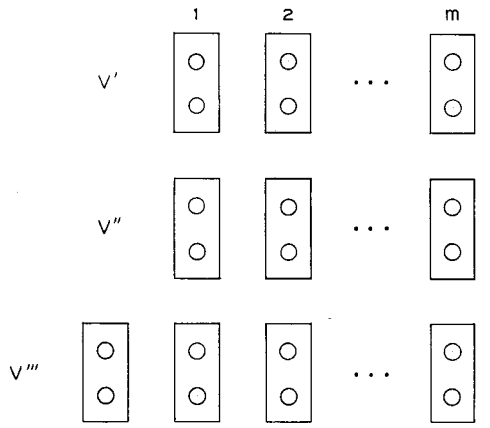


Fig. 1—Symbolic arrangement of vertex locations for addition.

$0 + (2^m - 1) = 2^m - 1$ and $1 + (2^m - 1) = 2^m$. The initial configurations differ in only *two* positions. The final configurations differ in at least $m + 1$ however, since $2^m - 1 = \overbrace{11 \cdots 1}^{m}{}_{(2)}$ and $2^m = \overbrace{100 \cdots 0}^{m}{}_{(2)}$. Thus, by the NDO theorem we get a contradiction and our assertion is proved.

We give another example of a program which does not exist. If $e = (a, b)$ is a command and $a, b \in X$ then $X^e = X$. In the case that a and b are both in X, we say that there is *interference* as e acts on X. (We can think of the bubble at b as interfering with the attempted movement of the bubble at a to vertex b.) Similarly, if $P = (e_1, \cdots, e_n)$ we say that there is interference as P acts on X if for some i there is interference as e_i acts on $X^{e_1 \cdots e_i}$. We note

Fact 2: If P acts on X with no interference then

$$X^P = \bigcup_{x \in X} \{x\}^P.$$

Proof: It is sufficient to establish this for the case $P = e = (a, b)$. In this case

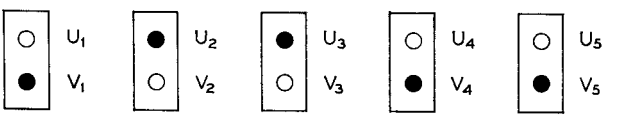


Fig. 2—A typical configuration representing an integer.

$$\{x\}^P = \begin{cases} b, & \text{if } x = a; \\ x, & \text{otherwise.} \end{cases}$$

Thus

$$\bigcup_{x \in X} \{x\}^P = \begin{cases} X - \{a\} \cup \{b\}, & \text{if } a \in X; \\ X, & \text{otherwise.} \end{cases}$$

But by the hypothesis of no interference, we cannot have both a and $b \in X$. Thus

$$X^P = \begin{cases} X - \{a\} \cup \{b\}, & \text{if } a \in X \\ X, & \text{otherwise} \end{cases} = \bigcup_{x \in X} \{x\}^P.$$

and the fact is established.

Fact 3: For $X = \{a, b, c, z\}$, there does not exist a program P such that

$$\{a, b\}^P = \{c, z\},$$
$$\{b, c\}^P = \{a, z\},$$
$$\{c, a\}^P = \{b, z\}.$$

Proof: Suppose such a P exists. If P acts on these sets with no interference then we would have by Fact 2,

$$\{c, z\} = \{\{a\}^P, \{b\}^P\},$$
$$\{a, z\} = \{\{b\}^P, \{c\}^P\},$$
$$\{b, z\} = \{\{c\}^P, \{a\}^P\},$$

which is impossible since the union of the left-hand sides of the equations cannot equal the union of the right-hand sides. Thus, if $P = (e_1, \cdots, e_n)$ we may assume that there is a *least* i, $1 \leqq i \leqq n$, with $P_{i-1} = (e_1, \cdots, e_{i-1})$ such that e_i acts on at least one of the sets $\{a, b\}^{P_{i-1}}$, $\{b, c\}^{P_{i-1}}$, $\{c, a\}^{P_{i-1}}$ with interference. To be specific, assume that it is the set $\{a, b\}^{P_{i-1}}$, that is, $e_i = (\{a\}^{P_{i-1}}, \{b\}^{P_{i-1}})$ (the other two cases are similar). By Fact 2 we have

$$\{a, b\}^{P_{i-1}} = \{\{a\}^{P_{i-1}}, \{b\}^{P_{i-1}}\},$$
$$\{b, c\}^{P_{i-1}} = \{\{b\}^{P_{i-1}}, \{c\}^{P_{i-1}}\},$$
$$\{c, a\}^{P_{i-1}} = \{\{c\}^{P_{i-1}}, \{a\}^{P_{i-1}}\}.$$

Therefore

$$\{b, c\}^{P_i} = \{\{b\}^{P_{i-1}}, \{c\}^{P_{i-1}}\}^{e_i} = \{\{b\}^{P_{i-1}}, \{c\}^{P_{i-1}}\}$$

and

$$\{c, a\}^{P_i} = \{\{c\}^{P_{i-1}}, \{a\}^{P_{i-1}}\}^{\bullet_i} = \{\{c\}^{P_{i-1}}, \{b\}^{P_{i-1}}\}.$$

Hence,

$$\{a, z\} = \{b, c\}^{P} = \{c, a\}^{P} = \{b, z\}$$

which is a *contradiction.* This proves the Fact 3.

Note that the nonexistence of the program of Fact 3 does not follow directly from Fact 1 or the NDO theorem. A similar argument can be given to show that for $X = (a, b, c, d, A, B, C, D, z)$ there is no program P such that

$$\{a, c\}^{P} = \{A, z\},$$

$$\{a, d\}^{P} = \{B, z\},$$

$$\{b, c\}^{P} = \{C, z\},$$

$$\{b, d\}^{P} = \{D, z\}.$$

IV. THE 2-VALUED BOOLEAN FUNCTIONS

Our attention will now be focussed on the positive aspects of the model. In particular we shall be concerned with the problem of representing the Boolean functions of m variables with appropriate programs. The way in which a function is to be represented is as follows. Suppose $m = 2$ and consider the function f: $\{0, 1\} \times \{0, 1\} \to \{0, 1\}$ by

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

If the values 1 and 0 are interpreted as "true" and "false", respectively, then f is just the truth function of the familiar operation of alternation. V will be the set of six vertices (x_0 , x_1 , y_0 , y_1 , f_0 , f_1) which we indicate in Fig. 3. It is not difficult to show that no generality is lost by assuming there are no additional vertices. In fact, by using the pair of positions x_0 , x_1 in which to observe the result of the program, instead of providing the separate positions f_0 , f_1 , it is true that if a Boolean function of $m \geqq 2$ variables can be represented by a program in this general way, then it can be represented using just $2m$ vertices. The program $P(f)$

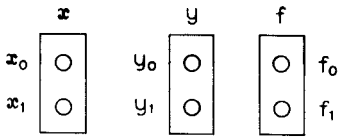


Fig. 3—Symbolic arrangement of vertex locations for computing Boolean functions of two variables.

which represents f is required to have the property that

$$f_0 \in \{x_0 , y_0\}^{P(f)}, \qquad f_1 \notin \{x_0 , y_0\}^{P(f)},$$
$$f_0 \in \{x_0 , y_1\}^{P(f)}, \qquad f_1 \notin \{x_0 , y_1\}^{P(f)},$$
$$f_0 \in \{x_1 , y_0\}^{P(f)}, \qquad f_1 \notin \{x_1 , y_0\}^{P(f)},$$
$$f_0 \notin \{x_1 , y_1\}^{P(f)}, \qquad f_1 \in \{x_1 , y_1\}^{P(f)}.$$

The correspondence between the indices of the vertices of V and the values of the variables of f is immediate. In terms of bubbles, one may think of the configurations shown in Fig. 4 as representing a 0 and 1 respectively (compare Fig. 2); $P(f)$ is required to map each of the four possible initial states of the x_i-pair and y_i-pair into the correct value in the f_i-pair.

It is not difficult in this case to find an appropriate $P(f)$, for example, we can take

$$P(f) = (x_0 , y_0)(x_0 , f_0)(x_1 , y_1)(y_1 , f_1).$$

This is easily checked, as shown in Table I. We can write the preceding result in the shorthand form

f	$P(f)$
(0, 0, 0, 1)	$(x_0 , y_0)(x_0 , f_0)(x_1 , y_1)(y_1 , f_1)$.

Note that if $\bar{f}$ is defined by $\bar{f}(x, y) = 1 - f(x, y)$, that is, $\bar{f}$ is the complement of f, then we can take

$$P(\bar{f}) = P(f)(x_0 , x_1)(x_0 , y_1)(x_0 , y_2)(f_1 , x_0)(f_0 , f_1)(x_0 , f_0)$$

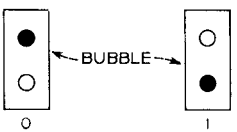


Fig. 4—Configurations which represent 0 and 1.

TABLE I—CUMULATIVE EFFECT OF $P(f)$

(x, y)	(x_0, y_0)	(x_0, f_0)	(x_1, y_1)	(y_1, f_1)	$f(x, y)$
$(0, 0) \leftrightarrow \{x_0, y_0\}$	$\{x_0, y_0\}$	$\{f_0, y_0\}$	$\{f_0, y_0\}$	$\{f_0, y_0\}$	$\leftrightarrow$ 0
$(0, 1) \leftrightarrow \{x_0, y_1\}$	$\{y_1, y_0\}$	$\{y_1, y_0\}$	$\{y_1, y_0\}$	$\{f_1, y_0\}$	$\leftrightarrow$ 1
$(1, 0) \leftrightarrow \{x_1, y_0\}$	$\{x_0, y_0\}$	$\{x_1, y_0\}$	$\{y_1, y_0\}$	$\{f_1, y_0\}$	$\leftrightarrow$ 1
$(1, 1) \leftrightarrow \{x_1, y_1\}$	$\{x_1, y_1\}$	$\{x_1, y_1\}$	$\{x_1, y_1\}$	$\{x_1, f_1\}$	$\leftrightarrow$ 1

as a program which represents $\bar{f}$ (we leave this to the reader to verify). Table II, together with this remark about $\bar{f}$, show that *all* of the 16 possible 2-valued Boolean functions of two variables can be represented by programs.

A question which naturally arises at this point is whether *all* Boolean functions of m variables can be represented by programs in this manner. For $m = 1$, the answer is in the affirmative (the specific programs are left to the reader to discover); for $m = 2$, we have given the required 16 programs; for $m = 3$, the answer is in the affirmative but the number ($2^{2^3} = 256$) of programs prohibits their listing here; for $m = 4$, the answer is once again in the affirmative but the calculations necessary to establish this are much too long to be exhibited (there are, after all, $2^{2^4} = 65536$ functions to consider). The cases $m = 3$ and $m = 4$ were established by J. H. Spencer.[9]

One may note that since all Boolean functions of two variables can be represented, then in particular the Sheffer stroke function given by

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

TABLE II—PROGRAMS FOR BOOLEAN FUNCTIONS OF 2 VARIABLES

f	$P(f)$
(0, 0, 0, 0)	(x_0, f_0) (x_1, f_0)
(0, 0, 0, 1)	(x_1, y_1) (x_1, f_1) (x_0, f_0) (y_0, f_0)
(0, 0, 1, 0)	(x_1, y_0) (x_1, f_1) (x_0, f_0) (y_1, f_0)
(0, 0, 1, 1)	(x_1, f_1) (x_0, f_0)
(0, 1, 0, 0)	(x_0, y_1) (x_0, f_1) (x_1, f_0) (y_0, f_0)
(0, 1, 0, 1)	(y_0, f_0) (y_1, f_1)
(0, 1, 1, 0)	(x_0, y_0) (x_0, f_0) (x_1, y_1) (y_1, y_0) (x_1, f_0) (y_1, f_1)
(0, 1, 1, 1)	(x_0, y_0) (x_0, f_0) (x_1, f_1) (y_1, f_1)

can also be represented. It is well known that any Boolean function of m variables can be generated by expressions containing just the variables and the stroke function.[10] Hence, one is tempted to conclude that any Boolean function is representable by a program. The flaw in this line of reasoning is that in order to express a particular Boolean function in terms of the stroke function, many occurrences of the stroke function and the variables are usually required. This in turn requires many "copies" of the variables to be available to the program in order to represent f. But we initially have only one pair of positions which indicates the value of any particular variable and by the NDO theorem we have seen that there cannot exist a "replication" program which would form extra copies of the values of the variables. Hence, within this model, we cannot use this technique to generate all the Boolean functions. It is certainly true however that if the model were extended to include bubble interactions which would allow replication of configurations (and such are known to exist physically), then all Boolean functions of m variables could be represented exactly in the manner described.

These initial results create considerable optimism concerning the possibility of representing all the Boolean functions of m variables. Such hopes are shattered however by the result (which we later prove) that *there exists a Boolean function of 11 variables which cannot be represented by any program of this type.* In fact, even though the fraction of the total number of Boolean functions of 11 variables which *can* be represented by programs can be shown to be $< 10^{-163}$, the author is currently unable to exhibit any specific function which cannot be represented. Clearly, our understanding of this is less than complete. It is not unreasonable to hope that the representable functions could eventually be effectively characterized.

We now restrict ourselves (without loss of generality) to representing the Boolean functions of m variables in the following way. We shall take $V = \{x_1 , x_1' , x_2 , x_2' , \cdots x_m , x_m'\}$ to be a set of $2m$ vertices which we imagine to be arranged in pairs as illustrated in Fig. 5. As before, a bubble in the $x_i(x_i')$ location of the pair (x_i , x_i') will denote that the ith variable of the function f has the value 0(1). The way in which a program $P(f)$ represents f is as follows. Choose a distinguished vertex $\alpha \in V$. There is an obvious 1-1 correspondence between $\{0, 1\}^m$ and the class $\bar{C}$ of all subsets $X \subseteq V$ such that X intersects each $\{x_i , x_i'\}$ in exactly one element given by

$$a = (a_1 , \cdots , a_m) \leftrightarrow \{y_i \in V : y_i = x_i \text{ if } a_i = 0,$$
$$y_i = x_i' \text{ if } a_i = 1,\ 1 \leqq 1 \leqq m\} = X.$$

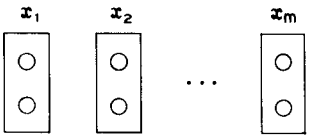


Fig. 5—Symbolic arrangement of vertex locations for computing Boolean functions of m variables.

Let $A_i \subseteq \{0, 1\}^m$, $i = 0, 1$, be the set of all $a \in \{0, 1\}^m$ such that $f(a) = i$, and let $\bar{C}_i$ be the corresponding subsets of $\bar{C}$. Our object is to find a program $P(f)$ which distinguishes between the sets $\bar{C}_0$ and $\bar{C}_1$. (Note that $\bar{C}_0 \cup \bar{C}_1 = \bar{C}$.) Specifically we shall say that $P(f)$ represents f if

$$\alpha \in X^{P(f)} \quad \text{for all} \quad X \in \bar{C}_0 ,$$

$$\alpha \notin X^{P(f)} \quad \text{for all} \quad X \in \bar{C}_1 .$$

Let C denote the subset of all subsets $x \subseteq V$ with $| x | = m$ and for x and y distinct elements of V, let $C(x)$ be the set of elements of C which contain x with $C(y)$ defined similarly.† Consider the effect of the command (x, y) on the members of $C(x)$ and $C(y)$. There are four cases:

(*i*) $X \in C(x)$, $X \in C(y)$.
Then $X^{(x,y)} = X$ and $X^{(x,y)} \in C(x)$, $X^{(x,y)} \in C(y)$.
(*ii*) $X \in C(x)$, $X \notin C(y)$.
Then $X^{(x,y)} = X - \{x\} \cup \{y\}$ and $X^{(x,y)} \notin C(x)$, $X^{(x,y)} \in C(y)$.
(*iii*) $X \notin C(x)$, $X \in C(y)$.
Then $X^{(x,y)} = X$ and $X^{(x,y)} \notin C(x)$, $X^{(x,y)} \in C(y)$.
(*iv*) $X \notin C(x)$, $X \notin C(y)$.
Then $X^{(x,y)} = X$ and $X^{(x,y)} \notin C(x)$, $X^{(x,y)} \notin C(y)$.

Hence, after the application of (x, y) to all the sets in C, the new sets $C'(x)$, $C'(y)$ (which now consist of all the subsets in C which contain x and y respectively) are related to $C(x)$ and $C(y)$ by

$$C'(x) = C(x) \cap C(y),$$

$$C'(y) = C(x) \cup C(y).$$

Stated in these terms, the object of the program $P(f)$ is finally to have $C'^{\cdots'}(\alpha) \cap \bar{C} = \bar{C}_0$ after it has been applied to all the sets in C.

We give an example which illustrates these concepts. Let f be the Boolean function of three variables defined by:

† This approach was first suggested by J. H. Spencer.[9]

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$V = \{x_1 , x_1' , x_2 , x_2' , x_3 , x_3'\} \quad \text{and we take} \quad \alpha = x_1' .$$

$$\bar{C}_0 = \{\{x_1 , x_2 , x_3\}, \{x_1 , x_2' , x_3\}, \{x_1 , x_2' , x_3'\}, \{x_1' , x_2 , x_3\}\},$$

$$\bar{C}_1 = \{\{x_1 , x_2 , x_3'\}, \{x_1' , x_2 , x_3'\}, \{x_1' , x_2' , x_3\}, \{x_1' , x_2' , x_3'\}\}.$$

A program $P(f)$ which achieves the separation is

$$P(f) = (x_1' , x_2)(x_1' , x_3)(x_2' , x_3)(x_1 , x_3)(x_1 , x_1').$$

That is,

$$X \in \bar{C}_0 \Rightarrow \alpha = x_1' \in X^{P(f)},$$

$$X \notin \bar{C}_0 \Rightarrow x_1' \notin X^{P(f)}.$$

If $C(x)$ denotes the initial subset of C consisting of all the sets in C which contain x then we may conveniently record the sequential changes which occur in each current $C(x)$ in terms of the original $C(y)$'s as the successive commands of $P(f)$ are applied as shown in Table III. A little computation shows that the final set in the x_1'-row, the final $C(x_1')$, when intersected with $\bar{C}$ gives exactly

$$\{\{x_1 , x_2 , x_3\}, \{x_1 , x_2' , x_3\}, \{x_1 , x_2' , x_3'\}, \{x_1' , x_2 , x_3\}\}$$

which equals $\bar{C}_0$ as required.

In general the problem of representing Boolean functions reduces to the following problem. We start with the $2m$ classes $C^{(0)}(y) = C(y) \cap \bar{C}$, $y \in V$. We are then allowed to replace two of the classes $C^{(0)}(y)$ and $C^{(0)}(y')$ by two (possibly) new classes $C^{(0)}(y) \cap C^{(0)}(y')$ and $C^{(0)}(y) \cup C^{(0)}(y')$. We can repeat this operation as many times as desired with any pair of classes currently in the list. Our objective is to eventually generate a specified subset C^* of $\bar{C}$.

We have already mentioned that for $m = 1, 2, 3$ and 4 it is possible

TABLE III—CUMULATIVE EFFECT OF $P(f)$

		(x_1', x_2)	(x_1', x_3)
x_1:	$C(x_1)$	$C(x_1)$	$C(x_1)$
x_1':	$C(x_1')$	$C(x_1') \cap C(x_2)$	$C(x_1') \cap C(x_2) \cap C(x_3)$
x_2:	$C(x_2)$	$C(x_1') \cup C(x_2)$	$C(x_1') \cup C(x_2)$
x_2':	$C(x_2')$	$C(x_2')$	$C(x_2')$
x_3:	$C(x_3)$	$C(x_3)$	$(C(x_1') \cap C(x_2)) \cup C(x_3)$
x_3':	$C(x_3')$	$C(x_3')$	$C(x_3')$

		(x_2', x_3)	(x_1, x_3)
x_1:	$C(x_1)$		$C(x_1) \cap (C(x_2') \cup (C(x_1') \cap C(x_2)) \cup C(x_3))$
x_1':	$C(x_1') \cap C(x_2) \cap C(x_3)$		$C(x_1') \cap C(x_2) \cap C(x_3)$
x_2:	$C(x_1') \cup C(x_2)$		$C(x_1') \cup C(x_2)$
x_2':	$C(x_2') \cap ((C(x_1') \cap C(x_2)) \cup C(x_3))$		$C(x_2') \cap ((C(x_1') \cap C(x_2)) \cup C(x_3))$
x_3:	$C(x_2') \cup (C(x_1') \cap C(x_2)) \cup C(x_3)$		$C(x_1) \cup C(x_2') \cup (C(x_1') \cap C(x_2)) \cup C(x_3)$
x_3':	$C(x_3')$		$C(x_3')$

		(x_1, x_1')
x_1:	$C(x_1) \cap (C(x_2') \cup (C(x_1') \cap C(x_2)) \cup C(x_3)) \cap C(x_1') \cap C(x_2) \cap C(x_3)$	
x_1':	$(C(x_1) \cap (C(x_2') \cup (C(x_1') \cap C(x_2))) \cup C(x_3)) \cup (C(x_1') \cap C(x_2) \cap C(x_3))$	
x_2:	$C(x_1') \cup C(x_2)$	
x_2':	$C(x_2') \cap ((C(x_1') \cap C(x_2)) \cup C(x_3))$	
x_3:	$C(x_1) \cup C(x_2') \cup (C(x_1') \cap C(x_2)) \cup C(x_3)$	
x_3':	$C(x_3')$	

to generate *any* subset of C in this manner. We proceed to show that for $m = 11$, there is a subset of C which cannot be generated. We first need several preliminary observations.

To begin with, for $a, b \in V$, let A and B denote the current sets $C^{(i)}(a)$ and $C^{(i)}(b)$, respectively, after the ith command of the program P has been executed. In other words, at this point in time $C^{(i)}(a)$ is the class of all the original subsets of C which now contain a. For example, if $a = x_2'$ in the preceding example, then after the fifth (and final) command of $P(f)$, $C^{(5)}(x_2')$ is $C^{(0)}(x_2') \bigcap (C^{(0)}(x_2') \bigcap C^{(0)}(x_2) \bigcup C^{(0)}(x_3))$. It is immediate that if $C^{(i)}(a) \subseteq C^{(i)}(b)$ then the application of the command (a, b) as the $(i + 1)$-st command of the program changes nothing. Hence *we can assume that we only use commands* (a, b) *for which at the time of their application* $C^{(i)}(a) \not\subseteq C^{(i)}(b) \not\subseteq C^{(i)}(a)$ (we say that $C^{(i)}(a)$ and $C^{(i)}(b)$ are incomparable).

Initially all the starting classes $C^{(0)}(x)$, $x \in V$, are mutually incomparable. In general suppose we have a family of classes $D = \{A_i ; 1 \leqq i \leqq t\}$, $A_i \subseteq \bar{C}$, with exactly r of the $\binom{t}{2}$ pairs of A_i being comparable and assume A_1 and A_2 are incomparable. Consider the family $D' =$

$D - \{A_1\} - \{A_2\} \cup \{A_1 \cap A_1\} \cup \{A_1 \cup A_2\}$. We wish to determine how many pairs of the classes of D' are comparable. By definition $D' = \{A_1 \cap A_2 , A_1 \cup A_2 , A_3 , A_4 , \cdots, A_t\}$. Of course for $i, j \geqq 3$, the comparability between the classes A_i and A_j remains unchanged. There are several cases:

(*i*) $A_i \supseteq A_1 , A_i \supseteq A_2$.
Then $A_i \supseteq A_1 \cup A_2 , A_i \supseteq A_1 \cap A_2$.
(*ii*) $A_i \supseteq A_1 , A_i \not\supseteq A_2$.
Then $A_i \supseteq A_1 \cap A_2$.
(*iii*) $A_i \not\supseteq A_1 , A_i \supseteq A_2$.
Then $A_i \supseteq A_1 \cap A_2$.
(*iv*) $A_i \subseteq A_1 , A_i \subseteq A_2$.
Then $A_i \subseteq A_1 \cap A_2 , A_i \subseteq A_1 \cup A_2$.
(*v*) $A_i \subseteq A_1 , A_i \not\subseteq A_2$.
Then $A_i \subseteq A_1 \cup A_2$.
(*vi*) $A_i \not\subseteq A_1 , A_i \subseteq A_2$.
Then $A_i \subseteq A_1 \cup A_2$.

Finally, we have a most important *new* comparability in D', namely $A_1 \cap A_2 \subseteq A_1 \cup A_2$. Thus, at least $r + 1$ pairs of classes of D' are comparable. An immediate consequence of this observation is

Fact 4: We can assume that no program $P(f)$ consists of more than $\binom{2m}{2}$ commands.

Proof: Since after i (nontrivial) commands of a program $P(f)$ have been applied, we must have (by induction) at least i pairs of the classes $C^{(i)}(x)$, $x \in V$, being comparable and since there are just $2m$ classes and therefore $\binom{2m}{2}$ pairs of classes, then $P(f)$ must have $\leqq \binom{2m}{2}$ commands.

Theorem. There exists a Boolean function of 11 variables which cannot be represented by a program.

Proof: It is sufficient to show that for $m = 11$, there is a subset C^* of $\tilde{C}$ which cannot be generated by starting with the $2m$ classes $C^{(0)}(x)$, $x \in V$, and recursively applying the transformation $A, B \to A \cap B$, $A \cup B$. Consider a typical program $P = (e_1 , e_2 , \cdots , e_t)$ and the corresponding expressions $C^{(i)}(t)$, presented in Table IV.

In choosing the ith command e_i of P there are at most $\binom{2m}{2} - i + 1$ possibilities for e_i since after $(e_1 , \cdots , e_{i-1})$ has been applied, at least $i - 1$ of the pairs $C^{(i-1)}(x)$, $C^{(i-1)}(y)$ are comparable and thus neither (x, y) nor (y, x) can be the next command e_i. Therefore there are at most

$$\prod_{i=1}^{\binom{2m}{2}} \left[\binom{2m}{2} - i + 1\right] = [m(2m - 1)]!$$

TABLE IV—CUMULATIVE EFFECT OF P

P:		e_1	e_2		e_i		e_t
x_1:	$C^{(0)}(x_1)$	$C^{(1)}(x_1)$	$C^{(2)}(x_1)$	$\cdots$	$C^{(i)}(x_1)$	$\cdots \cdots$	$C^{(t)}(x_1)$
x_1':	$C^{(0)}(x_1')$	$C^{(1)}(x_1')$	$C^{(2)}(x_1')$	$\cdots$	$C^{(i)}(x_1')$	$\cdots \cdots$	$C^{(t)}(x_1')$
.	.	.	.	.	.		.
.	.	.	.	.	.		.
.	.	.	.	.	.		.
x_m:	$C^{(0)}(x_m)$	$C^{(1)}(x_m)$	$C^{(2)}(x_m)$	$\cdots$	$C^{(i)}(x_m)$	$\cdots \cdots$	$C^{(t)}(x_m)$
x_m':	$C^{(0)}(x_{m1}')$	$C^{(1)}(x_m')$	$C^{(2)}(x_m')$	$\cdots$	$C^{(i)}(x_m')$	$\cdots \cdots$	$C^{(t)}(x_m')$

choices for the sequence of e_i, since $t \leqq \binom{2m}{2} = m(2m-1)$ by Fact 4. Also, for $i \geqq 1$, each column $C^{(i)}(x)$, $x \in V$, contains at most two new classes which did not occur in the preceding column since only two classes are changed at each step. Hence there are at most

$$[m(2m-1)]!\binom{2m}{2} + 2m$$

classes which can be generated by these rules where the additional term $+2m$ comes from the $2m$ initial sets $C^{(0)}(x)$, $x \in V$. On the other hand, since $\bar{C}$ contains 2^m sets $X \subseteq V$, then there are 2^{2^m} subsets of $\bar{C}$ which we must try to generate. We are doomed to failure however since

$$\left\{[m(2m-1)]!\binom{2m}{2} + 2m\right\} \Big/ 2^{2^m} \to 0$$

as $m \to \infty$. We list these expressions for several small values of m in Table V. Thus, not only are we guaranteed a single Boolean function of 11 variables which cannot be represented by a program, but in fact we have at least 10^{615} of them. It seems quite likely that there exist Boolean functions of five variables which cannot be represented. However, at present, no specific example of a Boolean function is known which cannot be represented by a program.

TABLE V—BOUNDS ON THE NUMBER OF BOOLEAN FUNCTIONS WHICH CAN BE GENERATED

m	$[m(2m-1)]!\binom{2m}{2} + 2m$	2^{2^m}
2	4324	16
3	19615115520006	256
10	$>10^{355}$	$<10^{309}$
11	$<10^{453}$	$>10^{616}$

V. SOME REMARKS

A number of partial results are known concerning the preceding problems which we shall only mention briefly here.

The generation of Boolean functions as described has the following very natural geometrical interpretation. For a fixed integer n, consider the set of the 2^n vertices of an n dimensional cube C^n and let $A_1, \cdots, A_{2n}$ represent the $2n$ sets of 2^{n-1} vertices which each lie on one $(n-1)$-dimensional "face". In other words, if the vertices of C^n are labelled by binary n-tuples in the usual way, then each A_i corresponds to a set of 2^{n-1} n-tuples in which some component is constant. As before, we are allowed to replace any two sets A and B in the class of $2n$ sets of $A \cap B$ and $A \cup B$. We can repeat this transformation as often as desired. The question is: which subsets $X \subseteq C^n$ can be generated in this manner. We have shown that there exists a set $X \subseteq C^{11}$ which cannot be so generated.

More generally, suppose we start with a class of n formal sets $X_1, \cdots, X_n$ and ask which formal expressions in the X_i can be generated using the transformation $X, Y \to X \cap Y, X \cup Y$ iteratively. It can be shown[11] for example, that all the elementary symmetric functions (using $\cap$ and $\cup$ in place of the usual $\cdot$ and $+$) can be generated. Let us call a well-formed expression E in the X_k's *symmetric in X_i and X_j* if the substitution $X_i \to X_j$, $X_j \to X_i$, leaves E unchanged. Thus we can write E in the form

$$E = (X_i \cap X_j \cap W_1) \cup ((X_i \cup X_j) \cap W_2) \cup W_3$$

where the W_i are well-formed (possibly empty) expressions in the X_k's *not* involving X_i or X_j. We say that we *collapse X_i and X_j in E* if we apply the transformation $X_i \cap X_j \to X_i$, $X_i \cup X_j \to X_j$, to form

$$E' = (X_i \cap W_1) \cup (X_j \cap W_2) \cup W_3 .$$

Certainly, if E can be generated using the transformations $X, Y \to X \cap Y, X \cup Y$ starting from $X_1, \cdots, X_n$, then there is a sequence of collapses starting with E and ending with some single variable X_i. A basic theorem can be proved which asserts that if it is possible to generate E, then no matter how we collapse symmetric variables starting with the expression E we must reach some single variable X_j. In other words in attempting to collapse E to a single variable, we can never make a "bad" move. Once the structure of the expressions E which can be generated is sufficiently well understood, perhaps the representable subsets of C^n can then be determined.

Another line of research suggested by this bubble model is in the

following direction. For binary sequences x and y, define $d(x, y)$, the (Hamming) distance between x and y, to be the number of positions in which the sequences x and y differ. The fact which prevented the existence of a program which could add two integers expressed to the base 2 was the fact that there are pairs of additions in which the binary expansions of the corresponding summands are close together (in the metric d) but whose sums are not close, thus conflicting with the NDO theorem. What we would like is a mapping $m \rightarrow \tau(m)$ of integers into binary sequences for which we have

$$d(\tau(m), \tau(n)) + d(\tau(m'), \tau(n')) \geqq d(\tau(m + n), \tau(m' + n')).$$

With only this constraint there are trivial solutions, for example,

$$m \rightarrow \underbrace{111 \cdots 1}_{m}.$$

With this mapping we are essentially expressing m to the base 1 (well-known by many cultures to be inefficient for representing large numbers, say, those exceeding 10). Hence, we might require in addition that the number of binary sequences of length t which are in the range of the mapping τ to be at least α^t for some fixed $\alpha > 1$. Is it possible to find a suitable τ for which an addition program *is* possible in this model of bubble interactions?

Finally, we have just considered just one rather simple model in this paper. Physically, many other bubble interactions are possible (although some presently operate with significantly smaller margins than others) and this of course would lead to other models. It would be very interesting to understand the corresponding questions in some of these other models.

VI. ACKNOWLEDGMENTS

The author wishes to express his indebtedness to W. Shockley, A. H. Bobeck, D. E. Knuth, A. A. Thiele and especially to J. H. Spencer for many interesting ideas and discussions on this subject.

REFERENCES

1. Bobeck, A. H., "Properties and Device Applications of Magnetic Domains in Orthoferrites," B.S.T.J., *46*, No. 8 (October 1967), pp. 1901–1925.
2. Galt, J. K., "Motion of Individual Domain Walls in a Nickel-Iron Ferrite," B.S.T.J., *33*, No. 5 (September 1954), pp. 1023–1054.
3. Gyorgy, E. M., and Hagedorn, F. B., "Analysis of Domain Wall Motion in Canted Antiferromagnets," J. Appl. Phys., *39*, No. 1 (January 1968), pp. 88–90.

4. Kooy, C., and Enz, U., "Experimental and Theoretical Study of the Domain Configuration in Thin Layers of $BaFe_{12}O_{19}$," Philips Res. Rep., *15*, No. 1 (February 1960), pp. 7–29.
5. Michaelis, P. C., "A New Method of Propagating Domains in Thin Ferromagnetic Films," J. Appl. Phys., *39*, No. 2, Part 2 (February 1968), pp. 1224–1226.
6. Thiele, A. A., "The Theory of Cylindrical Magnetic Domains," B.S.T.J., *48*, No. 10 (December 1969), pp. 3287–3336.
7. Smith, D. O., "Proposal for Magnetic Domain-Wall Storage and Logic," I.R.E. Trans. on Elec. Computers, *10*, No. 4 (December 1961), pp. 709–711.
8. Dillon, J. F., Jr., "Domains and Domain Walls," *Magnetism*, Vol. III, New York: Academic Press, (1963), pp. 450–453.
9. Spencer, J. H., unpublished work.
10. Graham, R. L., "On n-Valued Functionally Complete Truth Functions," Jour. Sym. Logic, *32*, No. 2 (June 1967), pp. 190–195.
11. Kurshan, R. P., "All Terminal Bubbles Programs Yield the Elementary Symmetric Polynomials," B.S.T.J., this issue, pp. 1995–1998.