BIT 14 (1974), 1–13

PROPERTIES OF BOOLEAN FUNCTIONS WITH A TREE DECOMPOSITION*

G. V. BOCHMANN AND W. W. ARMSTRONG

Abstract.

Boolean functions that have a multiple disjoint decomposition scheme in the form of a tree are considered. Properties of such functions are given for the case that the functions are increasing, unate, and/or have no vacuous variables. The functions with a binary decomposition scheme are of special interest. The modulus of sensitivity is defined, and evaluated for some classes of functions. The modulus of sensitivity is interesting from the point of view of semantic information processing. It is found that the sensitivity for the class of functions with a given disjoint binary decomposition scheme is much smaller than for the unrestricted class of boolean functions. This indicates that these functions are potentially useful in pattern recognition of discrete data.

1. Introduction.

This paper treats boolean functions with a multiple disjoint decomposition in the form of a tree. Properties of simple disjoint decompositions have been given by Ashenhurst [1] who points out the existence of a most decomposed form for any given boolean function. Some extensions to functions with non-disjoint decompositions are given by Curtis [2]. In this paper, we present a description of disjoint tree decompositions of boolean functions, and study properties of such tree functions which are significant from the point of view of semantic information processing.

Our principal interest is in functions associated with a binary tree. The value of such a function has a strong tendency to remain unchanged when a few of the inputs are inverted, as is shown in section 5. This represents a kind of "continuity" which makes the functions potentially useful for applications in pattern recognition. In fact, the discriminant functions used in pattern classification [3, 4] are almost always smoothly varying functions, such as linear or polynomial functions, and similarly, the distribution functions used in stochastic approximation methods are

* The authors gratefully acknowledge the financial support of the National Research Council of Canada through a postdoctoral fellowship and an operating grant respectively.

Received December 18, 1972. Revised September 13, 1973.

simple polynomials. This is important when a system trained on one set of data is to generalize its performance to a test set. Boolean tree functions exhibit an analogous insensitivity for discrete arguments. The authors have developed an adaptation algorithm for increasing tree functions [5], which satisfies a theorem analogous to the perceptron convergence theorem [4] for linearly separable functions. Generalization from a training set to a test set has been very satisfactory when this method was applied to handwritten numerals and measurements on iris plants, as will be described in a later paper.

The problem of finding an optimal realization of minimal cost for a given function is greatly simplified if the function has a decomposition. A family of tree functions, as defined in this paper, has been used to find an optimal realization of some given function by Roth and Wagner [6].

In this paper we consider only disjoint decompositions of functions, that is, within the decomposed form each variable appears only once. We note that every boolean function can be realized by a (large enough) non-disjoint decomposition, and that properties of the sensitivity similar to those described for disjoint tree functions in section 5, would also hold for a limited amount of non-disjointness (see [5], introduction).

In section 2, we give the basic definitions and show in theorem 2.3 some algebraic properties of multiple disjoint decompositions. In section 3, we define the concept of a tree function for a given decomposition scheme, and study its realizations. We then restrict ourselves to tree functions for binary trees, in section 4, and describe several classes of such functions.

The number of tree functions on a fixed tree is determined. Finally, in section 5, we introduce the modulus of sensitivity as a measure for the relative frequency of change of the function value when some arguments are perturbed. We determine its value for some classes of binary tree functions, and find it to be much smaller than for tree functions of higher order or for the unrestricted class of boolean functions of the given number of inputs.

2. Tree decomposition of Boolean functions.

In the following, we generally consider a tree, defined as usual by a set of *nodes* K, each node $k \in K$ having a certain number n_k of *successors* denoted by $s_{k1}, s_{k2}, \ldots, s_{kn_k}$. The *terminal* nodes with no successor form the set K_T, the *nonterminal* nodes form the set K_N. The root node of the tree is denoted by r.

We associate a set of distinct variables X with the terminal nodes K_T of the tree by a one-to-one correspondence between the variables $x_k \in X$ and the nodes $k \in K_T$. Together, tree and variables are characterized by the triple (K,s,X), where s is the successor function.

2.1. *Definition of a tree composition of boolean functions.*

Given a tree with variables (K,s,X) and boolean functions

$$g_k\colon \{0,1\}^{n_k} \to \{0,1\}$$

for each nonterminal node $k \in K_N$, we define recursively, on the tree and on its subtrees, the composite functions

$$f_k\colon \{0,1\}^{X_k} \to \{0,1\}\,.$$

X_k is the set of variables of the terminal nodes of the subtree of the node k. More precisely

$$X_k = \{x_k\} \qquad \text{for } k \in K_T$$

$$X_k = \bigcup_{i=1,\ldots,n_k} X_{s_{ki}} \quad \text{for } k \in K_N$$

and

$$f_k = x_k \qquad \text{for } k \in K_T\,,$$

$$f_k = g_k(f_{s_{k1}}, f_{s_{k2}}, \ldots, f_{s_{kn_k}}) \quad \text{for } k \in K_N\,.$$

The function f_r is the composite function on the entire tree, and the functions g_k are called the *node functions* of f_r.

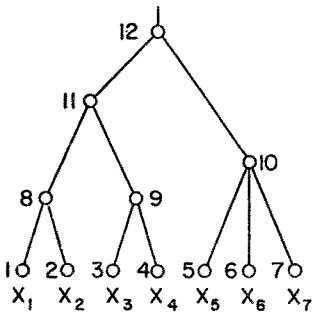

Fig. 1. A tree with terminal nodes 1,2,... 7, nonterminal nodes 8, 9, 10, 11, 12, and root node 12.

2.2. As an example we take the tree of fig. 1. With node functions g_k ($k=8,9,\ldots 12$), the composite function f_{12} is given by

$$f_{12}(x_1,\ldots,x_7) = g_{12}\big(g_{11}(g_8(x_1,x_2),g_9(x_3,x_4)),g_{10}(x_5,x_6,x_7)\big).$$

We now consider the following properties for a boolean function f (see for example [7]):

(a) f has no vacuous variables, i.e. f depends on all its variables X.
(b) f is a unate function, i.e. for each variable $x_i \in X$, f is either positive or negative.
(c) f is an increasing function (logically passive, positive unate function), i.e. f is positive in all variables.

2.3 THEOREM. *Given a tree with variables (K,s,X), and a composite function f_r defined by the node functions $g_k(k \in K_N)$, we have:*

(a) *f_r has no vacuous variables iff all node functions g_k have none.*
(b) *If all g_k are unate then f_r is unate, and if f_r is unate with no vacuous variables then all g_k are unate.*
(c) *If all g_k are increasing then f_r is increasing, and if f_r is increasing with no vacuous variables and $g_k(0)=0$ for all g_k then all g_k are increasing.*

In order to keep this article short we refer to [8] for the proof.

3. Boolean tree functions.

3.1. *Definition of tree functions* (TF).

Given a tree with variables $t=(K,s,X)$ and a boolean function $f\colon \{0,1\}^X \to \{0,1\}$, we say that f is *realizable by* t or that f is a *tree function* on t iff there exists an assignment of boolean functions $g_k(k \in K_N)$ such that the composite function f_r is equal to f. We say that the function assignment is a *realization* of f. We write F_t for the set of all TF realizable on t.

We mention that a given TF may have several distinct realizations on a given tree, as in the following examples:

(a) $g(x_1,OR(x_2,x_3))=g(x_1,AND(x_2,x_3))=g(x_1,0)$ if the second argument of g is vacuous.
(b) $b_3(x_1,b_1(x_2,x_3))=b_2(x_1,\bar{b}_1(x_2,x_3))$ where the functions $b_i(i=1,\ldots,3)$ are defined in Table 1 and the bar stands for complementation.

Table I. *The zero preserving* (ZP) *boolean functions of two variables.*

Symbol	Expression	Comment
$b_1(x,y)$	$x+y$	OR
$b_2(x,y)$	xy	AND
$b_3(x,y)$	$x\bar{y}$	
$b_4(x,y)$	$\bar{x}y$	
$b_5(x,y)$	$x \oplus y$	exclusive OR
$b_6(x,y)$	x	left connection
$b_7(x,y)$	y	right connection
$b_8(x,y)$	0	constant zero

A given boolean function f may be realizable on different trees. For example, the overall AND function is realized whenever all node functions are AND functions.

3.2. *The* ZP *realization.*

A realization of a TF is called a ZP (*zero-preserving*) realization iff $g_k(0)=0$ for all $k \in K_N$. (We note that for a ZP realization the values of all subtree functions $f_k(k \in K_N)$ are zero if all their arguments are zero.)

We have the following lemmas, the proof of which again can be found in [8]:

(a) Given a tree with variables $t=(K,s,X)$ and a tree function $f \in F_t$, there exists a ZP realization of f iff $f(0)=0$.
(b) If f has no vacuous variables then this ZP realization is unique.

4. Tree functions for binary trees.

In the following we consider only binary trees, i.e. trees in which each nonterminal node has exactly two successors. However, many of the statements that follow can be generalized to arbitrary trees. We consider only boolean functions f with $f(0)=0$. This does not significantly restrict the generality, since we would otherwise consider $\bar{f}$.

4.1 *The boolean functions of two variables.*

A realization of a TF on a binary tree consists of boolean functions of two variables. A short summary of these functions is given in table I. Besides the eight functions in the table, there are eight other functions of two variables which are obtained from $b_1,\ldots,b_8$ by complementation. We ignore them here because they do not appear in ZP realizations.

We note that of these eight

(a) only $b_1, b_2, \ldots, b_5$ have no vacuous variables,
(b) only b_5 is not unate,
(c) only b_3, b_4, b_5 are not increasing.

4.2. *Some classes of tree functions.*

We now define classes of TF which satisfy certain conditions. Later we give some properties of these classes, such as their interrelations, number of elements, and moduli of sensitivity.

Given a binary tree with variables $t=(K,s,X)$, we denote by $F_t^{(i)}$ the class of all TF that have a ZP realization using as node functions only $b_1, b_2, \ldots, b_i$. We note that

(i) $F_t^{(8)}$ consists of all functions $f \in F_t$ with $f(0)=0$. From 3.2 we have

$$F_t = F_t^{(8)} \cup \overline{F_t^{(8)}} \text{ (disjoint)} .$$

Furthermore, from theorem 2.3, we have the following:

(ii) $F_t^{(5)}$ consists of all functions $f \in F_t$ with $f(0)=0$ and no vacuous variables.

(iii) $F_t^{(4)}$ is the set of all unate functions in $F_t^{(5)}$.

(iv) $F_t^{(2)}$ is the set of all increasing functions in $F_t^{(5)}$.

4.3 *Some examples.*

(a) We consider the class of boolean functions of three variables which are increasing and without vacuous variables. This class consists of the following functions:

$$\begin{aligned}
f_1 &= AND(x_1, AND(x_2, x_3)) &&= x_1x_2x_3 \\
f_2 &= AND(x_1, OR(x_2, x_3)) &&= x_1x_2 + x_1x_3 \\
f_3 &= AND(x_2, OR(x_3, x_1)) &&= x_2x_3 + x_2x_1 \\
f_4 &= AND(x_3, OR(x_1, x_2)) &&= x_3x_1 + x_3x_2 \\
f_5 &= OR(x_1, AND(x_2, x_3)) &&= x_1 + x_2x_3 \\
f_6 &= OR(x_2, AND(x_3, x_1)) &&= x_2 + x_3x_1 \\
f_7 &= OR(x_3, AND(x_1, x_2)) &&= x_3 + x_1x_2 \\
f_8 &= OR(x_1, OR(x_2, x_3)) &&= x_1 + x_2 + x_3 \\
f_9 &= \text{(nondisjoint tree)} &&= x_1x_2 + x_1x_3 + x_2x_3
\end{aligned}$$

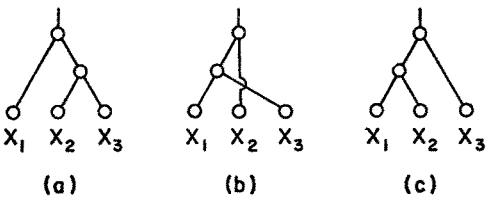

Fig. 2. The binary trees with three terminal nodes.

The possible binary trees with three terminal nodes are shown in fig. 2. One finds that for the tree

$$\text{of fig. 2(a)}: \; F_t^{(2)} = \{f_1, f_2, f_5, f_8\}$$

$$\text{for fig. 2(b)}: \; F_t^{(2)} = \{f_1, f_3, f_6, f_8\}$$

$$\text{for fig. 2(c)}: \; F_t^{(2)} = \{f_1, f_4, f_7, f_8\}.$$

We see that all functions of the class except f_9 are realizable by some tree.

(b) The function f_9 above and

$$f = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$$

which are increasing, without vacuous variables, symmetric, and linearly separable (i.e. threshold functions [7]) have *no* (disjoint) tree realizations. On the other hand, the function

$$f = (x_1 + x_2)(x_3 + x_4)$$

has a tree realization but is not linearly separable, i.e. there exist no real numbers $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta$ such that $f(\xi) = 1$ iff $\sum_i \alpha_i \xi_i \geqq \theta$. (For, if so, $\alpha_1 + \alpha_3 \geqq \theta$ and $\alpha_2 + \alpha_4 \geqq \theta$, which would imply either $\alpha_1 + \alpha_2 \geqq \theta$ or $\alpha_3 + \alpha_4 \geqq \theta$ which would give a wrong value for f).

We note that any boolean function can be realized by a non-disjoint decomposition if a sufficiently large decomposition tree is chosen and variables at different terminal nodes are identified.

4.4 LEMMA. *The set of all unate* TFs *is identical to the set of all functions obtained from increasing* TFs *by complementing some non-vacuous variables.*

The proof can be found in [8]. We note that a similar statement holds without the restriction to tree-realizable functions, since increasing functions are unate functions which are positive in each non-vacuous variable [7].

4.5 LEMMA. *Reduction of a* TF *with vacuous variables.*

Let $t=(K,s,X)$ *be a binary tree with* n *terminal nodes, and* $f \in F_t$. *If* f *has* $j \geqq 1$ *vacuous variables, then* f *is equal (up to the obvious projection* $\{0,1\}^n \to \{0,1\}^m$*) to a* TF *on a reduced tree with* $m=n-j$ *terminal nodes which depends on all* m *variables. (An example is shown in fig.* 3). *For the proof see* [8].

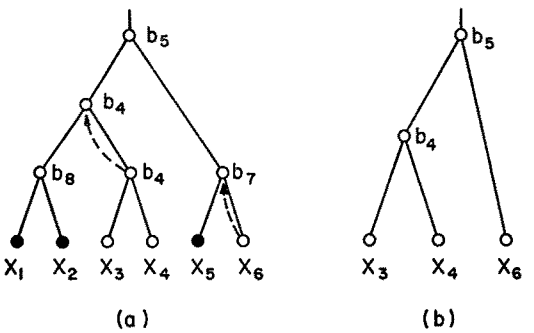

Fig. 3. (a) A realization of the tree function

$$f(x_1,\ldots x_6) = b_5(b_4(b_8(x_1,x_2),b_4(x_3,x_4)),b_7(x_5,x_6))$$

which has three vacuous variables x_1, x_2, and x_5. The replacements for a tree reduction (section 4.6) are shown (dotted arrows). The functions b_i are defined in Table I.

(b) The tree function

$$f_r(x_3,x_4,x_6) = b_5(b_4(x_3,x_4),x_6)$$

is obtained from (a) after reduction.

4.6 *Theorem on the number of tree functions.*

Given a fixed binary tree with n *terminal nodes, the number*

(i) *of functions* $(j=5)$

(ii) *of unate functions* $(j=4)$

(iii) *of increasing and decreasing functions* $(j=2)$ *which have a realization on* t, *is given by*

$$N_j = 2\left\{1+\frac{1}{j}[(j+1)^n-1]\right\}$$

with the value of j *as indicated above.*

PROOF.

(a) The factor 2 is due to the two cases $f(0)=0$ and $f(0)=1$.

(b) The leftmost 1 accounts for the constant functions $f\equiv 0$ and $f\equiv 1$.

(c) $(1/j)[(j+1)^n - 1] = \sum_{m=1}^{n} \binom{n}{m} j^{m-1}$ is the number of non-constant functions. To see this we note that there are $\binom{n}{m}$ different sets of m variables (out of n) on which a function can depend. Each set corresponds to a reduced tree of m terminal nodes (see Lemma 4.5). The number of functions in $F^{(j)}$ on a reduced tree is equal to j^{m-1} for $j \leqq 5$ by 3.2(b), since these functions depend on all m variables. Section 4.2 justifies the choice of values for j.

We note that the above number of TFs are calculated for a given binary tree. For a given set of terminal nodes, these numbers do not depend on the form of the tree, however, the classes of functions do. (See example 4.3(a)).

A comparison with the number 2^{2^n} of boolean functions of n variables shows that for large n only a small fraction of the boolean functions are TF on a given tree.

5. The modulus of sensitivity for tree functions.

In this section we introduce the concept of modulus of sensitivity of a boolean function. We shall show that the probability of a change in the value of an average tree function is rather small when some argument values are complemented, compared to the probability of change for an arbitrary boolean function. This property is inherent in the structure of tree functions, and indicates that tree functions may be appropriate for certain applications where "well-behaved" functions are required as approximants, such as for decision functions in pattern recognition. Tree functions have the potential to provide a certain kind of generalization from a training set to a test set of patterns.

5.1 DEFINITION. We define the modulus of sensitivity μ for a class C of boolean functions of n variables as the function $\mu\colon \{0,1,\ldots,n\} \to [0,1]$ given by

$$\mu(m) = \left\{ \operatorname{card}(C) 2^n \binom{n}{m} \right\}^{-1} \sum_{f \in C} \sum_{\xi \in \{0,1\}^n} \operatorname{card}\{\xi' \mid d(\xi,\xi') = m; f(\xi) \neq f(\xi')\}$$

which represents the average probability over all functions of C and over all argument vectors that a change of m argument values gives rise to a change of the function value (d = Hamming distance).

For the set of all boolean functions of n variables we find $\mu(m) = 0.5$ for $m = 1,2,\ldots,n$.

For $C = \{b_k \mid k = 1,\ldots,i\}$ where the functions b_k are listed in the table I, we find by direct calculation

$$\mu(1) = 0.5, \quad \mu(2) = 0.5 \quad \text{for } i = 2,4;$$

$$\mu(1) = 0.6, \quad \mu(2) = 0.4 \quad \text{for } i = 5 .$$

We denote these numbers by $\mu_1^{(i)}$ and $\mu_2^{(i)}$.

5.2 *The modulus of sensitivity of the tree functions for a balanced binary tree.*
In general it may be quite complicated to calculate $\mu(m)$ for a class C of functions and values $m = 1,2,\ldots,n$. In [8] we give a lemma which allows recursive calculation of $\mu(m)$ for the classes $F_l^{(i)}$ $(i = 2,4,5)$. In

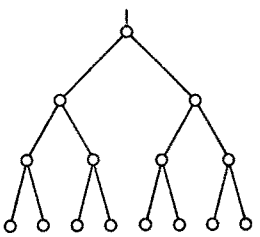

Fig. 4. A balanced binary tree with $l = 3$ levels.

the case of a balanced binary tree t (see fig. 4), the left and right subtrees for any given node are isomorphic, so we can characterize a subtree by its level l. We then have for the class $F_l^{(i)}$ $(i = 2,4,5)$ and $m = 0,1,\ldots,2^l$ the recursion formula:

$$\mu_{(l)}(m) = \binom{2^l}{m}^{-1} \sum_{m_s=0}^{m} \binom{2^{l-1}}{m_s} \binom{2^{l-1}}{m-m_s} \cdot$$

$$\{\mu_1^{(i)}[\mu_{(l-1)}(m_s) + \mu_{(l-1)}(m-m_s)] + [\mu_2^{(i)} - 2\mu_1^{(i)}]\mu_{(l-1)}(m_s)\mu_{(l-1)}(m-m_s)\}$$

One finds, for example, $\mu_{(l)}(1) = (\mu_1^{(i)})^l$ which is the sensitivity with respect to the complementation of one variable.

5.3 *Limiting values for large trees.*
For balanced trees, $\mu_{(l)}(m)$ approaches a limit as the level l of the tree goes to infinity and the proportion of complementations m/n, where $n = 2^l$, remains constant. The value of this limit is independent of m/n. We have a limit $\mu_{\lim}(m/n)$ of $\mu_{(l)}(m)$ if from one level $(l-1)$ to the next level l, the equilibrium condition

$$\mu_{(l)}(m) = \mu_{(l-1)}(m/2) \equiv \mu_{\lim}(m/n)$$

is satisfied. In [8] it is shown that

$$(2\mu_1^{(i)} - 1)\mu_{\lim}(m/n) + (\mu_2^{(i)} - 2\mu_1^{(i)})[\mu_{\lim}(m/n)]^2 = 0$$

with the stable solution

$$\mu_{\lim} = \frac{2\mu_1^{(i)} - 1}{2\mu_1^{(i)} - \mu_2^{(i)}}.$$

Inserting values for $\mu_j^{(i)}$, we find the limit of the modulus of sensitivity for $l \to \infty$ to be

$$\mu_{\lim}(m/n) = 0.25 \text{ for the class } F_l^{(5)}$$

and

$$\mu_{\lim}(m/n) = 0.0 \text{ for the classes } F_l^{(4)} \text{ and } F_l^{(2)}.$$

We note that this limit is independent of the relative number m/n of complementations of variable values.

We have also evaluated the modulus of sensitivity μ for the classes $F_l^{(i)}$ $(i = 2, 4, 5)$ on balanced trees of levels up to $l = 9$, using the formula of section 5.2. The results are shown in fig. 5a, b.

6. Conclusions.

We have described properties of boolean functions that have a multiple disjoint decomposition scheme in the form of a tree. We have specially mentioned the classes of tree functions that are increasing, or unate, and/or which have no vacuous variables. For the class of all binary TF on a large, balanced tree, the probability that the complementation of a number of variables induces a changed function value, is very small. For very large trees, the modulus of sensitivity is almost 0.25 for the class of functions with no vacuous variables, and 0.00 for the classes of unate and of increasing functions with no vacuous variables. Surprisingly enough, this is the case even when almost all of the variables are complemented. The difference in the values arises since the exclusive OR function b_5 is not allowed as node function in a unate tree function. In contrast to this, the average of the modulus of sensitivity over all boolean functions of n variables is $\mu(d) = 0.5$ for $1 \leqq d \leqq n$. The relatively small sensitivity of tree functions is inherent in the decomposition property, and is particularly remarkable for binary trees. For example, in the case of tree functions with decomposition into node functions of three variables (in contrast to two variables in binary trees), the limiting value of μ for large balanced trees is $\mu_{\lim} = 0.18$ [8] for the class of unate functions with no vacuous variables as compared to 0.0 for binary trees.

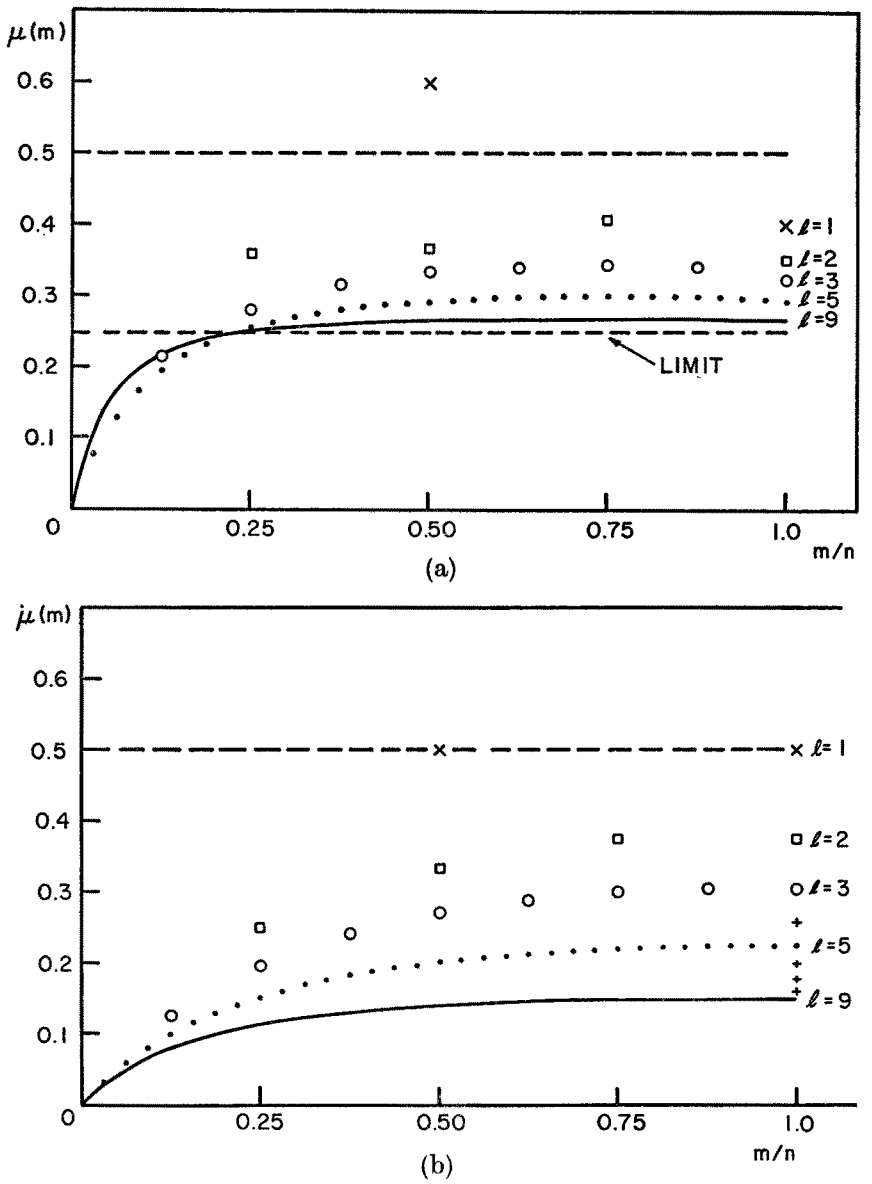

Fig. 5. The modulus of sensitivity $\mu_{(l)}(m)$ for the class $F_t^{(i)}$ of tree functions on *balanced* trees with l levels and $n=2^l$ variables. The abscissa m/n represents the relative number of complementations of variables, i.e. the proportion of perturbed input signals, and the ordinate represents the probability of the output changing due to this perturbation.

(a) $i=5$: the class of tree functions with no vacuous variables.

(b) $i=2$ and 4: the classes of increasing and unate tree functions with no vacuous variables. The limit is constant 0.

It is clear that some insensitivity is also present if we allow some non-disjointness in the decomposition schemes. This kind of insensitivity of binary tree functions has been found useful in the construction of discriminant functions where generalisation (i.e. insensitive extrapolation) from a training set to a test set is required.

REFERENCES

1. R. L. Ashenhurst, in Proceedings of an International Symposium on the Theory of Switching, 1957, Vol. 29 of Annals of Computation Laboratory of Harvard University, p. 74, 1959.
2. H. A. Curtis, *A new approach to the design of switching circuits*, D. Van Nostrand Inc., Princeton, N.J., 1962.
3. J. M. Mendel and K. S. Fu, *Adaptive, learning, and pattern recognition systems*, Academic Press, 1970.
4. N. J. Nilsson, *Learning Machines*, McGraw–Hill, 1965.
5. W. W. Armstrong and G. V. Bochmann, *A convergence theorem for logical network adaptation*. Publication #95, Département d'Informatique, Université de Montréal, 1972.
6. J. P. Roth and E. G. Wagner, *Algebraic topological methods for synthesis of switching systems: Minimisation of non-singular boolean trees*, IBM J. of Res. and Dev., 4 (October 1959), 326–344.
7. R. E. Miller, *Switching theory*, Vol. 1, Wiley, New York, 1965.
8. G. V. Bochmann and W. W. Armstrong, *Properties of Boolean Functions with a Tree Decomposition*, Publication #87, Département d'Informatique, Université de Montréal, May 1972.

DEPARTEMENT D'INFORMATIQUE
UNIVERSITE DE MONTREAL
CANADA