A note on symmetric Boolean functions

By P. Ecsedi—Tóth,* F. Móricz** and A. Varga**

To the memory of Professor Géza Fodor

Introduction

The notion of a symmetric function can be found in any textbook on switching theory or logical design. It is well-known (Shannon [1]) that the truth-value of a symmetric function depends only on the number of literals for which the truth-value TRUE is substituted. More precisely, the following theorem holds.

Theorem (Shannon). *Let φ be a Boolean function of n variables. φ is a symmetric function if and only if there exists a set of integers $\{n_1, n_2, \ldots, n_k\}$* (called the Shannon set of φ) *($k \leqq n$, $0 \leqq n_i \leqq n$ for $i \leqq k$) such that the truth-value of φ is* TRUE *iff for exactly n_i of the literals* TRUE *is substituted.*

The proof of this theorem gives no idea how to determine the set $\{n_1, n_2, \ldots, n_k\}$. Since symmetric functions have nice properties, it is important to decide whether a given function φ is symmetric or not. As far as we know, there are only trivial methods (i.e., to test all possible cases) for the solution of this problem.

In this paper we present an effective algorithm to determine the Shannon set of a Boolean function if it exists. The method is based on the tree-representation of Boolean functions used by the present authors [2] to get irredundant normal-forms as representation of them. In particular, we associate a number — the number of negative literals — to each path of this tree. Then by a simple comparison of the endnodes of the paths and of the associated numbers, we can collect the Shannon set provided it exists.

1. The tree-representation of Boolean functions

To make the paper self-contained, we present here the tree-construction algorithm, too. A more detailed explanation and the basic results can be found in [2].

Let a Boolean function φ be given in which at least one variable occurs. Choose a variable of φ according to some rule (a so-called selection function), fixed previously. First, substitute the truth-values TRUE and FALSE, respectively, for the chosen variable. Then eliminate the truth-values from both expressions obtained by using the following transformation rules:

$$
\begin{array}{ll}
\varphi\wedge 1 \leftrightarrow \varphi; & 1\wedge\varphi \leftrightarrow \varphi \\
\varphi\wedge 0 \leftrightarrow 0; & 0\wedge\varphi \leftrightarrow 0 \\
\varphi\vee 1 \leftrightarrow 1; & 1\vee\varphi \leftrightarrow 1 \\
\varphi\vee 0 \leftrightarrow \varphi; & 0\vee\varphi \leftrightarrow \varphi \\
\varphi\rightarrow 1 \leftrightarrow 1 & \\
\varphi\rightarrow 0 \leftrightarrow \bar{\varphi} & \\
1\rightarrow \varphi \leftrightarrow \varphi & \\
0\rightarrow \varphi \leftrightarrow 1 & \\
(\varphi\leftrightarrow 1)\leftrightarrow \varphi; & (1\leftrightarrow\varphi)\leftrightarrow\varphi \\
(\varphi\leftrightarrow 0)\leftrightarrow \bar{\varphi}; & (0\leftrightarrow\varphi)\leftrightarrow\bar{\varphi} \\
\varphi\wedge\varphi \leftrightarrow \varphi & \\
\varphi\wedge\bar{\varphi} \leftrightarrow 0; & \bar{\varphi}\wedge\varphi \leftrightarrow 0 \\
\varphi\vee\varphi \leftrightarrow \varphi & \\
\varphi\vee\bar{\varphi} \leftrightarrow 1; & \bar{\varphi}\vee\varphi \leftrightarrow 1 \\
\varphi\rightarrow\varphi \leftrightarrow 1 & \\
(\varphi\leftrightarrow\varphi)\leftrightarrow 1 & \\
\bar{\bar{\varphi}} \leftrightarrow \varphi & \\
\bar{1} \leftrightarrow 0 & \\
\bar{0} \leftrightarrow 1 &
\end{array}
$$

As a result of the elimination process we come to one of the following two cases:

(i) The expression obtained contains at least one variable. Then let us choose a variable in it according to our rule, and repeat the substitution and the elimination described above.

(ii) The expression obtained is a single truth-value. Then the algorithm stops.

We note that the function φ, together with a selection function, determines its tree uniquely up to isomorphism, and conversely, every binary tree determines a Boolean function uniquely up to logical equivalence.

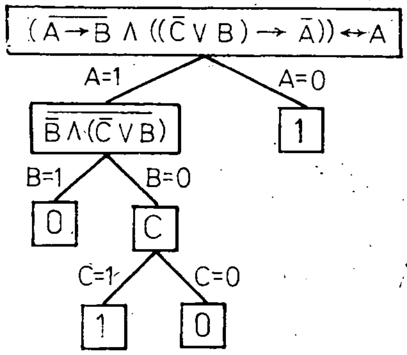
The following example illustrates the method. We use the usual logical connectives ($\wedge$ for conjunction, $\vee$ for disjunction, $\rightarrow$ for implication, $\leftrightarrow$ for equivalence, and $^-$ (bar) for negation). 1 and 0 will denote the truth-values TRUE and FALSE, respectively.

Example. Let the Boolean function φ be as follows:

$$\varphi\colon\ ((\overline{A \to B})\wedge((\bar{C}\vee B) \to \bar{A})) \leftrightarrow A.$$

Let us choose the variables alphabetically. The substituted and "simplified" expressions can be arranged in a tree as indicated in Fig. 1.

As it was proved in [2, Corollary 6], the function φ and also its sub-functions can be omitted since they are obtainable from the shape of the tree, so it is enough to draw the simpler form as indicated in Fig. 2.



$(\overline{A \to B} \wedge ((\bar{C} \vee B) \to \bar{A})) \leftrightarrow A$
A=1
A=0
$\bar{B} \wedge (\bar{C} \vee B)$
1
B=1
B=0
0
C
C=1
C=0
1
0

Fig. 1

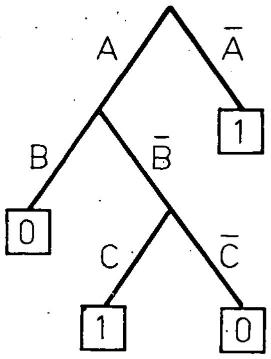


Fig. 2

The concept of a complete tree was introduced also in [2]. A tree of a Boolean function is *complete* iff all paths from the root to an endnode of the tree have the same length, which is equal to the number of the variables of φ. The reader can easily verify the following two assertions.

Lemma 1. *Let φ be a Boolean function of n variables. Then one can find a Boolean function φ' with the same variables, the tree of which is complete and φ' is logically equivalent to φ. φ' and its complete tree are uniquely determined.*

In practice, it is very easy to get a complete tree from any incomplete one as Fig. 3 shows.

Lemma 2. *Let φ be a Boolean function of n variables and suppose that the tree of φ is complete. Then there exist exactly 2^n paths in the tree of φ.*

Convention. In the rest of this paper we shall assume that every tree is drawn in such a way that the positive sub-expressions (those which can be obtained by substituting TRUE for a variable) are drawn on the left-hand side, while the negative sub-expressions are drawn on the right-hand side of the tree. Observe that trees in Fig. 1—3 correspond to this convention.

Definition. Let φ be a Boolean function of n variables. Then its *complete tree* is the tree of φ' determined by Lemma 1.

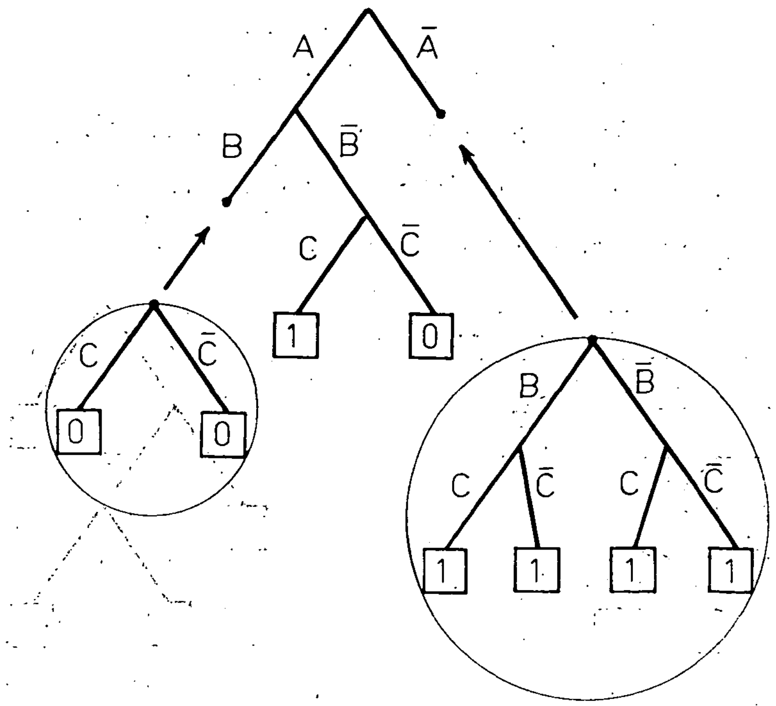


Fig. 3

2. The sequence ξ

Definition. Let us define the sequence of non-negative integers $\langle\xi_k\rangle$ by ξ_k=the number of 1 in the binary expansion of $k-1$ $(k=1, 2, \ldots)$.

Definition. Let $\langle\zeta_k\rangle$ be defined by the following recurrence:

$$\zeta_1 = 0,$$
$$\zeta_{2k-1} = \zeta_k,$$
$$\zeta_{2k} = \zeta_k+1.$$

Lemma 3. *We have* $\xi_k=\zeta_k$ *for every* $k=1, 2, \ldots$.

Proof. If $k=1$, then the lemma holds by definition. For every $k\geqq 2$ there exists exactly one non-negative integer n such that $2^n<k\leqq 2^{n+1}$. We proceed by induction on n.

Let n be fixed. Assume $2^n<k\leqq 2^{n+1}$ and that $l\leqq 2^n$ implies $\xi_l=\zeta_l$.

Let $k=2l-1$. Obviously, $k-1=2l-2$ is even and $l\leqq 2^n$, so

$$\zeta_k = \zeta_l = \xi_l = \xi_k.$$

Note that the last equation holds, since multiplication by 2 simple means a shifting in the binary expansion of $k-1$.

Let $k=2l$. Obviously $k-1=2l-1$ is odd and $l \leqq 2^n$, so

$$\zeta_k = \zeta_{2l} = \zeta_l + 1 = \xi_l + 1 = \xi_{2l-1} + 1,$$

where again the last equation holds by the shifting property mentioned above. We have to prove that

$$\xi_{2l-1} + 1 = \xi_{2l}.$$

However, this readily follows by definition from the fact that $2l-1$ is odd and $2l-1=2l-2+1$.

The proof of Lemma 3 is complete.

Definition. For each non-negative integer n we define $\xi_{2^n}(k)$ by the following recurrence:

(i) $\xi_{2^0}(1) = 0,$

(ii) $\xi_{2^{n+1}}(k) = \begin{cases} \xi_{2^n}(k) & \text{if } 0 < k \leqq 2^n, \\ \xi_{2^n}(l) & \text{if } 2^n < k \leqq 2^{n+1} \text{ and } k = 2^n + l. \end{cases}$

Lemma 4. *We have* $\zeta_k = \xi_{2^n}(k)$ *provided* $0<k \leqq 2^n$.

Proof. It is enough to prove that

$$\xi_{2^{n+1}}(2k-1) = \xi_{2^n}(k)$$

and (1)

$$\xi_{2^{n+1}}(2k) = \xi_{2^n}(k)+1,$$

since if we assume that $0<k\leqq 2^n$ entails

$$\zeta_k = \xi_{2^n}(k), \tag{2}$$

then if $l=2k-1$ $(2^n<l\leqq 2^{n+1})$, then

$$\zeta_l = \zeta_{2k-1} = \zeta_k = \xi_{2^n}(k)$$

by (2); and if $l=2k$ $(2^n<l\leqq 2^{n+1})$, then

$$\zeta_l = \zeta_{2k} = \zeta_k + 1 = \xi_{2^n}(k)+1.$$

We prove (1) by induction on n. If $n=0$, then (1) trivially holds. If $n\neq 0$, then we prove that

$$\xi_{2^{n+2}}(2k-1) = \xi_{2^{n+1}}(k)$$

and

$$\xi_{2^{n+2}}(2k) = \xi_{2^{n+1}}(k)+1 \quad (k = 1, 2, \dots, 2^{n+1}).$$

In each case two subcases will be distinguished.

1) If k is odd and $2k-1\leqq 2^n$, then

$$\xi_{2^{n+2}}(2k-1) = \xi_{2^{n+1}}(2k-1) = \xi_{2^n}(k) = \xi_{2^{n+1}}(k).$$

2) If k is odd and $2k-1>2^n$, then

$$2k-1 = 2^n+l,$$

where $l\leqq 2^n$ and l is odd, thus

$$2k-1 = 2^n+2m-1.$$

We have also for $k=2^n+m$,

$$\xi_{2^{n+2}}(2k-1) = \xi_{2^{n+2}}(2^n+l) = \xi_{2^{n+1}}(l)+1 =$$
$$= \xi_{2^n}(m)+1 = \xi_{2^{n+1}}(2^{n-1}+m) = \xi_{2^{n+1}}(k).$$

3) If k is even and $2k\leqq 2^n$, then

$$\xi_{2^{n+2}}(2k) = \xi_{2^{n+1}}(2k) = \xi_{2^n}(k)+1 = \xi_{2^{n+1}}(k)+1.$$

4) If k is even and $2k>2^n$, then

$$\xi_{2^{n+2}}(2k) = \xi_{2^{n+2}}(2^n+l) = \xi_{2^{n+1}}(l)+1 = \xi_{2^{n+1}}(2m)+1 =$$
$$= \xi_{2^n}(m)+2 = \xi_{2^{n+1}}(2^{n-1}+m)+1 = \xi_{2^{n+1}}(k)+1.$$

The proof of Lemma 4 is complete.

The sequence $\xi_{2^n}(k)$ can be easily generated, so by Lemma 3 and Lemma 4 we have a "fast" algorithm to obtain the sequence $\langle\xi_k\rangle$. The use of this sequence is shown by the following

Theorem 5. *Let φ be a Boolean function of n variables. Let us number the endnodes in its complete tree by $k=1, 2, \dots, 2^n$ from the left to the right. Then ξ_k means the number of the negative literals in the path, the endnode of which is numbered by k.*

Proof. Denote by $n(k)$ the number of the negative literals in the path labelled by k. Actually, one can prove by induction on the number of the variables in φ that

$$n(k) = \zeta_k.$$

3. Symmetric functions

Definition. Let T be the set of indices of those paths, whose endnodes are TRUE.

Corollary 6. *Let φ be an n-ary symmetric function and let m be an arbitrary non-negative integer such that $m\leqq n$. Then m is an element of the Shannon set of φ if and only if*

$$\{k\,|\,n-\xi_k = m\} \subseteqq T.$$

Proof. It is quite easy by Theorem 5.

* RESEARCH GROUP ON MATHEMATICAL LOGIC
AND THEORY OF AUTOMATA OF THE
HUNGARIAN ACADEMY OF SCIENCES
H-6720 SZEGED, HUNGARY
SOMOGYI U. 7.

** BOLYAI INSTITUTE OF THE
JÓZSEF ATTILA UNIVERSITY
H-6720 SZEGED, HUNGARY
ARADI VÉRTANÚK TERE 1.

References

[1] Shannon, C. E., A symbolic analysis of relay and switching circuits, *Trans. A.I.E.E.*, v. 57, 1938, pp. 713—723.

[2] Varga, A., P. Ecsedi-Tóth and F. Móricz, An effective method for minimizing Boolean polynomials, *Research Report* (Research Group on Mathematical Logic and Theory of Automata of the Hungarian Academy of Sciences), 1977, p. 40.

(Received Oct. 13, 1977)