The complexity of modular decomposition of Boolean functions

Jan C. Bioch

Department of Econometrics, Erasmus University Rotterdam, FEW, PO Box 1738, 3000 DR Rotterdam, Netherlands

Received 30 March 2002; received in revised form 14 November 2003; accepted 11 December 2003

Available online 10 May 2005

Abstract

Modular decomposition is a thoroughly investigated topic in many areas such as switching theory, reliability theory, game theory and graph theory. We propose an $O(mn)$-algorithm for the recognition of a modular set of a monotone Boolean function $f$ with $m$ prime implicants and $n$ variables. Using this result we show that the computation of the modular closure of a set can be done in time $O(mn^2)$. On the other hand, we prove that the recognition problem for general Boolean functions is coNP-complete. Moreover, we introduce the so-called generalized Shannon decomposition of a Boolean function as an efficient tool for proving theorems on Boolean function decompositions.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Boolean functions; Committees; Computational complexity; Modular decomposition; Modular sets; Substitution decomposition; Game theory; Reliability theory; Switching theory

1. Introduction

Substitution decomposition has been thoroughly studied by researchers in many different contexts such as switching theory, game theory, reliability theory, network theory, graph theory and hypergraph theory. Möhring and Radermacher [21,22] give an excellent survey for the various applications of substitution decomposition and connections with combinatorial optimization. They also present a framework for the algebraic and algorithmic aspects of substitution decomposition for a number of discrete structures. Substitution
decomposition (disjunctive and non-disjunctive decomposition) for general Boolean functions and partially defined Boolean functions in switching theory is mainly developed by Ashenhurst, Singer, Curtis and Hu [1,2,16–18]. Nowadays decomposition of Boolean functions is an important design-methodology in automatic synthesis for Field Programmable Gate Arrays (FPGAs), see e.g. [27,23]. Recently [11,10,19] the complexity of non-disjunctive decompositions of partially defined Boolean functions has been determined for various classes of Boolean functions. In this direction the recent paper of Zupan et al. [31] on concept hierarchies deserves further attention. In this paper a recursive decomposition of partially defined discrete function is used to obtain structural information of a data set. Decomposition for monotone Boolean functions has been studied in several contexts: game theory (decomposition of n-person games [29]), reliability theory (decomposition of coherent systems [8]) and set systems (clutters [3]). The concepts decomposition and modular set are very basic in many contexts and applications. Not surprisingly, the concept of a modular set is rediscovered several times under various names: bound sets, autonomous sets, closed sets, stable sets, clumps, committees, externally related sets, intervals, nonsimplifiable subnetworks, partitive sets and modules, see [12,22] and references therein. In all these contexts the collection of all modular sets is efficiently represented by the so-called decomposition tree introduced by Shaply in [29]. In graph theory efficient algorithms are known to compute this tree [12,20,14]. The notion of a module in a graph has been recently generalized to hypergraphs in [9]. A unified treatment of all algorithms (up to 1990) related to modular sets known in game theory, reliability theory and set systems (clutters) is given by Ramamurthy [25]. A systematic account using Boolean function theory based on the idea of ‘generalized Shannon decomposition’ is developed in our accompanying paper [4].

In this paper we are interested in the algorithmic complexity of the decomposition of Boolean functions given in DNF. After introducing some definitions and concepts in Section 2, we introduce in Section 3 the useful concept of ‘generalized Shannon decomposition’ and we argue that this concept can be used to simplify decomposition theory. In Section 4 we will show that the complexity of decomposition for general Boolean functions is coNP-complete. Decompositions of monotone Boolean functions, modular sets and the modular closure are discussed in Section 5. In Section 6 we discuss the computational aspects of decomposing positive functions and we prove that for a positive function \( f \) the recognition problem of the modularity of a set can be solved in time \( O(mn) \), where \( n \) is the number of variables of \( f \) and \( m \) is the number of prime implicants of \( f \). Moreover, we show that the modular closure of set can be computed in time \( O(mn^2) \). The last section contains the conclusions and topics for further research.

2. Definitions and notations

A Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) is called monotone (positive) on \( N = \{1, 2, \ldots, n\} \), if \( x \leq y \Rightarrow f(x) \leq f(y) \). A Boolean function \( f \) is constant if: \( f \equiv 0 \) (denoted by \( f = \bot \)) or \( f \equiv 1 \) (denoted by \( \top \)). A variable \( x_j \) of \( f \) is called essential if the restrictions respectively defined by: \( f(x_j = 0) = f(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n) \) and \( f(x_j = 1) = f(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) \), are not identical. The dual of a function \( f \) is defined
by: $f^d(x) = \bar{f}(\bar{x})$. Given a function $f$ in DNF, then the dual is obtained by interchanging $\land$ and $\lor$.

2.1. Disjunctive decompositions

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function and $A = \{1, 2, \ldots, n\}$. Identify each $i \in A$ with the variable $x_i$. Then $f$ is said to be a function defined on $A$. Furthermore, if $A = A_1 \cup A_2 \cup \cdots \cup A_n$ is a partition of $A (A_i \cap A_j = \emptyset, i \neq j)$, then we will denote this by $x_A = (x_{A_1}, \ldots, x_{A_n})$ and $f(x_A) = f(x_{A_1}, \ldots, x_{A_n})$. Let $F(y_A')$ and $g_i(x_{B_i})$ be Boolean functions defined on the mutually disjoint sets $A' = \{1, \ldots, m\}$ and $B_i, i \in A'$, and let $A = \bigcup_{j=1}^m B_j$. Then the Boolean function defined by

$$f(x_A) = F(g_1(x_{B_1}), \ldots, g_m(x_{B_m})), $$

is called the composition of the functions $f$ and $g_i, i \in A'$, obtained by substitution of the variables $y_i$ in $f$ by the functions $g_i, i \in A'$. This composition is denoted by $F[g_i, i \in A']$. A composition is called proper if $|A'| > 1$ and $|B_i| > 1$ for some $i \in A'$. A Boolean function is said to be decomposable if it has a representation as a proper composition. Otherwise, the function $f$ is called indecomposable or prime. If $F[g_i, i \in A']$ is a decomposition of the function $f$ then the partition $\pi = \{B_i, i \in A'\}$ is called a congruence partition and $f$ is called the quotient of $f$ modulo $\pi$ and is denoted by $f/\pi$. From the definition of decomposition it easily follows that

$$f = F[g_i, i \in A'] \Leftrightarrow f^d = F^d[g_i^d, i \in A'].$$

Therefore, we have $F = f/\pi \Leftrightarrow F^d = f^d/\pi$. Moreover, it is well-known that the functions $g_i, i \in A'$, are determined modulo complementation of the functions, and that the quotient $f$ is determined modulo complementation of the variables. The algebraic properties of congruence partitions are discussed in [22,21]. It is known that each decomposition of a Boolean function $f$ can be obtained by a series of the so-called simple disjunctive decompositions. These are decompositions of the form

$$f(x_A) = F(x_B, g(x_C)),$$

where $\pi = \{B, C\}$ is a partition of $A$.

**Definition 1.** Let $f$ be a Boolean function defined on $A$. Then $C \subseteq A$ is called a modular set of $f$ if $f$ has a simple disjunctive decomposition of the form $f(x_A) = F(x_B, g(x_C))$. The function $g$ is called a component of $f$.

3. Generalized Shannon decomposition

Let $f$ be a Boolean function on $A$. Then for all $j \in A$ the following decomposition holds:

$$f = \bar{x}_j f(x_j = 0) \lor x_j f(x_j = 1). $$

(1)
Eq. (1) is known as a Shannon decomposition of $f$. Now consider the simple disjunctive decomposition

$$f(x_A) = F(x_B, g(x_C)).$$

(2)

Then using Eq. (1) we have

$$f(x_A) = \bar{g}(x_C) F_0(x_B) \lor g(x_C) F_1(x_B),$$

(3)

where $F_0(x_B) = F(x_B, 0)$ and $F_1(x_B) = F(x_B, 1)$.

Conversely, let $g$ and $h_0, h_1$ be arbitrary Boolean functions defined respectively on $C$ and $B$ such that $f = \bar{g} h_0 \lor g h_1$, and let the function $f$ be defined by $F(x_B, y) := \tilde{y} h_0 \lor y h_1$. Then $f(x_A) = F(x_B, g(x_C))$ is a simple disjunctive decomposition of $f$, where $F_0(x_B) = h_0$ and $F_1(x_B) = h_1$. Therefore, we have proved the following fundamental lemma:

**Lemma 1.** Let $f$ be a Boolean function on $A$. Then $C \subseteq A$ is a modular set of $f$ iff there exists a Boolean function $g$ on $C$ and functions $h_0$ and $h_1$ on $B = A \setminus C$ such that $f = \bar{g} h_0 \lor g h_1$.

We call the decomposition in Eq. (3) a generalized Shannon decomposition associated with the simple disjunctive decomposition (2). If $C$ is a modular set of the function $f$ such that $C$ contains at least one essential variable of $f$, then it follows from the decomposition

$$f = \bar{g} h_0 \lor g h_1$$

(4)

that the function $g$ is non-degenerate and that the functions $h_0$ and $h_1$ are not identical. Therefore, there exists a binary vector $b_0$ such that either $g(x_C) = f(b_0, x_C)$ or $\bar{g} = f(b_0, x_C)$. This shows that we may assume that the function $g$ is a subfunction of $f$.

In general, Eq. (4) shows that if $b$ is a fixed vector then the function $f(b, x_C)$ is either degenerate or identical to $g$ of identical to $\bar{g}$. It is not difficult to see that the converse holds also. Therefore, the following theorem holds:

**Theorem 1.** Let $f$ be a Boolean function defined on $A$. If $C \subseteq A$ contains at least one essential variable of $f$, then the following statements are equivalent:

(a) $C$ is modular.
(b) There exists a vector $b_0$ such that the function $g(x_C) := f(b_0, x_C)$ is non-degenerate and for all fixed $b$ the function $f_b := f(b, x_C)$ is degenerate or identical to either $g$ or $\bar{g}$.

The following theorem is fundamental:

**Theorem 2.** Let $f$ be a general Boolean function. Suppose $A$ and $B$ are incomparable modular sets such that $A \cap B \neq \emptyset$. Then $A \tilde{B}$, $A \cap B$, $\tilde{A}B$ and $A \cup B$ are modular sets of $f$, and $f(x_{A \cup B}) = f(x_{\tilde{A}B}) \circ f(x_{A \cap B}) \circ f(x_{A \cup B})$, where $\circ$ is either $\land$, $\lor$ or $\oplus$. If $f$ is monotone, then $\circ$ is either $\land$ or $\lor$. 


Remark 1. Theorem 2 is a famous result called the *Three Modules Theorem* of Ashenhurst [2], reproved in game theory and reliability theory [25,4]. But as far as we know this result is due to Singer [30].

The ‘three modules theorem’ is proved in the literature by considering Ashenhurst decomposition charts, expansions of Boolean functions or differential calculus [1,2,16,18,17]. In our accompanying paper [4] we show that the theory on decomposition can be more easily developed by using the concept ‘generalized Shannon decomposition’ discussed in this section.

4. Complexity of decomposition for general Boolean functions

In this section we prove that for general Boolean functions the problem of recognizing modular sets (called MODULAR) is coNP-complete. In switching theory this complexity has not been discussed. In this context modular sets and decompositions are based on the evaluation of Ashenhurst decomposition charts or by using differential calculus [1,2,16,18,17]. However, here we will study the complexity of the recognition problem of Boolean functions given in DNF-form. In particular we will discuss the following problem:

**Problem MODULAR**
*Given:* A Boolean function $f$ in DNF defined on $V$ and a set $C \subseteq V$ that contains at least one essential variable of $f$.
*Question:* Is $C$ a modular set of $f$?

We relate this problem to the following recognition problem:

**Problem COMPLEMENT**
*Given:* Boolean functions $f$ and $g$ in DNF.
*Question:* $f = \overline{g}$?

It is easy to see that this problem is (polynomial) equivalent to the problem whether two functions $f$ and $g$ are mutually dual: $f = g^d$. It is well known that this problem is coNP-complete, see e.g. [5]. It can also be shown that problem COMPLEMENT remains coNP-complete if we assume that $f, g \notin \{\bot, \top\}$.

**Theorem 3.** Problem MODULAR is coNP-complete.

**Proof.** Suppose $g_1$ and $g_2$ are non-constant Boolean functions given in DNF on $A = \{x_1, \ldots, x_n\}$. Let $\{x, y\} \cap A = \emptyset$. Define the function $f$ on $A \cup \{x, y\}$ as

$$f = xg_1 \lor yg_2. \quad (5)$$

If $g_2 = \overline{g}_1$, then according to Lemma 1 $A$ is a modular set of $f$. Moreover, since $g_1, g_2 \notin \{\bot, \top\}$, $A$ also contains at least one essential variable. Conversely, suppose $A$ is modular
and \( A \) contains essential variables of \( f \). Then there exists a pair of binary values \((x_0, y_0)\) such that the function \( g \) defined by \( g = f(x_0, y_0, x_A) \) is non-trivial. Furthermore, according to Theorem 1 for all fixed \( x \) and \( y \) the function \( h(x_A) = f(x, y, x_A) \) is constant or identical to the function \( g \) or its complement. From Eq. (5) it follows that \( h \in \{ \perp, g_2, g_1, g_1 \lor g_2 \} \). Therefore, we have \( g_2 = \bar{g}_1 \). Conclusion: \( g_2 = \bar{g}_1 \iff A \) is modular. This shows that the problem MODULAR is coNP-hard. To prove that this problem is in coNP we note that according to Theorem 1 \( A \) is a modular set of \( f \) iff for all binary vectors \( b \) the function \( f_b := f(x_A, b) \in \{ \top, \perp, g, \bar{g} \} \), where \( g \) is a component of \( f \) on \( A \). Therefore, the set \( A \) is not modular iff there exist binary vectors \( b_1 \) and \( b_2 \) such that \( f_{b_1}, f_{b_2} \not\in \{ \top, \perp \} \) and \( f_{b_1} \neq f_{b_2} \). Equivalently, the set \( A \) is not modular iff there exist three different binary vectors \( a, a_1, a_2 \), and two different vectors \( b_1, b_2 \) such that \( f_{b_1}(a) = f_{b_2}(a) \neq f_{b_1}(a_1) = f_{b_2}(a_2) \). This shows that problem MODULAR is in coNP. \( \square \)

5. Decomposition of monotone Boolean functions

Let \( f \) be a positive function defined on \( N \). Then a subset \( A \subseteq N \) will be represented frequently by its characteristic vector \( a := char(A) \in \{0, 1\}^n \), with \( n = |N| \). If \( A = \emptyset \) then this will be denoted by \( a = 0 \), where \( 0 \) is the all-zero vector. If \( A \subseteq N \), then the functions \( f(a=0) \) and \( f(a=1) \) are the restrictions of \( f \) defined on the set \( A \) by setting all variables in \( A \) to 0 respectively 1. Similarly, the function \( f(\bar{a} = 1) \) is the restriction of \( f \) to \( A \) defined by setting all variables in \( A \) to 1, see Example 2. However, where needed we will consider all these restrictions of \( f \) as functions defined on \( N \) by adding dummy (non-essential) variables. Furthermore, the set of all essential variables of \( f \) is called the support set of \( f \). This set is denoted by \( S(f) \), and the vector \( char(S(f)) \) is denoted by \( \sigma(f) \). As known a positive Boolean function has a unique irredundant DNF consisting of all prime implicants. The set of prime implicants correspond to the set of minimal true vectors of \( f \), denoted by \( \min T(f) \). It is well known that \( \min T(f^d) \) represents the set of minimal transversals of \( \min T(f) \). The complement of a false vector is a transversal: \( f(x) = 0 \iff f^d(\bar{x}) = 1 \). If \( v, w \in \{0, 1\}^n \), then \( v \lor w \) (also denoted by \( vw \) ), and \( v \lor w \) denote respectively the vectors obtained by applying component-wise the and-operation and the or-operation to the vectors \( v \) and \( w \). Finally, we will denote the variables of a positive function by their index and + denotes the \( \lor \)-operation.

**Example 1.** Let \( f \) be the function defined by \( f(x) = x_1x_2 \lor x_2x_3 \). Then: \( f \) is denoted as: \( f = 12 + 23 \). Furthermore, \( f^d = (1 + 2)(2 + 3) = 2 + 13 \), \( \min T(f) = \{110, 011\} \), and \( \min T(f^d) = \{010, 101\} \) is the set of the minimal transversals of \( \min T(f) \). Moreover, \( 001 \) is a false vector of \( f \) and its complement \( 110 \) is a transversal of \( \min T(f) \).

5.1. Modular sets

**Definition 2.** Let \( f \) be a positive function defined on \( N \) and let \( A \subseteq N \). If \( f \) depends on \( A \) (i.e. \( \sigma(f) \land a \neq \emptyset \) ), then the positive function \( f^a \) on \( A \) is defined by: \( \min T(f^a) = \{v \mid v \in \min T(f), v \land a \neq \emptyset\} \), where \( a = char(A) \). Otherwise \( f^a := \perp \).
From this definition it follows that every positive Boolean function \( f \) can be decomposed as

\[
f = f(a = 0) \lor f^a,
\]

where \( A \subseteq N \).

Furthermore, for a monotone Boolean function \( f \) Shannon’s decomposition has the form

\[
f(x) = f(x_j = 0) \lor x_j f(x_j = 1).
\]

**Definition 3.** Let \( f \) be a positive function defined on \( N \), and \( A \subseteq N \). Then the contraction \( f_a \) of \( f \) on \( N \) is defined by

\[
f_a(x_A) = f_a(\bar{a} = 1)(x_A),
\]

where \( a = \text{char}(A) \).

**Example 2.** Let \( f \) be the positive function on \( \{1, 2, \ldots, 6\} \) defined by:

\[
f = 1245 + 126 + 2345 + 236 + 46
\]

and let \( A = \{1, 2, 3\} \). Then \( a = \text{char}(A) = 111000 \), \( f(a = 0) = 46 \), \( f^a = 1245 + 126 + 2345 + 236 \), and \( f_a = 12 + 23 \).

The following theorem proved in our paper [4], shows that if \( f \) is a positive function and if \( A \) is a modular set of \( f \), then the component \( g(x_A) \) of \( f \) is just the contraction of \( f \) on \( A \).

**Theorem 4.** Let \( f \) be a positive Boolean function defined on \( N \) and let \( A \subseteq N \). Then \( A \) is modular iff \( f^a = f^a(a = 1) f_a \).

**Remark 2.** Note, that according to [25] the problem of deciding whether a set \( A \) is modular or not can be solved in time \( O(m^2 n^2) \). The preceding theorem shows already that this problem can be easily solved by checking the equation \( f^a = f^a(a = 1) f_a \! \)!

**Example 3.** Consider the function \( f \) of Example 2, and let \( A = \{1, 2, 3\} \). Then:

\[
f^a = f^a(a = 1) f_a = (45 + 6)(12 + 23).
\]

The following characterizations of a modular set are well known (cf. [25,4]):

**Theorem 5.** Suppose that \( f \) is a positive function defined on \( N \), and \( A \subseteq N \). Furthermore, let \( \sigma(f) \land a \neq 0 \), where \( a = \text{char}(A) \). Then the following assertions are equivalent:

(a) \( A \) is a modular set of \( f \).
(b) \( A \) is a modular set of \( f^a \).
(c) \( \forall v, w \in \min T(f^a) : f(va \lor w\bar{a}) = 1 \).
(d) \( \min T(f^a) = \{va \lor w\bar{a} | v, w \in \min T(f^a)\} \).

5.2. The modular closure

Unless stated otherwise we assume that a positive function \( f \) depends on all its variables. A central step in the determination of the modular tree of a positive function is the computation
of the modular closure of a set. Since a non-empty intersection of two modular sets of a Boolean function is again modular, each subset $A$ of variables is contained in a smallest modular set called the modular closure of $A$.

**Definition 4 (Billera [3]).** Let $f$ be a Boolean function defined on $N$. The closure of $A \subseteq N$ is defined by $Cl(f)(A) = \cap\{B \mid A \subseteq B, \ B \text{ is a modular set of } f\}$.

**Definition 5.** Suppose $\exists u, v \in \min T(f^a)$ such that $f(ua \lor v\bar{a}) = 0$. Then we call the vector $ua \lor v\bar{a}$ a culprit of $f$ with respect to $a$.

In [4] we proved the following fundamental theorem which is a variation of a theorem in [25]:

**Theorem 6.** Let $f$ be a positive function. Suppose $t$ is the complement of a culprit of $f$ with respect to $a$. Then $U = \{u \in \min T(f^a) \mid uta = 0\} \neq \emptyset$. Furthermore, if $u_0 \in \arg\min_{u \in U} |ut|$, then $0 \neq u_0t = u_0t\bar{a} \leq Cl_f(a)$.

The vector $u_0t$ can be determined in $O(mn)$ time. Therefore, we have the following corollary:

**Corollary 1.** If a culprit is known, then an element in $Cl_f(A) \setminus A$ can be determined in time $O(mn)$.

### 6. Computational aspects

We have already seen that the recognition problem MODULAR for general Boolean functions is coNP-complete. For positive Boolean functions the situation is quite different. Various decomposition algorithms (in different contexts) are known. Therefore, we briefly discuss the computational aspects of the decomposition of positive Boolean functions. A unified treatment of all algorithms (up to 1990) related to modular sets known in game theory, reliability theory and set systems (clutters) is given by Ramamurthy [25,4].

Let $f$ be a positive function defined on the set $N$, where $|N| = n$, and let $m$ be the number of prime implicants of $f$. Then according to Möhring and Radermacher [22] the modular tree can be computed in time $O(n^3T(m, n))$, where $T(m, n)$ is the complexity of computing the modular closure of a set $A \subseteq N$. The first known algorithm to compute the modular closure due to Billera [3] is based on computing the dual of $f$. Although this problem is NP-hard in general, for positive functions the complexity of the dualization problem is still not known, although this problem is unlikely to be NP-hard, see e.g [5]. An improvement of Billera’s algorithm by Ramamurthy and Parthasarathy [26] also based on dualization has a similar complexity. The first polynomial algorithm given by Möhring and Radermacher [22] reduced the complexity to $T(m, n) = O(m^3n^4)$. Subsequently, the complexity was further reduced by Ramamurthy and Parthasarathy [26] and Ramamurthy [25] to respectively $T(m, n) = O(m^3n^2)$ and $T(m, n) = O(m^2n^2)$. It is easy to see that the
determination of the modular closure can be solved by solving $O(n)$ times the following problem:

**Problem PMODULAR**

*Input*: A Boolean function $f$ with $m$ prime implicants defined on $N$, where $|N| = n$ and $A \subseteq N$.

*Output*: “$A$ is modular” if $A$ is modular. An element $x \in \text{Closure}(A) \setminus A$ otherwise.

In the next subsection we show that the search problem PMODULAR can be solved in time: $O(mn)$. Therefore, the modular closure of a set can be determined in time $T(m, n) = O(mn^2)$.

6.1. Solving PMODULAR in time $O(mn)$

Before we solve problem PMODULAR we first show that for positive functions the recognition problem whether a set $A$ is modular or not can be solved in time $O(mn)$.

6.1.1. Recognition of modular sets

Let $f$ be positive Boolean function $f$ on $N$, $\emptyset \neq A \subseteq N$, and $a = \text{char}(A)$. Then we denote $M = \min T(f^a) = \{v_1, \ldots, v_m\}$, $S = \{va \mid v \in M\}$, $T = \{v\bar{a} \mid v \in M\}$, $p = |S|$ and $q = |T|$. Furthermore, without loss of generality we may assume that $M \neq \emptyset$ and that $\forall v \in M = \min T(f^a)$ we have $v \not\leq a$. For each $v \in M$ we can write $v = va \lor v\bar{a}$ as a $2n$-vector: $(va \mid v\bar{a})$. Note, that by assumption both vectors $va$ and $v\bar{a}$ are non-zero. We now consider the list of all (column-)vectors:

\[
\begin{bmatrix}
v_1a & v_2a & \cdots & v_ma \\
v_1\bar{a} & v_2\bar{a} & \cdots & v_m\bar{a}
\end{bmatrix}
\]

According to [28], the set of all these $2n$-vectors can be lexicographically sorted in time $O(mn)$.

**Example 4.** Let $f = 15 + 16 + 245 + 35 + 36 + 46$, and $A = \{1, 2, 3, 4\}$. Then $f^a = f$ and the sorted list is given by

\[
\mathcal{E} = \begin{bmatrix}
1 & 1 & 24 & 3 & 3 & 4 \\
5 & 6 & 5 & 6 & 6
\end{bmatrix}
\]

Note here, that the $2n$-vector $(va \mid v\bar{a})$ is denoted by a pair of subsets, e.g. the third column-vector $(010100 \mid 000010)$ is denoted by $(24 \mid 5)$.

**Theorem 7.** $A$ is modular iff the sorted list of all $2n$-vectors has the following structure:

\[
\mathcal{E} = \begin{bmatrix}
s_1 \cdots s_1 & s_2 \cdots s_2 & \cdots & s_p \cdots s_p \\
t_1 \cdots t_q & t_1 \cdots t_q & \cdots & t_1 \cdots t_q
\end{bmatrix},
\]

where $s_i \in S$ and $t_j \in T$, and we have: $S = \min T(f^a)$ and $T = \min T(f^a(a = 1))$. So if $A$ is modular, then the list $\mathcal{E}$ consists of $p$ segments of length $q$, and $m = pq$. 
Therefore, \( A \in \mu(f) \iff f^a = f_a f(a=1) \Rightarrow S = \min T(f_a) \) and \( T = \min T(f^a(a=1)) \). Furthermore, if \( v_1, v_2, w_1, w_2 \in \min T(f^a) \), then \( v_1a \lor w_1\bar{a} = v_2a \lor w_2\bar{a} \iff v_1a = v_2a \) and \( w_1\bar{a} = w_2\bar{a} \). \( \square \)

**Example 5.** Let \( f \) be the function of Example 2, and let \( A = \{1, 2, 3\} \). Then we have \( f^a = 126 + 236 + 1245 + 2345 \), and the sorted list is given by

\[
\mathcal{S} = \begin{pmatrix} 12 & 12 & 23 & 23 \\ 45 & 6 & 45 & 6 \end{pmatrix}.
\]

Therefore, \( A \) is a modular set of \( f \) and \( p = q = 2 \). Similarly, it can be checked that \( \{1, 3\} \) is a modular set of \( f \).

It is easy to see that the structure \( \mathcal{S} \) can be identified in time \( O(mn) \), by scanning the list \( \mathcal{S} \) from left to right. Therefore, it can be determined in time \( O(mn) \) whether a set \( A \) is modular or not. However, the more difficult part is to detect an element \( x \in \text{Closure}(A) \) in time \( O(mn) \) if \( A \) is not modular. According to Theorem 4 this can be done in time \( O(mn) \) if we can find a culprit in time \( O(mn) \).

### 6.1.2. Finding a culprit in time \( O(mn) \)

Recall that the vector \( v_a \lor w\bar{a} \), with \( v, w \in \min T(f^a) \) is called a culprit with respect to \( A \) if \( f(v_a \lor w\bar{a}) = 0 \). Note, that in order to find a culprit we have to scan and compare the rows of the structure \( \mathcal{S} \) separately. This is actually the reason why the algorithm of Ramamurthy [25] finds a culprit in time \( O(m^2n) \). However, it can be shown that the next basic lemma can be used several times in order to find a culprit if it exists in time \( O(mn) \). In this lemma the following notations are used: \( v \sim w \iff (v < w \text{ or } v > w) \), and \( v \asymp w \iff (v \leq w \text{ or } v \geq w) \).

**Lemma 2.** Let \( (s_1|t_1) \) and \( (s_2|t_2) \) denote any two different columns of the list \( \mathcal{S} \). Then:

(a) \( s_1 = s_2 \implies t_1 \not\asymp t_2 \).

(b) \( t_1 = t_2 \implies s_1 \not\asymp s_2 \).

(c) If \( s_1 \sim s_2 \) then either \( s_1 \lor t_2 \) or \( s_2 \lor t_1 \) is a culprit.

(d) If \( t_1 \sim t_2 \), then either \( s_1 \lor t_2 \) or \( s_2 \lor t_1 \) is a culprit.

(e) If the \( 2n \)-vector \((s_1|t_2)\) does not occur in the list \( \mathcal{S} \) and \( s_1 \) and \( t_2 \) are minimal, then \( s_1 \lor t_2 \) is a culprit.

**Proof.** Let \( v \) and \( w \) be minimal vectors of \( f^a \) such that \( s_1 = v_a, s_2 = w_a, t_1 = v\bar{a} \) and \( t_2 = w\bar{a} \).

(a) Suppose \( s_1 = s_2 \) so \( v_a = w_a \). Then obviously \( v\bar{a} \neq w\bar{w} \), otherwise we would have \( v = w \). Therefore, \( t_1 \neq t_2 \). Now assume that \( t_1 \sim t_2 \), e.g. \( v\bar{a} < w\bar{a} \). Then \( v = v_a \lor v\bar{a} < w_a \lor w\bar{a} = w \), contrary to our assumption that \( N \) and \( w \) are minimal vectors of \( f^a \).

(b) This is proved similar to (a).
(c) Suppose \( s_1 \sim s_2 \), e.g. \( va > wa \). Then \( v = va \vee v\bar{a} > wa \vee v\bar{a} \). Since \( v \) is a minimal vector of \( f^a \), the vector \( wa \vee v\bar{a} \) is a culprit: \( f(wa \vee v\bar{a}) = 0 \), see Theorem (5.f).

(d) This assertion is proved similar to (c).

(e) Suppose that the vector \( va \vee w\bar{a} \) is not a culprit. Then \( f(va \vee w\bar{a}) = 1 \). Hence, there exists a vector \( u \in \min T(f^a) \) such that \( u \leq va \vee w\bar{a} \). This implies \( ua \leq va \) and \( u\bar{a} \leq w\bar{a} \).

Since by assumption \( Na \) and \( w\bar{a} \) are minimal, we have \( ua = va \) and \( u\bar{a} = w\bar{a} \). Therefore, the vector \( (va|w\bar{a}) = (ua|u\bar{a}) \) is a column-vector of \( \mathcal{F} \), contrary to our assumption. So the vector \( va \vee w\bar{a} \) is a culprit. \( \square \)

Suppose that \((s_1|t_2)\) does not occur in the list \( \mathcal{F} \). Then we can check in \( O(mn) \) time whether the elements \( s_1 \) and \( t_2 \) are minimal. If both elements are minimal then we can apply assertion (e) of Lemma 2. Otherwise, we can apply either (c) or (d). Therefore, we have the following corollary:

**Corollary 2.** If \((s_1|t_2)\) does not occur in the list \( \mathcal{F} \), then a culprit can be found in time \( O(mn) \).

**Example 6.** Consider the sorted list in Example 4:

\[
\mathcal{F} = \begin{bmatrix}
1 & 1 & 24 & 3 & 3 & 4 \\
5 & 6 & 5 & 5 & 6 & 6 \\
\end{bmatrix}.
\]

Then the first segment has length \( q = 2 \). Since the first element of the fourth column is not equal to 24 we detect that the column \((24|6)\) is not in \( \mathcal{F} \). However, 246 \((= 010101)\) is not a culprit, because the element 24 is not minimal. By scanning the first row we discover that 4 is comparable with 24. Hence, by Lemma 2(c) applied to the third and last column, either 246 or 45 is not a true vector of \( f^a \). In this case 45 \((= 000110)\) is a culprit, because \((4|5)\) is not in \( \mathcal{F} \) (see Lemma 2(a)) and the elements 4 and 5 are minimal.

In [4] we have shown that Lemma 4 can be used to prove our final result:

**Theorem 8.** Problem PMODULAR is solvable in time \( O(mn) \).

7. Conclusions and future research

For monotone Boolean functions the recognition of modular sets and therefore the computation of the modular closure and the modular tree can be reduced with a factor \( O(m) \). On the other hand, we have proved that for general Boolean functions the recognition problem is coNP-complete. We also argued that the generalized Shannon representation of a disjunctive decomposition is an effective tool to study decompositions of Boolean functions [4]. Compared with the set theoretic approach used in the literature it appears that the Boolean function approach is more transparent. Since partially defined Boolean functions [11,10,19] play an important role in many data mining tasks and in switching theory we consider decomposition theory in data mining also as an important task for further research. Finally decompositions with components restricted to a certain class, e.g. self-dual functions [6]
(committees in game theory), matroids [15], regular functions etc. are an interesting topic for future research.

Acknowledgements

I would like to thank Peter Hammer, Endre Boros and Yves Crama for some stimulating discussions and pointers to the literature. We also acknowledge the referees for the valuable remarks that improved this paper.

References


