MINORS OF FINITE OPERATIONS

SLAVCHO SHTRAKOV

ABSTRACT. In this paper we present a new class of complexity measures, induced by a new data structure for representing k-valued functions (operations), called minor decision diagram. The results are presented in terms of Multi-Valued Logic circuits (MVL-circuits), ordered decision diagrams, formulas and minor decomposition trees. We prove that, if a function f has non-trivial arity gap ($gap(f) \geqslant 2$), then all sets of essential variables in f are separable. We define equivalence relations which classify the functions of k-valued logic into classes with the same minor complexities.

A *logic gate* is a physical device that realizes a Boolean function. A *logic circuit* is a directed acyclic graph in which all vertices except input vertices carry the labels of gates. When realizing functions are taken from the k-valued logic the circuit is called the (k, n)*-circuit or Multi-Valued Logic circuit (MVL-circuit).*

To move from logical circuits to MVL-circuits, researchers attempt to adapt CMOS (complementary metal oxide semiconductor), I^2L (integrated injection logic) and ECL (emitter-coupled logic) technologies to implement the many-valued and fuzzy logics gates. The MVL-circuits offer more potential opportunities for the improvement of present VLSI circuit designs. For instance, MVL-circuits are well-applied in memory technology as flash memory, dynamic RAM, and in algebraic circuits [5].

1. Subfunctions and minors of functions

Let k be a natural number with $k \geqslant 2$. Let Z_k denote the set $Z_k = \{0, 1, \ldots, k-1\}$. The operations addition " $\oplus$ " and product "." modulo k constitute Z_k as a ring. An *n-ary k-valued function (operation) on Z_k* is a mapping $f : Z_k^n \to Z_k$ for some natural number n, called *the arity* of f. P_k^n denotes the set of all n-ary k-valued functions and $P_k = \bigcup_{n=1}^{\infty} P_k^n$ is called *the algebra of k-valued logic.*

For a given variable x and $\alpha \in Z_k$, x^α is defined as follows:

$$x^\alpha = \begin{cases} 1 & if \quad x = \alpha \\ 0 & if \quad x \neq \alpha. \end{cases}$$

Let $f \in P_k^n$ and let $var(f) = \{x_1, \ldots, x_n\}$ be the set of all variables, which occur in f. We say that the i-th variable $x_i \in var(f)$ is *essential* in f, or f *essentially depends* on x_i, if there exist values $a_1, \ldots, a_n, b \in Z_k$, such that

$$f(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n).$$

The set of all essential variables in the function f is denoted $Ess(f)$ and $ess(f) = |Ess(f)|$. The variables from $var(f)$ which are not essential in $f \in P_k^n$ are called *inessential* or *fictive*.

2010 *Mathematics Subject Classification.* Primary: 94C10; Secondary: 06E30, 68Q25, 68Q15.
Key words and phrases. identification minor, minor decision diagram, minor complexity.

Let x_i be an essential variable in f and let c be a constant from Z_k. The function $g = f(x_i = c)$ obtained from $f \in P_k^n$ by assigning the constant c to the variable x_i is called a *simple subfunction of* f (sometimes termed a *cofactor* or a *restriction*). When g is a simple subfunction of f we write $f \succ g$. The transitive closure of $\succ$ is denoted $\succeq$. $Sub(f) = \{g \mid f \succeq g\}$ is the set of all subfunctions of f and $sub(f) = |Sub(f)|$.

We say that each subfunction g of f is a reduction to f via the *subfunction relationship*.

A non-empty set M of essential variables in the function f is called *separable* in f if there exists a subfunction g, $f \succeq g$ such that $M = Ess(g)$. $Sep(f)$ denotes the set of all the separable sets in f and $sep(f) = |Sep(f)|$.

Let x_i and x_j be two distinct essential variables in f. The function h is obtained from $f \in P_k^n$ by *identifying (collapsing) the variables* x_i *and* x_j, if

$$h(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) = f(a_1, \ldots, a_{i-1}, a_j, a_{i+1}, \ldots, a_n),$$

for all $(a_1, \ldots, a_n) \in Z_k^n$.

Briefly, when h is obtained from f, by identifying the variable x_i with x_j, we write $h = f_{i\leftarrow j}$ and h is called *a simple identification minor of* f [4]. Clearly, $ess(f_{i\leftarrow j}) < ess(f)$, because $x_i \notin Ess(f_{i\leftarrow j})$, but it has to be essential in f. When h is a simple identification minor of f we write $f \rhd h$. The transitive closure of $\rhd$ is denoted $\trianglerighteq$. $Mnr(f) = \{h \mid f \trianglerighteq h\}$ is the set of all distinct minors of f and $mnr(f) = |Mnr(f)|$. Let h, $f \trianglerighteq h$ be an identification minor of f. The natural number $r = ess(f) - ess(h)$, $r \geqslant 1$ is called the *order* of the minor h of f.

We say that each minor h of f is a reduction to f via the *minor relationship*.

Let $Mnr_m(f)$ denote the set $Mnr_m(f) = \{g \mid g \in Mnr(f) \ \& \ ess(g) = m\}$ and let $mnr_m(f) = |Mnr_m(f)|$, for all m, $m \leqslant n - 1$.

Let $f \in P_k^n$ be an n-ary k-valued function. The *essential arity gap* (shortly *arity gap* or *gap*) of f is defined as follows

$$gap(f) = ess(f) - \max_{h \in Mnr(f)} ess(h).$$

Let $2 \leqslant p \leqslant m$. We let $G_{p,k}^m$ denote the set of all k-valued functions which essentially depend on m variables whose arity gap is equal to p, i.e.

$$G_{p,k}^m = \{f \in P_k^n \mid ess(f) = m \ \& \ gap(f) = p\}.$$

We say that the arity gap of f is *non-trivial* if $gap(f) \geqslant 2$. It is natural to expect that the functions with "huge" gap, have to be more simple for realization by MVL-circuits and functional schemas when computing by identifying variables. For more results about the arity gap we refer [3, 4, 9, 11, 13, 14, 17].

Definition 1.1. Two functions g and h are called *equivalent (non-distinct as mappings)* (written $g \equiv h$) if g can be obtained from h by permutation of variables, introduction or deletion of inessential variables.

Many computations, constructions, processes, translations, mappings and so on, can be modeled as stepwise transformations of objects known as reduction systems. *Abstract Reduction Systems (ARS)* play an important role in various areas such as abstract data type specification, functional programming, automated deductions, etc. [7, 15] The concepts and properties of ARS also apply to other rewrite systems such as string rewrite systems (Thue systems), tree rewrite systems, graph grammars, etc. For more detailed facts about ARS we refer to J. W. Klop and Roel de Vrijer [7]. An ARS in P_k^n is a structure $W = \langle P_k^n, \{\rightarrow_i\}_{i\in I}, \rangle$, where $\{\rightarrow_i\}_{i\in I}$ is a family of binary relations on P_k^n, called *reductions or rewrite relations.* For a reduction $\rightarrow_i$ the transitive and reflexive closure is denoted $\twoheadrightarrow_i$. A function $g \in P_k^n$ is a *normal form* if there is no $h \in P_k^n$ such that $g \rightarrow_i h$. In all different branches of rewriting two basic concepts occur, known as termination (guaranteeing the existence of normal forms) and confluence (securing the uniqueness of normal forms).

A reduction $\rightarrow_i$ has the *unique normal form property* (UN) if whenever $t, r \in P_k^n$ are normal forms obtained by applying the reductions $\rightarrow_i$ on a function $f \in P_k^n$ then t and r are equivalent (non-distinct as mappings).

The computations on functions proposed in the present paper can be regarded as an ARS, namely: $W = \langle P_k^n, \{\succ, \rhd\}\rangle$. Next, we show that $\rhd$ completes the reduction process with unique normal form, whereas $\succ$ has not unique normal form property.

A reduction $\rightarrow$ is *terminating* (or *strongly normalizing* SN) if every reduction sequence $f \rightarrow f_1 \rightarrow f_2 \dots$ eventually must terminate. A reduction $\rightarrow$ is *weakly confluent* (or *has weakly Church-Rosser property* WCR) if $f \rightarrow r$ and $f \rightarrow v$ imply that there is $w \in P_k^n$ such that $r \twoheadrightarrow w$ and $v \twoheadrightarrow w$.

Theorem 1.2.

(i) *The reduction* $\rhd$ *is UN;*
(ii) *The reduction* $\succ$ *is not WCR, but it is SN.*

Lemma 1.3. *Let* $N \in Sep(f)$. *If there exist* m *constants* $c_1, \dots, c_m \in Z_k$ *such that* $N \cap Ess(g_i) = \emptyset$ *where* $g_i = f(x_i = c_i)$ *for* $1 \leqslant i \leqslant m$ *then* $M \cup N \in Sep(f)$ *for all* $M \neq \emptyset$, $M \subseteq \{x_1, \dots, x_m\}$.

Corollary 1.4. *Let* x_i *and* x_j *be two distinct essential variables in* f. *If there is a constant* c, $c \in Z_k$ *such that* $f(x_i = c)$ *does not essentially depend on* x_j *then* $\{x_i, x_j\} \in Sep(f)$.

Next, we turn our attention to relationship between essential arity gap and separable sets in functions.

Theorem 1.5. *Let* $f \in P_k^n$. *If* $gap(f) \geqslant 2$ *then all non-empty sets of essential variables are separable in* f.

2. Minor decision diagrams of functions

Intuitively, it seems that a function f has high complexity if all its sets of essential variables are separable, because the variables from separable sets remain essential after assigning constants to other variables (see [12]). For example, when assigning Boolean constants to some variables of a Boolean function, then a natural complexity measure is the size of its Binary Decision Diagrams (BDDs), which also depend on the variable ordering (see [1, 2]). Each path from the root (function node) to a terminal node (leaf) of BDD is called an *implementation* of f. In [12] we count the subfunction complexities $imp(f)$, $sub(f)$ and $sep(f)$ of all implementations obtained under all $n!$ variable orderings, subfunctions, and separable sets of n-ary Boolean functions for n, $n \leqslant 5$.

Roughly spoken, the complexity of functions, is a mapping (evaluation) $Val : P_k^n \rightarrow \mathbb{N}$ with $Val(x) = c$ for all $x \in X$ and for some natural number $c \in \mathbb{N}$, called the *initial value* of the complexity, and $Val(f) \geqslant c$ for all $f \in P_k^n$.

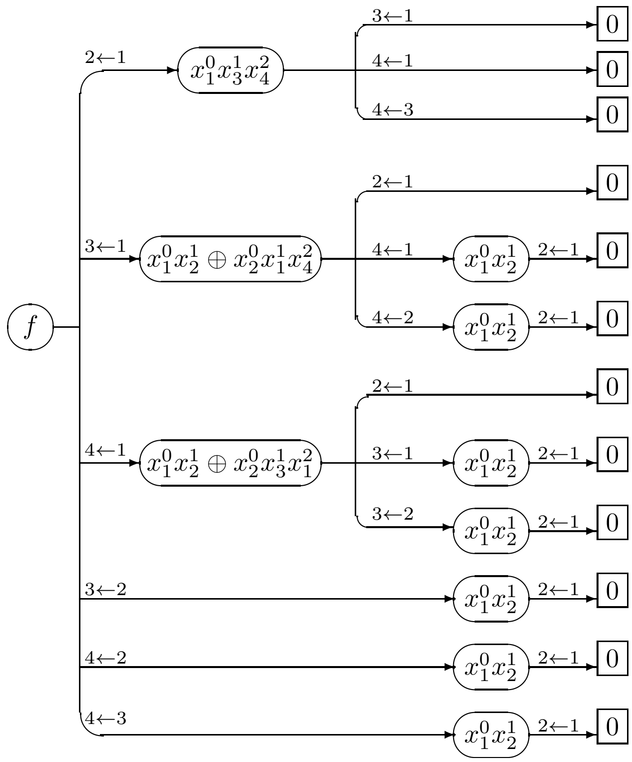


FIGURE 1. Minor decomposition tree of $f = x_1^0x_2^1 \oplus x_2^0x_3^1x_4^2$ $(mod\ 3)$.

The concept of complexity of functions is based on the "difficulties" when computing several resulting objects as subfunctions, implementations, separable sets, values, superpositions, etc.

As mentioned, we have used the computational complexities $sub(f)$, $imp(f)$ and $sep(f)$ in [12] to classify the functions from the algebra P_k^n. These complexities are invariants under the action of the groups SB_k^n, IM_k^n and SP_k^n.

Figure 1 shows the minor decomposition tree, constructed for the function $f = x_1^0x_2^1 \oplus x_2^0x_3^1x_4^2$ $(mod\ 3)$, which essentially depends on all of its four variables x_1, x_2, x_3 and x_4.

Let f be a k-valued function. The *minor decision diagram (MDD)* of f is obtained from the corresponding MDT by *reductions* of its nodes and edges applying of the following rules, starting from the MDT and continuing until neither rule can be applied:

Reduction rules

- If two edges have equivalent (as mappings) labels of their nodes they are merged.
- If two nodes have equivalent labels, they are merged.

Each edge $e = (v_1, v_2)$ in the diagram is supplied with a label $l(v_1, v_2)$, (written as bold in Figure 2 A), which presents the number of the merged edges of the MDT, connecting the nodes v_1 and v_2 in MDT. If two nodes in MDT are connected with unique edge then this edge is presented in MDD without label, for brevity.

For example, such pairs in Figure 1 are $(f, f_{2\leftarrow 1})$, $(f, f_{3\leftarrow 1})$, $(f, f_{4\leftarrow 1})$, $(f_{3\leftarrow 1}, 0)$, $(f_{3\leftarrow 2}, 0)$ and $(f_{4\leftarrow 1}, 0)$.

So, the MDD of f is an acyclic directed graph, with unique function node and according to Theorem 1.2, with unique terminal node. Clearly, the MDD and MDT are uniquely determined by the function f.

Example 2.1. Let us build the MDDs of the following two functions $f = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^2 \ (mod\ 3)$ and $g = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^1 \ (mod\ 3)$, using the reduction rules.

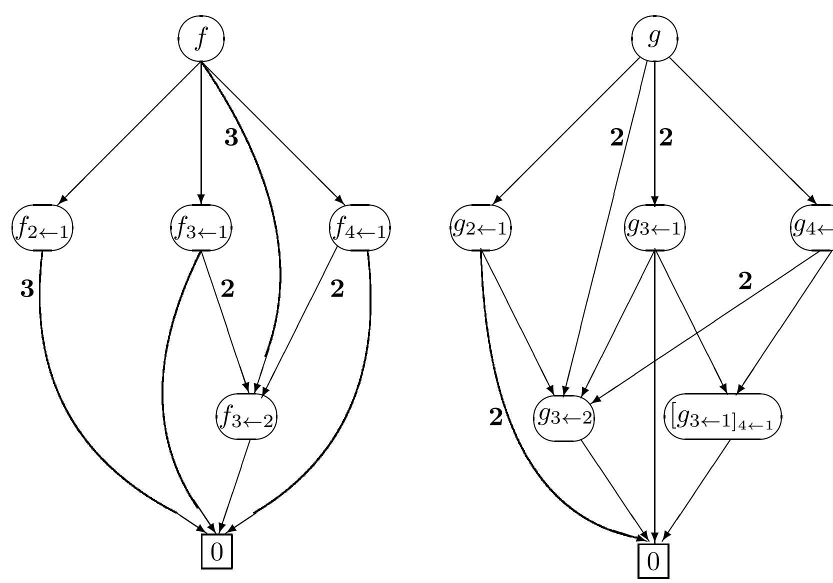
Figure 2 A) shows the MDD of the function f, obtained from its MDT, given in Figure 1, after applying the reduction rules. Figure 2 B) presents the MDD of g. The identification minors of f and g are:

$f_{2\leftarrow 1} = x_1^0 x_3^1 x_4^2 \ (mod\ 3)$, $f_{3\leftarrow 1} = x_1^0 x_2^1 \oplus x_2^0 x_1^1 x_4^2 \ (mod\ 3)$,
$f_{4\leftarrow 1} = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_1^2 \ (mod\ 3)$, $f_{3\leftarrow 2} = x_1^0 x_2^1 \ (mod\ 3)$,

$g_{2\leftarrow 1} = x_1^0 x_3^1 x_4^1 \ (mod\ 3)$, $g_{3\leftarrow 1} = x_1^0 x_2^1 \oplus x_2^0 x_1^1 x_4^1 \ (mod\ 3)$,
$g_{4\leftarrow 1} = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_1^1 \ (mod\ 3)$, $g_{4\leftarrow 3} = x_1^0 x_2^1 \oplus x_2^0 x_3^1 \ (mod\ 3)$,
$[g_{2\leftarrow 1}]_{4\leftarrow 3} = x_1^0 x_3^1 \ (mod\ 3)$, $[g_{3\leftarrow 1}]_{4\leftarrow 1} = x_1^0 x_2^1 \oplus x_2^0 x_1^1 \ (mod\ 3)$,
$g_{3\leftarrow 2} = x_1^0 x_2^1 \ (mod\ 3)$.

Clearly, $f_{3\leftarrow 2} = f_{4\leftarrow 2} = f_{4\leftarrow 3} = [f_{4\leftarrow 1}]_{3\leftarrow 1} = [f_{4\leftarrow 1}]_{3\leftarrow 2} = [f_{3\leftarrow 1}]_{4\leftarrow 1} = [f_{3\leftarrow 1}]_{4\leftarrow 2} = x_1^0 x_2^1 \ (mod\ 3)$. The minors $f_{i\leftarrow 1}$ for $i = 2, 3, 4$ are of order 1, and the last minor $f_{3\leftarrow 2}$ is of order 2.

The label of the edge $(f, f_{3\leftarrow 2})$ is **3** because there are three identification minors, namely $f_{3\leftarrow 2}$, $f_{4\leftarrow 2}$ and $f_{4\leftarrow 3}$ of f which are equivalent to $f_{3\leftarrow 2}$ (see the last three branches of the MDT in Figure 1).



A) $f = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^2 \ (mod\ 3)$ B) $g = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^1 \ (mod\ 3)$

Figure 2. Minor decision diagrams

The size of the MDD and the minor complexities are determined by the function, being represented. The "scalability" of the diagram is an important measure of the computational complexity of the function. We are going to formalize this problem and establish a method for classification of functions by the minor complexities. Our goal is to classify functions in finite algebras by these complexities.

Definition 2.2. Let $f \in P_k^n$ be a k-valued function. Its *cmr-complexity* $cmr(f)$ is defined as follows:

(i) $cmr(f) = 1$ if $ess(f) \leqslant 1$;
(ii) $cmr(f) = 2$ if $ess(f) = 2$;
(iii) $cmr(f) = \sum_{j<i,\ x_i, x_j \in Ess(f)} cmr(f_{i\leftarrow j})$ if $ess(f) \geqslant 3$.

The minors $f_{i\leftarrow j}$ with $i < j$ are excluded because $f_{i\leftarrow j} \equiv f_{j\leftarrow i}$. The minor complexity cmr can be inductively calculated using the MDDs of the functions as it is shown in Algorithm 1, below.

Example 2.3. We now count the cmr-complexity of the functions f and g from Example 2.1, using their MDDs given in Figure 2 A) and B), respectively. There is one minor $(f_{3\leftarrow 2})$ of order 2 and three minors of order 1. Thus we have $cmr(f_{3\leftarrow 2}) = 2$, $cmr(f_{2\leftarrow 1}) = 1.3 = 3$, $cmr(f_{3\leftarrow 1}) = 1.1+2.2 = 5$, and $cmr(f_{4\leftarrow 1}) = 1.1+2.2 = 5$. Again, using (ii) and (iii) of Definition 2.2 we obtain $cmr(f) = 3+5+5+3.2 = 19$ and $cmr(g) = 1.4 + 2.2 + 2.5 + 1.6 = 24$. We clearly have: $mnr(f) = 5$ and $mnr(g) = 6$. Note that the set $M = \{x_1, x_3, x_4\}$ is inseparable in both f and g.

Theorem 2.4. *Let $f \in P_k^n$ with $2 \leqslant ess(f) = n \leqslant k$. Then*

(i) $\frac{n(n-1)}{2} \leqslant cmr(f) \leqslant \frac{n!(n-1)!}{2^{n-2}}$;

(ii) $1 \leqslant mnr(f) \leqslant \frac{n!(n-1)!}{2^{n-2}}$.

3. Equivalence relations with respect to minor complexities

Many of the problems in the applications of the k-valued logic are compounded because of the large number of the functions, namely k^{k^n}. Techniques which involve enumeration of functions can only be used if k and n are trivially small. A common way for extending the scope of such enumerative methods is to classify the functions into equivalence classes by some natural equivalence relation.

Let S_A denote the symmetric group of all permutations of the non-empty set A, and let S_m denote the group $S_{\{1,\ldots,m\}}$ for a natural number m, $m \geqslant 1$.

A *transformation* $\psi : P_k^n \longrightarrow P_k^n$ is an n-tuple of k-valued functions $\psi = (g_1, \ldots, g_n)$, $g_i \in P_k^n$, $i = 1, \ldots, n$ acting on any function $f = f(x_1, \ldots, x_n) \in P_k^n$ as follows $\psi(f) = f(g_1, \ldots, g_n)$. Then the composition of two transformations ψ and $\phi = (h_1, \ldots, h_n)$ is defined as follows

$$\psi\phi = (h_1(g_1, \ldots, g_n), \ldots, h_n(g_1, \ldots, g_n)).$$

The set of all transformations of P_k^n is the *universal monoid* Ω_k^n with unity - the identical transformation $\epsilon = (x_1, \ldots, x_n)$. When taking only invertible transformations we obtain the *universal group* C_k^n isomorphic to the symmetric group $S_{Z_k^n}$. The groups consisting of invertible transformations of P_k^n are called *transformation groups* (sometimes termed *permutation groups*).

Let $\simeq$, $\simeq \subseteq P_k^n \times P_k^n$ be an equivalence relation on the algebra P_k^n. Since P_k^n is a finite algebra of k-valued functions, the equivalence relation $\simeq$ makes a partition of the algebra in a finite number equivalence classes $P_1, \ldots, P_r$.

A mapping $\varphi : P_k^n \longrightarrow P_k^n$ is called a *transformation preserving* $\simeq$ if $f \simeq \varphi(f)$ for all $f \in P_k^n$. Taking only invertible transformations which preserve $\simeq$, we get the group $G_\simeq$ of all transformations preserving $\simeq$. The *orbits* (also called $G_\simeq$*-types*) of this group are exactly the classes $P_1, \ldots, P_r$. The number r of orbits of a group $G_\simeq$ of transformations is denoted $t(G_\simeq)$.

Let $f \in P_k^n$ and let $nof(f)$ denote the normal form obtained by applying the reduction $\rhd$ on f. According to Theorem 1.2, the normal form $nof(f)$ is unique and $nof(f) \in P_k^1$. Thus, our first natural equivalence is defined as follows:

Definition 3.1. Let f and g be two functions from P_k^n. We say that f and g are nof-equivalent (written $f \simeq_{nof} g$) if $nof(f) = nof(g)$.

The transformation group induced by nof-equivalence is denoted NF_k^n. The transformations in NF_k^n preserves $\simeq_{nof}$, i.e. $nof(g) = nof(\psi(g))$ for all $g \in P_k^n$ and $\psi \in NF_k^n$. Since the atomic minors (labels of terminal nodes in MDD) depend on at most one essential variable, it follows that $t(NF_k^n) = |P_k^1| = k^k$. These transformations involve permuting variables, only (see Theorem 3.6, below).

By analogy with the ordered decision diagrams [2, 12], we define several equivalence relations in P_k^n, which allow us to classify the functions by the complexity of their MDDs.

Definition 3.2. Let f and g be two functions from P_k^n. We say that f and g are cmr-equivalent (written $f \simeq_{cmr} g$) iff:

(i) $ess(f) \leqslant 1 \implies ess(f) = ess(g)$;
(ii) $ess(f) \geqslant 2 \implies ess(f) = ess(g)$ and there exists a permutation σ of the set $\{1, \ldots, n\}$ such that $f_{i\leftarrow j} \simeq_{cmr} g_{\sigma(i)\leftarrow\sigma(j)}$ for all j, i, with $x_i, x_j \in Ess(f)$, $j < i$.

Let CM_k^n denote the transformation group preserving the equivalence $\simeq_{cmr}$. Note that if $ess(f) \leqslant 1$ then $Mnr(f) = \emptyset$. Hence, if $ess(f) = ess(g) \leqslant 1$ then $f \simeq_{mnr} g$. MN_k^n denotes the transformation group which preserves the equivalence $\simeq_{mnr}$.

Definition 3.3. Let f and g be two functions from P_k^n. We say that f and g are *mnr-equivalent* (written $f \simeq_{mnr} g$) if $mnr_m(f) = mnr_m(g)$ for all m, $0 \leqslant m \leqslant ess(f) - 1$.

Theorem 3.4.

(i) $f \simeq_{cmr} g \implies cmr(f) = cmr(g)$;
(ii) $f \simeq_{cmr} g \implies f \simeq_{mnr} g$.

It is naturally to ask which groups among "traditional" transformation groups are subgroups of the groups NF_k^n or CM_k^n and which of these groups include NF_k^n, MN_k^n or CM_k^n as their subgroups.

Let $\sigma : Z_k \longrightarrow Z_k$ be a mapping and $\psi_\sigma : P_k^n \longrightarrow P_k^n$ be a transformation of P_k^n generated by σ as follows $\psi_\sigma(f)(\overline{a}) = \sigma(f(\overline{a}))$ for all $\overline{a} \in Z_k^n$.

Theorem 3.5. *The transformation ψ_σ preserves $\simeq_{cmr}$ if and only if σ is a permutation of Z_k, $k > 2$.*

Let $\pi \in S_n$ and $\phi_\pi : P_k^n \longrightarrow P_k^n$ be a transformation of P_k^n defined as follows $\phi_\pi(f)(a_1, \ldots, a_n) = f(a_{\pi(1)}, \ldots, a_{\pi(n)})$ for all $(a_1, \ldots, a_n) \in Z_k^n$.

Theorem 3.6. *The transformation ϕ_π preserves the equivalence relations $\simeq_{cmr}$, $\simeq_{mnr}$ and $\simeq_{nof}$ for all $\pi \in S_n$.*

We deal with "natural" equivalence relations which involve variables of functions. Such relations induce permutations of the domain Z_k^n of the functions. These mappings form a transformation group whose number of equivalence classes can be determined. The restricted affine group (RAG) is defined as a subgroup of the symmetric group on the direct sum of the module Z_k^n of arguments of functions and the ring Z_k of their outputs. The group RAG permutes the direct sum $Z_k^n + Z_k$ under restrictions which preserve single-valuedness of all functions from P_k^n [6, 8, 16].

In the model of RAG an affine transformation ψ operates on the domain or space of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ to produce the output $\mathbf{y} = \mathbf{x}\mathbf{A} \oplus \mathbf{c}$, which might be used as an input in the function f. Its output $f(\mathbf{y})$ together with the function variables $x_1, \ldots, x_n$ are linearly combined by a range transformation which defines the image $g = \psi(f)$ of f as follows:

$$\begin{aligned} g(\mathbf{x}) = \psi(f)(\mathbf{x}) = f(\mathbf{y}) \oplus a_1x_1 \oplus \ldots \oplus a_nx_n \oplus d = \\ f(\mathbf{x}\mathbf{A} \oplus \mathbf{c}) \oplus \mathbf{a^t}\mathbf{x} \oplus d, \end{aligned} \tag{1}$$

where d and a_i for $i = 1, \ldots, n$ are constants from Z_k. Such a transformation belongs to RAG if $\mathbf{A}$ is a non-singular matrix.

S_k^n denotes the transformation group induced by permuting of variables.

Boolean functions of two variables are classified into twelve S_2^2-classes [6], as it is shown in Table 1.

Table 1. The twelve classes in P_2^2 under the permutation of arguments.

$[0]$,	$[x_1^0x_2^0]$,	$[x_1^0x_2,\ x_1x_2^0]$,	$[x_1,\ x_2]$,
$[x_1 \oplus x_2]$,	$[x_1 \oplus x_2^0]$,	$[x_1^0 \oplus x_1x_2,\ x_2^0 \oplus x_1x_2]$,	$[x_1 \oplus x_1^0x_2]$,
$[x_1^0 \oplus x_1x_2^0]$,	$[x_1x_2]$,	$[x_1^0,\ x_2^0]$,	$[1]$.

M. Harrison has determined the cycle index of the group S_2^n and using Polya's counting theorem he has counted the number of equivalence classes under permuting arguments (see [6] and Table 3, below).

The subgroups of RAG, defined according to (1) are determined by equivalence relations, where $\mathbf{P}$ denotes a permutation matrix, $\mathbf{I}$ is the identity matrix, $\mathbf{b}$ and $\mathbf{c}$ are n-dimensional vectors over Z_k^n and $d \in Z_k$.

It is naturally to ask which subgroups of RAG are subgroups of the group NF_k^n, CM_k^n and MN_k^n. Theorem 3.5 and Theorem 3.6 show that CF_k^n and S_k^n are subgroups of CM_k^n. Theorem 3.4 shows that they must also be subgroups of MN_k^n. Clearly, $S_k^n \leqslant NF_k^n$.

Example 3.7. Let $f = x_1 \oplus x_2 \oplus x_3 (mod\ 3)$ and $g = x_1x_2 \oplus x_1x_3 \oplus x_2x_3 (mod\ 3)$. Then we have $f_{i \leftarrow j} = 2x_j \oplus x_m (mod\ 3)$ and $g_{i \leftarrow j} = 2x_jx_m \oplus x_jx_j (mod\ 3)$ where $\{i, j, m\} = \{1, 2, 3\}$. Clearly, $f_{i \leftarrow j_{j \leftarrow m}} = g_{i \leftarrow j_{j \leftarrow m}} = 0$, and hence $f \simeq_{cmr} g$ and $f \simeq_{nof} g$. One can show that there is no transformation $\psi \in RAG$, defined as in (1), for which $g = \psi(f)$. Consequently, $CM_k^n \not\leqslant RAG$, $NF_k^n \not\leqslant RAG$ and $MN_k^n \not\leqslant RAG$.

Example 3.8. Let $f = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^2 (mod\ 3)$ and $g = x_1^0 x_2^1 \oplus x_2^0 x_3^1 x_4^1 (mod\ 3)$ be the functions from Example 2.3 whose MDDs are given in Figure 2. Let

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Then clearly, $f(\mathbf{x}) = g(\mathbf{Ax})$ and hence, f and g belong to the same equivalence class under the transformation groups LG_3^4. Let $\mathbf{c} = (0, 0, 0, 1)$. Then we have $f(\mathbf{x}) = g(\mathbf{x}.\mathbf{I} \oplus \mathbf{c})$, which shows that f and g belong to the same equivalence class under the transformation group CA_3^4. One can show that $f \simeq_{nof} g$. Example 2.3 shows that $f \not\simeq_{mnr} g$. Consequently, $NF_k^n \not\leqslant MN_k^n$, $LG_k^n \not\leqslant MN_k^n$ and $CA_k^n \not\leqslant MN_k^n$. Theorem 3.4 shows that $NF_k^n \not\leqslant CM_k^n$, $LG_k^n \not\leqslant CM_k^n$ and $CA_k^n \not\leqslant CM_k^n$.

Example 3.9. Let $f = x_1^0 x_2 \oplus x_1^1 x_3 \oplus x_1^2 x_2^1 x_3^0 (mod\ 3)$ and $g = x_1^0 x_2 \oplus x_1^1 x_3 (mod\ 3)$ be two functions. It is easy to see that $f_{i \leftarrow j} = g_{i \leftarrow j}$ for all i, j with $1 \leqslant j < i \leqslant 3$. Hence, $f \simeq_{cmr} g$ and $f \simeq_{nof} g$. Now, it is clear that each set of essential variables in f is separable in f, but $\{x_2, x_3\} \notin Sep(g)$ which shows that $f \not\simeq_{sep} g$, i.e. $CM_k^n \not\leqslant SP_k^n$ and $NF_k^n \not\leqslant SP_k^n$.

So, the next theorem summarizes results which determine the positions of the groups NF_k^n, CM_k^n and MN_k^n, with respect to the subgroups of RAG and the groups induced by subfunction complexities [12].

Theorem 3.10.
(i) $CF_k^n \leqslant CM_k^n$; *(ii)* $S_k^n \leqslant CM_k^n$; *(iii)* $S_k^n \leqslant NF_k^n$;
(iv) $NF_k^n \not\leqslant RAG$; *(v)* $CM_k^n \not\leqslant RAG$; *(vi)* $LG_k^n \not\leqslant MN_k^n$;
(vii) $CA_k^n \not\leqslant MN_k^n$; *(viii)* $CA_k^n \not\leqslant NF_k^n$; *(ix)* $LG_k^n \not\leqslant NF_k^n$;
(x) $CM_k^n \not\leqslant SP_k^n$; *(xi)* $CM_k^n \not\leqslant SP_k^n$; *(xii)* $CM_k^n \not\leqslant NF_k^n$
(xiii) $NF_k^n \not\leqslant MN_k^n$.

Theorem 3.10 is well-illustrated by Figure 3, in the case of Boolean functions.

4. Classification of Boolean functions by minor complexities

Table 2 shows the four classes in P_2^2 under the equivalence $\simeq_{cmr}$. The $\simeq_{cmr}$-classes are represented as union of several classes under the permuting arguments, according to Theorem 3.10 (ii) (see Tables 1 and Table 2).

Table 2. The four classes in P_2^2 under the *cmr*-complexity.

[0, 1],	$[x_1^0 x_2,\ x_1 x_2^0,\ x_1 \oplus x_2,\ x_1 \oplus x_2^0,\ x_1^0 \oplus x_1 x_2,\ x_2^0 \oplus x_1 x_2]$,
$[x_1,\ x_2,\ x_1^0,\ x_2^0]$,	$[x_1 x_2,\ x_1 \oplus x_1^0 x_2,\ x_1^0 \oplus x_1 x_2^0,\ x_1^0 x_2^0]$.

The number of types $t(S_2^n)$ under permuting arguments, is an upper bound of the number of equivalence classes induced by the relations $\simeq_{nof}$, $\simeq_{cmr}$ and $\simeq_{mnr}$ for $n \leqslant 6$ (see Table 3). Theorem 3.10 shows that the transformation groups NF_k^n, CM_k^n and MN_k^n are not comparable with the groups IM_k^n, SB_k^n and SP_k^n, determined by subfunction complexities.

Figure 3 presents the subgroups of RAG and transformation groups whose invariants are subfunction and minor complexities of Boolean functions of n-variables. According to Theorem 3.10 the group CM_2^n has three subgroups from RAG, namely:

Table 3. Number of equivalence classes in P_2^n under transformation groups.

n	S_2^n	CM_2^n	MN_2^n	IM_2^n	SB_2^n	SP_2^n
1	4	2	2	2	2	2
2	12	4	3	4	4	3
3	80	11	5	13	11	5
4	3984	*	*	104	74	11
5	37 333 248	*	*	*	*	38
6	25 626 412 338 274 304	*	*	*	*	*

S_2^n - the group of permuting arguments, trivial group, consisting of the identity map, only and CF_2^n- the group of complementing outputs. The groups NF_2^n and MN_2^n are not subgroups of any subgroup of RAG and also, they are not subgroups of any group among IM_2^n, SP_2^n and SB_2^n.

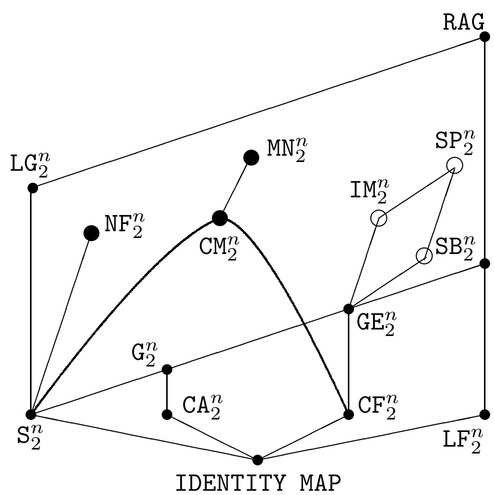


Figure 3. Transformation groups in P_2^n.

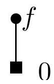
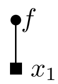
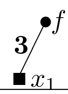
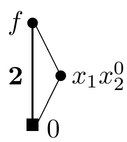
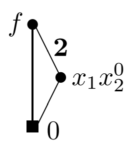
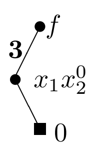
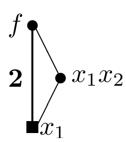
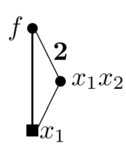
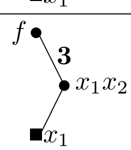
Table 4 presents a full classification of the Boolean functions of tree variables by the minor complexities cmr and mnr.

5. Appendix

Table 4 presents classification of ternary Boolean functions under the equivalences $\simeq_{cmr}$ and $\simeq_{mnr}$, including the catalogue of the equivalence classes (last column).

We also provide an algorithm to find the complexity $cmr(f)$ of an arbitrary k-valued function f. Similar algorithms for manipulation of Boolean functions are presented in [2, 10]. We shall express our algorithm in a pseudo-Pascal notation. The main data structure describes the nodes in the MDD of f. Each node is represented by a record declared as follows:

TABLE 4. Minor classification of ternary Boolean functions.

cmr-class		cmr	Functions per class		mnr	Repres. function	Catalogue
	MDD						
1	const	1	2			0	0
2	var	1	6	8	0	x_1	15,51,85
3		2	18	18	1	$x_1 x_2^0$	10,12,34,48,60,68, 80,90,102
4		2	12			$x_1 x_2$	3,5,17, 63,95,119
5		3	8	20	1	$x_1 \oplus x_2$ $\oplus x_3$	43,77, 105,113
6		4	18			$x_1 x_2 x_3^0$	2,4,8, 16,24,32, 36,64,66
7		5	36			$x_1 x_2^0 x_3$ $\oplus x_1 x_2 x_3^0$	6,18,20,26,28,38, 40,44,52,56,70,72, 74,82,88,96, 98,100
8		6	54	108	2	$x_1 x_2^0$ $\oplus x_1 x_2 x_3^0$	14,22,30,42,46,50, 54,58,62,76,78,84, 86,92,94,104,106, 108,110,112,114,116, 118,120,122,124,126
9		4	50			$x_1 x_2$ $\oplus x_1 x_2^0 x_3$	7,11,13,19,21,23, 31,35,41,47,49,55, 59,69,73,79,81,87, 93,97,107,109,115, 117,121
10		5	36			$x_1 x_2 x_3$ $\oplus x_1 x_2^0 x_3^0$	9,27,29,33,39,45, 53,57,65,71,75,83, 89,99,101, 111,123,125
11		6	16	102	3	$x_1 x_2 x_3$	1,25,37, 61,67,91, 103,127

```
type minor=record
   ess:  1..n;
   val:  0..k^{k^n} - 1;
end;
```

The first field named `ess` presents the number of essential variables in f and the second field `val` is a natural number whose k-ary representation is the last column $\mathbf{b}$ of the truth table (of size k^n) of f.

Algorithm 1 Counting cmr(f)

type minor=record
    ess: 1..n;
    val: 0..$k^{k^n} - 1$;
  end;
var f:minor;
  cmr:integer;
function GetMinor(g:minor; i,j:integer): minor; ▷ Getting minor
  var A,H: array[1..k^N, 1..N] of integer (mod k);
    B,L; array[1..k^N] of integer (mod k);
    h: minor;
  $n := g.ess$;
  Create truth table $A_{k^n \times n}B$ of g;
  Create truth table $H_{k^{n-1} \times n}L$ of $h := g_{i \leftarrow j}$; ▷ Use truth table of g
  Calculate - $h.ess$ and $h.val$ from table HL;
  $GetMinor := h$;
end function;
function Complexity(g:minor):integer; ▷ Counting complexity
  $n := g.ess$;
  if $n > 2$ **then**
    for $j,\ 1 \leqslant j \leqslant n-1$ **do**
      for $i,\ j+1 \leqslant i \leqslant n$ **do**
        $h := GetMinor(g, i, j)$;
        $Complexity := Complexity + Complexity(h)$;
      end for
    end for
  else ▷ Basis of recursion
    if $n = 2$ **then**
      $Complexity := 2$
    else
      $Complexity := 1$
    end if
  end if
end function
Input k; f.ess; f.val;
cmr:=Complexity(f).;
Print cmr.

References

[1] H. Andersen. An introduction to binary decision diagrams, 1997. Lecture notes for 49285 Adv. Algorithms.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[3] K. Chimev. Separable sets of arguments of functions. *MTA SzTAKI Tanulmanyok*, 80:1–173, 1986.

[4] M. Couceiro and E. Lehtonen. Generalizations of Swierczkowski's lemma and the arity gap of finite functions. *Discrete Mathematics*, 309:5905–5912, 2009.

[5] E. Dubrova. Multiple-valued logic in VLSI: challenges and opportunities. In *Proceedings of NORCHIP'99*, pages 340–350, Oslo, Norway, 1999.

[6] M. Harrison. Counting theorems and their applications to classification of switching functions. In A. Mikhopadhyay, editor, *Recent Developments in Switching Theory*, pages 85–120. NY, Academic Press, 1971.

[7] J. Klop and R. de Vrijer. *Term rewriting systems.* Cambridge University Press, 2003.

[8] R. J. Lechner. Harmonic analysis of switching functions. In A. Mikhopadhyay, editor, *Recent Developments in Switching Theory*, pages 121–228. NY, Academic Press, 1971.

[9] A. Salomaa. On essential variables of functions, especially in the algebra of logic. *Annales Academia Scientiarum Fennicae*, Ser. A(333):1–11, 1963.

[10] J. E. Savage. *Models of computation, Exploring the Power of Computing.* Brown University, http://cs.brown.edu/people/jes/book/, 2008.

[11] Sl. Shtrakov. Essential arity gap of boolean functions. *Serdica, Journal of Computing*, 2(3):249–266, 2008.

[12] Sl. Shtrakov and I. Damyanov. On the computational complexity of finite operations. *International Journal of Foundations of Computer Science*, 27(01):15–38, 2016.

[13] Sl. Shtrakov and J. Koppitz. On finite functions with non-trivial arity gap. *J. Discussiones Mathematicae - General Algebra and Applications*, 30:217–245, 2010.

[14] Sl. Shtrakov and J. Koppitz. Finite symmetric functions with non-trivial arity gap. *Serdica, Journal of Computing*, 6(4):419–436, 2012.

[15] Sl. Shtrakov and J. Koppitz. Stable varieties of semigroups and groupoids. *Algebra Universalis*, 75(1):85–106, 2016.

[16] I. Strazdins. On fundamental transformation groups in the algebra of logic. In B. Csakany and I. Rosenberg, editors, *Finite algebra and multiple-valued logic*, volume 28, pages 669–691, Szeged, 1979. Colloquia Mathematica Societatis János Bolyai.

[17] R. Willard. Essential arities of term operations in finite algebras. *Discrete Mathematics*, 149:239–259, 1996.

E-mail address: `shtrakov@swu.bg`
URL: `http://shtrakov.swu.bg/`

Department of Computer Science, South-West University, Blagoevgrad