Bi-Decomposition of Discrete Function Sets

Bernd Steinbach, Christian Lang, and Marek Perkowski †
Freiberg Univ. of Mining and Technology, Inst. of Computer Science
09599 Freiberg, Germany, steinb@informatik.tu-freiberg.de
† Portland State Univ., EE Dept., Portland, OR 97207
mperkows@ee.pdx.edu

Abstract

This paper extends the bi-decomposition of Boolean functions by generalizing the notion of Incompletely Specified Functions (ISFs) to the new concept of **function sets**. In particular, the relation between EXOR-decomposition and a special class of function sets, C-ISFs, is discussed, and the respective decomposition algorithms for C-ISFs are presented. It is shown that decompositions of better quality are obtained when using C-ISFs instead of ISFs. Decomposition of function sets is extended to sets of multi-valued functions, where especially good results have been obtained for highly unspecified benchmark functions from the area of Data Mining.

1 Introduction

Decomposition has many applications in circuit design or machine learning. The background on decomposition and bi-decomposition can be found in [1, 2, 3]. Usually, decomposition is performed recursively over several stages. Each stage makes some decisions about the decomposition process based on local knowledge, but with global consequences. Usually, heuristics are used to make these decisions. In functional decomposition for instance, a function $f(X) = h(A, g(B))$ is decomposed into component functions g and h. However, there exists not just one pair of functions (g, h) that satisfy the above equation. Many decomposition algorithms select a promising candidate pair from the set of all pairs of functions $G = \{(g, h)|f = h(A, g(B))\}$ for further decomposition by some heuristic [12]. This method is efficient if there is only a small number of functions to choose from. Machine learning benchmarks, however, frequently contain incompletely specified functions with an extremely high percentage of don't cares. Therefore, there is a very large number of functions from which the heuristic has to choose from. In such case it is difficult to find good heuristics especially for the early stages of the decomposition process. In some cases however, it is possible to postpone such a decision until more information becomes available. Decomposition of Incompletely Specified Functions (ISFs) is one example [5, 9, 15, 14]. This paper is a continuation of [17] where standard incompletely specified functions were discussed (Here we improve on the concepts and ideas introduced there, repeating only the minimum notations

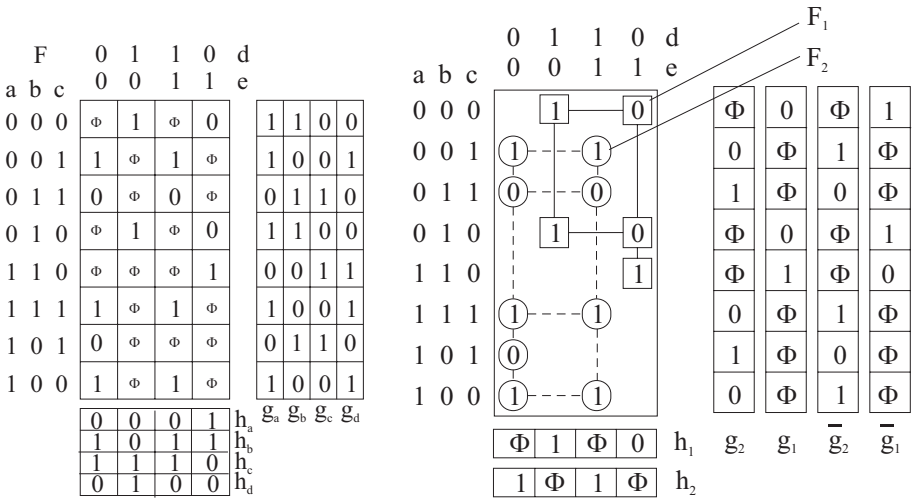

Figure 1: a) EXOR decomposable ISF b) Decomposition graph.

necessary to understand the paper.) Many pairs of functions (g_i, h_i) can be selected from the set G if one decomposes ISFs instead of completely specified functions. The decision which particular function is designed will be made by further decomposition steps. Here, ISFs can be considered as a special class of **function sets**. In many algorithms however, other sets of functions than ISFs arise from the decision step. Usually, these sets are approximated by a heuristically chosen ISF. Consider the ISF from Fig. 1a). EXOR bi-decomposition of this ISF gives four possible pairs of functions $(g_a, h_a), (g_b, h_b), (g_c, h_c), (g_d, h_d)$ where set $G = \{g_a, g_b, g_c, g_d\}$ cannot be described by an ISF. The algorithm from [14] would choose function g_a instead of g_b which has a more expensive total realization $f_a = g_a \oplus h_a$ than the realization $f_b = g_b \oplus h_b$, as shown in Fig. 2. To select the best solution (g_i, h_i) we will suggest here a new strategy of postponing the selection process. When we have to select among several functions, we compute all these functions and **pass the set of functions to a suitable algorithm for further decomposition**. One problem with this strategy is the complexity of the function sets. There are 2^{2^n} Boolean functions of n variables, but $2^{2^{2^n}}$ sets of Boolean functions. Because of this high number, representations of function sets must be problem-specific and exploit the structure of sets created by the algorithm. Efficient data structures must be applied to store large sets of functions.

This paper introduces and investigates C-ISF; a special class of function sets which allows to store and process function sets arising from EXOR-decompositions of ISF. In many problems of EXOR logic and decomposition theory there is a need to represent a function as an EXOR of two functions that satisfy certain properties, for instance have the smallest literal cost. Our introduced here methods can find applications in many of those problems, although here we will concentrate on bi-decomposition for Data Mining [17].

Section 2 will introduce the Function Sets and discuss some of their properties. Section 3 discusses the decomposition of Function Sets, and section 4 the EXOR decomposition of ISFs. In section 5 we introduce the crux of this paper - the decomposition of C-ISFs. These

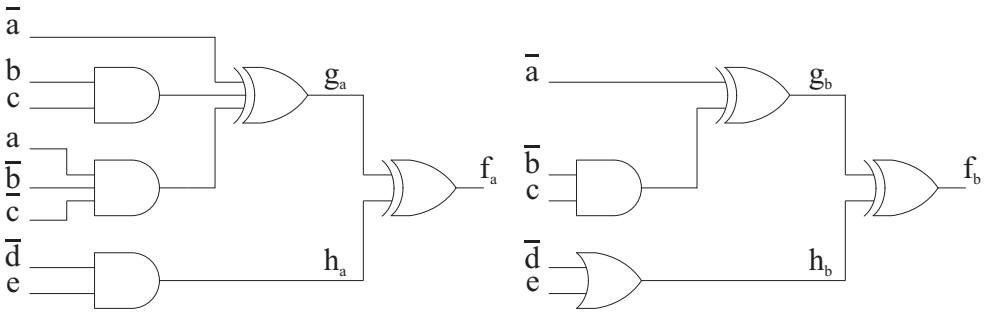

Figure 2: Two realizations of the EXOR-decomposition from Fig. 1a) with different cost.

results are next generalized in section 6 to multiple-valued C-ISFs. Section 7 presents our experimental results and section 8 introduces the new concept of **Vector Relation Sets** that are more general than both the C-ISFs introduced here and the Boolean and Multi-Valued Relations introduced in [4, 17]. Finally, section 9 concludes the paper.

2 Definition and Properties of Function Sets

A *Boolean function* is a mapping $B^n \rightarrow B^1$ and is denoted $f(\underline{x})$, where $\underline{x}$ is a vector of n Boolean variables. An ISF can be understood as a subset of $B^n \times B^1$. Here, we define an ISF as a set of Boolean functions.

Definition 1. An *ISF* $F < f(\underline{x}), \varphi(\underline{x}) >$ with $f(\underline{x}) \wedge \varphi(\underline{x}) = 0$ is the set of functions $g(\underline{x})$ for which $f(\underline{x}) \leq g(\underline{x}) \leq f(\underline{x}) \vee \varphi(\underline{x})$. The Boolean function $f(\underline{x})$ is called the on-set and $\varphi(\underline{x})$ the don't care set of F.

An ISF $F < f, \varphi >$ consists of 2^m functions, where $m = |\varphi|$. Note that, by this definition, a Boolean function is a special case of an ISF with $\varphi = 0$. To denote an ISF by a single function, we define the characteristic function of an ISF that assumes the constant value Φ for the don't cares of the ISF.

Definition 2. The *characteristic function* $F_C(\underline{x})$ of an ISF $F < f(\underline{x}), \varphi(\underline{x}) >$ is a mapping $B^n \rightarrow \{0, 1, \Phi\}$ with

$$F_C(\underline{x}) = \begin{cases} f(\underline{x}) & \text{if } \varphi(\underline{x}) = 0, \\ \Phi & \text{otherwise.} \end{cases} \tag{1}$$

There is an one-to-one mapping between ISFs and characteristic functions. Therefore, we will use the same notation for an ISF and its characteristic function.

Example 1. Fig. 3 gives an example of an ISF $F < f, \varphi >$ with the on-set f, the don't care set φ and the characteristic Function F_C. The ISF contains the 4 functions f_1, f_2, f_3, and f_4.

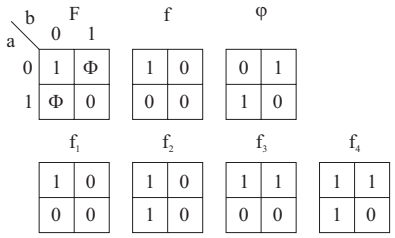

Figure 3: ISF $F < f, \varphi >$ and its element functions f_1, f_2, f_3, f_4.

To decide whether function $g(\underline{x})$ is an element of an ISF F it is sufficient to compare the values of g with the characteristic function $F_C(\underline{x})$ for all the cares of F.

Lemma 1. *A Boolean function $g(\underline{x})$ is element of an ISF F if $g(\underline{x}_0) = F(\underline{x}_0)$ for all $\underline{x}_0$ with $F(\underline{x}_0) \neq \Phi$.*

The value of a element function of an ISF can be flipped at the don't cares, and the result still is an element of the ISF. To define this property symbolically, we introduce the delta function.

Definition 3. The *delta function* $\delta_{\underline{x}_0}(\underline{x})$ is a Boolean function whose value is 1 for $\underline{x} = \underline{x}_0$ and 0 otherwise [20].

Lemma 2. *Let F be an ISF and function $g(\underline{x}) \in F$ be an element of F. If $F(\underline{x}_0) = \Phi$, then $g'(\underline{x}) = g(\underline{x}) \oplus \delta_{\underline{x}_0}(\underline{x})$ is also an element of F, denoted by $g'(\underline{x}) \in F$.*

Because there are sets of Boolean functions that cannot be described by an interval of functions as in the case of ISF, we will generalize ISFs and the notion of their characteristic function.

Definition 4. A *function set* $S(\underline{x})$ is a non-empty subset of the set of all Boolean functions $\mathbb{F}(\underline{x})$ that depend on the vector of variables $\underline{x}$. Its *characteristic function* $S_C(\underline{x})$ is defined as a mapping $B^n \to \{0, 1, \Phi, \Delta\}$, with

$$S_C(\underline{x}_0) = \begin{cases} 1 & \text{if } \forall f \in S : f(\underline{x}_0) = 1 \\ 0 & \text{if } \forall f \in S : f(\underline{x}_0) = 0 \\ \Phi & \text{if } \forall f \in S : f(\underline{x}) \oplus \delta_{\underline{x}_0}(\underline{x}) \in S \\ \Delta & \text{otherwise} \end{cases} \tag{2}$$

where Δ is a new constant value which we call a *dependent care*.

Example 2. Consider the functions f_1, f_2, f_3, and f_4 in Fig. 4. Assume there is an ISF F that contains exactly these 4 functions. Consider the value of F for $\underline{x}_0 = \overline{a}\,\overline{b}\,c$. This value cannot be 1 because of function f_4, it cannot be 0 because of f_1, and it cannot be Φ because, by Lemma 2, $f_5 = f_1 \oplus \delta_{\overline{a}\,\overline{b}\,c}$ would be an additional element of F. This contradicts the assumption that there is an ISF F containing only $f_1, \ldots, f_4$.

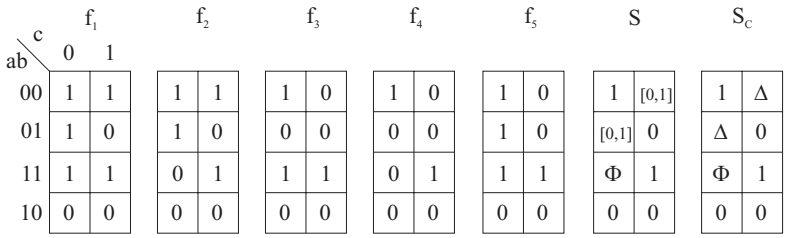

Figure 4: Function set $S = \{f_1, f_2, f_3, f_4\}$ and characteristic function S_C.

Fig. 4 also shows the function set $S = \{f_1, f_2, f_3, f_4\}$ and its characteristic function S_C. The values 0, 1, and Φ can be interpreted as in the case of an ISF. For the dependent cares (cells $\overline{a}\,\overline{b}\,c$ and $\overline{a}\,b\overline{c}$), the Karnaugh map of S shows vectors of values and the characteristic function S_C assumes value Δ. The interpretation of the vectors of values is that any component of the vector can be chosen as the value of an element function of S, but the components with the same index must be chosen for every vector. For function set S for instance, we can choose $f(\overline{a}\,\overline{b}\,c) = f(\overline{a}\,b\overline{c}) = 0$ (first component), or $f(\overline{a}\,\overline{b}\,c) = f(\overline{a}\,b\overline{c}) = 1$ (second component).

Note that an ISF is a special case of a function set. Because function sets are more general than ISFs, we need to introduce another constant value Δ for the characteristic function that describes the **dependent cares**. It follows from Lemma 2 that the characteristic function of an ISF does not assume value Δ, and Definition 4 is consistent with Definition 2 of the characteristic function for ISF. Note that there is a many-to-one relation between function sets and characteristic functions. Two different function sets can have the same characteristic function with different interpretations of the Δ-values. Therefore, we have to distinguish between a function set S and its characteristic function S_C.

The characteristic function gives some information about the possible choices for a certain function value. A value $S_C(\underline{x}_0) = \Phi$ indicates, as in the case of ISF, that the value of an element function can be chosen arbitrarily for $\underline{x}_0$. For a value $S_C(\underline{x}_0) = \Delta$, the element function can be 0 or 1 at $\underline{x}_0$, but the choice of values for other $\underline{x}_1 \neq \underline{x}_0$ is restricted by this decision.

Example 3. Consider the function set S from Fig. 4. The value at $ab\overline{c}$ can be 0 or 1 without additional restrictions for other function values. For $f(\overline{a}\,\overline{b}\,c)$ we have a dependency with $f(\overline{a}\,b\overline{c})$ of the form $f(\overline{a}\,\overline{b}\,c) = f(\overline{a}\,b\overline{c})$.

It would be possible to store function sets by enumerating all element functions described by them. The number of element functions, however, can become very large, as in the case of ISFs for example. Therefore, we compose functions sets from ISFs by using set operations on them. Because the function sets are sets themselves, the union and intersection operations on them are well defined.

Definition 5. The *intersection* of two function sets, $S_1 \cap S_2$, is the set of functions S_3 which are elements of both S_1 and S_2.

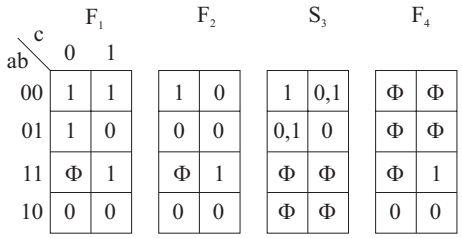

Figure 5: Function sets for union and intersection.

Definition 6. The *union* of two function sets, $S_1 \cup S_2$, is the set of functions which are elements of S_1, or S_2, or both.

The intersection of function sets is either empty or a function set. The union is always a function set. It is important to distinguish between these set operations on sets of functions and the conjunction and disjunction operations that can be applied to incomplete Boolean functions. Conjunction and disjunction are defined for Boolean functions and the result is a Boolean function. Intersection and union are set operations over function sets. Their results are sets of functions.

Example 4. Consider the Boolean functions f_1, f_2, f_3, and f_4 from Fig. 3. Their disjunction is the Boolean function $f_1 \vee f_2 \vee f_3 \vee f_4 = f_4 = \overline{a} \vee \overline{b}$, their union $f_1 \cup f_2 \cup f_3 \cup f_4 = \{f_1, f_2, f_3, f_4\} = F$ is the ISF F shown in Fig. 3. Their conjunction is the Boolean function $f_1 \wedge f_2 \wedge f_3 \wedge f_4 = f_1 = \overline{a} \wedge \overline{b}$. Because f_1 and f_2 are different functions, the intersection is empty; $f_1 \cap f_2 \cap f_3 \cap f_4 = \emptyset$.

In this paper we only deal with two special cases of intersection and union of function sets.

1. Intersection of functions sets where the cares and dependent cares of one function set lie completely in the don't care area of the other function set: In this case the don't cares of the first function are specified by the care and dependent care values of the other function.

2. Union of ISFs with the same dont't care sets: Different care values become dependent care values with two choices. The cares of the first function set form the first choice, and the cares of the other function form the second choice.

Example 5. For the first special case, see Fig. 5, the intersection $S = S_3 \cap F_4$ is the function set S shown in Fig. 4. The cares of F_4 replaced the Φ values of S_3.

For the second special case assume ISFs $F_1 = \{f_1, f_2\}$ and $F_2 = \{f_3, f_4\}$ (compare Fig. 4). Therefore, $S = \{f_1, f_2, f_3, f_4\} = F_1 \cup F_2$. The cares of the ISFs F_1 and F_2 differ for $\overline{a}\, b \overline{c}$ and $\overline{a}\, \overline{b}\, c$. These are the dependent cares of S. The first choice $f(\overline{a}\, b \overline{c}\,) = f(\overline{a}\, \overline{b}\, c) = 0$ has the care values of F_2. The second choice $f(\overline{a}\, b \overline{c}\,) = f(\overline{a}\, \overline{b}\, c) = 1$ has the care values of F_1.

We sometimes only want to verify whether or not the intersection of two function sets is empty. A value Φ in the characteristic function of a function set does not put restrictions

on other values of the element functions of a function set. Therefore, intersection with Φ always gives non-empty results.

Lemma 3. *The intersection of function sets S_1 and S_2; $S_1 \cap S_2$ is non-empty if their characteristic functions $S_{C1}(\underline{x})$ and $S_{C2}(\underline{x})$ only combine values $\neq \Phi$ with values Φ in the other function, i.e. for all $\underline{x}_0$ there is $S_{C2}(\underline{x}_0) = \Phi$ if $S_{C1}(\underline{x}_0) \neq \Phi$.*

There is the identity $x \oplus y = \overline{x} \oplus \overline{y}$. Therefore, we have to compute the set of all functions whose complement is an element of F.

Definition 7. The *negation* $\overline{F}$ of an ISF F is the set of functions f' whose negation is an element of F, $\overline{F} = \{f' | \overline{f'} \in F\}$.

The next Lemma shows how the negation of an ISF can be computed.

Lemma 4. *The negation $F_2 < f_2, \varphi_2 >= \overline{F_1}$ of an ISF $F_1 < f_1, \varphi_1 >$ is an ISF with on-set $f_2 = \overline{f_1} \wedge \overline{\varphi_1}$ and the don't care set $\varphi_2 = \varphi_1$.*

The negation of an ISF can be thought of an ISF with the same don't care set where the cares are negated. The column to the right of Fig. 1b) shows an ISF $g_1(a,b,c)$ and its negation $\overline{g_1(a,b,c)}$.

3 Decomposition of Function Sets

Bi-decomposition is the decomposition of a function into two subfunctions $f(X) = g(A,C) \circ h(B,C)$ where (A,B,C) is a partition of X, and operation $\circ$ is an arbitrary binary operation. Depending on operation $\circ$, we distinguish OR-decomposition ($\vee$), AND-decomposition ($\wedge$), and EXOR-decomposition ($\oplus$). If $C = \emptyset$, we call the decomposition disjoint. Here, we will discuss only the case of disjoint decompositions. The more general case of non-disjoint bi-decompositions can be reduced to disjoint decomposition of function sets with repeated variables [9].

The definition of bi-decomposition can be extended to function sets. Generally, if S is a function set then there are many pairs of functions g and h so that $g \circ h \in S$.

Definition 8. Given a function set $S(X)$ and a partition (A,B) of X, a *decomposition relation* $D^{\circ}_{AB}(g,h)$ is a relation whose domain is the set $\mathbb{F}(A)$ of all functions that depend on A and whose codomain is the set of all functions $\mathbb{F}(B)$ that depend on B. Functions $g(A)$ and $h(B)$ have relation $D^{\circ}_{AB}(g,h) = 1$ if and only if $g \circ h \in S$.

Example 6. Consider the ISF F from Fig. 3. The decomposition relation for $A = \{a\}$ and $B = \{b\}$ for OR-Decomposition ($\circ \rightarrow \vee$) is shown in Fig. 6a. For $g(a) = 0$ and $h(b) = \overline{b}$ we have $g \vee h = f_2 \in F$. Therefore, $D^{\vee}_{ab}(0, \overline{b}\,) = 1$. The only other elements of $D^{\vee}_{ab}$ are $D^{\vee}_{ab}(\overline{a}\,, 0) = 1$ and $D^{\vee}_{ab}(\overline{a}\,, \overline{b}\,) = 1$.

Consider EXOR-decomposition of the ISF F from Fig. 1a with respect to $A = \{a,b,c\}$ and $B = \{d,e\}$. There are four pairs of functions, (g_a, h_a), (g_b, h_b), (g_c, h_c), and (g_d, h_d) with $g(a,b,c) \oplus h(d,e) \in S$. The domain of $D^{\oplus}_{AB}$ is the set of the 256 functions $g(a,b,c)$ that

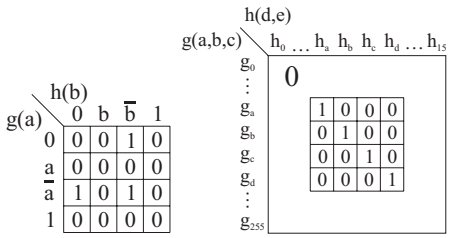

Figure 6: Decomposition relation for a) OR-Decomposition of F from Fig. 3, b) EXOR-Decomposition of the ISF from Fig. 1a.

depend on the variables $\{a, b, c\}$, the codomain are the 16 functions $h(d, e)$ that depend on $\{d, e\}$. The decomposition relation has $D^{\oplus}_{AB} = 1$ only for the above mentioned four pairs of functions.

Definition 9. Given a function set S with a decomposition relation D°_{AB}. The *image* of a function set $S_g(A)$ is a set of functions $S_h(B)$ so that for every $h(B) \in S_h$ there is a function $g(A) \in S_g$ with $D^{\circ}_{AB}(g, h) = 1$. The *decomposition set* $G(A)$ is the image of the set of all the functions that depend on B, $\mathbb{F}(B)$.

The decomposition set G of a function set S is the largest set of functions such that for every function $g \in G$ there is a decomposition $g \circ h \in S$. The decomposition set will be passed to the next stage of recursive decomposition. The remainder of this section and the next section will show how the decomposition set of EXOR-decomposition of an ISF can be computed.

Example 7. The image of $S_g(a) = \{g_a, g_c\}$ with respect to decomposition relation $D^{\oplus}_{AB}$ in Fig. 6b is the set $S_h = \{h_a, h_c\}$ because $D^{\oplus}_{AB}(g_a, h_a) = 1$ and $D^{\oplus}_{AB}(g_c, h_c) = 1$. The decomposition sets of S are $G = \{g_a, g_b, g_c, g_d\}$ and $H = \{h_a, h_b, h_c, h_d\}$.

To find the structure of a decomposition set, we partition the function set into independent parts. For this reason we depict function sets in a decomposition chart, which is a Karnaugh map with variables from A for the rows and variables from B for the columns, see Fig. 1a with $A = \{a, b, c\}$ and $B = \{d, e\}$. Using this chart we can define decomposition graph.

Definition 10. A *decomposition graph* of function set S is a graph that has one vertex for each care or dependent care value of $S_C(X)$, where S_C is the characteristic function of S. The vertex is labeled with the value of S_C. There is an edge between any two vertices from the same row or the same column of the decomposition chart of $S_C(X)$. A *row(column) cover set* is the set of all rows(columns) where the decomposition graph has at least one vertex.

The decomposition graph of a function set can be partitioned to independent components, each of them being a connected component. No connected component is connected to another.

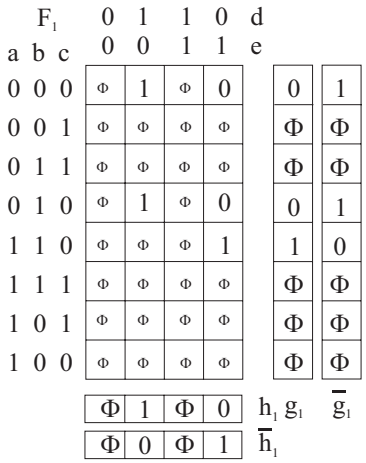

Figure 7: Component ISF F_1 of ISF F from Fig. 1.

Example 8. Fig. 1b depicts the decomposition chart of the ISF F from Fig. 1a with respect to the variable sets $A = \{a, b, c\}$ and $B = \{d, e\}$ (transitive edges in the same row or column are not shown). There is one vertex for each care value of F. There are edges between the vertices in the same rows and columns. The graph of F has two independent connected components, F_1 and F_2, shown by circle and square vertices in Fig 1b, respectively. Each of the independent components is connected by itself. There is no edge between the (independent) components F_1 and F_2. The row cover set of F_1 consists of rows $R = \{\overline{a}\,\overline{b}\,\overline{c}\,, \overline{a}\,b\overline{c}\,, ab\overline{c}\,\}$. The column cover set of F_1 consists of columns $C = \{d\overline{e}\,, \overline{d}\,e\}$.

Our EXOR-decomposition algorithm for ISFs first finds the decomposition sets of the **independent components**, and then combines these sets to compute the decomposition set of the whole ISF. We first show that an ISF can be divided into its independent components.

Definition 11. Let F be an ISF and $P_1, \ldots, P_n$ be the independent components of the decomposition graph of F. A *component ISF* F_i of F is the ISF that has a care with the same label for each vertex of P_i and is Φ otherwise.

Theorem 1. *An ISF F is the intersection of all its component ISFs $F_1, \ldots, F_n$: $F = \bigcap_{i=1}^{n} F_i$.*

Proof. The care sets of the component ISFs are disjoint. Their intersection is the combination of the cares. Because each care of F is the care of one component ISF F_i, the intersection of all component ISFs gives ISF F. □

The component ISF F_1 of F from Fig. 1b is depicted in Fig. 7. Comparison of Fig. 1a and Fig. 1b shows that $F = F_1 \cap F_2$.

4 EXOR Decomposition of ISF

In bi-decomposition it is sufficient to consider the cases of EXOR- and OR-decomposition [13]. Other types of bi-decomposition can be reduced to these two types. For the case

of OR-decomposition of ISF, it has been shown that the decomposition set is an ISF [5]. However, the decomposition set has a more complicated structure for EXOR decomposition. The decomposition set G of a fully specified decomposable function $f = g \oplus h = \overline{g} \oplus \overline{h}$ consists of the two functions $G = \{g, \overline{g}\}$. The structure of the decomposition set of EXOR-decomposable ISFs is the topic of this section.

We start with ISFs whose decomposition graph is connected.

Theorem 2. *Let $F(X)$ be an ISF whose decomposition graph is connected. Then, the decomposition set $G(A)$ for EXOR-decomposition of F is the union of an ISF F_G and its negation, $G = F_G \cup \overline{F_G}$. We call F_G the* decomposition ISF *of F.*

Proof. Let be $\underline{x}_{a0} \in R$ be an arbitrary but fixed element of R, where $R(A)$ is the row cover set of F. We first show that the set G_0 of all decomposition functions $g \in G$ with $g(\underline{x}_{a0}) = 0$ is an ISF. Let H_0 be the image of G_0 with respect to the decomposition relation $D^{\oplus}_{AB}$. We can compute the values of functions in G_0 and H_0 successively. For a care of F in row $\underline{x}_{a0}$ and column $\underline{x}_{b0}$ of the decomposition chart, we obtain $h(\underline{x}_{b0}) = F(\underline{x}_{a0}, \underline{x}_{b0}) \oplus g(\underline{x}_{a0})$. Similarly, if $h(\underline{x}_{b0})$ has been computed, we can compute $g(\underline{x}_{a1}) = F(\underline{x}_{a1}, \underline{x}_{b0}) \oplus h(\underline{x}_{b0})$ for a care of F in row $\underline{x}_{a1}$ and column $\underline{x}_{b0}$. Because the decomposition graph of F is connected, all values of G_0 for the row cover set of F can be computed. For values $\underline{x}_{a3} \notin R$ by definition of the row cover set, it holds $F(\underline{x}_{a3}, \underline{x}_b) = \Phi$. Therefore, the value for $g(\underline{x}_{a3})$ can be chosen arbitrarily, and $G_0(\underline{x}_{a3}) = \Phi$. Note that G_0 does not have dependent cares. Therefore, G_0 is an ISF.

Similarly, it can be shown that G_1, the set of all functions $g \in G$ with $g(\underline{x}_{a0}) = 1$ is an ISF. Because of $g \oplus h = \overline{g} \oplus \overline{h}$, we have $G_0 = \overline{G_1}$, and $G = G_0 \cup \overline{G_0}$. □

Example 9. We show the generation of the decomposition set for EXOR-decomposition of ISF F_1 from Fig. 7. Choose $\underline{x}_{a0} = \overline{a}\,\overline{b}\,\overline{c}$ as an element of the row cover set of F and compute the ISF G_0 of all decomposition functions g with $g(\overline{a}\,\overline{b}\,\overline{c}) = 0$. Now $h(\overline{d}\,e) = F_1(\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}\,e) \oplus g(\overline{a}\,\overline{b}\,\overline{c}) = 0 \oplus 0 = 0$ can be derived. With $h(\overline{d}\,e)$ computed $g(\overline{a}\,b\overline{c}) = F_1(\overline{a}\,b\overline{c}\,\overline{d}\,e) \oplus h(\overline{d}\,e) = 0$ is found. Similar computation gives the values for g and h as shown in Fig. 7. The decomposition set of F consists of the function set $G = G_0 \cup \overline{G_0}$.

Existing algorithms for EXOR-decomposition, like [10, 14], compute the decomposition set for component ISFs very efficiently. We will now show how the results for the component ISFs should be combined to produce the decomposition set of general ISFs. With the help of the next Lemma, the theorem about the combination of the decomposition sets of the component ISFs will be formulated.

Lemma 5. *Let $S(X)$ be a function set with the decomposition set $G(A)$ and the column cover $C(B)$. The image of a function set $S_g(A) \subseteq G$ with respect to the decomposition relation D°_{AB} is a function set $S_h(B)$ whose characteristic function $S_{Ch}(\underline{x}_{b0}) = \Phi$ for all $\underline{x}_{b0} \notin C$.*

Proof. We have to show that if $h \in S_h$ then $h' = h \oplus \delta_{\underline{x}_{b0}} \in S_h$ for all $\underline{x}_{b0} \notin C$. Because S_g is a subset of G and S_h is a image of S_g there is a function $g(A)$ so that $g \circ h \in S$. It follows from Def. 3 that $h(\underline{x}_{b1}) = h'(\underline{x}_{b1})$ for all $\underline{x}_{b1} \in C$ and $g(A) \circ h(\underline{x}_{b1}) = g(A) \circ h'(\underline{x}_{b1})$. By definition of the column cover set, $S_C(A, \underline{x}_{b0}) = \Phi$. Therefore, for each $\underline{x}_b$ either $g(A) \circ h(\underline{x}_b) = g(A) \circ h'(\underline{x}_b)$ or $S_C(A, \underline{x}_b) = \Phi$, and $g \circ h' \in S$. Therefore, that h' is element of S_h. □

Theorem 3. *Let S_1 and S_2 be two function sets with disjoint row covers R_1 and R_2 and disjoint column covers C_1 and C_2. Let G_1 and H_1 be the decomposition sets of S_1; and G_2 and H_2 be the decomposition sets of S_2. Then, the decomposition sets of $S_3 = S_1 \cap S_2$ are $G_3 = G_1 \cap G_2$ and $H_3 = H_1 \cap H_2$.*

Proof. We first show that $H_3 \subseteq H_1 \cap H_2$. Obviously, $S_3 \subseteq S_1$. Therefore, each decomposition function $h \in H_3$ is also element of H_1. Similarly, $H_3 \subseteq H_2$, and $H_3 \subseteq H_1 \cap H_2$.

We now show that $H_1 \cap H_2 \subseteq H_3$. That means it should be shown that for all $h \in H_1 \cap H_2$ there is a function $g(A)$ so that $g \circ h \in S_3$. Let $S_{g1}(A)$ be the image of h with respect to decomposition relation D°_{1AB} of S_1. Then, by Lemma 5, we get, for the characteristic function S_{Cg1} of S_{g1}, $S_{Cg1}(\underline{x}_{a1}) = \Phi$ for all $\underline{x}_{a1} \notin R_1$, where R_1 is the row cover of set S_1. Similarly one obtains $S_{Cg2}(\underline{x}_{a2}) = \Phi$ for all $\underline{x}_{a2} \notin R_2$, where S_{g2} is the image of h with respect to the decomposition relation D°_{2AB} of S_2, and R_2 is the row cover set of S_2. Because the row covers R_1 and R_2 are disjoint, we get $S_{g1} \cap S_{g2} \neq \emptyset$ by Lemma 3. Let $g \in S_{g1} \cap S_{g2}$. Since $g \circ h \in S_1$ and $g \circ h \in S_2$ then $g \circ h \in S_3$.

Because $H_3 \subseteq H_1 \cap H_2$ and $H_1 \cap H_2 \subseteq H_3$ then $H_3 = H_1 \cap H_2$. □

The next lemma shows that component ISFs satisfy the assumption of disjoint row and column cover sets in Theorem 3.

Lemma 6. *Let P be the decomposition graph of a function set S, and P_1 and P_2 be two independent connected components of S (that are not connected to each other). Then, the column cover sets of P_1 and P_2 are disjoint. Similarly, the row cover sets of P_1 and P_2 are disjoint.*

Proof. Assume the column cover sets of P_1 and P_2 were not disjoint. Then, there would be a column where both graphs had a vertex. By definition of the decomposition graph there would be an edge between these vertices and the graphs would be connected. A similar proof holds for the row cover. □

Now the theorem about the structure of the decomposition set of an ISF for EXOR-decomposition can be finally formulated.

Theorem 4. *The decomposition set G of EXOR-decomposition of an ISF F is given by*

$$G = (F_{G1} \cup \overline{F_{G1}}) \cap (F_{G2} \cup \overline{F_{G2}}) \ldots (F_{Gn} \cup \overline{F_{Gn}}), \tag{3}$$

where $F_{G1}, \ldots, F_{Gn}$ are the decomposition ISFs of the component ISFs of F.

Proof. By Theorem 2, the decomposition set of each component ISF F_i is the union $F_{Gi} \cup \overline{F_{Gi}}$. By Lemma 6 the row and column cover sets of the component ISFs $F_1, \ldots, F_n$ are mutually disjoint, and $F = F_1 \cap \ldots F_n$ because of Theorem 1. Because of Theorem 3, the decomposition set of F is given by the intersection of the decomposition sets of its component ISFs F_i. □

Example 10. Fig. 1b shows the component ISFs F_1 and F_2 of the ISF F from Fig. 1a. The decomposition set of F_1 is the union $G_1 = g_1 \cup \overline{g_1}$. The decomposition set of F_2 is the union $G_2 = g_2 \cup \overline{g_2}$ also shown in the Figure. Every decomposition function g_a, g_b, g_c and g_d of F

is an element of the decomposition sets G_1 and G_2. That is, each decomposition function g_i is an element of either g_1 or $\overline{g_1}$ and an element of either g_2 or $\overline{g_2}$. For instance, $g_a \in \overline{g_1}$ and $g_a \in \overline{g_2}$.

To describe the decomposition set of EXOR decomposition of ISF, a special class of function sets will be defined; the so-called **Combinational ISFs**, or C-ISFs.

Definition 12. A *combinational ISF, or C-ISF* $S_F < F_1, \ldots, F_n >$ is the set of functions defined by

$$S_F = (F_1 \cup \overline{F_1}) \cap (F_2 \cup \overline{F_2}) \cap \cdots \cap (F_n \cup \overline{F_n}) \tag{4}$$

By Theorem 4, the decomposition set of EXOR-decomposition of ISF is a C-ISF. We will now show an algorithm for the computation of a C-ISF. There are several algorithms that compute the largest ISF approximation of the decomposition set of an ISF [10, 14]. These algorithms can be used to compute the decomposition ISF of the component ISFs. This vector of decomposition ISFs is then returned as an C-ISF. This leads to Algorithm 1.

Algorithm 1. EXOR-Decomposition of ISF.
Input: ISF $F(A, B)$
Output: C-ISF $G < \underline{F_G}(A) >$ that is the decomposition set of F
1. Compute the decomposition graph F.
2. Find the component ISFs $F_1, \ldots, F_n$ of F.
3. For $i = 1, \ldots, n$
   Find the decomposition ISF $F_{Gi}(A)$ of F_i as introduced in Theorem 2.
4. Return $G < F_{G1}, \ldots, F_{Gn} >$.

Example 11. We demonstrate the above algorithm for a decomposition of the ISF F from Fig. 1a. The algorithm finds the component ISFs F_1 and F_2. EXOR-decomposition of the ISFs results in ISFs g_1 and g_2 or their negations. Because the C-ISF also contains the negation of g_i, it does not matter if we obtain g_i or $\overline{g_i}$ from the decomposition. The result is the C-ISF

$$\begin{aligned} G < g_1, g_2 > &= (g_1 \cup \overline{g_1}) \cap (g_2 \cup \overline{g_2}) \\ &= (g_1 \cap g_2) \cup (g_1 \cap \overline{g_2}) \cup (\overline{g_1} \cap g_2) \cup (\overline{g_1} \cap \overline{g_2}). \end{aligned} \tag{5}$$

Comparison with Fig. 1a shows that $g_a = \overline{g_1} \cap \overline{g_2}$, $g_b = \overline{g_1} \cap g_2$, $g_c = g_1 \cap g_2$, and $g_d = g_1 \cap \overline{g_2}$.

5 Decomposition of C-ISF

The previous section demonstrated that the decomposition sets of ISFs can be described by C-ISFs. In this section it will be shown how to decompose C-ISFs. By rewriting (4), the set of functions described by a C-ISF $S_F < \underline{F} >$ can be written as

$$\begin{array}{lcccc} S_F = & (\overline{F_1} \cap \overline{F_2} \cap \ldots \overline{F_n}) & \cup (F_1 \cap \overline{F_2} \cap \ldots \overline{F_n}) & \cup \ldots & (F_1 \cap F_2 \cap \ldots F_n) \\ = & F_{N1} & \cup F_{N2} & \cup \ldots & F_{N2^n} \end{array} \tag{6}$$

where the union runs over all 2^n possible patterns of negation of $F_1, \ldots, F_n$. Because the intersection of ISFs is an ISF, the terms in parentheses are ISFs F_{Ni}. A naive decomposition algorithm computes all these 2^n ISFs and checks their decomposability by decomposition algorithms for ISFs. This algorithm is not very efficient because even for small n the number of ISFs that must be checked becomes very large. There is no efficient and exact algorithm known that computes OR- or AND-decompositions of C-ISF. We suggest here a heuristic approach. Our greedy algorithm can be applied to AND-, OR-, and EXOR-decomposition.

Algorithm 2. Or-Decomposition of C-ISF.
Input: C-ISF $S_F < \underline{F} = (F_1, \ldots, F_n) >$, variable sets A and B
Output: OR-decomposition of S_F with respect to A and B
1. $F_R = \Phi$
2. for $i = 1$ to n
3. if $((F_R \cap F_i)$ is OR-decomposable)
4. $F_R = F_R \cap F_i$
5. else
6. if $((F_R \cap \overline{F_i}\,)$ is OR-decomposable)
7. $F_R = F_R \cap \overline{F_i}$
8. else
9. return (no decomposition found)
10. return (OR-decomposition of F_R)

The algorithm successively selects either F_i or $\overline{F_i}$ for the ISF F_R. Whether F_i or its complement is selected, depends on the decomposablility of the resulting ISF. After the termination of the loop in line 2, it holds that $F_R = \tilde{F}_1 \cap \tilde{F}_2 \cap \ldots \tilde{F}_n$, where $\tilde{F}_i$ is either F_i or $\overline{F_i}$. Because of (6) we have $F_R \in S_F$. Therefore, if a decomposition is returned in line 10, this is a valid decomposition for the C-ISF S_F. This algorithm is applicable to any binary operation for which there is a ISF decomposition algorithm. It is possible however, that S_F is decomposable, and Algorithm 2 returns "no decomposition found".

Example 12. Consider the C-ISF $S < F_A, F_B, F_C, F_D >$ from Fig. 8. The cares of F_A, F_B, F_C and F_D are labeled with letters A, B, C, and D in the Karnaugh map. The C-ISF S can be thought of an ISFs where the cares can be complemented. If a care is complemented however, all cares labeled with the same letter must be complemented. The Kmap of F in Fig. 8 gives an example where the cares with labels A and B are complemented.

Algorithm 2 searches a suitable pattern of negation so that the resulting ISF is OR-decomposable. An ISF F is OR-decomposable if every 1-care in the decomposition chart is an element of either a row with only 1s and don't cares, or an element of a column with only 1s and don't cares Φ. The algorithm starts with $F_R = \Phi \cap F_A = F_A$. ISF F_A is not OR-decomposable because the value **1** at $\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}$ is not an element of either a row, or a column with only 1s and Φ. The complement $\overline{F_A}$ shown in Fig. 8 is OR-decomposable. Therefore, $F_{R1} = \overline{F_A}$ after the first iteration of the for-loop. The second iteration checks $\overline{F_A} \cup F_B$ which is not decomposable because of the 1 at $\overline{a}\,b\overline{c}\,\overline{d}$. The intersection $\overline{F_A} \cap \overline{F_B}$ is decomposable, hence $F_{R2} = \overline{F_A} \cap \overline{F_B}$ after the second iteration. Two more iterations give $F_{R4} = \overline{F_A} \cap \overline{F_B} \cap F_C \cap F_D$ with the decomposition shown in Fig. 8.

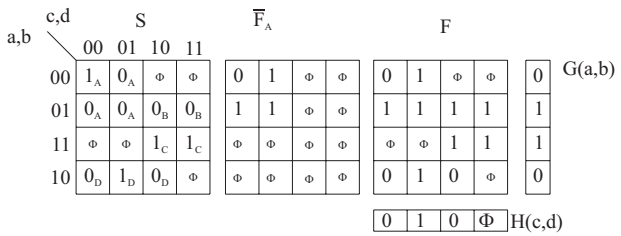

Figure 8: OR-Decomposition of a C-ISF.

Algorithm 2 does not always find a decomposition for a given decomposable C-ISF. To find all possible decompositions, we should implement a search strategy with back-tracking. Such a strategy can be very time consuming because to check ISFs for decomposability is computationally expensive, and the number n of component ISFs can become very large. Therefore, we only implemented Algorithm 2. In the case of EXOR-decomposition, properties of the EXOR-operation can be used to avoid the search among all patterns of negation in (6). For a C-ISF $S_F < F_1, \ldots, F_n >$, the EXOR-decomposition algorithm first finds the component ISFs F_{ij} for the ISFs F_i. The decomposition sets of these components can then be combined by a system of linear equations. The details can be found in [16].

6 Decomposition of Multi-Valued C-ISFs

The notion of function sets can be extended to multi-valued functions. The equivalent of EXOR-decomposition in the multi-valued case is MODSUM-decomposition, $f(X) = g(A) +_n h(B)$, where $+_n$ denotes addition modulo n, $a +_n b = (a + b) mod\, n$. The equivalents of OR- and AND-decomposition are MAX-decomposition, $f(X) = max(g(A), h(B))$, and MIN-decomposition, $f(X) = min(g(A), h(B))$, respectively. Efficient algorithms for MAX- and MIN-decomposition of multi-valued ISFs can be found in [17].

To show the main ideas of MODSUM-decomposition, the difference modulo n operator $-_n$ is defined as the inverse operation of addition modulo n: $a -_n b = c$ if and only if $a = b +_n c$. The decomposition set G of a connected multi-valued ISF F can be found as shown in Example 9. First, we compute G_0 the set of all decomposition functions with $g(\underline{x}_{a0}) = 0$ for a chosen $\underline{x}_{a0}$. Next the values of G_0 and its image H_0 are sucessively computed; by $H_0(\underline{x}_{bj}) = F(\underline{x}_{ai}, \underline{x}_{bj}) -_n G_0(\underline{x}_{ai})$ and $G_0(\underline{x}_{ar}) = F(\underline{x}_{ar}, \underline{x}_{bs}) -_n H_0(\underline{x}_{bs})$ where $F(\underline{x}_{ai}, \underline{x}_{bj})$ and $F(\underline{x}_{ar}, \underline{x}_{bs})$ are cares of F, and $G_0(\underline{x}_{ai})$ and $H_0(\underline{x}_{bs})$ are cares of G_0 and H_0 that have been computed. The identity $g \oplus h = \overline{g} \oplus \overline{h}$ translates to the identity $g +_n h = (g +_n i) +_n (h -_n i)$ for every integer i in the multi-valued case. Therefore, the decomposition set G of ISF F is the union

$$G = G_0 \cup (G_0 +_n 1) \cup (G_0 +_n 2) \cup \ldots (G_0 +_n (n-1)) = \bigcup_{i=0}^{n-1} (G_0 +_n i). \qquad (7)$$

The decomposition set of MODSUM-decomposition of ISFs can be described by the intersection of the decomposition sets of their component ISFs. Finally, a multi-valued C-ISF is

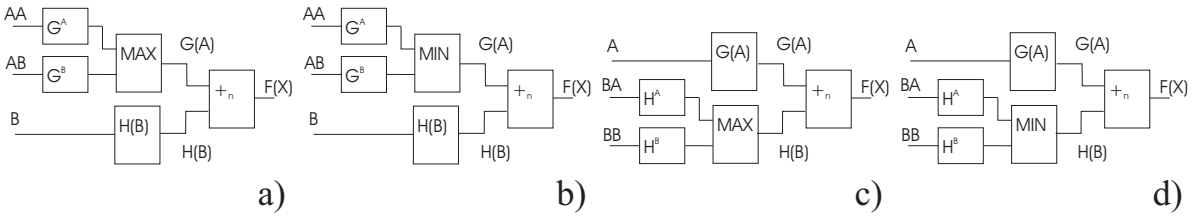

Figure 9: Two-Level decompositions used for the benchmark functions.

created.

$$S_F < F_1, \ldots, F_m >= \left(\bigcup_{i_1=0}^{n-1} (F_1 +_n i_1) \right) \cap \left(\bigcup_{i_2=0}^{n-1} (F_2 +_n i_2) \right) \cap \ldots \left(\bigcup_{i_m=0}^{n-1} (F_m +_n i_m) \right). \quad (8)$$

The extension of Algorithm 2, decomposition of C-ISF, is shown in Algorithm 3. The algorithm now checks the decomposability of the component ISFs for addition of every integer $0 \leq i < n$.

Algorithm 3. MAX-Decomposition of multi-valued C-ISF.
Input: multi-valued C-ISF $S_F < \underline{F} = (F_1, \ldots, F_m) >$, variable sets A and B
Output: MAX-decomposition of S_F with respect to A and B
1. $F_R = \Phi$
2. for $j = 1$ to m
3. for $i = 0$ to $n - 1$
3. if $(F_R \cap (F_j +_n i)$ is MAX-decomposable)
4. $F_R = F_R \cap F_i$
5. break
6. if (i==n)
9. return (no decomposition found)
10.return (MAX-decomposition of F_R)

7 Experimental Results

To verify that C-ISFs can improve the quality of function minimization, we compared decomposition using ISFs with decomposition using C-ISFs. The functions are taken from POLO directory and come from the machine-learning domain [17]. All functions have multi-valued inputs and multi-valued outputs. The benchmarks are strongly unspecified functions. We did two experiments. The first experiment used ISFs and the second experiment used C-ISFs. In both experiments, decomposition was two-level, see Fig. 9a). The first level was MODSUM-decomposition. In the first experiment we applied the algorithm from [10] producing $F(X) = G_{ISF}(A) +_n H_{ISF}(B)$, where G_{ISF} and H_{ISF} are ISFs. In the second experiment we applied Algorithm 1 producing $F(X) = G_{C-ISF}(A) +_n H_{C-ISF}(B)$, where G_{C-ISF} and H_{C-ISF}

	#inp	#inp2	R^G_{max}	R^G_{min}	R^H_{max}	R^H_{min}	time[s]
monks3tr	6	8.8	-	-	1.00	1.00	93
monks1tr	6	8.8	1.00	1.00	0.19	1.00	15
post-operative	8	11.9	-	-	1.00	1.00	168
bridges1	9	14.8	0.42	0.75	0.75	1.00	428
cloud	6	17.7	1.00	1.00	-	-	1734
sleep	9	43.4	2.89E-04	5.79E-04	1.00	1.00	2785
trains	32	64.4	1.00	1.00	1.00	1.00	881

Table 1: Comparison of DFC ratio of 2-level bi-decomposition using C-ISF or ISF.

are ISFs. The function sets G_{ISF} and G_{C-ISF} were then MAX-decomposed to $G_{ISF}(A) = \max(G^A_{ISF}(AA), G^B_{ISF}(AB))$ and $G_{C-ISF}(A) = \max(G^A_{C-ISF}(AA), G^B_{C-ISF}(AB))$ in the first and second experiment respectively. We applied the algorithm from [17] in the first experiment and Algorithm 2 in the second experiment. To compare the complexity of the solution, we compared the complexities of the subfunctions G^A_{ISF} and G^B_{ISF} with the complexities of G^A_{C-ISF} and G^B_{C-ISF} by means of the ratio

$$R^G_{max} = \frac{DFC(G^A_{C-ISF}) + DFC(G^B_{C-ISF})}{DFC(G^A_{ISF}) + DFC(G^B_{ISF})}, \tag{9}$$

with $DFC(F) = m_{out}{}^{m_{in1} * \ldots m_{inn}}$ where m_{out} is the output mulitplicity and $m_{in1}, \ldots m_{inn}$ are the input mulitplicities of F. The ratio R^G_{max} for various benchmark functions is shown in Table 1. A value of smaller than 1 indicates that a better solution has been found using C-ISFs in the decomposition process. A '-' in the table indicates the cases where neither algorithm found a decomposition. The other columns contain the results when MIN-decomposition instead of MAX-decomposition was used, see Fig. 9b) (R^G_{min}), and when function set $H(B)$ was decomposed in the second level instead of function set $G(A)$, see Figures 9c) and d) (R^H_{max} and R^H_{min}). Column (#inp) in Table 1 shows the number of input variables, column (#inp2) shows the equivalent number of binary input variables (inputs are multi-valued). The last column (time) contains the total computation time on a 90MHz Pentium PC for the decomposition of the benchmark by both methods, ISF and C-ISF decomposition.

8 The Concept of Vector Relation Sets

We believe that the introduced above concept of function sets can be further extended in several ways and that it can find many practical applications. Especially in the area of EXOR logic we often have to decompose some function to an EXOR of few functions [11, 19]. Various new classes of function sets arise from new types of applications. It can be shown also that Ashenhurst decomposition of ISFs, or the OR-decomposition of C-ISFs produce decomposition sets that cannot be described by either ISFs or C-ISFs. Curtis decomposition of ISFs, or the decomposition of relations [15] produce vector functions which

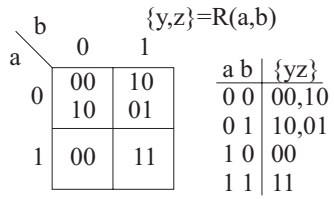

Figure 10: Example of a relation.

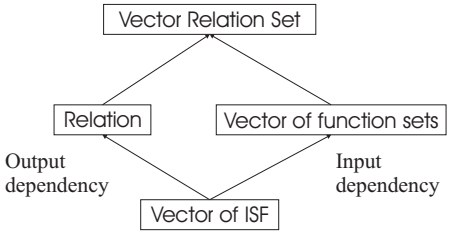

Figure 11: Hierarchy of sets of vector functions.

can be described by relations. In Example 3 we demonstrated that function sets impose dependencies between values of their element functions **for different inputs**. This behavior could not be described by ISFs. Relations extend vectors of ISFs in a similar way **for the outputs**. Consider the relation $\{y, z\} = R(a, b)$ in Fig. 10. There is a dependency of different outputs for the same input of the form $y(\overline{a}\, b) = z(\overline{a}\, b)$. Such dependencies cannot be expressed by vectors of ISFs, or vectors of function sets. A further generalization of relations and function sets leads therefore to *vector relation sets*, as shown in Fig. 11. Vector relation sets can not only express dependencies between different input values, but also dependencies between different output values. Thus they are the way of describing very general **constraints** on discrete data.

Example 13. Fig. 12 shows two vector functions $\{y, z\} = f_1(a, b)$ and $\{y, z\} = f_2(a, b)$. The vector relation set $V = \{f_1, f_2\}$ has a dependency between different outputs for different

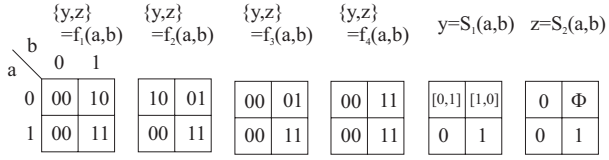

Figure 12: Functions of a vector relation set.

inputs of the form $y(\overline{a}\,\overline{b}\,) = z(\overline{a}\, b)$. There is no relation, or vector of function sets that can express this "vector relation set". For instance, consider the relation R from Fig. 10. This relation contains the functions f_1 and f_2, but also the function $f_3 \notin V$ from Fig. 12. On the other hand, the vector of function sets $[y = S_1, z = S_2]$ in Fig. 12b also contains f_1 and

f_2, but also $f_4 \notin V$. Thus the vector relation set V = $\{f_1,\ f_2\}$ can be not represented by a relation, nor can it be represented by a vector of function sets. This clearly demonstrates the natural need to introduce such a new powerful concept to the research areas of Logic Synthesis and Machine Learning.

Concluding, vectors of ISFs and C-ISFs are the result of decomposition of vectors of ISFs. Relations can be used to describe problems in machine learning and finite state machine design or are a result of function decomposition. Vector relation sets arise in decomposition of general relations as in [15]. To our knowledge, vector relation sets are the most powerful generalizations of binary and mutli-valued functions introduced so far in the literature.

9 Conclusion

The concept of function sets was presented as a generalization of ISFs. Because of the huge number of function sets created even during decomposition of functions with small numbers of variables, the algorithms to process sets of functions have to correctly exploit specific structures of the sets. We demonstrated the creation of C-ISFs, a special class of function sets, during EXOR-decomposition of an ISF. A decomposition algorithm for C-ISF was also developed. Using highly unspecified multi-valued benchmarks we showed that better decompositions can be found with algorithms that use C-ISFs instead of ISFs. These results can be useful not only for EXOR and MODSUM gates but also for any gate that has similar to them **linear properties** [18].

References

[1] R. M. Karp, "Functional Decomposition and Switching Circuit Design", *J. S.I.A.M.*, Vol. 11, pp. 291–335, 1963.

[2] H. A. Curtis, *The Design of Switching Circuits*. Van Nostrand, 1962.

[3] M. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*. McGraw-Hill, (Chapter 11), 1978.

[4] R. Brayton and F. Somenzi, "An Exact Minimizer for Boolean Relations", *Proc. of ICCAD*, pp. 316–320, 1989.

[5] D. Bochmann, F. Dresig, and B. Steinbach, "A new decomposition method for multilevel circuit design", *Proc. European Design Automation Conference*, pp. 374–377, 1991.

[6] T. Luba, J. Kalinowski, and K. Jasinski, "PLATO—a CAD tool for logic synthesis based on decomposition", *Proc. European Design Automation Conference*, pp. 65–69, 1991.

[7] T. Sasao, "FPGA design by generalized functional decomposition", in Logic Synthesis, ed. T. Sasao, Kluwer Academic Publishers, Boston, pp. 1–31, 1993.

[8] T. Q. Le, "Dekomposition und ihre Anwendung bei der Synthese mehrstufiger Schaltungen", *Proc. Workshop Boolesche Probleme*, pp. 81–89, Freiberg, 1994.

[9] M. A. Perkowski, "A new representation of strongly unspecified switching functions and its application to multi-level AND/OR/EXOR synthesis", *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion*, pp. 143–151, 1995.

[10] B. Steinbach and A. Wereszczynski "Synthesis of multi-level circuits using EXOR-gates", *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion*, pp. 161–168, 1995.

[11] M. A. Perkowski, I. Schaefer, A. Sarabi, M. Chrzanowska-Jeske, "Multi-level Logic Synthesis Based on Kronecker Decision Diagrams and Boolean Ternary Decision Diagrams for Incompletely Specified Functions," *VLSI Design* , 1995, Vol. 3., Nos. 3-4, pp. 301-313.

[12] Y-T. Lai, K-R. Pan and M. Pedram, "OBDD-based functional decomposition: algorithms and implementation", *IEEE Trans. on Computer-Aided Design*, Vol. 15, No. 8 August 1996.

[13] T. Sasao and J. T. Butler, "On Bi-Decomposition of Logic Functions", *Proc. of IWLS '97*, Tahoe City, 1997.

[14] A. Zakrevskij, "On a special kind decomposition of weakly specified Boolean functions", *Computer-Aided Design of Discrete Devices*, pp. 36–45, Minsk, 1997.

[15] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. S. Zhang, "Decomposition of Multiple-Valued Relations", *Proc. ISMVL '97*, pp. 13–18, Halifax, Nova Scotia, Canada, May 1997.

[16] B. Steinbach and Ch. Lang, "A General Data Structure for EXOR-Decomposition of Sets of Switching Functions", *Proc. 3rd Workshop on Boolean Problems*, pp. 59–66, Freiberg, 1998.

[17] B. Steinbach, M. A. Perkowski, and Ch. Lang, "Bi-Decompositions of Multi-Valued Functions for Circuit Design and Data Mining Applications", *Proc. ISMVL '99*, Freiburg, Germany, May 17-21, 1999.

[18] U. Kalay, D. Hall, and M. Perkowski, "Highly Testable Boolean Rings," *Proc. ISMVL'99,* Freiburg, Germany, May 17-21 1999.

[19] M. Perkowski, B. Falkowski, M. Chrzanowska-Jeske, and R. Drechsler, "Efficient Algorithms for Linearly Independent Decision Diagrams," *submitted.*

[20] R. Barthel, *Grundlagen einer Booleschen Signaltheorie*, Wissenschaftliche Schriftenreihe 13/1984, TH Karl-Marx-Stadt, p. 21, Karl-Marx-Stadt, 1984.