Bi-Kronecker Functional Decision Diagrams: A Novel Canonical Representation of Boolean Functions

Xuanxiang Huang, Kehang Fang, Liangda Fang,* Qingliang Chen, Zhao-Rong Lai, Linfeng Wei
Department of Computer Science, Jinan University
Guangzhou 510632, China
cshxx@stu2016.jnu.edu.cn; fkh@stu2014.jnu.edu.cn; {fangld,tpchen,laizhr,twei}@jnu.edu.cn

Abstract

In this paper, we present a novel data structure for compact representation and effective manipulations of Boolean functions, called Bi-Kronecker Functional Decision Diagrams (BKFDDs). BKFDDs integrate the classical expansions (the Shannon and Davio expansions) and their bi-versions. Thus, BKFDDs are the generalizations of existing decision diagrams: BDDs, FDDs, KFDDs and BBDDs. Interestingly, under certain conditions, it is sufficient to consider the above expansions (the classical expansions and their bi-versions). By imposing reduction and ordering rules, BKFDDs are compact and canonical forms of Boolean functions. The experimental results demonstrate that BKFDDs outperform other existing decision diagrams in terms of sizes.

Introduction

A representation of Boolean functions, which has a compact size and supports effective manipulations, plays a dominant role in the area of artificial intelligence, *e.g.*, planning (Palacios et al. 2005; Huang 2006), diagnosis (Huang and Darwiche 2005; Siddiqi and Huang 2007), and probabilistic inference (Chavira and Darwiche 2008; Fierens et al. 2015). The most notable representation is *Binary Decision Diagrams (BDDs)* (Bryant 1986) that are based on *the Shannon expansion* (Boole 1854; Shannon 1938). The Shannon expansion splits a function into two cofactors w.r.t. the variable x and its complement $\bar{x}$. Moreover, *the (positive and negative) Davio expansions*[1] are more suitable to serve as the basis of decomposing XOR-intensive functions than the Shannon one (Kebschull, Schubert, and Rosenstiel 1992). Based on the Davio expansion, *Functional Decision Diagrams (FDDs)* are proposed in (Kebschull, Schubert, and Rosenstiel 1992; 1993; Drechsler, Theobald, and Becker 1996). However, it is shown that either of BDDs or FDDs alone is inefficient for representing some classes of Boolean functions (Becker and Drechsler 1995b). To address this deficit, Drechsler and Becker (1998) proposed *Kronecker Functional Decision Diagrams (KFDDs)* that integrate the aforementioned three expansions. In both theory and practices, KFDDs turns out to be more compact than BDDs and FDDs.

The notion of expansions is the theoretical foundation of decision diagrams. A generalization to the Shannon expansion (Kerntopf 2001) has been proposed for a long time. This generalization decomposes a function into two cofactors w.r.t. an auxiliary function g and its negation $\bar{g}$. However, identifying a suitable auxiliary function for the generalized Shannon expansion is hard, which hinders it serving as the core theory of decision diagrams that enjoys effective operations. Recently, a simple fragment, called *the bi-Shannon expansion*, restricting the auxiliary function to be a single variable, has been proposed in (Amarú, Gaillardon, and Micheli 2014). Based on the above expansion, *Biconditional Binary Decision Diagrams (BBDDs)* are developed accordingly. Thanks to the simplicity of the bi-Shannon expansion, Boolean manipulations on BBDDs are efficient.

It can be observed that (1) the more expansions the decision diagram is empowered by, the more compact it is, and (2) the simpler expansions the decision diagram uses, the more effective it is. In this paper, we thus aim to design a decision diagram that fuses as many expansions as possible under the principle of adopting simple ones.

The main contributions of this paper are as follows:

1. Inspired by the idea of the bi-Shannon expansion, we propose two invariants of Davio expansion, called *the bi-positive and bi-negative Davio expansions*.
2. By combining the above two expansions together with the existing ones (Shannon, bi-Shannon, and positive and negative Davio), we develop a novel representation of Boolean functions, called *bi-Kronecker Functional Decision Diagrams (BKFDDs)*. Due to the diversity of the expansions, BKFDDs are generalizations of the existing decision diagrams: BDDs, FDDs, KFDDs and BBDDs. Interestingly, we show that, under certain conditions, only some essential expansions are necessary, *i.e.*, the other ones can indeed be reduced to one of them.
3. We prove that BKFDDs are the canonical form of Boolean functions via enforcing the ordering and reduction rules. In order to further reduce the size of BKFDDs, we provide an additional reduction rule that

*Corresponding author

[1]The Davio expansion is also called the Reed-Muller expansion (Reed 1954; Muller 1954).

compresses chain structures in decision diagrams into single nodes.

4. We also introduce the algorithms for Boolean manipulations on BKFDDs, and analyze their complexities.
5. Experiments are carried out to compare BDDs, KFDDs, BBDDs and BKFDDs on the MCNC benchmarks[2]. The empirical results show that BKFDDs outperform other decision diagrams in terms of sizes.

Preliminaries

In this section, we briefly review some background knowledge on decision diagrams.

Throughout this paper, we fix a set $\mathbf{X}$ with n variables $x_1, \cdots, x_n$. We say a Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ is *simple*, if it is a Boolean constant $\mathbf{0}$ or $\mathbf{1}$, a variable x_i or its negation $\bar{x}_i$. It is well-known that any Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ can be represented by BDDs (Bryant 1986). A BDD is a rooted directed acyclic graph with two types of nodes: internal and terminal nodes. Each terminal node denotes a constant $\mathbf{1}$ or $\mathbf{0}$. Each internal node, labeled with a variable x_i, represents a function according to the Shannon expansion:

$$f = \bar{x}_i \cdot f_{x_i=\mathbf{0}} + x_i \cdot f_{x_i=\mathbf{1}} \text{ (S)}$$

where $f_{x_i=\mathbf{0}}$ (resp. $f_{x_i=\mathbf{1}}$) is the function obtained from f by replacing x_i with $\mathbf{0}$ (resp. $\mathbf{1}$).

A Boolean function has many different BDD representations. To ensure the canonicity of BDDs, *i.e.*, every function has a unique BDD, two constraints are imposed on BDDs (Bryant 1986). Given a variable order π, a BDD is *ordered*, if each variable appears at most once on each path from the root to a terminal node, and the variables appear following the order π on all such paths. A BDD is *reduced*, if it contains neither isomorphic subgraphs nor nodes with isomorphic children. Ordered and reduced BDDs are canonical representations for Boolean functions.

The Davio expansion defined below (Kebschull, Schubert, and Rosenstiel 1992) is more suitable for XOR-intensive Boolean functions than the Shannon one.

$$f = f_{x_i=\mathbf{0}} \oplus x_i \cdot (f_{x_i=\mathbf{0}} \oplus f_{x_i=\mathbf{1}}) \text{ (pD)}$$

$$f = f_{x_i=\mathbf{1}} \oplus \bar{x}_i \cdot (f_{x_i=\mathbf{0}} \oplus f_{x_i=\mathbf{1}}) \text{ (nD)}$$

Based on the positive Davio expansion, a variant of BDDs, called FDDs, is proposed in (Kebschull, Schubert, and Rosenstiel 1992; 1993). As before, by imposing reduction and ordering rules, FDDs are canonical.

For certain classes of Boolean functions, their BDD representations are exponentially larger than their FDD ones, and vice versa (Becker and Drechsler 1995b). Drechsler and Becker (1998) proposed KFDDs that combines the above three expansions. With both characteristics of BDDs and FDDs, KFDDs are more compact and general representations for Boolean functions.

Later on, it was shown that the Shannon and Davio expansions can be generalized as follows (Kerntopf 2001):

$$f = (\bar{x}_i \oplus g) \cdot f_{x_i=g} + (x_i \oplus g) \cdot f_{x_i=\bar{g}} \text{ (gS)}$$

$$f = f_{x_i=g} \oplus (x_i \oplus g) \cdot (f_{x_i=g} \oplus f_{x_i=\bar{g}}) \text{ (gpD)}$$

$$f = f_{x_i=\bar{g}} \oplus (\bar{x}_i \oplus g) \cdot (f_{x_i=g} \oplus f_{x_i=\bar{g}}) \text{ (gnD)}$$

It is obvious that the generalized Shannon expansion degenerates into the classical one if g is substituted by $\mathbf{0}$. Nevertheless, finding a proper auxiliary Boolean function g is a hard problem. Recently, Amarú, Gaillardon, and Micheli (2014) considered the simple case where the auxiliary function is a single variable x_j, and proposed the bi-Shannon expansion:

$$f = (\bar{x}_i \oplus x_j) \cdot f_{x_i=x_j} + (x_i \oplus x_j) \cdot f_{x_i=\bar{x}_j} \text{ (bS)}$$

Based on this expansion, Amarú, Gaillardon, and Micheli (2014) developed BBDDs, and showed that BBDDs are more compact than BDDs for two famous Boolean functions: Majority and Adder.

Bi-Kronecker Functional Decision Diagrams

In this section, we introduce a novel decision diagram, called *Bi-Kronecker Functional Decision Diagram (BKFDD)*. We first extend both positive and negative Davio expansions to their bi-variants, namely bi-positive Davio (bpD) and bi-negative Davio (bnD). By integrating the six expansions: S, pD, nD, bS, bpD and bnD, we obtain the BKFDDs that are the generalizations of the existing decision diagrams: BDDs, FDDs, KFDDs, and BBDDs. Then, we show that BKFDDs are canonical under the ordering and reduction rules. Interestingly, with the help of complemented edges, it is found that the above six expansions are different only when the auxiliary function is simple. Finally, the algorithms for Boolean operations on BKFDDs are given.

The structure

Inspired by the bi-Shannon expansion, we naturally provide the bi-version of Davio expansions as follows:

$$f = f_{x_i=x_j} \oplus (x_i \oplus x_j) \cdot (f_{x_i=x_j} \oplus f_{x_i=\bar{x}_j}) \text{ (bpD)}$$

$$f = f_{x_i=\bar{x}_j} \oplus (\bar{x}_i \oplus x_j) \cdot (f_{x_i=x_j} \oplus f_{x_i=\bar{x}_j}) \text{ (bnD)}$$

For better illustration, we classify the six expansions (S, pD, nD, bS, bpD and bnD) into two groups. The first three expansions are called *classical expansions* while the latter three ones are called *bi-expansions*.

We hereafter introduce BKFDD that integrates the above six expansions. The formal definition of decision diagrams, which serves as the basis of BKFDD, is given as follows.

Definition 1. A *decision diagram (DD)* is a rooted directed acyclic graph consisting of two types of nodes: internal and terminal nodes. Each internal node v is labeled with a pair (x_i, g) where $x_i \in \mathbf{X}$ and g is a Boolean function, and has two successors: $\mathsf{low}(v)$ and $\mathsf{high}(v)$. Each terminal node v is labeled with a constant $\mathbf{1}$ or $\mathbf{0}$ and has no successor.

For an internal node v labeled with a pair (x_i, g), we call x_i the *decision variable* of v, and g the *auxiliary function* of v. We use $|D|$ for the size of a decision diagram D, *i.e.*, the number of nodes in D. The definition of DDs in this paper is slightly different from Definition 1 in (Drechsler and Becker 1998), since each internal node in our definition contains not

[2] https://ddd.fit.cvut.cz/prj/Benchmarks/

only the decision variable but also the auxiliary function. As mentioned before, it is difficult to identify a suitable auxiliary function for internal nodes. In this paper, the choice of auxiliary functions is relatively easy since we only consider the classical expansions and bi-expansions. Hence, the auxiliary function g is either a Boolean constant $\mathbf{0}$ or a distinct variable x_j (*i.e.*, $i \neq j$). We remark that it is not necessary to consider the cases where g is $\mathbf{1}$ or $\bar{x}_j$ as they are redundant which will be explained later.

Just as BDDs and FDDs are characterized by a variable order, BKFDDs are characterized by a variable order augmented with expansion types (OET).

Definition 2. A *variable order with expansion types (OET)* π is a sequence of n pairs (x, e) where $x \in \mathbf{X}$ and $e \in \{S, pD, nD, bS, bpD, bnD\}$ such that the variables of any two distinct pairs are different.

For a pair (x, e), x is called the variable of the pair, and e is called the expansion type. For an OET π, we use π^x for the variable order following π, π_i for the i-th element of π, π_i^x for the variable of π_i, and π_i^e for the expansion type of π_i. For example, suppose that π is an OET $[(x_2, pD), (x_3, bS), (x_1, bnD), (x_4, bS)]$. Following the OET π, we get that π^x is a variable order $[x_2, x_3, x_1, x_4]$, $\pi_3 = (x_1, bnD)$, $\pi_3^x = x_1$, and $\pi_3^e = bnD$.

To determine the Boolean function to which a DD corresponds, it is necessary to identify the auxiliary function and expansion type of each internal node. To achieve this, we present the *respecting relation* between DDs and OETs.

Definition 3. We say a DD G *respects* an OET π, if the following conditions hold for each internal node v labeled with (π_i^x, g),

1. $g = \mathbf{0}$ when $\pi_i^e \in \{S, pD, nD\}$ or $i = n$;
2. $g = \pi_{i+1}^x$ when $\pi_i^e \in \{bS, bpD, bnD\}$ and $i \neq n$.

For an arbitrary internal node v whose decision variable is π_i^x, if v is associated with one of the classical expansions, then its auxiliary function is $\mathbf{0}$. If the expansion type of v is a bi-expansion, and $i < n$, then its auxiliary function is π_{i+1}^x. We note that if the decision variable of v is the last element of π^x, then the node associated with a bi-expansion degenerates into a node associated with the corresponding classical expansion, and thus its auxiliary function becomes $\mathbf{0}$.

Example 1. An OET $\pi : [(x_1, bS), (x_2, bS), (x_3, pD), (x_4, S)]$ is given in Figure 1. Thus, $\pi^x = [x_1, x_2, x_3, x_4]$ and $\pi_1 = (x_1, bS)$. In Figure 1(a), the decision variable of node 1 is x_1. It follows that the auxiliary function of node 1 is the variable x_2. Similarly, the auxiliary function of node 4 is $\mathbf{0}$.

We say a DD G respecting an OET π is a BKFDD G_π. Then we are ready to give the correspondence between BKFDDs and Boolean functions.

Definition 4. Let G be a BKFDD respecting an OET π. The Boolean function $f_G : \mathbf{B}^n \to \mathbf{B}$ represented by G is defined as follows.

1. If G consists of only one terminal node labeled with $\mathbf{1}$ (resp. $\mathbf{0}$), then $f_G = \mathbf{1}$ (resp. $\mathbf{0}$).

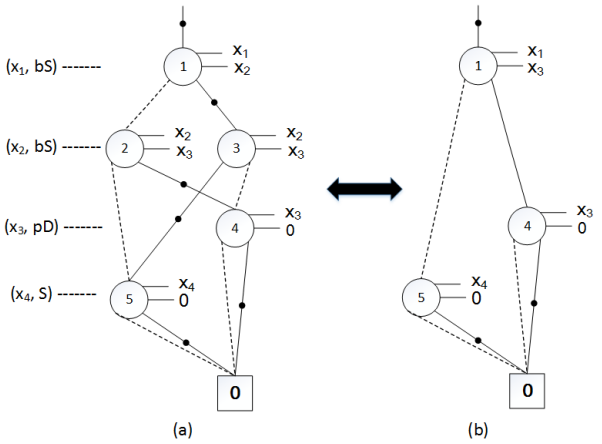

Figure 1: Graphical depiction of the weak ROBKFDD (a) and strong ROBKFDD (b) for the function $f = \bar{x}_1 \cdot x_3 + (\bar{x}_1 \oplus x_3) \cdot \bar{x}_4$. The dashed line denotes the low-edge and the solid line denotes the high-edge. The dotted line denotes the complemented edge.

2. If G is associated with the root node v labeled with (x, g), then
$$f_G = \begin{cases} (\bar{x} \oplus g) f_{\mathsf{low}(v)} + (x \oplus g) f_{\mathsf{high}(v)} & \text{if } e \in \{S, bS\} \\ f_{\mathsf{low}(v)} \oplus (x \oplus g) f_{\mathsf{high}(v)} & \text{if } e \in \{pD, bpD\} \\ f_{\mathsf{low}(v)} \oplus (\bar{x} \oplus g) f_{\mathsf{high}(v)} & \text{if } e \in \{nD, bnD\} \end{cases}$$
where $(x, e) \in \pi$, $f_{\mathsf{low}(v)}$ and $f_{\mathsf{high}(v)}$ are the Boolean functions represented by the BKFDD rooted by $\mathsf{low}(v)$ and $\mathsf{high}(v)$, respectively.

We hereafter introduce the ordering rule for BKFDDs.

Definition 5. Given an OET π, we say a BKFDD is an *ordered* BKFDD (OBKFDD), if for each path from the root to a terminal node, we have

1. For any two distinct internal nodes on the path, their decision variables are different.
2. The decision variables of internal nodes on such path appear following π^x.
3. For any internal node v on the path, if its auxiliary function is a variable π_i^x and its successors are internal nodes, then the decision variables of successors of v is π_j^x where $j \geq i$.

We remark that the constant $\mathbf{0}$ is considered as a variable that is the last one following any OET, since auxiliary functions of some nodes of bi-expansion types are $\mathbf{0}$.

For example, in Figure 1(a), three internal nodes labeled with 1, 2 and 5 are in a path from the root node to the terminal node $\mathbf{0}$. The decision variables of them are x_1, x_2 and x_4 respectively. It is easy to observe that their decision variables are different, and appear following the order $\pi^x : [x_1, x_2, x_3, x_4]$. In addition, the auxiliary functions of node 1 is x_2, and node 2 is a successor of node 1. Thus, the decision variable of node 2 is x_2.

A node labeled with (π_i^x, g) in a BKFDD G_π is called a π_i^e-node. For example, a node in a BKFDD is called an S-node, if it is interpreted by the Shannon expansion, *e.g.*,

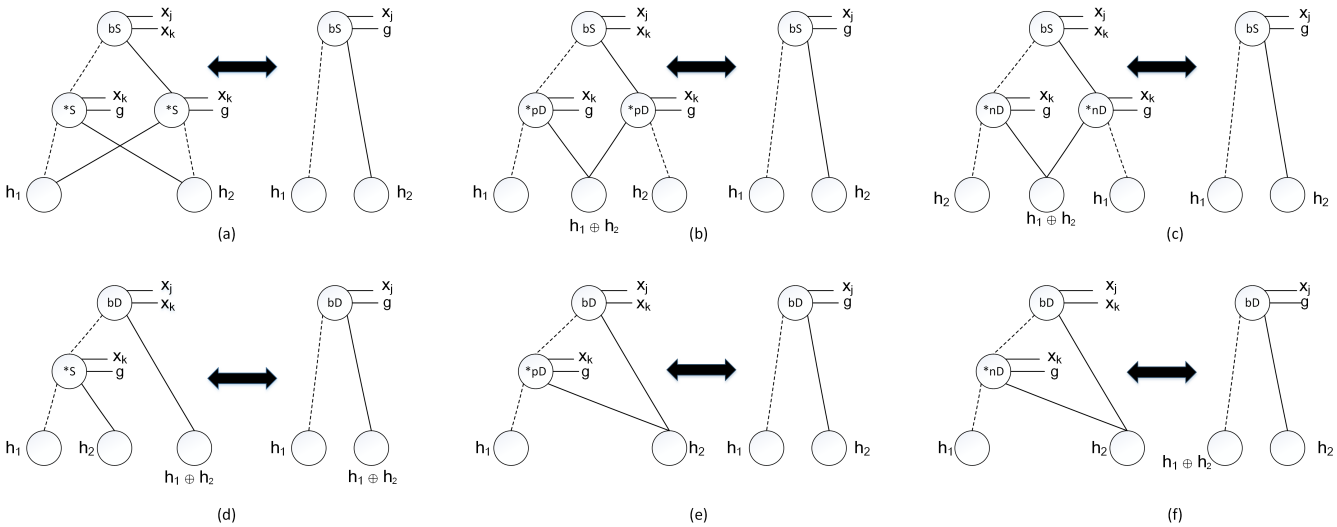

Figure 2: The rule **RC** where a node labeled with "*S" is an S- or bS-node, a node labeled with "bD" is a bpD- or bnD-node, a node labeled with "*pD" is a pD- or bpD-node, and a node labeled with "*nD" means this node is a nD- or bnD-node.

node 5 in Figure 1(a) is an S-node. Furthermore, it is easily verified that BKFDDs are generalizations of the four DDs: BDDs, FDDs, KFDDs, and BBDDs. For example, BKFDDs degenerates into KFDDs by only considering classical expansions.

Reduction rules and canonicity

In order to compactly represent Boolean functions, BKFDDs should be compressed to another one denoting the same function according to a set of reduction rules. In the following, we propose reduction rules for BKFDDs that are inherited from KFDDs (Drechsler and Becker 1998), and obtain the canonicity theorem for OBKFDDs.

RI Eliminate a node v isomorphic to a distinct node v', and redirect all incoming edges of v to v'.

RS Eliminate a node v whose two successors are v', and redirect all incoming edges of v to v'.

RD Eliminate a node v such that the successor $\mathsf{high}(v)$ points to the terminal node **0**, and redirect all incoming edges of v to $\mathsf{low}(v)$.

The rule **RI** can be applied in any node of OBKFDDs. However, the rule **RS** is applied only in S- and bS-nodes while the rule **RD** is applied only in pD-, nD-, bpD- and bnD-nodes. We say an OBKFDD is a *weak reduced* OBKFDD (weak ROBKFDD), if none of the rules **RI**, **RS** and **RD** can be applied in it.

By repeated applications of the above three rules, we obtain a canonical representation for Boolean functions.

Theorem 1. *For any Boolean function f and OET π, there is exactly one weak ROBKFDD such that it respects π and represents f.*

Thanks to the simplicity of auxiliary functions and bi-expansions, BKFDDs can be further compressed by new reduction rules. Let us illustrate it in the following example.

Example 2. Let $f = \bar{x}_1 \cdot x_3 + (\bar{x}_1 \oplus x_3) \cdot \bar{x}_4$. Figure 1(a) depicts a weak ROBKFDD D representing the function f with 6 nodes. Interestingly, the nodes on the second level can be eliminated according to the following equation:

$$\begin{aligned} f=&(\bar{x}_1 \oplus x_3) \cdot \bar{x}_4 + (x_1 \oplus x_3) \cdot x_3 \\ =&[(\bar{x}_1 \oplus x_2) \oplus (x_2 \oplus x_3)] \cdot \bar{x}_4 + [(x_1 \oplus x_2) \oplus (x_2 \oplus x_3)] \cdot x_3 \\ =&(\bar{x}_1 \oplus x_2) \cdot (\bar{x}_2 \oplus x_3) \cdot \bar{x}_4 + (\bar{x}_1 \oplus x_2) \cdot (x_2 \oplus x_3) \cdot x_3 + \\ & (x_1 \oplus x_2) \cdot (\bar{x}_2 \oplus x_3) \cdot x_3 + (x_1 \oplus x_2) \cdot (x_2 \oplus x_3) \cdot \bar{x}_4 \end{aligned}$$

The compression process is as follows:

1. Replace the auxiliary function of the root node with x_3;
2. Eliminate nodes 2 and 3;
3. Redirect the low (resp. high) edge of the root node to node 4 (resp. 5).

The new BKFDD D', shown in Figure 1(b), contains only 4 nodes, and thus its size is less than $|D|$.

Motivated by Example 2, we propose a new reduction rule, called *chain reduction* (**RC**). We present the chain reduction by six cases in Figure 2. Due to the space limit, we only elaborate the detailed operations of the cases shown in Figure 2(a) and 2(d). The operations for other cases can be constructed according to other subfigures in Figure 2.

In Figure 2(a), the root node v is a bS-node labeled with (x_j, x_k). Its successor $\mathsf{low}(v)$ and $\mathsf{high}(v)$ are S- or bS-nodes whose decision variable is x_k. The low successor of $\mathsf{low}(v)$, representing the function h_1, is identical to the high successor of $\mathsf{high}(v)$; and the high successor of $\mathsf{low}(v)$, representing the function h_2, is the same as the low successor of $\mathsf{high}(v)$. The node v denotes the function f as follows:

$$\begin{aligned} f=&[(\bar{x}_j \oplus x_k) \cdot (\bar{x}_k \oplus g)] \cdot h_1 + [(\bar{x}_j \oplus x_k) \cdot (x_k \oplus g)] \cdot h_2 + \\ & [(x_j \oplus x_k) \cdot (\bar{x}_k \oplus g)] \cdot h_2 + [(x_j \oplus x_k) \cdot (x_k \oplus g)] \cdot h_1 \\ =&(\bar{x}_j \oplus g) \cdot h_1 + (x_j \oplus g) \cdot h_2 \end{aligned}$$

So the the two nodes labeled with (x_k, g) can be eliminated. The compression process consists of replacing the

auxiliary function of the node v with g, and redirecting the low (resp. high) edge of v to the node $\mathsf{low}(\mathsf{low}(v))$ (resp. $\mathsf{high}(\mathsf{low}(v))$).

In Figure 2(d), the root node v is a bpD- or bnD-node. Its successor $\mathsf{low}(v)$ are S- or bS-nodes whose decision variable is x_k. The function denoted by $\mathsf{high}(v)$ is equal to the exclusive disjunction of the two functions h_1 and h_2 represented by $\mathsf{low}(\mathsf{low}(v))$ and $\mathsf{high}(\mathsf{low}(v))$ respectively. Suppose that v is a bpD-node. The node v denotes the function f as follows:

$$\begin{aligned} f &= [(\bar{x}_k \oplus g) \cdot h_1 + (x_k \oplus g) \cdot h_2] \oplus (x_j \oplus x_k) \cdot (h_1 \oplus h_2) \\ &= [h_1 \oplus (x_k \oplus g) \cdot (h_1 \oplus h_2)] \oplus (x_j \oplus x_k) \cdot (h_1 \oplus h_2) \\ &= h_1 \oplus (x_j \oplus g) \cdot (h_1 \oplus h_2) \end{aligned}$$

Similar to Figure 2(a), the node labeled with (x_k, g) can be removed. The compression process for this case is similar to that in Figure 2(a) except we only redirect the low edge of v to the node $\mathsf{low}(\mathsf{low}(v))$.

We note that the new DD does not respect the original OET π after applying the rule **RC**. When the expansion type of the node v is a bi-expansion, the auxiliary function is not the variable next to its decision variable. For example, the auxiliary function of node 1 in Figure 1(b) is x_3 but not x_2. We therefore weaken the respecting relation between DDs and OETs.

Definition 6. We say a DD G *semi-respects* an OET π, if the following conditions hold for each internal node labeled with (π_i^x, g),

1. $g = \mathbf{0}$ when $\pi_i^e \in \{S, pD, nD\}$;
2. $g = \pi_j^x$ or $g = \mathbf{0}$ where $i < j \leq n$ when $\pi_i^e \in \{bS, bpD, bnD\}$.

The difference between the *semi-respecting* relation and the respecting relation is that the former permits the auxiliary function of v to be the variable that is after its decision variable w.r.t. π^x, if the internal node v is interpreted by a bi-expansion.

We say an OBKFDD is a *strong reduced* OBKFDD (strong ROBKFDD), if none of rules **RI**, **RS**, **RD** and **RC** can be applied in it. By the four reduction rules, we obtain another canonical representation for Boolean functions.

Theorem 2. *For any Boolean function f and OET π, there is exactly one strong ROBKFDD such that it semi-respects π and represents f.*

Complemented edges

Complemented edge (Brace, Rudell, and Bryant 1990) is an improvement of the implementation for DDs that can be used for further reducing the size of DDs. A node can represent a function and its negation simultaneously via complemented edges. Suppose that the node v denotes a function f. This node with a complemented edge denotes the function $\bar{f}$. It can be proven that the ROBKFDD representation with complemented edges remains canonical if complemented edges only appear at the high edge of each internal node, and only one terminal node $\mathbf{0}$ is available.

Theorem 3. *Weak and strong ROBKFDDs with complemented edges are canonical.*

Algorithm 1: $f \oplus h$

Input : Two BKFDD nodes f and h
Output: The BKFDD node r for $f \oplus h$

```
if f = 0 or h = 0 or f = h or f = h̄ then
    return pre-defined result
else
    i ← the top level in the OET π for f and h
    g ← the auxiliary function of the i-level according to π
    f' ← the π_i^e-node labeled with (π_i^x, g) equivalent to f
    h' ← the π_i^e-node labeled with (π_i^x, g) equivalent to h
    The label of r ← (π_i^x, g)
    low(r) ← low(f') ⊕ low(h')
    high(r) ← high(f') ⊕ high(h')
    if π_i^e ∈ {S, bS} and low(r) = high(r) or
       π_i^e ∈ {pD, nD, bpD, bnD} and high(r) = 0 then
        r ← low(r)
    if the rule RC can be applied on r then
        Apply the rule RC on r
    return r
```

In the following, we consider the question: how many expansion types do we need when incorporating complemented edges and requiring only simple auxiliary functions? The answer is six, and the six expansion types are the Shannon and (positive and negative) Davio expansion as well as their bi-versions as we show next.

The *dependent set* of a Boolean function f, written $\mathsf{dep}(f)$, is the set of variables on which f depends (*i.e.*, $\{x \mid f_{x=\mathbf{0}} \neq f_{x=\mathbf{1}}\}$). An expansion type can be considered as a ternary Boolean operator.

Definition 7. An expansion type $op \in \mathbf{B}_3$ is a ternary operator s.t. for all functions f, g, and all variables x_i s.t. $x_i \notin \mathsf{dep}(g)$, there are unique functions h_1, h_2 s.t. $x_i \notin \mathsf{dep}(h_1) \cup \mathsf{dep}(h_2)$ and $f = op((x_i \oplus g), h_1, h_2)$.

By adjusting the proof in (Becker and Drechsler 1995a), we show that for a fixed function g, it is enough to consider three expansion types: gS, gpD, and gnD when incorporating complemented edges. The simple auxiliary function $\mathbf{1}$ and $\bar{x}_j$ are superfluous. This is because the gS expansion with the auxiliary function g is equivalent to that with $\bar{g}$, and the gpD expansion with g is equivalent to the gnD expansion with $\bar{g}$ and vice versa. For example, according to the equation (gS), if $g = \mathbf{1}$, then

$$\begin{aligned} f &= (\bar{x}_i \oplus \mathbf{1}) \cdot f_{x_i=\mathbf{1}} + (x_i \oplus \mathbf{1}) \cdot f_{x_i=\mathbf{0}} \\ &= (\bar{x}_i \oplus \mathbf{0}) \cdot f_{x_i=\mathbf{0}} + (x_i \oplus \mathbf{0}) \cdot f_{x_i=\mathbf{1}} \end{aligned}$$

The auxiliary function of the latter equation is $\mathbf{0}$.

We therefore can draw a conclusion that it is sufficient to only consider the six expansions (S, pD, nD, bS, bpD and bnD) for simple auxiliary functions.

Theorem 4. *Suppose that the auxiliary function is simple. There are only six non-equivalent expansion types (S, pD, nD, bS, bpD and bnD) for decision diagrams with complemented edges.*

Boolean operations on BKFDDs

We now provide three Boolean operations: exclusive disjunction (XOR), conjunction (AND), and negation (NOT) on BKFDDs that serve as the basis for other Boolean operations.

For the sake of simplicity, we do not differentiate Boolean functions and nodes in BKFDDs. We assume that BKFDDs are ordered and strong reduced. Suppose that two Boolean functions f and h are labeled with (x_i, g) and expanded by the S- or bS-expansion. The exclusive disjunction of f and h can be obtained by the following equation:

$$f \oplus h = (\bar{x}_i \oplus g) \cdot (f_{x_i=g} \oplus h_{x_i=g}) \oplus (x_i \oplus g) \cdot (f_{x_i=\bar{g}} \oplus h_{x_i=\bar{g}}) \quad (1)$$

This equation is the key to implement a recursive algorithm for the XOR operation. According to Equation 1, generating a node representing $f \oplus h$ reduces to computing the exclusive disjunction of $f_{x_i=g}$ and $h_{x_i=g}$, and that of $f_{x_i=\bar{g}}$ and $h_{x_i=\bar{g}}$. The following equation is given for pD- or bpD-nodes.

$$f \oplus h = (f_{x_i=g} \oplus h_{x_i=g}) \oplus (x_i \oplus g) \cdot (f_{x_i \oplus g} \oplus h_{x_i \oplus g}) \quad (2)$$

where $f_{x_i \oplus g}$ denotes the function $f_{x_i=g} \oplus f_{x_i=\bar{g}}$.

The algorithm of the XOR-operation for nD- and bnD nodes is performed similarly. Algorithm 1 provides a sketch of the code for generating a BKFDD node representing $f \oplus h$. Lines 1 and 2 handle the base cases where (1) f or h is the constant $\mathbf{0}$; (2) f is identical to h or its negation $\bar{h}$. In these cases, the resulting node r can be computed directly. For example, if $f = \mathbf{0}$, then the result should be h. When it is not the base cases, the algorithm will enter the inductive cases (Lines 4 - 15). Firstly, the decision variables or auxiliary functions of f and h may be different, and Equations 1 and 2 do not work on this case. To address this deficit, Lines 4 - 7 create two nodes f' and h' with the same label denoting the Boolean function f and h respectively via reversely applying reduction rules **RS**, **RD** and **RC** if necessary. For example, suppose that π is an OET, and f and h are two S-nodes labeled with $(x_i, \mathbf{0})$ and $(x_j, \mathbf{0})$ where x_i is before x_j according to the OET π. The top level for f and h is i, and the auxiliary function g is $\mathbf{0}$. The node h' will be created where the label of h' is $(x_i, \mathbf{0})$, and both its low and high successors are h. The information of the resulting node r (including the decision variable, auxiliary function, and successors) are assigned in Lines 8 - 10. The resulting node may violate the reduction rules **RS**, **RD** and **RC**. However, they will be enforced in Lines 11 - 14. Finally, the time complexity of Algorithm 1 is $O(|f| \cdot |h|)$.

For S- and bS-nodes, the AND-operation can be executed as the XOR-operation does. The computation of the AND-operation is a bit complicated for pD-, nD-, bpD- and bnD-nodes. The following equation holds for pD- and bpD-nodes, and the equation for nD- and bnD-nodes can be similarly obtained.

$$\begin{aligned} f \cdot h = {} & (f_{x_i=g} \cdot h_{x_i=g}) \oplus \\ & (x_i \oplus g) \cdot [(f_{x_i \oplus g} \cdot h_{x_i \oplus g}) \oplus (f_{x_i=g} \cdot h_{x_i \oplus g}) \oplus (f_{x_i \oplus g} \cdot h_{x_i=g})] \end{aligned} \quad (3)$$

A recursive algorithm for the AND-operation can be developed similar to Algorithm 1, with exponential running time in the worst case. Fortunately, if the number of pD-, nD-, bpD- and bnD-expansions of the OET is bounded, then the runtime becomes polynomial.

Due to complemented edges, the negation of a function can be computed via inverting the complemented edge on the node denoting the function. It follows that the negation operation can be executed in constant time.

Any Boolean operation can be realized via AND, XOR and NOT operations. For example, the disjunction (OR) of f and g can be achieved by first computing negations of f and g, then conjoining them, and finally generating the negation of the conjunction, *i.e.*, $f + g = \overline{(\bar{f} \cdot \bar{g})}$.

Experimental Results

In this section, we compare four decision diagrams BDDs, KFDDs, BBDDs, and BKFDDs on the MCNC benchmark in terms of sizes and running time. We implement a package that integrates BKFDDs as well as BDDs and KFDDs using C programming language. In the implementation, we store the decision variables, the auxiliary functions, the pointers to both low and high, and the flag denoting whether the edge points to high successors is complemented for each node. The nodes with the same decision variable share the same expansion type. Hence, we only store an OET as a global variable, and it is sufficient to determine the expansion type of each node via the OET. The package for BBDDs we use are originated from (Amarú, Gaillardon, and Micheli 2014)[3]. We compile each test case of the benchmark into different decision diagrams. The compilation process starts with the initial variable order declared in the file of the test case, and all expansion types in the initial OET are the classical Shannon expansion. The reordering algorithm for all decision diagrams is based on the original sifting algorithm (Rudell 1993). For KFDDs, we use the DTL-sifting algorithm proposed in (Drechsler and Becker 1998) which searches not only a better variable order but also a better type for each level. The dynamic reordering used in BKFDDs is a bit different from the DTL-sifting algorithm. The former considers not only the classical Shannon and Davio expansions but also the bi-versions of expansions. Hence, six different expansions will be introduced during the reordering algorithm. For most test cases, the compilation process does not trigger the dynamic reordering algorithm since the size of intermediate decision diagrams are too small. Therefore, we invoke one additional call to the dynamic reordering algorithm for the compiled decision diagrams after the compilation process. In addition, it is difficult to implement the variable swapping algorithm for the strong reduced OBKFDDs, which is the basis of the dynamic reordering algorithm. Thus, intermediate BKFDDs, generated during the compilation process, are all weak reduced, and finally we apply the chain reduction rule on the compiled weak reduced BKFDDs. We also utilize the ABC tools[4], and CUDD[5] to verify the correctness of the compiled decision diagrams. The machine running the benchmark is

[3] https://lsi.epfl.ch/BBDD

[4] https://people.eecs.berkeley.edu/~alanmi/abc/

[5] http://vlsi.colorado.edu/~fabio/

Table 1: Experimental results for BDDs, KFDDs, BBDDs and BKFDDs. BBDD failed to compile C2670 since it is out of the memory (24GB).

Bench.	BDD		KFDD		BBDD		BKFDD		Best	
	size	time	size	time	size	time	size	time	size	time
C1355	29562	**6.92**	17897	487.48	299669	16606.56	**13862**	461.28	**13862**	**6.92**
C2670	6182	**64.8**	**2373**	126.75	-	-	4810	31417.46	**2373**	**64.8**
C432	**1252**	**0.24**	**1252**	0.84	16517	17.37	**1252**	11.25	**1252**	**0.24**
C499	29562	**7.73**	17897	535.25	299669	16807.38	**13862**	460.93	**13862**	**7.73**
C1908	7560	**1.58**	5166	4.29	26085	24.21	**4482**	166.86	**4482**	**1.58**
C5315	2515	**20.58**	**2113**	82.66	30363	924.01	2115	12277.25	**2113**	**20.58**
C880	12178	**5.14**	9627	16.4	76946	7098.17	**5731**	674.58	**5731**	**5.14**
C3540	187494	81.68	62236	**81.07**	324549	10170.63	**33398**	3376.77	**33398**	**81.07**
misex3	**604**	**0.09**	632	0.37	739	1.58	632	1.26	**604**	**0.09**
too_large	819	**1.52**	**381**	5.6	1235	26.63	542	135.04	**381**	**1.52**
amd	283	**0.02**	228	0.09	353	0.72	**225**	1.14	**225**	**0.02**
apex2	783	**0.44**	**551**	1.45	861	2.09	555	34	**551**	**0.44**
apex3	899	**1.22**	**845**	5.35	3704	8.26	858	170.42	**845**	**1.22**
apex6	657	**1.83**	**632**	8.75	10750	170.91	644	949.79	**632**	**1.83**
b2	594	**0.1**	560	0.28	841	2.37	**546**	5.75	**546**	**0.1**
bc0	536	**0.1**	441	0.64	1257	2.29	**437**	7.05	**437**	**0.1**
chkn	**319**	**0.14**	**319**	0.81	566	0.74	324	5.38	**319**	**0.14**
dalu	841	**2.59**	619	6.92	9553	169.58	**509**	290.25	**509**	**2.59**
des	3138	**32.36**	3033	128	95505	4477.4	**2477**	42904.04	**2477**	**32.36**
dist	121	**0.03**	121	0.15	182	0.64	**119**	0.31	**119**	**0.03**
duke2	354	**0.07**	**342**	0.28	712	1.14	370	3.59	**342**	**0.07**
e64	728	**0.61**	728	2.81	**208**	1.71	728	64.98	**208**	**0.61**
ex5	242	**0.05**	**235**	0.13	374	0.9	**235**	0.5	**235**	**0.05**
frg2	1471	**4.41**	**1294**	20.04	13014	167.18	1307	1452.77	**1294**	**4.41**
gary	**302**	**0.05**	319	0.31	435	0.7	317	3.75	**302**	**0.05**
in0	**302**	**0.05**	319	0.31	435	0.85	317	3.68	**302**	**0.05**
in1	594	**0.16**	560	0.28	841	2.4	**546**	5.73	**546**	**0.16**
in2	299	**0.08**	205	0.43	488	0.79	**197**	4.78	**197**	**0.08**
in3	309	**0.15**	292	0.92	1205	2.19	**290**	17.16	**290**	**0.15**
in4	559	**0.14**	**504**	0.65	1185	1.95	539	10.7	**504**	**0.14**
intb	693	**0.07**	656	0.34	848	1.5	**585**	2.3	**585**	**0.07**
jbp	436	**0.14**	355	0.98	927	2	**343**	15.94	**343**	**0.14**
m3	**130**	**0.03**	**130**	0.15	186	0.43	**130**	0.27	**130**	**0.03**
m4	**174**	**0.07**	**174**	0.15	236	0.8	**174**	0.28	**174**	**0.07**
mainpla	2398	**0.23**	**1890**	0.83	3019	12.11	2022	16.73	**1890**	**0.23**
max1024	244	**0.04**	244	0.2	**141**	1.1	247	0.62	**141**	**0.04**
max512	**145**	**0.04**	**145**	0.18	166	0.68	**145**	0.36	**145**	**0.04**
mlp4	135	**0.04**	112	0.14	187	0.55	**109**	0.34	**109**	**0.04**
prom1	1782	**0.09**	1782	0.22	1840	9.76	**1780**	0.84	**1780**	**0.09**
prom2	834	**0.07**	834	0.19	958	4.24	**833**	0.54	**833**	**0.07**
seq	2118	**0.53**	1128	1.63	2837	9.48	**983**	64.32	**983**	**0.53**
soar	601	**0.69**	559	3.8	1553	4	**553**	166.67	**553**	**0.69**
t481	21	**0.06**	**19**	0.34	51	1.63	**19**	1.12	**19**	**0.06**
table3	769	**0.07**	**748**	0.32	892	2.49	766	1.91	**748**	**0.07**
table5	724	**0.08**	681	0.42	1080	2.31	**668**	3.53	**668**	**0.08**
vda	489	**0.13**	466	0.79	774	1.29	**448**	4.9	**448**	**0.13**
x3	657	**1.83**	**632**	8.71	10921	159.7	644	949.38	**632**	**1.83**
x4	577	**0.97**	**495**	4.73	3134	30.15	520	168.01	**495**	**0.97**
x6dn	243	**0.2**	**239**	1.18	857	1.47	**239**	15.23	**239**	**0.2**
x7dn	524	**0.48**	581	2.65	5330	24.7	**456**	68.27	**456**	**0.48**
Average	6095	**4.8**	2871.8	30.95	25595.5	1162.5	**2077.8**	1928	**2006.2**	**4.8**

equipped with an Intel Core i7-8086K 4GHz CPU and 32GB RAM.

Table 1 shows the experimental results for the four decision diagrams. The name of test cases are reported in the first four columns. The columns "BDD", "KFDD", "BBDD" and "BKFDD" denote the results of the corresponding decision diagrams respectively. The columns "size" denote the number of nodes of DDs and "time" the total runtime (sec.). The last column "best" is the best result among the four decision diagrams. And the best results are shown as bolded text.

As we can see in Table 1, adding more expansion types leads to a more compact decision diagrams. Since KFDDs employ two more expansion types (positive and negative Davio expansions) than BDDs, the average size of KFDDs is about 47.18% of that of BDDs. Furthermore, thanks to the integration of the classical expansions and their bi-versions, BKFDDs have average 27.65%, 65.91% and 91.88% smaller size than those of KFDDs, BDDs and BBDDs. We also observe that BKFDDs outperform the other three decision diagrams in 31 test cases (out of 50 test cases). In particular, for C3540, the size of BKFDD is 33398, which is much less than those of BDD, KFDD and BBDD that are 187494, 62236 and 324549, respectively. On the other side, the reordering algorithm we use finds the best OET for a given test case of benchmarks such that the size of the BKFDD is the smallest. This is because it is inherently a heuristic approach. This explains that the sizes of the corresponding BKFDDs are not the best in the rest 19 cases. However, there are 15 cases where its sizes are close to the best results (the difference is below 7%). Compared to the best results among the four decision diagrams, the sizes of BKFDDs are a bit larger by 3.4%. For some test cases, the sizes of BKFDDs are close to the best result even if BKFDDs are not the best one.

Currently, the compilation time of BKFDDs is 402, 62 and 1.6 times than that of BDDs, KFDDs and BBDDs on average. This is because that our implementation of BKFDDs package is still not optimized, and that the reordering algorithm is time consuming. In the future, we will improve our package with some existing optimization techniques, and develop more efficient reordering algorithms for BKFDDs.

Conclusions

We have proposed a new canonical representation BKFDDs for Boolean functions. Thanks to the combination of the Shannon, positive and negative Davio expansions as well as their bi-versions, BKFDDs can embrace the advantages of existing well-known Boolean representations: BDDs, FDDs, KFDDs, and BBDDs, and thus are the generalizations of them. By incorporating complemented edges and imposing the simple auxiliary function conditions, we have also proved that the above six expansions are sufficient enough. Furthermore, we have provided the algorithms for Boolean operations on BKFDDs. Moreover, we have presented the new rule **RC** that can further reduce the size of BKFDDs. Finally, the experimental results have justified that BKFDD is a more compact representation of Boolean functions than the existing DDs.

Acknowledgments

We are grateful to Biqing Fang for his help on the paper. This work was supported by the Natural Science Foundation of China (Nos. 61603152, 61463044, 61472369, 61572234, 61703182, 61772232 and 61773179) and Guangxi Key Laboratory of Trusted Software project (Nos. kx201604 and kx201606). Liangda Fang is also affiliated to Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China.

Supplemental Proofs

Proof of Theorem 1

Proof. Without of generality, we assume that the variable order is $x_1, \cdots, x_n$. Let i be the least number s.t. $x_i \in \mathsf{dep}(f)$. We remark that $\mathsf{dep}(f) = \emptyset$ if f is the constant function $\mathbf{0}$ or $\mathbf{1}$. In this case, we assume that $i = n + 1$. We prove by induction on $n + 1 - i$.

Base case ($i = n+1$): Suppose that $f = \mathbf{0}$. It can be seen that, in weak ROBKFDD, only terminal node labeled with 0 denotes the function f. The case where $f = \mathbf{1}$ is similar.

Inductive case: Suppose that G_1 and G_2 are two weak ROBKFDDs representing the function f. Let v_1 and v_2 be the root nodes of G_1 and G_2 respectively. Assume that the expansion types of v_1 and v_2 are the Shannon expansion.

We first verify that v_1 and v_2 are labeled with (x_i, g). This can be done by contradiction. If the node v_1 is labeled with (x_j, g) where $j < i$ or $j > i$. Suppose that $j < i$. The low and high successors of v_1 represent the function f, and hence they are identical. This violates the rule **RI**. Suppose that $j > i$. Then, there is no node in G_1 s.t. its decision variable is x_i. This contradicts the fact that $x_i \in \mathsf{dep}(f)$.

We now prove that G_1 and G_2 are identical. It is easy to verify that $f_{x_i=\mathbf{0}}$ and $f_{x_i=\mathbf{1}}$ are unique. So the low successors of G_1 and G_2 represent the same function $f_{x_i=\mathbf{0}}$. By the induction hypothesis, the low successors of G_1 and G_2 are identical. Similarly, G_1 and G_2 share the same high successors. Hence, G_1 and G_2 are identical.

The proofs of the other cases where the expansion type of the root node is one of the pD-, nD-, bS-, bpD- and bnD-expansion are similar. □

Proof of Theorem 2

Proof. Without of generality, we assume that the variable order is $x_1, \cdots, x_n$. Let i be the least number s.t. $x_i \in \mathsf{dep}(f)$. We remark that $\mathsf{dep}(f) = \emptyset$ if f is the constant function $\mathbf{0}$ or $\mathbf{1}$. In this case, we assume that $i = n + 1$. We prove by induction on $n + 1 - i$.

The proof of the base case ($i = n + 1$) is the same as that of Theorem 1. We only prove the inductive case. Suppose that G_1 and G_2 are two strong ROBKFDDs representing the function f. Let v_1 and v_2 be the root nodes of G_1 and G_2 respectively. Assume that the expansion types of v_1 and v_2 are the bi-Shannon expansion (the proofs of the other cases where the expansion type of the root node is one of the pD-, nD-, bS-, bpD- and bnD- expansion is similar).

We first verify that v_1 and v_2 share the same decision variable x_i and the same auxiliary function g. By the proof

of Theorem 1, we get that the decision variables of v_1 and v_2 are x_i. It remains to show that v_1 and v_2 share the same auxiliary function g. Suppose that x_j and x_k are auxiliary functions of v_1 and v_2, and $j < k$. Without of generality, we assume that $\pi_j^e = \pi_k^e = bS$ (the other cases where $\pi_j^e \in \{S, pD, nD, bS, bpD, bnD\}$ can be shown in the same way). According to the bS-expansion, $f = (\bar{x}_i \oplus x_j) \cdot f_{x_i=x_j} + (x_i \oplus x_j) \cdot f_{x_i=\bar{x}_j}$ and $f = (\bar{x}_i \oplus x_k) \cdot f_{x_i=x_k} + (x_i \oplus x_k) \cdot f_{x_i=\bar{x}_k}$. It is easy to check that $f_{x_i=x_j} = (\bar{x}_i \oplus x_k) \cdot f_{x_i=x_j} + (x_i \oplus x_k) \cdot f_{x_i=\bar{x}_k}$ and $f_{x_i=\bar{x}_j} = (\bar{x}_i \oplus x_k) \cdot f_{x_i=\bar{x}_k} + (x_i \oplus x_k) \cdot f_{x_i=x_j}$. This violates the fact that G_2 is a strong ROBKFDD.

Finally, in very much the same way, G_1 and G_2 being identical can be justified as Theorem 1 illustrates. □

Proof of Theorem 3

Proof. We only consider the weak ROBKFDD case and the strong ROBKFDD case can be shown in the same way.

We now introduce complemented edges and require that complemented edges only appear at the high edge of each internal node, and only one terminal node $\mathbf{0}$ is available.

Without of generality, we assume that the variable order is $x_1, \cdots, x_n$. Let i be the least number s.t. $x_i \in \mathsf{dep}(f)$. We remark that $\mathsf{dep}(f) = \emptyset$ if f is the constant function $\mathbf{0}$ or $\mathbf{1}$. In this case, we assume that $i = n + 1$. We prove by induction on $n + 1 - i$.

Base case ($i = n + 1$): If f is a constant function $\mathbf{0}$, then the weak ROBKFDD G is a terminal node $\mathbf{0}$ without any complemented edge. If $f = \mathbf{1}$, then G contains only one terminal node $\mathbf{0}$ with complemented edges.

Inductive case: Now we construct the weak ROBKFDD G s.t. it represents f and it may be with complemented edges. Let v be the root node of G. Suppose that the expansion type of v is the Shannon expansion. According to the proof of Theorem 1, the decision variable of v is x_i, and $f = \bar{x}_i \cdot f_{x_i=\mathbf{0}} + x_i \cdot f_{x_i=\mathbf{1}}$. By the induction hypothesis, there are two unique weak ROBKFDDs G_1 and G_2 which represent $f_{x_i=\mathbf{0}}$ and $f_{x_i=\mathbf{1}}$ respectively. If G_1 is with complemented edges, then we set the edge pointing to G_1 to be regular, and the edge pointing to G to be complemented. In addition, we set the edge pointing to G_2 to be regular (resp. complemented), if the edge is initially complemented (resp. regular). Finally, we let the low successor of G be G_1 and the high successor of G be G_2 (the construction of the other the expansion types are similar). Since G_1 and G_2 are unique and G satisfies the two requirements that complemented edges only appear at the high edge of each internal node, and that only one terminal node $\mathbf{0}$ is available. So G is the unique weak ROBKFDD with the complemented edge representing f. □

References

Amarú, L.; Gaillardon, P.-E.; and Micheli, G. D. 2014. Biconditional Binary Decision Diagrams: A Novel Canonical Logic Representation Form. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 4(4):487–500.

Becker, B., and Drechsler, R. 1995a. How many Decomposition Types do we need? In *Proceedings of 1995 European Design and Test Conference (ED&TC-1995)*, 438–443.

Becker, B., and Drechsler, R. 1995b. On the Relation between BDDs and FDDs. *Information and Computation* 123(2):185–197.

Boole, G. 1854. *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. Dover Publications.

Brace, K. S.; Rudell, R. L.; and Bryant, R. E. 1990. Efficient implementation of a BDD package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC-1990)*, 40–45.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computer* 100(8):677–691.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7):772–799.

Drechsler, R., and Becker, B. 1998. Ordered Kronecker Functional Decision Diagrams – A Data Structure for Representation and Manipulation of Boolean Functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17(10):965–973.

Drechsler, R.; Theobald, M.; and Becker, B. 1996. Fast OFDD-Based Minimization of Fixed Polarity Reed-Muller Expressions. *IEEE Transactions on Computers* 45(11):1294–1299.

Fierens, D.; den Broeck, G. V.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and Raedt, L. D. 2015. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming* 15(3):358–401.

Huang, J., and Darwiche, A. 2005. On Compiling System Models for Faster and More Scalable Diagnosis. In *AAAI*, 300–306.

Huang, J. 2006. Combining Knowledge Compilation and Search for Conformant Probabilistic Planning. In *ICAPS*, 253–262.

Kebschull, U.; Schubert, E.; and Rosenstiel, W. 1992. Multilevel Logic Synthesis Based on Functional Decision Diagrams. In *Proceedings of the Third European Conference on Design Automation (EDAC-1992)*, 43–47.

Kebschull, U.; Schubert, E.; and Rosenstiel, W. 1993. Efficient Graph-Based Computation and Manipulation of Functional Decision Diagrams. In *Proceedings of the Fourth European Conference on Design Automation (EDAC-1993)*, 278–282.

Kerntopf, P. 2001. New Generalizations of Shannon Decomposition. In *International Workshop on Application of Reed-Muller Expansion in Circuit Design*, 109–118.

Muller, D. E. 1954. Application of Boolean Algebra to Switching Circuit Design and to Error Detection. *Transactions of the IRE Professional Group on Electronic Computers* 3(3):6–12.

Palacios, H.; Bonet, B.; Darwiche, A.; and Geffner, H. 2005. Pruning Conformant Plans by Counting Models on Compiled d-DNNF Representations. In *ICAPS*, 141–150.

Reed, I. S. 1954. A class of multiple-error-correcting codes and their decoding scheme. *Transactions of the IRE Professional Group on Information Theory* 4(4):38–49.

Rudell, R. 1993. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proceedings of the Sixth IEEE/ACM International Conference on Computer-Aided Design (ICCAD-1993)*, 42–47.

Shannon, C. E. 1938. A Symbolic Analysis of Relay and Switching Circuits. *Transactions of the American Institute of Electrical Engineers* 57(12):713–723.

Siddiqi, S. A., and Huang, J. 2007. Hierarchical Diagnosis of Multiple Faults. In *IJCAI*, 581–586.