Tagged Sentential Decision Diagrams: Combining Standard and Zero-suppressed Compression and Trimming Rules

Liangda Fang
Department of Computer Science
Jinan University
Guangzhou, China
fangld@jnu.edu.cn

Biqing Fang
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
fangbq3@mail2.sysu.edu.cn

Hai Wan*
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
wanhai@mail.sysu.edu.cn

Zeqi Zheng
Department of Computer Science
Jinan University
Guangzhou, China
97xintu@stu2016.jnu.edu.cn

Liang Chang
Guilin University of Elecronic Technology
Guilin, China
changl@guet.edu.cn

Quan Yu
School of Mathematics and Statistics
Qiannan Normal University for Nationalities
Duyun, China
yuquan1704@163.com

***Abstract*—The Sentential Decision Diagram (SDD) is a compact and canonical representation of Boolean functions that generalizes the Ordered Binary Decision Diagrams (OBDDs). A variant of SDDs, namely Zero-suppressed Sentential Decision Diagrams (ZSDDs), was proposed recently by using different trimming rules. SDDs are suitable for functions where adjacent input assignments have the same outcome, while ZSDDs are more compact for spare functions. In this paper, we introduce a novel canonical SDD variant, called the Tagged Sentential Decision Diagrams (TSDDs). The key insight of TSDDs is to combine both trimming rules of SDDs and ZSDDs. With both characteristics of SDDs and ZSDDs, the TSDD representation is at least as small as the SDD or ZSDD representation for any Boolean functions. This is also shown in our experimental evaluation.**

***Index Terms*—Boolean functions, Decision diagrams**

I. Introduction

A representation of Boolean functions, which has compact size and supports efficient manipulation, plays a dominant role in many fields of computer-aided design and verification, *e.g.*, logic synthesis [1], circuit verification [2] and testing [3]. One notable representation is *binary decision diagram (BDD)* [4] that is based on the Shannon decomposition [5]. Under two restrictions: ordering and reduction, BDDs enjoy a nice property: canonicity, *i.e.*, any Boolean function has a unique representation, and supports efficient Boolean operations. By using different reduction rules, [6] proposed a variant of BDDs, namely *zero-suppressed binary decision diagrams (ZBDD)*. In general ZBDDs tend to be smaller than BDDs when representing sparse Boolean functions. By combining reduction rules that originates from BDDs and ZBDDs, [7] proposed *tagged binary decision diagram (TBDD)*. Thanks to two reduction rules, TBDDs is a more compact representation than BDDs and ZDDs.

Instead of the Shannon decomposition, [8] proposed the structured decomposition, which splits a Boolean function according to a set of mutually exclusive subfunctions rather than a single variable. Based on this decomposition, [9] proposed a new decision diagram, called *sentential decision diagram (SDD)*. Just as Ordered BDDs (OBDDs) are characterized by a variable order, SDDs are characterized by a binary tree whose leaves are variables, called vtree. SDDs generalize OBDDs since structured decomposition and vtree are extensions to the Shannon decomposition and variable order respectively. Interestingly, SDDs are also canonical under restrictions similar to reductions in BDDs, and support efficient Boolean operations just like OBDDs. From both of practical and theoretical perspectives, SDDs are more compact than OBDDs [10], [11]. Recently, [12] proposed the zero-suppressed trimming rule, and hence obtaining a variant of SDD, namely *zero-suppressed sentential decision diagram (ZSDD)*. Compared to SDDs, ZSDDs are a more compact form for spare Boolean functions.

Inspired by combing two different trimming rules that from SDDs and ZSDDs, we develop a representation of Boolean functions, namely *tagged sentential decision diagram (TSDD)*. Due to the fuse of two trimming rules, TSDDs are more compact than SDDs and ZSDDs, and generalizations of TBDDs. TSDDs are proved to be the canonical form of Boolean functions under the compression and trimming rules. The algorithm for Boolean operations on TSDDs is provided, and their complexity is also given. Finally, we conduct some experiments on LGSynth89, iscas85, and iscas89 benchmark sets from CAD community. The experimental results reveals that TSDDs outperforms other decision diagrams in terms of size in most test cases.

The rest of this paper is organized as follows. Section 2

* Corresponding author

provides the preliminaries of Boolean function, SDDs, and ZSDDs. In Section 3, we illustrate the definition, canoncity theorem and boolean operations of TSDDs. Experimental evaluation for logical synthesis and comparison with the other five decision diagrams: BDDs, ZBDDs, TBDDs, SDDs and ZSDDs appears in Section 4. Finally, Section 5 concludes this paper.

II. Technical Preliminaries

This section first provides basic concepts of Boolean functions, and then introduces the syntax and semantics of structured decomposable diagrams which serves as the basis of SDDs and ZSDDs. This section finally presents the definition, and compression and trimming rules of SDDs and ZSDDs.

Throughout this paper, we use lower case letters (*e.g.*, x_1, x_2) to denote variables, and bold upper case letters (*e.g.*, $\mathbf{X}, \mathbf{Y}$) to denote sets of variables. For a variable x, $\bar{x}$ denotes its negation. A *literal* is a variable or a negated one. A *truth assignment* over $\mathbf{X}$ is a mapping $\sigma : \mathbf{X} \mapsto \{0,1\}$. We let $\Sigma_{\mathbf{X}}$ be the set of truth assignments over $\mathbf{X}$. We say $f(\mathbf{X})$ is a *Boolean function* over $\mathbf{X}$, which is a mapping: $\Sigma_{\mathbf{X}} \mapsto \{0,1\}$. We use the constant $\mathbf{1}$ (resp. $\mathbf{0}$) to denote the trivial function which maps all assignments to 1 (resp. 0). Given a literal l, the cofactor $f|_l$ of f w.r.t. l is a subfunction resulted by setting x to 1 (resp. 0) if l is x (resp. $\bar{x}$). We say a function f *depends* on a variable x if $f|_x \neq f|_{\bar{x}}$.

A Boolean function $f(\mathbf{X}, \mathbf{Y})$ with disjoint sets of variables $\mathbf{X}$ and $\mathbf{Y}$ can be represented as follows:

$$f = [p_1(\mathbf{X}) \wedge s_1(\mathbf{Y})] \vee \cdots \vee [p_n(\mathbf{X}) \wedge s_n(\mathbf{Y})].$$

The set $\{(p_1, s_1), \cdots, (p_n, s_n)\}$ is called an $(\mathbf{X}, \mathbf{Y})$-*decomposition* of f [13]. Each pair (p_i, s_i) is called an *element* of the decomposition, each p_i is called a *prime* and each s_i is called a *sub* [13]. For a prime p_i, we say s_i is the sub w.r.t. p_i. An $(\mathbf{X}, \mathbf{Y})$-decomposition is called an $(\mathbf{X}, \mathbf{Y})$-*partition* [9], if the following hold:

1) $p_i \neq \mathbf{0}$ for each prime p_i;
2) $p_i \wedge p_j = \mathbf{0}$ for any two distinct primes p_i and p_j;
3) $p_1 \vee \cdots \vee p_n = \mathbf{1}$.

A decomposition is *compressed* if its subs are pairwise different, *i.e.*, $s_i \neq s_j$ for any two distinct subs s_i and s_j. Interestingly, compressed $(\mathbf{X}, \mathbf{Y})$-partition is unique for a fixed Boolean function.

Theorem 1 ([9]). Let $\mathbf{X}$ and $\mathbf{Y}$ be two disjoint sets of variables. Any Boolean function $f(\mathbf{X}, \mathbf{Y})$ has a unique compressed $(\mathbf{X}, \mathbf{Y})$-partition.

A Boolean function can be graphically represented as a diagram by recursively applying $(\mathbf{X}, \mathbf{Y})$-decompositions based on the notion of vtrees, which generalize variable orders. A *vtree* is a full binary tree whose leaves are labeled with variables. Given a vtree $\mathbf{T}$, we use $v(\mathbf{T})$ to denote the set of variables appearing in leaves of $\mathbf{T}$, and use $\mathbf{T}_l$ and $\mathbf{T}_r$ to denote the left and right subtrees of $\mathbf{T}$ respectively. In addition, there is a special leaf node labeled by 0, which can be considered as a child of any vtree node, and $v(0) = \emptyset$. The notation $\mathbf{T}^1 \preccurlyeq \mathbf{T}^2$ denotes that $\mathbf{T}^1$ is a subtree of $\mathbf{T}^2$. In the following, to unify the definitions of SDDs and ZSDDs, we will provide the concept of the *structured decomposable diagram* and two versions of semantics for these diagrams.

Definition 1. Let $\mathbf{T}$ be a vtree. A structured decomposable diagram is a pair $(\mathbf{T}, \alpha)$ which is defined inductively as follows

- α is a terminal node labeled by one of the four symbols: $\mathbf{1}$, $\mathbf{0}$, ε, and $\bar{\varepsilon}$, and $\mathbf{T}$ is any vtree.
- α is a decomposition node $\{(p_1, s_1), \cdots, (p_n, s_n)\}$ satisfying the following conditions:
  1) each p_i is a structured decomposable diagram $(\mathbf{T}^1, \alpha)$ where $\mathbf{T}^1$ is a subtree of $\mathbf{T}_l$;
  2) each s_i is a structured decomposable diagram $(\mathbf{T}^2, \alpha)$ where $\mathbf{T}^2$ is a subtree of $\mathbf{T}_r$.

The size of α, denoted $|\alpha|$, is obtained by summing the sizes of all its decompositions. The notation ε denotes the conjunction of negative literals, and $\bar{\varepsilon}$ denotes the negation of ε.

To interpret structured decomposable diagrams, we define *the standard semantics*, *i.e.*, a mapping from structured decomposable diagrams together with vtrees into Boolean functions.

Definition 2. Let $\mathbf{T}$ and $\mathbf{T}'$ be two vtrees where $\mathbf{T}$ is a subtree of $\mathbf{T}'$. The *standard semantics* is inductively defined as follows:

- $\langle \mathbf{T}', (\mathbf{T}, \mathbf{1}) \rangle_s = \mathbf{1}$ and $\langle \mathbf{T}', (\mathbf{T}, \mathbf{0}) \rangle_s = \mathbf{0}$;
- $\langle \mathbf{T}', (\mathbf{T}, \varepsilon) \rangle_s = \bigwedge\limits_{x \in v(\mathbf{T})} \bar{x}$ and $\langle \mathbf{T}', (\mathbf{T}, \bar{\varepsilon}) \rangle_s = \bigvee\limits_{x \in v(\mathbf{T})} x$;
- $\langle \mathbf{T}', (\mathbf{T}, \{(p_1, s_1), \cdots, (p_n, s_n)\}) \rangle_s = \bigvee\limits_{i=1}^{n} (\langle \mathbf{T}_l, p_i \rangle_s \wedge \langle \mathbf{T}_r, s_i \rangle_s)$.

We remark that the vtree $\mathbf{T}'$ is not explicitly needed for the standard semantics. However, the zero-suppressed semantics, which will be given later, depends on the vtree $\mathbf{T}'$. To conform with the zero-suppressed semantics, we introduce an extra vtree in the standard semantics.

We are ready to present the concept of SDDs by imposing some conditions on structured decomposable diagrams.

Definition 3. A sentenial decision diagram $(\mathbf{T}, \alpha)$ is a structured decomposable diagram that satisfies the following:

1) if α is a terminal node labeled by $\mathbf{1}$ or $\mathbf{0}$, then $\mathbf{T} = 0$.
2) if α is a terminal node labeled by ε or $\bar{\varepsilon}$, then $\mathbf{T}$ is a leaf node.
3) if α is a decomposition node $\{(p_1, s_1), \cdots, (p_n, s_n)\}$, then the following hold:
   - $\langle \mathbf{T}_l, p_i \rangle_s \neq \mathbf{0}$ for $1 \leq i \leq n$;
   - $\langle \mathbf{T}_l, p_i \rangle_s \wedge \langle \mathbf{T}_l, p_j \rangle_s = \mathbf{0}$ for $i \neq j$;
   - $\bigvee\limits_{i=1}^{n} \langle \mathbf{T}_l, p_i \rangle_s = \mathbf{1}$.

Consider the vtree depicted in Figure 1(a). The node labeled with 4 is the root whose left subtree contains $\mathbf{X} = \{x_1, x_2\}$ and whose right subtree contains $\mathbf{Y} = \{x_3, x_4\}$. Decomposing the function $f : (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3)$, depicted

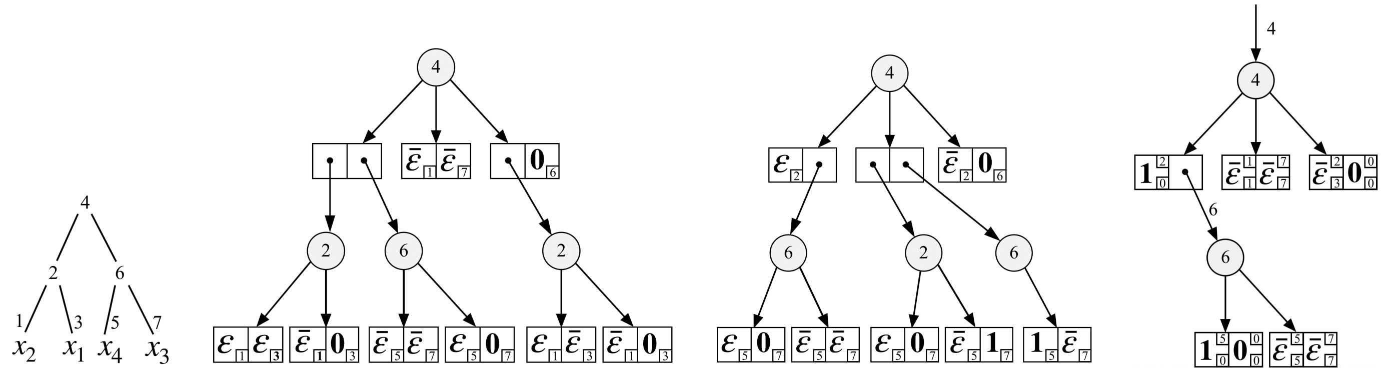
(a) A vtree (b) An SDD (c) A ZSDD (d) A TSDD

Fig. 1: The vtree and the SDD, ZSDD and TSDD representations of $f = (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3)$

in Figure 1(b), at node 1 leads to the following $(\mathbf{X}, \mathbf{Y})$-decomposition: $\{(\bar{x}_1 \wedge \bar{x}_2, x_3 \wedge \bar{x}_4), (x_2, x_3), (\bar{x}_1 \wedge x_2, \mathbf{0})\}$. This decomposition is represented by the root node of Figure 1(b). The circle node, labeled with 4, is a decomposition node with three pairs. Each prime or sub is either a terminal node or pointer to a decomposition node. Each terminal node $(\mathbf{T}, \alpha)$ is graphically represented by a box where α is shown in the center, and $\mathbf{T}$ is shown on the bottom-right corner. Then, the primes (resp. subs) will be decomposed recursively w.r.t. node 2 (resp. 6). This decomposition process continues until it reaches constants or literals.

In order to compactly represent Boolean functions, compression and trimming rules for SDDs are proposed in [9].

- Standard compression rule: if $\langle \mathbf{T}_r, s_i \rangle_s = \langle \mathbf{T}_r, s_j \rangle_s$, then replace $(\mathbf{T}, \{(p_1, s_1), \cdots, (p_i, s_i), \cdots, (p_j, s_j), \cdots, (p_n, s_n)\})$ with $(\mathbf{T}, \{(p_1, s_1), \cdots, (p', s_i), \cdots, (p_n, s_n)\})$ where $\langle \mathbf{T}_l, p' \rangle_s = \langle \mathbf{T}_l, p_i \rangle_s \vee \langle \mathbf{T}_l, p_j \rangle_s$.
- Standard trimming rule:
  1) replace $(\mathbf{T}, \{(p_1, (\mathbf{T}', \mathbf{1})), (p_2, (\mathbf{T}'', \mathbf{0}))\})$ with p_1;
  2) replace $(\mathbf{T}, \{((\mathbf{T}', \mathbf{1}), s)\})$ with s.

It is easily verified that applying the above two rules on an SDD α does not modify the Boolean function that α corresponds to under the standard semantics. An SDD is *compressed* (resp. *trimmed*), if no standard compression (resp. trimming) rule can be applied in it. Interestingly, compressed and trimmed SDDs are shown to be a canonical form of Boolean functions [9].

Similarly to SDDs, ZSDDs are a class of structured decomposable diagrams. Hence, SDDs and ZSDDs share the same syntactic definition. However, they differ in the semantics as well as compression and trimming rules.

The *zero-suppressed semantics* $\langle \mathbf{T}', \alpha \rangle_z$ is defined as the conjunction of the standard one and an extra Boolean function $\bigwedge_{x \in v(\mathbf{T}') \setminus v(\mathbf{T})} \bar{x}$. For example, $\langle \mathbf{T}', (\mathbf{T}, \varepsilon) \rangle_z = (\bigwedge_{x \in v(\mathbf{T})} \bar{x}) \wedge (\bigwedge_{x \in v(\mathbf{T}') \setminus v(\mathbf{T})} \bar{x}) = \bigwedge_{x \in v(\mathbf{T}')} \bar{x}$.

Definition 4. A zero-suppressed sentenial decision diagram $(\mathbf{T}, \alpha)$ is a structured decomposable diagram that satisfies the following:

1) if α is a terminal node labeled by $\mathbf{0}$ or ε, then $\mathbf{T} = 0$.
2) if α is a terminal node labeled by $\mathbf{1}$ or $\bar{\varepsilon}$, then $\mathbf{T}$ is a leaf node.
3) if α is a decomposition node $\{(p_1, s_1), \cdots, (p_n, s_n)\}$, then the following hold:
   - $\langle \mathbf{T}_l, p_i \rangle_z \neq \mathbf{0}$ for $1 \leq i \leq n$;
   - $\langle \mathbf{T}_l, p_i \rangle_z \wedge \langle \mathbf{T}_l, p_j \rangle_z = \mathbf{0}$ for $i \neq j$;
   - $\bigvee_{i=1}^{n} \langle \mathbf{T}_l, p_i \rangle_z = \mathbf{1}$.

Besides the semantics and syntax, SDDs and ZSDDs have different compression and trimming rules. In the following, we present the zero-suppressed compression and trimming rules for decomposition nodes.

- Zero-suppressed compression rule: if $\langle \mathbf{T}_r, s_i \rangle_z = \langle \mathbf{T}_r, s_j \rangle_z$, then replace $(\mathbf{T}, \{(p_1, s_1), \cdots, (p_i, s_i), \cdots, (p_j, s_j), \cdots, (p_n, s_n)\})$ with $(\mathbf{T}, \{(p_1, s_1), \cdots, (p', s_i), \cdots, (p_n, s_n)\})$ where $\langle \mathbf{T}_l, p' \rangle_z = \langle \mathbf{T}_l, p_i \rangle_z \vee \langle \mathbf{T}_l, p_j \rangle_z$.
- Zero-suppressed trimming rule:
  1) replace $(\mathbf{T}, \{(p_1, (\mathbf{T}_r, \varepsilon)), (p_2, (\mathbf{T}', \mathbf{0}))\})$ with p_1;
  2) replace $(\mathbf{T}, \{((\mathbf{T}_l, \varepsilon), s), ((\mathbf{T}_l, \bar{\varepsilon}), (\mathbf{T}', \mathbf{0}))\})$ with s.

Similarly to SDDs, after compressing or trimming a ZSDD $(\mathbf{T}, \alpha)$, the Boolean function which it denotes remains the same under the zero-suppressed semantics. An ZSDD is *compressed* (resp. *trimmed*), if no zero-suppressed compression (resp. trimming) rule can be applied in it. Similarly to SDDs, compressed and trimmed ZSDDs enjoys the canoncity property [12].

Figure 1(c) shows the ZSDD representation of the function f. For the same Boolean function, SDD and ZSDD may lead to different sizes. For example, the function $\bar{x}_1 \wedge \bar{x}_2$ is the prime of the first element of the $(\{x_1, x_2\}, \{x_3, x_4\})$-decomposition of f. It can been seen that $\bar{x}_1 \wedge \bar{x}_2$ is a sparse function. Its SDD representation is size of 2 and contains four terminal nodes. By contrast, the ZSDD representation with size 1 is more compact than the SDD representation. For the prime x_2 of the second element of the partition, SDDs are more suitable for this function than ZSDDs since the size of the former is smaller than the latter. We can observe that ZSDD are generally more compact than SDDs when representing sparse

Boolean functions, and SDDs suits for Boolean functions where adjacent truth assignments have the same outcome.

III. Tagged Sentential Decision Diagram

In this section, we first introduce the syntax and semantics of the variant of SDD, named *tagged sentential decision diagram* (TSDD), which combines both the standard and zero-suppressed compression and trimming rules, and thereby being a more compact representation than SDDs and ZSDDs. We then give the canonicity theorem and Boolean operations for TSDDs.

A. The syntax and semantics

By Definitions 3 and 4, any SDD or ZSDD is a pair of a vtree node $\mathbf{T}$ and a terminal/decomposition node α. To integrate the syntax of ZSDDs into that of SDDs, we incorporate an additional vtree node. The syntax and semantics of TSDD are defined together as follows:

Definition 5. Let $\mathbf{T}^1$ and $\mathbf{T}^2$ be two vtrees s.t. $\mathbf{T}^2 \preccurlyeq \mathbf{T}^1$. A tagged sentenial decision diagram is a tuple $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ defined inductively as follows

- α is a terminal node labeled by $\mathbf{0}$, $\mathbf{T}^1 = \mathbf{T}^2 = 0$.
  Semantics: $\langle(\mathbf{T}^1, \mathbf{T}^2, \mathbf{0})\rangle_t = \mathbf{0}$.
- α is a terminal node labeled by $\mathbf{1}$, and $\mathbf{T}^2 = 0$.
  Semantics: $\langle(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})\rangle_t = \bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}$;
- α is a terminal node labeled by $\bar{\varepsilon}$, and $\mathbf{T}^2$ is a leaf node.
  Semantics: $\langle(\mathbf{T}^1, \mathbf{T}^2, \bar{\varepsilon})\rangle_t = (\bigvee_{x \in v(\mathbf{T}^2)} x) \wedge (\bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x})$;
- α is a decomposition node $\{(p_1, s_1), \cdots, (p_n, s_n)\}$ satisfying the following conditions:
  1) each p_i is a tagged sentenial decision diagram $(\mathbf{T}^3, \mathbf{T}^4, \beta)$ where $\mathbf{T}^4 \preccurlyeq \mathbf{T}^3 \prec \mathbf{T}^2$;
  2) each s_i is a tagged sentenial decision diagram $(\mathbf{T}^5, \mathbf{T}^6, \gamma)$ where $\mathbf{T}^6 \preccurlyeq \mathbf{T}^5 \prec \mathbf{T}^2$;
  3) $\langle p_i\rangle_t \neq \mathbf{0}$ for $1 \leq i \leq n$;
  4) $\langle p_i\rangle_t \wedge \langle p_j\rangle_t = \mathbf{0}$ for $i \neq j$;
  5) $\bigvee_{i=1}^{n} \langle p_i\rangle_t = \mathbf{1}$.

  Semantics: $\langle(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \cdots, (p_n, s_n)\})\rangle_t =$
  $$[\bigvee_{i=1}^{n} (\langle p_i\rangle_t \wedge \langle s_i\rangle_t)] \wedge [\bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}].$$

For a TSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$, we call $\mathbf{T}^1$ the zero-suppressed vtree node of the TSDD, and $\mathbf{T}^2$ the standard vtree node. The semantics for a TSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ is a conjunction of two components that involve standard and zero-suppressed components. The standard component is similar to the standard semantics for structured decomposable diagrams. For example, in the case where $\alpha = \bar{\varepsilon}$, the first conjunct is $\bigvee_{x \in v(\mathbf{T}^2)} x$. The zero-suppressed component is $\bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}$. We remark that $\langle(\mathbf{T}^1, \mathbf{T}^2, \mathbf{0})\rangle_t$ also contains two components although it denotes the constant function $\mathbf{0}$. The zero-suppressed component of $\langle(\mathbf{T}^1, \mathbf{T}^2, \mathbf{0})\rangle_t = \bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}$ while the standard one is $\mathbf{0}$. The conjunction of these two components is therefore $\mathbf{0}$. Similarly, $\langle(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})\rangle_t$ contains two components. It can be verified that the sets of variables occurring in two components are disjoint.

With both characteristics of SDDs and ZSDDs, TSDDs are more compact representations for Boolean functions. Figure 1(d) shows a graphical representation of TSDD for the function $f : (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3)$. The first prime of $(\{x_1, x_2\}, \{x_3, x_4\})$-partition of f is $\bar{x}_1 \wedge \bar{x}_2$. This function is a spare function, hence its ZSDD representation is a terminal node ε with size 1, but its SDD representation is a decomposition node with two elements. The TSDD representation is a terminal node $(2, 0, \mathbf{1})$ whose size is 1. By contract, the second prime of partition is x_2, and it suits to SDDs more than ZSDDs. Its SDD representation is a terminal node $\bar{\varepsilon}$ while its ZSDD representation is a decomposition node. The TSDD representation is also a terminal node $(1, 1, \bar{\varepsilon})$. Hence, we can observe that TSDDs have advantages of both of SDDs and ZSDDs.

B. Canonicity

In the following, we address the canoncity of TSDDs. We will then use these lemmas to prove our main canonicity theorem. For two TSDDs F and G, we will write $F = G$ to mean that they are syntactically equal. We will write $F \equiv G$ to mean that they represent the same Boolean function: $\langle F\rangle_t = \langle G\rangle_t$. We first state some definitions and lemmas about TSDDs.

To reduce the size of TSDDs, we introduce the tagged version of compression and trimming rules.

- Tagged compression rule: if $\langle s_i\rangle_t = \langle s_j\rangle_t$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \cdots, (p_i, s_i), \cdots, (p_j, s_j), \cdots, (p_n, s_n)\})$ with $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \cdots, (p', s_i), \cdots, (p_n, s_n)\})$ where $\langle p'\rangle_t = \langle p_i\rangle_t \vee \langle p_j\rangle_t$.
- Tagged trimming rule (shown in Figure 2):
  (a) if $p_1 = (\mathbf{T}^2, \mathbf{T}^3, \alpha)$, $\langle s_1\rangle_t = \mathbf{1}$ and $\langle s_2\rangle_t = \mathbf{0}$, then replace $(\mathbf{T}^1, \mathbf{T}^1, \{(p_1, s_1), (p_2, s_2)\})$ with p_1;
  (b) if $\mathbf{T}^1 = \mathbf{T}^2$ or $\langle s\rangle_t = \mathbf{0}$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p, s)\})$ with s;
  (c) if $p_1 = (\mathbf{T}^2_l, \mathbf{T}^3, \alpha)$, $s_1 = (\mathbf{T}^2_r, 0, \mathbf{1})$ and $\langle s_2\rangle_t = \mathbf{0}$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}^3, \alpha)$;
  (d) if $p_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $s_1 = (\mathbf{T}^2_r, 0, \mathbf{1})$, $\langle s_2\rangle_t = \mathbf{0}$ and $\mathbf{T}^3 \preccurlyeq (\mathbf{T}^2_l)_l$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}^2_l, \{(p_1, (0, 0, \mathbf{1})), (p_2, s_2)\})$;
  (e) if $p_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $s_1 = (\mathbf{T}^2_r, 0, \mathbf{1})$, $\langle s_2\rangle_t = \mathbf{0}$ and $\mathbf{T}^3 \preccurlyeq (\mathbf{T}^2_l)_r$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}^2_l, \{((0, 0, \mathbf{1}), p_1)\})$;
  (f) if $p_1 = (\mathbf{T}^2_l, 0, \mathbf{1})$, $s_1 = (\mathbf{T}^2_r, \mathbf{T}^3, \alpha)$ and $\langle s_2\rangle_t = \mathbf{0}$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}^3, \alpha)$;
  (g) if $p_1 = (\mathbf{T}^2_l, 0, \mathbf{1})$, $s_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $\langle s_2\rangle_t = \mathbf{0}$ and $\mathbf{T}^3 \preccurlyeq (\mathbf{T}^2_r)_l$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with

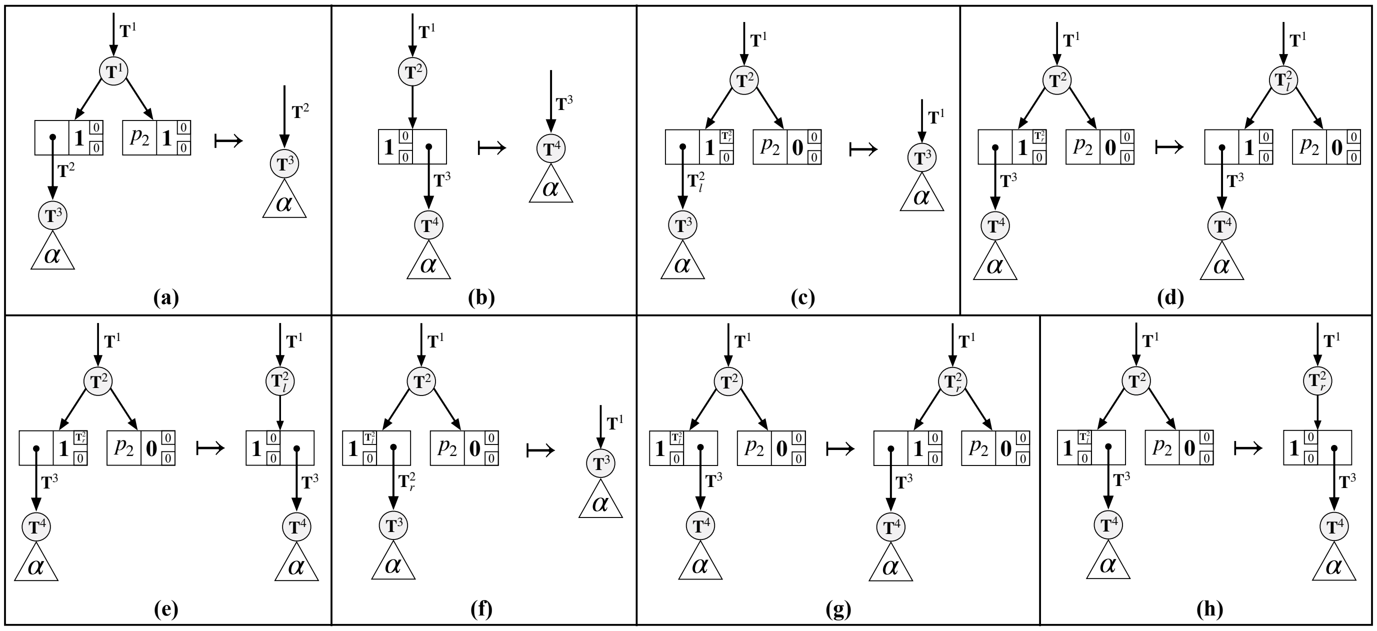
Fig. 2: Trimming rules

$(\mathbf{T}^1, \mathbf{T}^2_r, \{(s_1, (0, 0, \mathbf{1})), (p_3, s_2)\})$ where $\langle p_3\rangle_t = \neg\langle s_1\rangle_t$;

(h) if $p_1 = (\mathbf{T}^2_l, 0, \mathbf{1})$, $s_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $\langle s_2\rangle_t = \mathbf{0}$ and $\mathbf{T}^3 \preccurlyeq (\mathbf{T}^2_r)_r$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}^2_r, \{((0, 0, \mathbf{1}), s_1)\})$.

A TSDD is *compressed* (resp. *trimmed*), if no tagged compression (resp. trimming) rule can be applied in it. We hereafter prove the canonicity of compressed and trimmed TSDDs. Before that, we first give one definition and two lemmas for proving the canonicity theorem.

Definition 6. A Boolean function f *essentially depends on* a vtree node $\mathbf{T}$ if f is not trivial and if $\mathbf{T}$ is a deepest node that includes all variables that f depends on.

Lemma 1 ([9]). A non-trivial Boolean function essentially depends on exactly one vtree node.

Lemma 2. Let $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ be a compressed and trimmed TSDD s.t. $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$ is not trivial. Then, $\mathbf{T}^1$ and $\mathbf{T}^2$ are unique, and $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$ essentially depends on $\mathbf{T}^1$.

Proof. Let f be the Boolean function s.t. $f = \langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$. Let $\mathbf{X}$ be the set of variables on which f depends. Since $\langle\alpha\rangle$ is not trivial, $\mathbf{X} \neq \emptyset$. Let $(\mathbf{T}^3, \mathbf{T}^4, \beta)$ is a compressed and trimmed TSDD s.t. $f = \langle(\mathbf{T}^3, \mathbf{T}^4, \beta)\rangle_t$.

We first prove that $\mathbf{T}^1 = \mathbf{T}^3$. Suppose that $\mathbf{T}^1$ and $\mathbf{T}^3$ are incomparable. Thus, $v(\mathbf{T}^1) \cap v(\mathbf{T}^3) = \emptyset$. This contradicts that the fact that $v(\mathbf{T}^1)$ and $v(\mathbf{T}^3)$ include $\mathbf{X}$. We now assume without of loss generality that $\mathbf{T}^1 \prec \mathbf{T}^3$. Thus, $\mathbf{X} \subseteq v(\mathbf{T}^1) \subset v(\mathbf{T}^3)$. By Definition 5, the function $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle$ contains a zero-suppressed component $f_\alpha = \bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}$. Similarly, the zero-suppressed component f_β of $\langle(\mathbf{T}^3, \mathbf{T}^4, \beta)\rangle_t$ is $\bigwedge_{x \in v(\mathbf{T}^3)\setminus v(\mathbf{T}^4)} \bar{x}$. Suppose that $\mathbf{T}^3 \neq \mathbf{T}^4$. Thus, there is a variable $x \in v(\mathbf{T}^3) \setminus v(\mathbf{T}^4)$ s.t. $x \notin v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)$, and hence $f_\alpha \neq f_\beta$. This, together with the fact that the sets of variables occurring in the standard and zero-suppressed components are disjoint, imply that $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t \neq \langle(\mathbf{T}^3, \mathbf{T}^4, \beta)\rangle_t$. Suppose that $\mathbf{T}^3 = \mathbf{T}^4$. We assume that $\mathbf{T}^1 \preccurlyeq \mathbf{T}^3_r$. Then, $\beta = \{((0, 0, \mathbf{1}), \gamma)\}$ where $\langle\gamma\rangle_t = \langle\alpha\rangle_t$. We can apply the trimming rule (2) on $(\mathbf{T}^3, \mathbf{T}^3, \beta)$. Thus, $(\mathbf{T}^3, \mathbf{T}^3, \beta)$ is not trimmed. Similarly, the case where $\mathbf{T}^1 \preccurlyeq \mathbf{T}^3_l$ is also impossible. Hence, $\mathbf{T}^1 = \mathbf{T}^3$.

We now prove that $\mathbf{T}^2 = \mathbf{T}^4$. Suppose that $\mathbf{T}^2$ and $\mathbf{T}^4$ are incomparable. Let $\mathbf{T}^5$ be the least common ancestor of $\mathbf{T}^2$ and $\mathbf{T}^4$. We assume without of loss generality that $\mathbf{T}^2 \preccurlyeq \mathbf{T}^5_l$ and $\mathbf{T}^4 \preccurlyeq \mathbf{T}^5_r$. Let f_α and g_α be the zero-suppressed and standard components of $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$ respectively. Let f_β and g_β be the zero-suppressed and standard components of $\langle(\mathbf{T}^1, \mathbf{T}^4, \beta)\rangle_t$ respectively. By Definition 5, $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t = f_\alpha \wedge g_\alpha = f_\beta \wedge g_\beta$, and $f_\alpha = \bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^2)} \bar{x}$ and $f_\beta = \bigwedge_{x \in v(\mathbf{T}^1)\setminus v(\mathbf{T}^4)} \bar{x}$. Since g_α depends on $v(\mathbf{T}^2)$, $g_\alpha = \bigwedge_{x \in v(\mathbf{T}^2)} \bar{x}$. Thus, $\alpha = \{((\mathbf{T}^2_l, 0, \mathbf{1}), (\mathbf{T}^2_r, 0, \mathbf{1})), (p, (0, 0, \mathbf{0}))\}$ where $\langle g\rangle_t = \bigvee_{x \in v(\mathbf{T}^2_l)} x$. This violates the trimming rules (3) and (6). We now assume without of loss generality that $\mathbf{T}^2 \prec \mathbf{T}^4$. Suppose that $\mathbf{T}^2 \prec \mathbf{T}^4_l$. Thus, g_α contains a conjunction $\bigwedge_{x \in v(\mathbf{T}^4_r)} \bar{x}$, and $\beta = \{(p_1, (\mathbf{T}^4_r, 0, \mathbf{1})), (p_2, s_2)\}$ where $\langle s_2\rangle_t = \mathbf{0}$. This violates the trimming rules (3), (4) or (5). Hence, $\mathbf{T}^2 = \mathbf{T}^4$.

Finally, we prove that f essentially depends on $\mathbf{T}^1$. On the contrary, we assume that $\mathbf{T}^1$ is not the vtree node that f essentially depends on. If there is variables x s.t. f depends on x and $x \notin v(\mathbf{T}^1)$, then $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t \neq f$. So all variables that f depends on are in $v(\mathbf{T}^1)$. By Lemma 1, f essentially depends on a unique vtree node, and we let $\mathbf{T}^5$ be this node. Suppose that $\mathbf{T}^5 \prec \mathbf{T}^1_r$. Thus, $\mathbf{T}^1 = \mathbf{T}^2$ and $\alpha = \{((0, 0, \mathbf{1}), s)\}$ where $\langle s\rangle_t = f$. This violates the trimming rule (2). Hence, f essentially depends on $\mathbf{T}^1$. $\square$

It is ready to prove the canonicity theorem for TSDDs.

Theorem 2. Given a vtree $\mathbf{T}$ over $\mathbf{X}$, any Boolean $f(\mathbf{X})$ has a unique compressed and trimmed TSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ where $\mathbf{T}^1 \preccurlyeq \mathbf{T}$.

Proof. Suppose that $f = \mathbf{0}$. If α is a decomposition node $\{(p_1, s_1),$
$\cdots, (p_n, s_n)\}$, then $\langle s_i\rangle_t = \mathbf{0}$ for any s_i. Since $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ is compressed, α contains only one pair (p, s) where $\langle s\rangle_t = \mathbf{0}$. This contradicts the trimming rule (2). So α is a terminal node $(0, 0, \mathbf{0})$. Similarly, the only TSDD denoting $\mathbf{1}$ is $(0, 0, \mathbf{1})$.

We now assume that f is not trivial. By Lemma 2, $\mathbf{T}^1$ and $\mathbf{T}^2$ is unique. Let β be a node s.t. $\langle(\mathbf{T}^1, \mathbf{T}^2, \beta)\rangle_t = f$. Let f_α and f_β be the zero-suppressed components of $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$ and $\langle(\mathbf{T}^1, \mathbf{T}^2, \beta)\rangle_t$ respectively. Let g_α and g_β be the standard components of $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$ and $\langle(\mathbf{T}^1, \mathbf{T}^2, \beta)\rangle_t$ respectively. Since $\langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t = \langle(\mathbf{T}^1, \mathbf{T}^2, \beta)\rangle_t$, $f_\alpha = f_\beta$ and $g_\alpha = g_\beta$. Then, we prove by induction on vtrees.

Base case: $\mathbf{T}^2$ is a leaf node or $\mathbf{T}^2 = 0$. Thus, α and β are one of terminal nodes $\mathbf{1}$ and $\bar{\varepsilon}$. Since $g_\alpha = g_\beta$, $\alpha = \beta$.

Inductive step: $\mathbf{T}^2$ is an internal node. Thus, α and β are decomposition nodes. Let $\mathbf{X} = v(\mathbf{T}^2_l)$ and $\mathbf{Y} = v(\mathbf{T}^2_r)$. By Definition 5, $\{(\langle p_1\rangle_t, \langle s_1\rangle_t), \cdots, (\langle p_n\rangle_t, \langle s_n\rangle_t)\}$ and $\{(\langle q_1\rangle_t, \langle r_1\rangle_t), \cdots, (\langle q_m\rangle_t, \langle r_m\rangle_t)\}$ are $(\mathbf{X}, \mathbf{Y})$-partitions. By Theorem 1, we get that $m = n$, and there is a permutation h of $\{1, \cdots, n\}$ s.t. $\langle p_i\rangle_t = \langle q_{h(i)}\rangle_t$ and $\langle s_i\rangle_t = \langle r_{h(i)}\rangle_t$. By the induction hypothesis, $p_i = q_{h(i)}$ and $s_i = r_{h(i)}$ for $1 \leq i \leq n$. This implies that $\alpha = \beta$. $\square$

C. Boolean operations

In this subsection, we describe the algorithm of Boolean operations for TSDDs.

Suppose that we are given two TSDDs $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ and $(\mathbf{T}^3, \mathbf{T}^4, \beta)$. In the case where they share the same zero-suppressed and standard vtree nodes (*i.e.*, $\mathbf{T}^1 = \mathbf{T}^3$ and $\mathbf{T}^2 = \mathbf{T}^4$), the Boolean operation $\circ$ on them can be performed by induction as follows. If α and β are terminal nodes, then the results is $(\mathbf{T}^1, \mathbf{T}^2, \alpha \circ \beta)$. If they are decomposition nodes where $\alpha = \{(p_1, s_1), \cdots, (p_n, s_n)\}$ and $\beta = \{(q_1, r_1), \cdots, (q_m, r_m)\}$, then we let $\gamma = \{(p_i \wedge q_j, s_i \circ r_j) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}$, and $(\mathbf{T}^1, \mathbf{T}^2, \gamma)$ is the resulting TSDD. Although the resulting TSDD may be not compressed nor trimmed, but it can be transformed into a compressed and trimmed one according to the tagged compression and trimming rules that are illustrated in the previous section. In the case where $\mathbf{T}^1 \neq \mathbf{T}^3$ or $\mathbf{T}^2 \neq \mathbf{T}^4$, we first normalize them into another TSDDs which use the same vtree nodes. We let the zero-suppressed vtree node $\mathbf{T}'$ of the normalized TSDD be the least common ancestor of $\mathbf{T}^1$ and $\mathbf{T}^3$. We also let the standard vtree node $\mathbf{T}$ be the least common ancestor of $\mathbf{T}^2$ and $\mathbf{T}^4$ if $\mathbf{T}^1 = \mathbf{T}^3$, and let $\mathbf{T}$ be $\mathbf{T}'$ otherwise. Suppose that a TSDD $F = (\mathbf{T}^1, \mathbf{T}^2, \alpha)$. Tagged normalizing rules will generate a TSDD $(\mathbf{T}', \mathbf{T}, \beta)$ where $\langle(\mathbf{T}', \mathbf{T}, \beta)\rangle_t = \langle(\mathbf{T}^1, \mathbf{T}^2, \alpha)\rangle_t$.

(a) if $\mathbf{T}^1 \preccurlyeq \mathbf{T}_l$, then replace F with $(\mathbf{T}', \mathbf{T}, \{(F, (0, 0, \mathbf{1})), (p_1, (0, 0, \mathbf{0}))\})$ where $\langle p_1\rangle_t = \neg\langle F\rangle_t$;
(b) if $\mathbf{T}^1 \preccurlyeq \mathbf{T}_r$, then replace F with $(\mathbf{T}', \mathbf{T}, \{((0, 0, \mathbf{1}), F)\})$;
(c) if $\mathbf{T}^2 \preccurlyeq \mathbf{T}_l$, then replace F with $(\mathbf{T}', \mathbf{T}, \{((\mathbf{T}_l, \mathbf{T}^2, \alpha), (\mathbf{T}_r, 0, \mathbf{1})),$ $(p_1, (0, 0, \mathbf{0}))\})$ where $\langle p_1\rangle_t = \neg\langle(\mathbf{T}_l, \mathbf{T}^2, \alpha)\rangle_t$;
(d) if $\mathbf{T}^2 \preccurlyeq \mathbf{T}_r$, then replace F with $(\mathbf{T}', \mathbf{T}, \{((\mathbf{T}_l, 0, \mathbf{1}), (\mathbf{T}_r, \mathbf{T}^2, \alpha)),$ $(p_1, (0, 0, \mathbf{0}))\})$ where $\langle p_1\rangle_t = \neg\langle(\mathbf{T}_l, 0, \mathbf{1})\rangle_t$;

The main algorithm `Apply` is illustrated in Algorithm 1. We first check the terminal cases for two TSDDs (Lines 1 - 2). Then, we generate the vtree nodes that are used to normalize F and G (Lines 3 - 6). Lines 7 - 13 compute the resulting decomposition node γ for the normalized TSDDs of F and G. Finally, we apply the compression and trimming rules on the TSDD $(\mathbf{T}', \mathbf{T}, \gamma)$ (Lines 14 - 15). If we do not apply the compression rule, then the running time of `Apply` algorithm is $O(n \cdot |\alpha| \cdot |\beta|)$ where n is the number of subvtrees of $\mathbf{T}'$, $|\alpha|$ is the size of α, and $|\beta|$ is the size of β. If we require the resulting TSDD H to be compressed, then the algorithm has exponential worst case runtime w.r.t. $|\alpha|$ and $|\beta|$. Applying compressing rules on SDDs leads to canonical form of SDDs, and hence facilitating caching in practice. It is shown that compilation with compressed SDDs is much more efficient than with uncompressed SDDs even if the latter supports polytime `Apply` algorithm while the former is not [14]. This is because that applying compressing rules leads to a canonical form of SDDs and facilitates caching and dynamic reordering in practice [14]. TSDDs are extensions to SDDs and share many traits with the later. Hence, we focus on only compressed and trimmed TSDDs in the following section.

IV. Experimental Results

Based on the the theoretic work, we have implemented the TSDD package using C++ language. We adopt conjunctive normal forms (CNFs) of combinational circuits used in CAD community from LGSynth89, iscas85, and iscas89 benchmark sets. All experiments were performed on a Linux machine with Intel Core i7-9800X 3.80GHz CPU and 32GB RAM. We compare BDD, ZBDD, TBDD, SDD, ZSDD and TSDDs in terms of size. For the sake of comparison, we use the CUDD package for BDDs and ZBDDs [15], the TBDD package for TBDDs [16], the SDD package for SDDs [17] and the ZSDD package for ZSDDs [18]. The orders for BDDs, ZBDDs, and TBDDs were generated by the symmetric reordering algorithm of the CUDD package [19] while the vtrees for TSDDs, SDDs, and ZSDDs were generated by the dynamic reordering algorithm of the SDD package [10]. We exclude benchmarks

Algorithm 1: Apply(F, G, $\circ$)

```
Input  : $F$: a TSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$;
         $G$: a TSDD $(\mathbf{T}^3, \mathbf{T}^4, \beta)$;
         $\circ$: a Boolean operator.
Output: $H$: The resulting TSDD.
1  if $\mathbf{T}^1 = \mathbf{T}^3$ and $\mathbf{T}^2 = \mathbf{T}^4$ and $\alpha, \beta$ are terminal nodes then
2      return $(\mathbf{T}^1, \mathbf{T}^2, \alpha \circ \beta)$
3  $\mathbf{T}' \leftarrow$ Lca($\mathbf{T}^1, \mathbf{T}^3$)
4  if $\mathbf{T}^1 = \mathbf{T}^3$ then
5      $\mathbf{T} \leftarrow$ Lca($\mathbf{T}^2, \mathbf{T}^4$)
6  else  $\mathbf{T} \leftarrow \mathbf{T}'$
7  $\gamma \leftarrow \{\}$
8  foreach element $(p_i, s_i)$ in Normalized($\mathbf{T}', \mathbf{T}, F$) do
9      foreach element $(q_j, r_j)$ in Normalized($\mathbf{T}', \mathbf{T}, G$) do
10         $p \leftarrow$ Apply($p_i, q_j, \wedge$)
11         if $p$ is consistent then
12             $s \leftarrow$ Apply($s_i, r_j, \circ$)
13             add element $(p, s)$ to $\gamma$
14 $H \leftarrow$ Compress(($\mathbf{T}'$, $\mathbf{T}$, $\gamma$))
15 $H \leftarrow$ Trim($H$)
16 return $H$
```

if either SDD or TSDD compilations run out of memory or failed in two hours time limit.

Our experimental results are shown in Table I. Columns 1-3 report the names of circuits from three benchmarks, the variable number of the circuit in CNF format, and the number of clauses, respectively. Columns 4-9 indicate the sizes of nodes of BDDs, ZBDDs, TBDDs, SDDs, ZSDDs and TSDDs. The last column "Best" is the best result among the six decision diagrams, and the best results are shown as bolded text. "-" means the test case run out of memory or timeout. Furthermore, TSDDs have average 8.83%, 7.70%, 89.61%, 79.06% and 78.81% smaller size than those of SDDs, ZSDDs, BDDs, ZDDs, and TBDDs, respectively. We also observe that TSDDs outperform the other five decision diagrams in 31 test cases (out of 39 test cases), and TSDDs attain the smallest size in all test cases when compared to SDDs and ZSDDs. In particular, for test case pcler8, the size of TSDD is 1167, which is much less than those of SDD and ZSDD that are 1524 and 1346, respectively. For a given Boolean function, finding a good vtree of a specific decision diagram is of uttermost importance, and this is inherently a heuristic approach. This explains that the sizes of the corresponding TSDDs are not the best in 8 cases. In the future, we will investigate heuristic algorithms of dynamical minimization to find better vtrees that are more suitable for TSDDs.

V. Conclusion

In this paper, we have presented a new canonical repersentation for Boolean functions: tagged sentential decision diagrams (TSDDs), along with the Boolean operation algorithms. First of all, we have provided the theoretical foundation of TSDDs: the syntactic definition and the semantics which is a mapping TSDDs to Boolean functions. In addition, we have given the compression and trimming rules for TSDDs, and have proven that TSDDs under these two rules turn out to be a canonical form. Finally, the Boolean operations on TSDDs have been provided. We also conduct experiments on the benchmarks from CAD communicty. The experimental results reveal that compared to existing decision diagrams, TSDD is a more compact form.

There are many directions for future work. An obvious one is to provide a high-optimized implementation for TSDDs. Another direction is to investigate dynamic reordering algorithms of TSDDs and the knowledge compilation properties, *e.g.*, the polytime satisfiaibilty and entailment checking. Finally, BDDs are widespread used in solving problems in the area of computer-aided design and verification, *e.g.*, logical synthesis and circuit verification. As TSDDs are more compact than BDDs in both theoretical and practical sides, it is interesting to lift the approaches to logical synthesis and circuit verification based on BDDs to that based on TSDDs.

Acknowledgment

This paper was supported by the National Natural Science Foundation of China (Nos. 61603152, 61573386, 61703182, 61772232, 61773179, 61862051 and 61802147), National Key R&D Program of China (No. 2018YFC0830600), Guangdong Province Natural Science Foundation (Nos. 2016A030313292 and 2018A030310634), Guangdong Province Science and Technology Plan projects (No. 2016B0303 05007 and No. 2017B010110011), Guangzhou Science and Technology Project (No. 201804010435), the Foundation of Qiannan Normal University for Nationalities (Nos. QNSY2018JS010 and [2019]QNSYXM05), the Foundation of Science cooperation project of Qiannan (No. (2018)10).

References

[1] L. Amarú, P.-E. Gaillardon, and G. D. Micheli, "BDS-MAJ: a BDD-based Logic Synthesis Tool Exploiting Majority Logic Decomposition," in *DAC*, 2013, pp. 47:1–47:6.

[2] J. R. Burch, E. M. Clarke, and K. L. McMillan, "Sequential Circuit Verification Using Symbolic Model Checking," in *DAC*, 1990, pp. 46–51.

[3] B. Becker and R. Drechsler, "Synthesis for Testability: Circuits Derived from Ordered Kronecker Functional Decision Diagrams," in *ED&TC*, 1995, p. 592.

[4] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 677–691, 1986.

[5] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Transactions of the American Institute of Electrical Engineers*, vol. 57, no. 12, pp. 713–723, 1938.

[6] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," in *DAC*, 1993, pp. 272–277.

[7] T. van Dijk, R. Wille, and R. Meolic, "Tagged BDDs: Combining Reduction Rules from Different Decision Diagram Types," in *FMCAD*, 2017, pp. 108–115.

CNF	Vars	Clauses	BDD	ZDD	TBDD	SDD	ZSDD	TSDD	Best
b1	21	50	196	**72**	**72**	177	105	101	**72**
C17	17	30	112	70	**68**	148	106	98	**68**
cc	115	265	10198	4933	4912	2054	1663	**1561**	**1561**
cht	203	650	13034	6162	6091	4146	3352	**3100**	**3100**
cm138a	50	114	1286	768	768	547	431	**405**	**405**
cm150a	84	202	7150	3495	3463	1622	1300	**1233**	**1233**
cm151a	44	100	1752	980	952	709	552	**513**	**513**
cm152a	20	49	138	106	**52**	151	137	94	**52**
cm162a	73	173	2720	1358	1356	1007	779	**731**	**731**
cm163a	68	157	3630	1840	1820	1222	968	**924**	**924**
cm42a	48	110	818	509	508	626	451	**440**	**440**
cm82a	25	62	538	204	204	244	140	**134**	**134**
cm85a	77	176	5572	2660	2648	1328	1008	**952**	**952**
cmb	62	147	4010	2236	2203	1288	1087	**1024**	**1024**
comp	197	475	23062	11774	11774	2549	2030	**1899**	**1899**
count	184	425	23874	11431	11278	4027	3541	**3324**	**3324**
cu	94	235	8422	4565	4178	2038	1665	**1580**	**1580**
decod	41	122	728	**116**	**116**	395	181	173	**116**
f51m	108	511	9434	**2612**	**2612**	4940	3793	3696	**2612**
ldd	145	414	3824	512	**440**	2085	1350	1237	**440**
majority	14	35	142	65	**62**	136	76	71	**62**
mux	73	240	5260	2207	2006	2071	1746	**1623**	**1623**
parity	61	135	1172	582	582	671	520	**481**	**481**
pcle	66	156	2570	1046	989	1232	1061	**962**	**962**
pcler8	98	220	2908	1274	1189	1524	1346	**1167**	**1167**
pm1	105	245	6936	3543	3523	2431	2120	**2031**	**2031**
sct	175	508	49212	19835	19814	8078	7344	**7175**	**7175**
tcon	65	136	5866	3277	3211	680	482	**451**	**451**
x2	62	166	2178	737	701	827	619	**579**	**579**
z4ml	78	390	6282	1802	1802	2725	1847	**1781**	**1781**
c17	11	18	104	82	**76**	121	85	83	**76**
s208.1	122	285	5812	3261	3234	2023	1774	**1671**	**1671**
s27	18	30	248	149	134	106	72	**68**	**68**
s298	136	363	28238	13320	13320	3864	3424	**3322**	**3322**
s344	193	447	282924	148590	145700	3819	3407	**3227**	**3227**
s382	182	464	34194	17946	17946	5326	4853	**4684**	**4684**
s386	172	506	63248	24907	24907	8920	8303	**8134**	**8134**
s400	186	482	73320	41116	41116	4319	3843	**3697**	**3697**
s444	205	533	56034	30476	30476	4955	4501	**4333**	**4333**
Average	94.8	251.9	19157.6	9503.0	9392.4	2182.8	1847.7	**1763.1**	**1763.1**

TABLE I: Experimental results on benchmarks.

[8] K. Pipatsrisawat and A. Darwiche, "New Compilation Languages Based on Structured Decomposability," in *AAAI*, 2008, pp. 517–522.

[9] A. Darwiche, "SDD: A New Canonical Representation of Propositional Knowledge Bases," in *IJCAI*, 2011, pp. 819–826.

[10] A. Choi and A. Darwiche, "Dynamic Minimization of Sentential Decision Diagrams," in *AAAI*, 2013, pp. 187–194.

[11] S. Bova, "SDDs Are Exponentially More Succinct than OBDDs," in *AAAI*, 2016, pp. 929–935.

[12] M. Nishino, N. Yasuda, S. ichi Minato, and M. Nagata, "Zero-Suppressed Sentential Decision Diagrams," in *AAAI*, 2016, pp. 1058–1066.

[13] K. Pipatsrisawat and A. Darwiche, "A Lower Bound on the Size of Decomposable Negation Normal Form," in *AAAI*, 2010, pp. 345–350.

[14] G. V. den Broeck and A. Darwiche, "On the Role of Canonicity in Knowledge Compilation," in *AAAI*, 2015, pp. 1641–1648.

[15] F. Somenzi, "Cudd: Cu decision diagram package 3.0," http://vlsi.colorado.edu/~fabio, 2014.

[16] T. van Dijk, R. Wille, and R. Meolic, "The TBDD Package," http://fmv.jku.at/tbdd/, 2017.

[17] A. Choi and A. Darwiche, "The SDD Package Version 2.0," http://reasoning.cs.ucla.edu/sdd/, 2018.

[18] M. Nishino, N. Yasuda, S. ichi Minato, and M. Nagata, "ZSDD package," https://github.com/nsnmsak/zsdd, 2016.

[19] S. Panda, F. Somenzi, and B. F. Plessier, "Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams," in *ICCAD*, 1994, pp. 628–631.