

Variants of Tagged Sentential Decision Diagrams

Deyuan Zhong

Department of Computer Science
Jinan University
Guangzhou, China
zhongdeyuan@stu2021.jnu.edu.cn

Mingwei Zhang

Department of Computer Science
Jinan University
Guangzhou, China
mingweizhang@stu2022.jnu.edu.cn

Quanlong Guan

Department of Computer Science
Jinan University
Guangzhou, China
guanql@jnu.edu.cn

Liangda Fang

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

Zhaorong Lai

Department of Computer Science
Jinan University
Guangzhou, China
laizhr@jnu.edu.cn

Yong Lai

College of Computer Science and Technology
JiLin University
Changchun, China
laiy@jlu.edu.cn

Abstract—A recently proposed canonical form of Boolean functions, namely tagged sentential decision diagrams (TSDDs), exploits both the standard and zero-suppressed trimming rules. The standard ones minimize the size of sentential decision diagrams (SDDs) while the zero-suppressed trimming rules have the same objective as the standard ones but for zero-suppressed sentential decision diagrams (ZSDDs). The original TSDDs, which we call zero-suppressed TSDDs (ZTSDDs), firstly fully utilize the zero-suppressed trimming rules, and then the standard ones. In this paper, we present a variant of TSDDs which we call standard TSDDs (STSDDs) by reversing the order of trimming rules. We then prove the canonicity of STSDDs and present the algorithms for binary operations on TSDDs. In addition, we offer two kinds of implementations of STSDDs and ZTSDDs and acquire three variations of the original TSDDs. Experimental evaluations demonstrate that the four versions of TSDDs have the size advantage over SDDs and ZSDDs.

Index Terms—Boolean functions, Combination sets, Decision diagrams

I. INTRODUCTION

Knowledge compilation aims to transform a Boolean function into a tractable representation. Binary decision diagrams (BDDs) [1] is one of the most notable representations that is widely employed for numerous fields of computer science including computer-aided design [2], [3], cryptography [4], [5], formal method [6], [7]. Interestingly, BDDs are a canonical form under the two restrictions: ordering and reduction, that means, any Boolean function has a unique BDD representation. This property reduces the storage space of BDDs and enables an $O(1)$ time equality-test on BDDs.

Following the success of BDDs, a variant zero-suppressed BDDs (ZDDs) was proposed in [8]. ZDDs enjoy the same properties: canonicity and supporting polytime Boolean operations as BDDs. The main difference between BDDs and ZBDDs lies in their different reduction rules. Some applications inspire several extensions of BDD that combine the reduction rules of BDDs and ZBDDs, including tagged BDDs (TBDDs) [9], chain-reduced BDDs (CBDDs) [10], chain-reduced ZDDs (CZDDs) [10] and edge-specified-reduction BDDs (ESRBDDs) [11]. Thanks to the integration of two reduction rules, the above extensions are more compact representations than BDDs and ZDDs.

The theoretical foundation of BDDs is the Shannon decomposition [12], which splits a Boolean function into two subfunctions based on a single variable. Structured decomposition [13], an extension to the Shannon decomposition, splits a Boolean function according to a set of mutually exclusive subfunctions. By using structured decomposition instead of the Shannon decomposition, a novel decision diagram, namely sentential decision diagram (SDD), was developed in [14].

Just as BDDs are characterized by a total order of variables, SDDs are characterized by a variable tree (vtree), that is, a full and binary tree whose leaves are variables. The advantage of SDDs over BDDs is providing a more succinct representation in theory and practice [15], [16]. In addition, [17] proposed the zero-suppressed variant of SDDs (called ZSDDs), which is also based on structured decomposition, and applies the zero-suppressed trimming rules instead of the standard rules used in SDDs. ZSDDs offer a more compact form for sparse Boolean functions compared to SDDs. In contrast, SDDs are more suitable for homogeneous Boolean functions. In order to harness the relative strengths of SDDs and ZSDDs, [18] designed a novel decision diagram, namely *tagged SDDs (TSDDs)*, which combines the standard and zero-suppressed trimming rules.

In this paper, we investigate the variants of TSDDs. To distinguish it from its variants, we call the original TSDD zero-suppressed TSDD (ZTSDD). ZTSDD firstly fully utilizes the zero-suppressed trimming rules before adopting the standard ones. By reversing the order of the trimming rules, we propose the first variant, namely standard TSDD (STSDD). The syntactical definition of STSDD is the same as ZTSDD that is made up of two vtrees and a decomposition node. However, STSDD uses the standard trimming rules as the first rule and the zero-suppressed ones as the second rule. We also propose the semantics for STSDDs and design the trimming rules for STSDDs, obtaining the canonicity property of STSDDs. In addition, we implement these two types of TSDDs in two ways: *node-based* and *edge-based*. Basically, the node-based implementation specifies two vtrees and the decomposition node in a TSDD node. In contrast, the edge-based implementation only keeps the secondary vtree in a TSDD node and associate the edge pointing to each STSDD subnode of the decomposition node with its primary vtree. When a large number of nodes share the same secondary vtree and decomposable node, edge-based implementation utilizes less memory than node-based one. Node-based implementation, on the other hand, uses less memory space to save TSDDs. [18] developed only edge-based implementation of ZTSDDs using C++ language. Some critical data structures, such as unique table and cache table, were built directly on standard template library, making them less efficient. We provide more efficient implementations of four TSDD variations by rewriting such data structures in C language. We also compare SDDs and ZSDDs with the four TSDD variations in terms of size and compilation time of decision diagrams on an extensive set of benchmarks. The experimental results support the effectiveness of our implementation and the relative compactness of

TABLE I: Operations on combination sets.

Operation	Description	Definition
$\mathbf{Q} \cap \mathbf{Q}'$	intersection	$\{\tilde{\mathbf{X}} \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ and } \tilde{\mathbf{X}} \in \mathbf{Q}'\}$
$\mathbf{Q} \cup \mathbf{Q}'$	union	$\{\tilde{\mathbf{X}} \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ or } \tilde{\mathbf{X}} \in \mathbf{Q}'\}$
$\mathbf{Q} \setminus \mathbf{Q}'$	difference	$\{\tilde{\mathbf{X}} \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ and } \tilde{\mathbf{X}} \notin \mathbf{Q}'\}$
$\mathbf{Q} \sqcup \mathbf{Q}'$	orthogonal join	$\{\tilde{\mathbf{X}} \cup \tilde{\mathbf{X}}' \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ and } \tilde{\mathbf{X}}' \in \mathbf{Q}'\}$
$\text{Change}(\mathbf{Q}, x)$	change	$\{\tilde{\mathbf{X}} \cup \{x\} \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ and } x \notin \tilde{\mathbf{X}}\} \cup \{\tilde{\mathbf{X}} \setminus \{x\} \mid \tilde{\mathbf{X}} \in \mathbf{Q} \text{ and } x \in \tilde{\mathbf{X}}\}$

TSDDs over SDDs and ZSDDs on the majority of test-cases.

The rest of this paper is organized as follows. Section 2 provides the preliminaries of Boolean function, combination set, the standard and zero-suppressed trimming rules. In Section 3, we give the syntax of TSDDs and two semantics for TSDDs, obtaining two versions of TSDDs: STSDDs and ZTSDDs. We also design the compressness and trimming rules for STSDDs, gaining the canonicity property of STSDDs and offer two implementations for TSDDs. In Section 4, we develop the algorithm for binary operations of combination sets on STSDDs. Experimental evaluation for comparison among four variations of TSDDs with SDDs and ZSDDs appears in Section 5. Finally, Section 6 concludes this paper.

II. PRELIMINARIES

Throughout this paper, we use lower case letters (e.g., x_1, x_2) for variables, and bold upper case letters (e.g., $\mathbf{X}, \mathbf{Y}$) for sets of variables. For a variable x , we use $\bar{x}$ to denote the negation of x . 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 is a *Boolean function* over $\mathbf{X}$, which is a mapping: $\Sigma_{\mathbf{X}} \mapsto \{0, 1\}$. We use $\mathbf{1}$ (resp. $\mathbf{0}$) for the Boolean function that maps all assignments to 1 (resp. 0). A *combination* $\tilde{\mathbf{X}}$ on $\mathbf{X}$ is a subset of $\mathbf{X}$. Every combination $\tilde{\mathbf{X}}$ corresponds to exactly one truth assignment σ , that is, $x \in \tilde{\mathbf{X}}$ iff $\sigma(x) = 1$. A *combination set* $\mathbf{Q}$ over $\mathbf{X}$ is a collection of combinations on $\mathbf{X}$. It was shown that every combination set can be transformed into a Boolean function, and vice versa [8], [17]. The operations on combination sets include: union $\cup$, intersection $\cap$, difference $\setminus$, orthogonal join $\sqcup$ and change [17]. The definitions of the above operations are illustrated in Table I. We use $\mathbf{U}_{\mathbf{X}}$ for the universe set of combinations on $\mathbf{X}$. For example, $\mathbf{U}_{\{x_1, x_2\}} = \{\{x_1, x_2\}, \{x_1\}, \{x_2\}, \emptyset\}$. We remark that $\mathbf{U}_{\emptyset} = \{\emptyset\}$.

Let $\mathbf{X}$ and $\mathbf{Y}$ be two disjoint and non-empty sets of variables. We say the set $\{(\mathbf{P}_1, \mathbf{S}_1), \dots, (\mathbf{P}_n, \mathbf{S}_n)\}$ is an $(\mathbf{X}, \mathbf{Y})$ -*decomposition* of a combination set $\mathbf{Q}$, iff $\mathbf{Q} = [\mathbf{P}_1 \sqcup \mathbf{S}_1] \cup \dots \cup [\mathbf{P}_n \sqcup \mathbf{S}_n]$ where every $\mathbf{P}_i$ (resp. $\mathbf{S}_i$) is a combination set over $\mathbf{X}$ (resp. $\mathbf{Y}$). A decomposition is *compressed* iff $\mathbf{S}_i \neq \mathbf{S}_j$ for $i \neq j$. An $(\mathbf{X}, \mathbf{Y})$ -decomposition is called an $(\mathbf{X}, \mathbf{Y})$ -*partition*, iff (1) every $\mathbf{P}_i$ is non-empty, (2) $\mathbf{P}_i \cap \mathbf{P}_j = \emptyset$ for $i \neq j$, and (3) $\mathbf{P}_1 \cup \dots \cup \mathbf{P}_n = \mathbf{U}_{\mathbf{X}}$.

A *vtree* is a full binary tree whose leaves are labeled by variables, which generalizes variable orders. For a vtree $\mathbf{T}$, we use $v(\mathbf{T})$ for the set of variables appearing in leaves of $\mathbf{T}$, and $\mathbf{T}_l$ and $\mathbf{T}_r$ for the left and right subtrees of $\mathbf{T}$ respectively. There is a special leaf node labeled by 0 that can be considered as a child of any vtree node and $v(0) = \emptyset$. The notation $\mathbf{T}^1 \preceq \mathbf{T}^2$ denotes that $\mathbf{T}^1$ is a subtree of $\mathbf{T}^2$ and $\mathbf{T}^1 \prec \mathbf{T}^2$ means that $\mathbf{T}^1$ is a proper subtree.

Based on the notion of vtrees, a combination set can be graphically represented by a *structured decomposable diagram* [18].

Definition 1: A structured decomposable diagram is a pair $(\mathbf{T}, \alpha)$ where $\mathbf{T}$ is a vtree and α is recursively defined 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 $\{(p_1, s_1), \dots, (p_n, s_n)\}$ satisfying the following conditions:
 - 1) each p_i is a structured decomposable diagram $(\mathbf{T}_i^1, \beta)$ where $\mathbf{T}_i^1 \preceq \mathbf{T}_l$;
 - 2) each s_i is a structured decomposable diagram $(\mathbf{T}_i^2, \gamma)$ where $\mathbf{T}_i^2 \preceq \mathbf{T}_r$.

Every pair (p_i, s_i) of a decomposition node is called an *element* where p_i is called a *prime* and s_i is called a *sub*.

We hereafter provide two ways to interpret a structured decomposable diagram $(\mathbf{T}^2, \alpha)$ as a combination set, which we call *standard* and *zero-suppressed semantics*. Since the standard semantics depends on an extra vtree $\mathbf{T}^1$, it is a mapping from structured decomposable diagrams and vtrees into combination sets.

Definition 2: Let $\mathbf{T}^1$ be a vtree and $(\mathbf{T}^2, \alpha)$ be a structured decomposable diagram where $\mathbf{T}^2 \preceq \mathbf{T}^1$. The *standard semantics* $\langle \mathbf{T}^1, (\mathbf{T}^2, \alpha) \rangle_s$ is recursively defined as follows:

- $\langle \mathbf{T}^1, (\mathbf{T}^2, \mathbf{1}) \rangle_s = \mathbf{U}_{\mathbf{T}^1}$ and $\langle \mathbf{T}^1, (\mathbf{T}^2, \mathbf{0}) \rangle_s = \emptyset$;
- $\langle \mathbf{T}^1, (\mathbf{T}^2, \varepsilon) \rangle_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)}$ and $\langle \mathbf{T}^1, (\mathbf{T}^2, \bar{\varepsilon}) \rangle_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)} \sqcup (\mathbf{U}_{v(\mathbf{T}^2)} \setminus \{\emptyset\})$;
- $\langle \mathbf{T}^1, (\mathbf{T}^2, \{(p_1, s_1), \dots, (p_n, s_n)\}) \rangle_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)} \sqcup \left[\bigcup_{i=1}^n (\langle \mathbf{T}_l^1, p_i \rangle_s \sqcup \langle \mathbf{T}_r^1, s_i \rangle_s) \right]$.

The standard semantics $\langle \mathbf{T}^1, (\mathbf{T}^2, \alpha) \rangle_s$ contains two combination sets. The *main combination set* is based on $\mathbf{T}^2$ and α . The four terminal nodes $\mathbf{1}$, $\mathbf{0}$, ε and $\bar{\varepsilon}$ represent $\mathbf{U}_{v(\mathbf{T}^2)}$, $\emptyset$, $\{\emptyset\}$ and $\mathbf{U}_{v(\mathbf{T}^2)} \setminus \{\emptyset\}$, respectively. The decomposition node $\{(p_1, s_1), \dots, (p_n, s_n)\}$ denotes the combination set that is the union of the orthogonal join of $\langle p_i \rangle_s$ and $\langle s_i \rangle_s$ for every pair (p_i, s_i) . The *auxiliary combination set* is the universe set over $v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)$. The standard semantics is the orthogonal join of main and auxiliary combination sets. For example, the combination set of $(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})$ is $\mathbf{U}_{v(\mathbf{T}^2)} \sqcup \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)}$, and hence being $\mathbf{U}_{v(\mathbf{T}^1)}$.

The zero-suppressed semantics $\langle \mathbf{T}^1, (\mathbf{T}^2, \alpha) \rangle_z$ is only the main combination set, which can be easily defined. For example, $\langle \mathbf{T}^1, (\mathbf{T}^2, \{(p_1, s_1), \dots, (p_n, s_n)\}) \rangle_z = \bigcup_{i=1}^n (\langle p_i \rangle_s \sqcup \langle s_i \rangle_s)$. We introduce the extra vtree $\mathbf{T}^1$ in the zero-suppressed semantics in accordance with the standard semantics though it is not required for the zero-suppressed semantics.

Based on the standard semantics, we impose some restrictions on structured decomposable diagram and obtain the definition of sentential decision diagram (SDD).

Definition 3: A structured decomposable diagram $(\mathbf{T}, \alpha)$ is a sentential decision diagram, if one of the following holds:

- 1) α is a terminal node labeled by $\mathbf{1}$ or $\mathbf{0}$, and $\mathbf{T} = \mathbf{0}$.
- 2) α is a terminal node labeled by ε or $\bar{\varepsilon}$, and $\mathbf{T}$ is a leaf node.
- 3) α is a decomposition node $\{(p_1, s_1), \dots, (p_n, s_n)\}$, and all of the following hold:
 - $\langle \mathbf{T}_l, p_i \rangle_s \neq \emptyset$ for $1 \leq i \leq n$;
 - $\langle \mathbf{T}_l, p_i \rangle_s \cap \langle \mathbf{T}_l, p_j \rangle_s = \emptyset$ for $i \neq j$;
 - $\bigcup_{i=1}^n \langle \mathbf{T}_l, p_i \rangle_s = \mathbf{U}_{v(\mathbf{T}_l)}$.

The definition of zero-suppressed sentential decision diagram (ZSDD) is the same as SDD, except that (1) we require $\mathbf{T}$ to be the special vtree 0 when α is a terminal node labeled by ε ; (2) the vtree $\mathbf{T}$ can be any leaf node when α is labeled by $\mathbf{1}$; (3) we use the zero-suppressed semantics for the decomposition node.

An SDD can be transformed to an equivalent one with smaller size by the following the standard compressness and trimming rules.

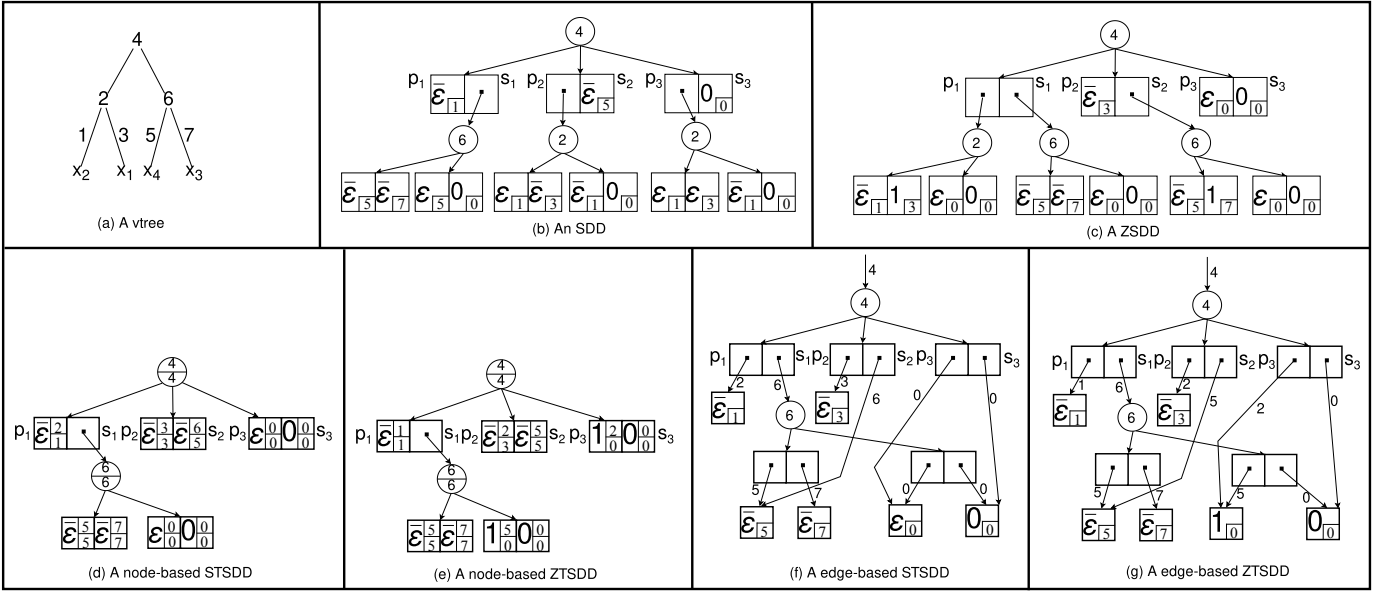


Fig. 1: The vtree and the SDD, ZSDD and TSDD representations of the combination set $\{\{x_1, x_2, x_3, x_4\}, \{x_2, x_3, x_4\}, \{x_1, x_3, x_4\}, \{x_1, x_4\}\}$.

- Standard compression rule (S-compression rule): if $\langle \mathbf{T}_r, s_i \rangle_s = \langle \mathbf{T}_r, s_j \rangle_s$, then replace $(\mathbf{T}, \{(p_1, s_1), \dots, (p_i, s_i), \dots, (p_j, s_j), \dots, (p_n, s_n)\})$ with $(\mathbf{T}, \{(p_1, s_1), \dots, (p', s_i), \dots, (p_n, s_n)\})$ where $\langle \mathbf{T}_l, p' \rangle_s = \langle \mathbf{T}_l, p_i \rangle_s \cup \langle \mathbf{T}_l, p_j \rangle_s$.
- Standard trimming rule (S-trimming rule):

- replace the diagram $(\mathbf{T}, \{(p, s)\})$ by the diagram s .
- if $\langle \mathbf{T}_r, s_1 \rangle_s = \mathbf{U}_{v(\mathbf{T}_r)}$ and $\langle \mathbf{T}_r, s_2 \rangle_s = \emptyset$, then replace the diagram $(\mathbf{T}, \{(p_1, s_1), (p_2, s_2)\})$ by the diagram p_1 .

The S-compression rule combines two elements (p_i, s_i) and (p_j, s_j) when s_i and s_j denotes the same combination set. The two S-trimming rules aim to remove the universe set over a subset of variables in a decomposition node. By repeatedly applying the S-compression and trimming rules, we can create the unique SDD.

Similarly, we can define the zero-suppressed compression and trimming rules for ZSDD.

- Zero-suppressed compression rule (Z-compression rule): if $\langle \mathbf{T}_r, s_i \rangle_z = \langle \mathbf{T}_r, s_j \rangle_z$, then replace $(\mathbf{T}, \{(p_1, s_1), \dots, (p_i, s_i), \dots, (p_j, s_j), \dots, (p_n, s_n)\})$ with $(\mathbf{T}, \{(p_1, s_1), \dots, (p', s_i), \dots, (p_n, s_n)\})$ where $\langle \mathbf{T}_l, p' \rangle_z = \langle \mathbf{T}_l, p_i \rangle_z \cup \langle \mathbf{T}_l, p_j \rangle_z$.
 - Zero-suppressed trimming rule (Z-trimming rule):
- if $\langle \mathbf{T}_l, p_1 \rangle_z = \{\emptyset\}$ (resp. $\langle \mathbf{T}_r, s_1 \rangle_z = \{\emptyset\}$) and $\langle \mathbf{T}_r, s_2 \rangle_z = \emptyset$, then replace the diagram $(\mathbf{T}, \{(p_1, s_1), (p_2, s_2)\})$ by the diagram s_1 (resp. p_1);
 - if $\langle \mathbf{T}_r, s \rangle_z = \emptyset$, then replace the diagram $(\mathbf{T}, \{(p, s)\})$ by the diagram s ;

The Z-compression rule is similar to the S-compression rule except that it uses the zero-suppressed semantics. The two Z-trimming rules seek to eliminate $\{\emptyset\}$. We acquire the canonical representation via utilizing the Z-trimming rule on ZSDDs.

Example 1: Fig. 1(a) shows the vtree $\mathbf{T}$ where its left subtree $\mathbf{T}_l$ involves $\mathbf{X} : \{x_1, x_2\}$ while its right one $\mathbf{T}_r$ involves $\mathbf{Y} : \{x_3, x_4\}$. Fig. 1(b) depicts an SDD representing the combination set $\mathbf{Q} = \{\{x_1, x_2, x_3, x_4\}, \{x_2, x_3, x_4\}, \{x_1, x_3, x_4\}, \{x_1, x_4\}\}$ based on $\mathbf{T}$. The $(\mathbf{X}, \mathbf{Y})$ -partition of $\mathbf{Q}$ contains three elements:

$$\underbrace{(\{x_1, x_2\}, \{x_2\})}_{\mathbf{P}_1}, \underbrace{(\{x_3, x_4\})}_{\mathbf{S}_1}, \underbrace{(\{x_1\}, \{x_3, x_4, x_4\})}_{\mathbf{P}_2}, \underbrace{(\{x_3, x_4\}, \{x_4\})}_{\mathbf{S}_2} \text{ and}$$

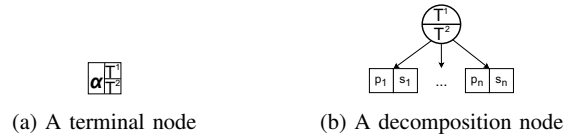


Fig. 2: Two types of nodes of ESDDs

$\underbrace{(\{\emptyset\}, \{\emptyset\})}_{\mathbf{P}_3}, \underbrace{(\{\emptyset\})}_{\mathbf{S}_3}$. Each combination subset $\mathbf{P}_i$ (resp. $\mathbf{S}_i$) corresponds to the node p_i (resp. s_i) of the SDD. The ZSDD representation for $\mathbf{Q}$ with smaller nodes than the SDD one is shown in Fig. 1(c). $\square$

III. TAGGED SENTENTIAL DECISION DIAGRAMS

In this section, we will first provide a general structure, namely extended structured decomposable diagram (ESDD), that is, the syntactic definition for TSDDs, and then with two different semantics, that are, a mapping from ESDDs to combination sets. The ESDD with the standard semantics is called standard TSDD (STSDD) while it is called zero-suppressed TSDD (ZTSDD) under the zero-suppressed semantics. We also present the trimming rules for STSDD so as to reduce the size of STSDD and obtain the canonicity theorem of STSDDs. Finally, we provide two implementations for TSDDs: node-based and edge-based. Hence, we obtain four versions of TSDDs.

A. The Syntax and semantics

In order to facilitate combining two types of trimming rules, we first provide a general structure, namely extended structured decomposable diagram (ESDD).

Definition 4: An ESDD is a tuple $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ s.t. $\mathbf{T}^2 \preceq \mathbf{T}^1$, which is recursively defined as:

- α is a terminal node labeled by one of the four symbols: $\mathbf{1}$, $\mathbf{0}$, ε and $\bar{\varepsilon}$;
- α is a decomposition node $\{(p_1, s_1), \dots, (p_n, s_n)\}$ satisfying the following:
 - each p_i is an ESDD $(\mathbf{T}^3, \mathbf{T}^4, \beta)$ where $\mathbf{T}^4 \preceq \mathbf{T}^3 \prec \mathbf{T}^2$;
 - each s_i is an ESDD $(\mathbf{T}^5, \mathbf{T}^6, \gamma)$ where $\mathbf{T}^6 \preceq \mathbf{T}^5 \prec \mathbf{T}^2$.

An ESDD $F = (\mathbf{T}^1, \mathbf{T}^2, \alpha)$ consists of three components: the primary vtree $\mathbf{T}^1$, the secondary vtree $\mathbf{T}^2$ and the terminal (or decomposition) node α . As seen in Fig. 2(a), when α is a terminal node, the above three components are represented by a square where α is shown in the left side of the square, $\mathbf{T}^1$ in the upper-right corner and $\mathbf{T}^2$ in the lower-right corner. When α is a decomposition node, the primary and secondary vtrees are displayed as a circle with outgoing edges pointing to the elements as shown in Fig. 2(b). Each element (p_i, s_i) is represented by a paired box where the left box represents the prime p_i and the right box stands for the sub s_i . We use $pv(F)$ for the primary vtree of F and $sv(F)$ for the secondary vtree. The size of α , denoted by $|\alpha|$, is the sizes of all of its decompositions.

To interpret ESDDs, we provide the semantics, that is, a mapping from ESDDs into combination sets.

Definition 5: Let $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ be an ESDD. The *standard semantics* $\|(\mathbf{T}^1, \mathbf{T}^2, \alpha)\|_s$ is recursively defined as:

- $\|(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})\|_s = \mathbf{U}_{v(\mathbf{T}^1)}$ and $\|(\mathbf{T}^1, \mathbf{T}^2, \mathbf{0})\|_s = \emptyset$;
- $\|(\mathbf{T}^1, \mathbf{T}^2, \varepsilon)\|_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)}$ and $\|(\mathbf{T}^1, \mathbf{T}^2, \bar{\varepsilon})\|_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)} \sqcup (\mathbf{U}_{v(\mathbf{T}^2)} \setminus \{\emptyset\})$;
- $\|(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \dots, (p_n, s_n)\})\|_s = \mathbf{U}_{v(\mathbf{T}^1) \setminus v(\mathbf{T}^2)} \sqcup \left[\bigcup_{i=1}^n (\|p_i\|_s \sqcup \|s_i\|_s) \right]$.

Since every ESDD involves an extra vtree $\mathbf{T}^1$ compared to structured decision diagrams, the standard semantics for ESDDs is similar to structured decision diagrams (cf. Definition 2).

A standard tagged sentential decision diagram (STSDD) is an ESDD with the following constraints.

Definition 6: An ESDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ is an STSDD, if one of the following holds:

- α is a terminal node labeled by $\mathbf{0}$ and $\mathbf{T}^1 = \mathbf{T}^2 = \mathbf{0}$.
- α is a terminal node labeled by ε and $\mathbf{T}^2 = \mathbf{0}$.
- α is a terminal node labeled by $\bar{\varepsilon}$ and $\mathbf{T}^2$ is a leaf node.
- α is a decomposable node $\{(p_1, s_1), \dots, (p_n, s_n)\}$ and $\{(\|p_1\|_s, \|s_1\|_s), \dots, (\|p_n\|_s, \|s_n\|_s)\}$ is an $(\mathbf{X}, \mathbf{Y})$ -partition where $\mathbf{X} = v(\mathbf{T}_l^1)$ and $\mathbf{Y} = v(\mathbf{T}_r^1)$.

We remark that we use the terminal node ε instead of $\mathbf{1}$ in STSDDs since $\|(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})\|_s = \|(\mathbf{T}^1, \mathbf{0}, \varepsilon)\|_s$ for any vtrees $\mathbf{T}^1$ and $\mathbf{T}^2$.

B. Canonicity

We hereafter design the standard tagged compression and trimming rules for reducing the size of STSDD and obtaining the canonicity property of STSDDs.

- Standard tagged compression rule (ST-compression rule):
if $\|s_i\|_s = \|s_j\|_s$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \dots, (p_i, s_i), \dots, (p_j, s_j), \dots, (p_n, s_n)\})$ with $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \dots, (p'_i, s_i), \dots, (p_n, s_n)\})$ where $\|p'_i\|_s = \|p_i\|_s \cup \|p_j\|_s$.
- Standard tagged trimming rule (ST-trimming rule) (Fig 3):
(a) if $p_1 = (\mathbf{T}^2, \mathbf{T}^3, \alpha)$, $\|s_1\|_s = \{\emptyset\}$ and $\|s_2\|_s = \emptyset$, or $\|p_1\|_s = \{\emptyset\}$, $s_1 = (\mathbf{T}^2, \mathbf{T}^3, \alpha)$ and $\|s_2\|_s = \emptyset$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^2, \mathbf{T}^3, \alpha)$;
- (b) if $\|p_1\|_s = \|s_1\|_s = \{\emptyset\}$, $\|s_2\|_s = \emptyset$ and $\mathbf{T}^2$ is $\mathbf{T}_l^1$ or $\mathbf{T}_r^1$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^3, \mathbf{0}, \varepsilon)$, where $\mathbf{T}^3 = \mathbf{T}_l^1$ when $\mathbf{T}^2 = \mathbf{T}_l^1$ and $\mathbf{T}^3 = \mathbf{T}_r^1$ when $\mathbf{T}^2 = \mathbf{T}_r^1$.
- (c) if $p = (\mathbf{T}_l^1, \mathbf{0}, \varepsilon)$ and $\|s\|_s = \emptyset$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p, s)\})$ with $(\mathbf{0}, \mathbf{0}, \mathbf{0})$.
- (d) if $p_1 = (\mathbf{T}_l^1, \mathbf{T}^3, \alpha)$, $s_1 = (\mathbf{T}_r^2, \mathbf{0}, \varepsilon)$ and $\|s_2\|_s = \emptyset$ (resp. $p = (\mathbf{T}_l^1, \mathbf{0}, \varepsilon)$ and $s = (\mathbf{T}_r^2, \mathbf{T}^3, \alpha)$), then replace $(\mathbf{T}^1, \mathbf{T}^2, \{((\mathbf{0}, \mathbf{0}, \varepsilon), s_1), (p_2, (\mathbf{0}, \mathbf{0}, \mathbf{0}))\})$ (resp. $(\mathbf{T}^1, \mathbf{T}^2, \{(p, s)\})$) with $(\mathbf{T}^1, \mathbf{T}^3, \alpha)$;

- (e) if $p_1 = (\mathbf{T}_l^1, \mathbf{0}, \varepsilon)$, $s_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$ and $\mathbf{T}^3 \preceq (\mathbf{T}_r^2)_l$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1)\})$ with $(\mathbf{T}^1, \mathbf{T}_r^2, \{(s_1, (\mathbf{0}, \mathbf{0}, \varepsilon)), (p_2, (\mathbf{0}, \mathbf{0}, \mathbf{0}))\})$ where $\|p_2\|_s = \mathbf{U}_{(\mathbf{T}_r^2)_l} \setminus \|s_1\|_s$;
- (f) if $p_1 = (\mathbf{T}_l^1, \mathbf{0}, \varepsilon)$, $s_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$ and $\mathbf{T}^3 \preceq (\mathbf{T}_r^2)_r$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1)\})$ with $(\mathbf{T}^1, \mathbf{T}_r^2, \{((\mathbf{0}, \mathbf{0}, \varepsilon), s_1), (p_2, (\mathbf{0}, \mathbf{0}, \mathbf{0}))\})$ where $\|p_2\|_s = \mathbf{U}_{(\mathbf{T}_r^2)_r} \setminus \{\emptyset\}$;
- (g) if $p_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $s_1 = (\mathbf{T}_r^2, \mathbf{0}, \varepsilon)$, $\|s_2\|_s = \emptyset$ and $\mathbf{T}^3 \preceq (\mathbf{T}_l^1)_l$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}_l^1, (p_1, (\mathbf{0}, \mathbf{0}, \varepsilon)), (p_3, s_2))$ where $\|p_3\|_s = \mathbf{U}_{(\mathbf{T}_l^1)_l} \setminus \|p_1\|_s$;
- (h) if $p_1 = (\mathbf{T}^3, \mathbf{T}^4, \alpha)$, $s_1 = (\mathbf{T}_r^2, \mathbf{0}, \varepsilon)$, $\|s_2\|_s = \emptyset$ and $\mathbf{T}^3 \preceq (\mathbf{T}_l^1)_r$, then replace $(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), (p_2, s_2)\})$ with $(\mathbf{T}^1, \mathbf{T}_l^1, \{((\mathbf{0}, \mathbf{0}, \varepsilon), p_1), (p_3, s_2)\})$ where $\|p_3\|_s = \mathbf{U}_{(\mathbf{T}_l^1)_r} \setminus \{\emptyset\}$.

The goal of ST-compression rule is to combine elements with the same subs. The ST-trimming rules are shown in Fig. 3. Rules (a) and (b) are used to eliminate the sub-diagram representing the set $\{\emptyset\}$ whereas rules (c) – (h) aim to reduce the sub-diagram denoting the universe set over a subset of variables. A STSDD is compressed (resp. trimmed), if no ST-compression (resp. trimming) rule can be applied in it. We hereafter state the important property of compressed and trimmed STSDDs.

Theorem 1: Given a vtree $\mathbf{T}$ over $\mathbf{X}$, for any combination set $\mathbf{Q}$ over $\mathbf{X}$, there is a unique compressed and trimmed STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ s.t. $\mathbf{T}^1 \preceq \mathbf{T}$ and $\|(\mathbf{T}^1, \mathbf{T}^2, \alpha)\|_s = \mathbf{Q}$.

Thanks to the additional vtree and the above trimming rules, STSDD has compactness advantages over both SDD and ZSDD.

Example 2: We continue to Example 1. The combination set $\mathbf{Q}$ in SDD, ZSDD and STSDD representations are shown in Fig. 1(b) – (d), respectively. The combination subsets $\mathbf{P}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ can be represented as terminal nodes in SDD while $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{S}_3$ can be in ZSDD. Hence, the combination set has SDD and ZSDD representations of size 9. All of the above 5 combination subsets are represented by terminal nodes in STSDD. The STSDD representation, in comparison, is only 5 in size smaller than SDD and ZSDD. $\square$

C. Zero-suppressed Variant

In a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$, S-trimming rules are applied from the primary vtree $\mathbf{T}^1$ to the secondary one $\mathbf{T}^2$ and Z-trimming rules are applied from the secondary vtree $\mathbf{T}^2$ to the primary vtree of each of the terminal node α , or the prime p_i and the sub s_i of the decomposition node α . We hereafter define a variant of STSDD by reversing the order of trimming rules, that is, Z-trimming rules are implied first and S-trimming rules second.

Definition 7: Let $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ be an ESDD. The *zero-suppressed semantics* $\|(\mathbf{T}^1, \mathbf{T}^2, \alpha)\|_z$ is recursively defined as:

- $\|(\mathbf{T}^1, \mathbf{T}^2, \mathbf{1})\|_z = \mathbf{U}_{v(\mathbf{T}^2)}$ and $\|(\mathbf{T}^1, \mathbf{T}^2, \mathbf{0})\|_z = \emptyset$;
- $\|(\mathbf{T}^1, \mathbf{T}^2, \varepsilon)\|_z = \{\emptyset\}$ and $\|(\mathbf{T}^1, \mathbf{T}^2, \bar{\varepsilon})\|_z = \mathbf{U}_{v(\mathbf{T}^2)} \setminus \{\emptyset\}$;
- $\|(\mathbf{T}^1, \mathbf{T}^2, \{(p_1, s_1), \dots, (p_n, s_n)\})\|_z = \bigcup_{i=1}^n [\mathbf{U}_{v(\mathbf{T}^2) \setminus (pv(p_i) \cup pv(s_i))} \sqcup \|p_i\|_z \sqcup \|s_i\|_z]$.

When α is the terminal node, the zero-suppressed semantics is only the main combination set of α . When α is the decomposition node, besides the main combination set, the zero-suppressed semantics contains an extra combination set, that is, the universal set of $v(\mathbf{T}^2) \setminus (pv(p_i) \cup pv(s_i))$ for each element (p_i, s_i) .

Based on the zero-suppressed semantics, we provide the zero-suppressed variant of TSDD, namely zero-suppressed TSDD (ZTSDD).

Definition 8: An ESDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ is an ZTSDD, if one of the following holds:

- α is a terminal node labeled by $\mathbf{0}$ and $\mathbf{T}^1 = \mathbf{T}^2 = \mathbf{0}$.

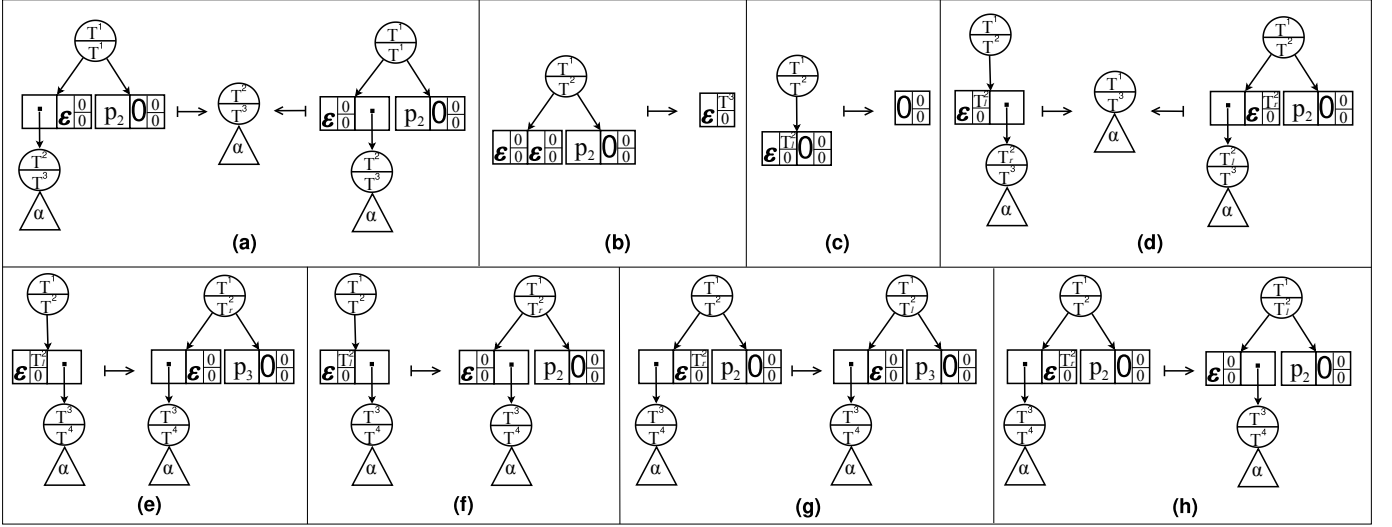


Fig. 3: Trimming rules for STSDD

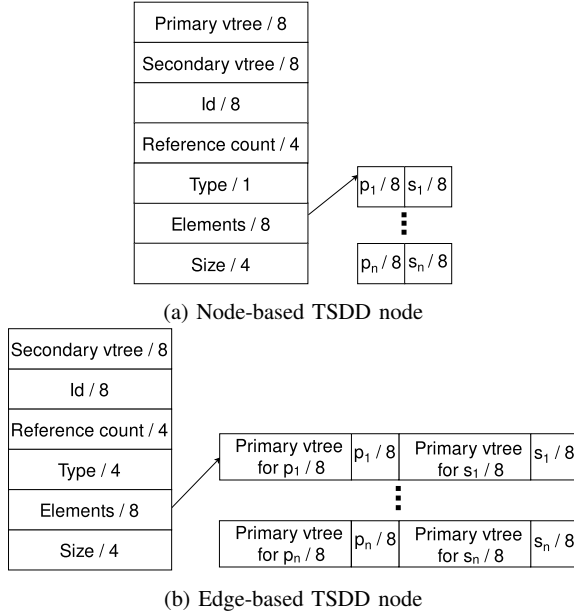


Fig. 4: Two implementations for TSDDs

- α is a terminal node labeled by $\mathbf{1}$ and $\mathbf{T}^2 = 0$.
- α is a terminal node labeled by $\bar{\epsilon}$ and $\mathbf{T}^2$ is a leaf node.
- α is a decomposable node $\{(p_1, s_1), \dots, (p_n, s_n)\}$ and $\{(\|p_1\|_z, \|s_1\|_z), \dots, (\|p_n\|_z, \|s_n\|_z)\}$ is an $(\mathbf{X}, \mathbf{Y})$ -partition where $\mathbf{X} = v(\mathbf{T}_l^2)$ and $\mathbf{Y} = v(\mathbf{T}_r^2)$.

We remark that the terminal node ϵ is omitted in ZTSDD due to the fact that $\|(\mathbf{T}^1, \mathbf{T}^2, \epsilon)\|_z = \|(\mathbf{T}^1, 0, \mathbf{1})\|_z$ for any vtrees $\mathbf{T}^1$ and $\mathbf{T}^2$. In addition, ZTSDD is a canonical form for combination set by applying zero-suppressed tagged compression and trimming rules.

Fig. 1(e) shows the ZTSDD for representing the example. Since ZTSDD both enjoy the advantages of SDD and ZSDD, it has smaller size 5 than SDD and ZSDD, which is the same as STSDD. We remark that ZTSDDs and STSDDs in general have different sizes for representing the same combination set given the same vtree.

D. Edge-based Variant

In Definition 6, both the primary vtree $\mathbf{T}^1$ and the secondary one $\mathbf{T}^2$ are kept in each TSDD node. Such TSDD node is called node-based TSDD. We now introduce the edge-based variant of TSDD. The main distinctions between node-based and edge-based TSDD are: (1) Each edge-based TSDD node only includes the secondary vtree $\mathbf{T}^2$ and the terminal/decomposition node α ; (2) Each element of a decomposition node consists of not only the prime p and the sub s but also two vtrees $\mathbf{T}_p$ and $\mathbf{T}_s$ that are the primary vtrees of p and s , respectively; (3) There is an extra edge pointing to the root node denoting the primary vtree of the root node.

The main data structures of node-based TSDDs and edge-based TSDDs are shown in Fig. 4, respectively. Suppose that the TSDD node has n pairs of primes and subs. We remark that $n = 0$ when the node is a terminal node. In the node-based TSDDs, each node requires at least $41 + 16n$ bytes: 8 bytes for the pointer to the primary vtree, 8 bytes for the pointer to the secondary vtree, 8 bytes for the id of the node in the unique table that ensures no two equivalent TSDD nodes are stored, 4 bytes for the reference count that is used to garbage collection, 1 byte for the type of this node: terminal node or decomposition node, 8 bytes for the pointer to a singly-linked list of pairs of primes and subs, and 4 bytes for the number of elements. The node of edge-based TSDD has the similar data structure with node-based TSDD. However, each edge-based TSDD node do not have the primary vtree and each element has two additional primary vtrees for the prime and the sub. The size of an edge-based node is $33 + 32n$.

Fig. 1(f) and (g) show the edge-based STSDD and ZTSDD representations for the same example. Node-based STSDD representation for $\mathbf{Q}$ needs 449 bytes whereas that of edge-based one requires 432 bytes. Due to this sharing mechanism, edge-based variant consumes less memory than node-based one when numerous nodes share the same secondary vtree and terminal (or decomposable) node. Otherwise, node-based variant is a data structure occupying less memory space for storing TSDDs.

IV. OPERATIONS ON STSDD

In this section, we will design the algorithms of STSDDs for achieving the five operations on combination sets. We first introduce a normalization algorithm that serves as the basis of the above algorithms, and then introduce a unified algorithm that accomplishes

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

Input : F : a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$;
 G : a STSDD $(\mathbf{T}^3, \mathbf{T}^4, \beta)$;
 $\circ$: an operator on combination sets $(\cap, \cup \text{ or } \setminus)$.

Output: H : The resulting STSDD $(\mathbf{T}^5, \mathbf{T}^6, \gamma)$.

```

1 if Some cases are satisfied then return predefined results
2 if Cache( $F, G, \circ$ )  $\neq \text{nil}$  then return Cache( $F, G, \circ$ )
3 if  $\mathbf{T}^1$  and  $\mathbf{T}^3$  are incomparable then
4    $\mathbf{T}^5 \leftarrow \text{Lca}(\mathbf{T}^1, \mathbf{T}^3)$  and  $\mathbf{T}^6 \leftarrow \mathbf{T}^5$ 
5    $F' \leftarrow \text{Normalize1}(F, \mathbf{T}^5)$  and  $G' \leftarrow \text{Normalize1}(G, \mathbf{T}^5)$ 
6 else if  $\mathbf{T}^1 \prec \mathbf{T}^3$  then
7    $\mathbf{T}^5 \leftarrow \mathbf{T}^3$  and  $\mathbf{T}^6 \leftarrow \mathbf{T}^3$ 
8    $F' \leftarrow \text{Normalize1}(F, \mathbf{T}^5)$  and  $G' \leftarrow \text{Normalize2}(G, \mathbf{T}^6)$ 
9 else if  $\mathbf{T}^3 \prec \mathbf{T}^1$  then
10   $\mathbf{T}^5 \leftarrow \mathbf{T}^1$  and  $\mathbf{T}^6 \leftarrow \mathbf{T}^1$ 
11   $F' \leftarrow \text{Normalize2}(F, \mathbf{T}^6)$  and  $G' \leftarrow \text{Normalize1}(G, \mathbf{T}^5)$ 
12 else
13   $\mathbf{T}^5 \leftarrow \mathbf{T}^1$  and  $\mathbf{T}^6 \leftarrow \text{Lca}(\mathbf{T}^2, \mathbf{T}^4)$ 
14   $F' \leftarrow \text{Normalize2}(F, \mathbf{T}^6)$  and  $G' \leftarrow \text{Normalize2}(G, \mathbf{T}^6)$ 
15  $\gamma \leftarrow \emptyset$ 
16 foreach element  $(p_i, s_i)$  of  $F'$  do
17   foreach element  $(q_j, r_j)$  of  $G'$  do
18      $p \leftarrow \text{Apply}(p_i, q_j, \cap)$ 
19     if  $\|p\|_s \neq \emptyset$  then
20        $s \leftarrow \text{Apply}(s_i, r_j, \circ)$ 
21       add element  $(p, s)$  to  $\gamma$ 
22  $H \leftarrow \text{Trim}(\text{Compress}(\mathbf{T}^5, \mathbf{T}^6, \gamma))$ 
23 Cache( $F, G, \circ$ )  $\leftarrow H$ 
24 return  $H$ 

```

the three operations: intersection, union and difference, and followed by two algorithms for orthogonal join and change operations.

A. Apply

The normalization rules can be considered as a reverse of trimming rules. There are two types of normalization rules. Given a vtree $\mathbf{T}^3$ s.t. $\mathbf{T}^1 \preceq \mathbf{T}^3$, the first one is to transform a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ into an equivalent one $(\mathbf{T}^3, \mathbf{T}^4, \beta)$, denoted by $\text{Normalize1}((\mathbf{T}^1, \mathbf{T}^2, \alpha), \mathbf{T}^3)$.

- (a) if $\mathbf{T}^1 = \mathbf{T}^3$, then $\mathbf{T}^4 = \mathbf{T}^2$ and $\beta = \alpha$.
- (b) if $\mathbf{T}^1 \prec \mathbf{T}_l^3$, then $\mathbf{T}^4 = \mathbf{T}^3$ and $\beta = \{((\mathbf{T}^1, \mathbf{T}^2, \alpha), (0, 0, \varepsilon)), (p, (0, 0, 0))\}$ where $\|p\|_s = \mathbf{U}_{\mathbf{T}_l^3} \setminus \|(\mathbf{T}^1, \mathbf{T}^2, \alpha)\|_s$.
- (c) if $\mathbf{T}^1 \prec \mathbf{T}_r^3$, then $\mathbf{T}^4 = \mathbf{T}^3$ and $\beta = \{((0, 0, \varepsilon), (\mathbf{T}^1, \mathbf{T}^2, \alpha)), (p, (0, 0, 0))\}$ where $\|p\|_s = \mathbf{U}_{\mathbf{T}_l^3} \setminus \{\emptyset\}$.

The second type of normalization rules takes a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$ and a vtree $\mathbf{T}^4$ where $\mathbf{T}^2 \preceq \mathbf{T}^4 \preceq \mathbf{T}^1$ as input, and outputs the resulting STSDD, denoted by $\text{Normalize2}((\mathbf{T}^1, \mathbf{T}^2, \alpha), \mathbf{T}^4)$, with the same combination set as $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$.

- (a) if $\mathbf{T}^2 = \mathbf{T}^4$, then $\mathbf{T}^3 = \mathbf{T}^1$ and $\beta = \alpha$.
- (b) if $\mathbf{T}^2 \prec \mathbf{T}_l^4$, then $\mathbf{T}^3 = \mathbf{T}^1$ and $\beta = \{((\mathbf{T}_l^4, \mathbf{T}^2, \alpha), (\mathbf{T}_r^4, 0, \varepsilon)), (p, (0, 0, 0))\}$ where $\|p\|_s = \mathbf{U}_{\mathbf{T}_l^4} \setminus \|(\mathbf{T}^1, \mathbf{T}^2, \alpha)\|_s$.
- (c) if $\mathbf{T}^2 \prec \mathbf{T}_r^4$, then $\mathbf{T}^3 = \mathbf{T}^1$ and $\beta = \{((\mathbf{T}_l^4, 0, \varepsilon), (\mathbf{T}_r^4, \mathbf{T}^2, \alpha))\}$.

The Apply algorithm, illustrated in Algorithm 1, aims to compute the binary operation $\circ$ on two STSDDs $F : (\mathbf{T}^1, \mathbf{T}^2, \alpha)$ and $G : (\mathbf{T}^3, \mathbf{T}^4, \beta)$ where $\circ$ is one of the three operations on combination sets: intersection $(\cap)$, union $(\cup)$ or difference $(\setminus)$. For some simple cases, we can directly return the predefined results (line 1). For example, if $F = (0, 0, 0)$ and $\circ = \cup$, then the resulting STSDD H is G . Now we consider the case where α and

Algorithm 2: OrthogonalJoin(F, G)

Input : F : a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$;
 G : a STSDD $(\mathbf{T}^3, \mathbf{T}^4, \beta)$;

Output: H : The resulting STSDD.

```

1 if  $F = (0, 0, 0)$  or  $G = (0, 0, 0)$  then return  $(0, 0, 0)$  /*  $\|F\|_s = \emptyset$ 
   or  $\|G\|_s = \emptyset$  */
2 if  $F = (0, 0, \varepsilon)$  then return  $G$  /*  $\|F\|_s = \{\emptyset\}$  */
3 if  $G = (0, 0, \varepsilon)$  then return  $F$  /*  $\|G\|_s = \{\emptyset\}$  */
4  $\mathbf{T} \leftarrow$  the least common ancestor of  $\mathbf{T}^1$  and  $\mathbf{T}^3$ 
5 if  $\mathbf{T}^1 \preceq \mathbf{T}_l$  then
6    $\tilde{F} \leftarrow$  an STSDD s.t.  $\|\tilde{F}\|_s = \mathbf{U}_{\mathbf{T}_l} \setminus \|F\|_s$ 
7    $H \leftarrow \text{Trim}((\mathbf{T}, \mathbf{T}, \{(F, G), (\tilde{F}, (0, 0, 0))\}))$ 
8 else
9    $\tilde{G} \leftarrow$  an STSDD s.t.  $\|\tilde{G}\|_s = \mathbf{U}_{\mathbf{T}_l} \setminus \|G\|_s$ 
10   $H \leftarrow \text{Trim}((\mathbf{T}, \mathbf{T}, \{(G, F), (\tilde{G}, (0, 0, 0))\}))$ 
11 return  $H$ 

```

β are decomposition nodes. In general, $\mathbf{T}^1 \neq \mathbf{T}^2$ and $\mathbf{T}^3 \neq \mathbf{T}^4$. Therefore, it is necessary to convert F and G into their equivalent STSDD F' and G' with the same primary vtree $\mathbf{T}^5$ and secondary vtree $\mathbf{T}^6$ via normalization rules (lines 3 – 14). If $\mathbf{T}^1$ and $\mathbf{T}^3$ are incomparable, then both $\mathbf{T}^5$ and $\mathbf{T}^6$ are the least common ancestor of $\mathbf{T}^1$ and $\mathbf{T}^3$. The transformed STSDDs F' and G' can be obtained via the first type of normalization rules. The other cases can be handled similarly. Let $F' = (\mathbf{T}^5, \mathbf{T}^6, \{(p_1, s_1), \dots, (p_n, s_n)\})$ and $G' = (\mathbf{T}^5, \mathbf{T}^6, \{(q_1, r_1), \dots, (q_m, r_m)\})$. It is easily verified that $H = (\mathbf{T}^5, \mathbf{T}^6, \gamma)$ where $\gamma = \{(p_i \cap q_j, s_i \circ r_j) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq m \text{ and } p_i \cap q_j \neq \emptyset\}$ (lines 16 – 21). Finally, compressing and trimming rules will be performed on H to gain the canonicity property (line 22). In addition, we use the cache table to avoid the recomputation on the same TSDDs and operation (lines 2 & 23). Let n be the number of subtrees of $\mathbf{T}^5$, and $|\alpha|$ and $|\beta|$ the size of α and β , respectively. The Apply algorithm runs in $O(n \cdot |\alpha| \cdot |\beta|)$ without the compression rules. When we consider compressing TSDDs, the time complexity is exponential in $|\alpha|$ and $|\beta|$ in the worst case. The above time complexity result of the Apply algorithm still holds for SDDs. It however was demonstrated in [19] that compiling any combination set into compressed SDDs is significantly more efficient than without compressed SDDs. The application of compression rules results in a canonical form of SDDs and hence stipulating that no two SDDs representing the same combination set are stored in the unique table, and facilitating caching in practice. As an extension to SDDs, TSDDs have many characteristics in common with SDDs. We focus on only compressed TSDDs in the remaining of this paper.

B. Orthogonal Join and Change

We begin by introducing the algorithm for orthogonal join, illustrated in Algorithm 2. Assume that we are given two STSDDs $F = (\mathbf{T}^1, \mathbf{T}^2, \alpha)$ and $G = (\mathbf{T}^3, \mathbf{T}^4, \beta)$. Algorithm 2 requires that F and G are orthogonal, that is, $\mathbf{T}^1$ and $\mathbf{T}^3$ are incomparable. The resulting STSDD is also the empty set, if one of the STSDDs F and G is the empty set $\emptyset$ (line 1). In the case where F (resp. G) denotes the combination set $\{\emptyset\}$, the outcome is G (resp. F) (lines 2 & 3). In general, we let $\mathbf{T}$ be the least common ancestor of $\mathbf{T}^1$ and $\mathbf{T}^3$. If $\mathbf{T}^1$ is a subtree of the left child of $\mathbf{T}$, then the result STSDD H is $(\mathbf{T}, \mathbf{T}, \{(F, G), (\tilde{F}, (0, 0, 0))\})$ where $\|\tilde{F}\|_s = \mathbf{U}_{\mathbf{T}_l} \setminus \|F\|_s$ (lines 5 – 7). The opposite direction can be similarly handled (lines 8 – 10). Algorithm 2 runs in a constant time.

Another basic operation for combination set is change. It takes a STSDD F and a variable x as inputs, and outputs a STSDD G

Algorithm 3: $\text{Change}(F, x)$

Input : F : a STSDD $(\mathbf{T}^1, \mathbf{T}^2, \alpha)$;
 x : a variable.

Output: G : The resulting TSDD.

```

1  $\mathbf{T}^3 \leftarrow$  the leaf node labeled by  $x$ 
2 if  $F = (0, 0, \varepsilon)$  then return  $(\mathbf{T}^3, \mathbf{T}^3, \bar{\varepsilon})$  /*  $\|F\|_s = \{\emptyset\}$  */
3 if  $F = (\mathbf{T}^3, \mathbf{T}^3, \bar{\varepsilon})$  then return  $(0, 0, \varepsilon)$  /*  $\|F\|_s = \{\{x\}\}$  */
4 if  $F = (0, 0, 0)$  or  $F = (\mathbf{T}^3, 0, \varepsilon)$  then return  $F$  /*  $\|F\|_s = \emptyset$  or  $\|F\|_s = \{\{x\}, \emptyset\}$  */
5 if  $\mathbf{T}^3 \prec \mathbf{T}^1$  and  $\mathbf{T}^3$  is not a subtree of  $\mathbf{T}^2$  then return  $F$ 
6 if  $\text{Cache}(F, x, \text{Change}) \neq \text{nil}$  then return  $\text{Cache}(F, x, \text{Change})$ 
7 if  $\mathbf{T}^1$  and  $\mathbf{T}^3$  are incomparable then
8    $G \leftarrow \text{OrthogonalJoin}(F, (\mathbf{T}^3, \mathbf{T}^3, \bar{\varepsilon}))$ 
9 else if  $\mathbf{T}^3 = \mathbf{T}^2$  then
10    $\mathbf{T}_p^2 \leftarrow$  the parent node of  $\mathbf{T}^2$ 
11   if  $\mathbf{T}^2 = (\mathbf{T}_p^2)_l$  then
12      $H \leftarrow$  an STSDD s.t.  $\|H\|_s = \mathbf{U}_{(\mathbf{T}_p^2)_l} \setminus \{\emptyset\}$ 
13      $G \leftarrow (\mathbf{T}^1, \mathbf{T}_p^2, \{((0, 0, \varepsilon), ((\mathbf{T}_p^2)_r, 0, \varepsilon)), (H, (0, 0, 0))\})$ 
14   else /*  $\mathbf{T}^2 = (\mathbf{T}_p^2)_r$  */
15      $G \leftarrow (\mathbf{T}^1, \mathbf{T}_p^2, \{(((\mathbf{T}_p^2)_l, 0, \varepsilon), (0, 0, \varepsilon))\})$ 
16 else /*  $\mathbf{T}^3 \prec \mathbf{T}^2$  */
17    $\gamma \leftarrow \emptyset$ 
18   foreach element  $(p_i, s_i)$  of  $\alpha$  do
19     if  $\mathbf{T}^3 \preceq \mathbf{T}_l^2$  then
20        $\text{add element } (\text{Change}(p_i, X), s_i)$  to  $\gamma$ 
21     else /*  $\mathbf{T}^3 \preceq \mathbf{T}_r^2$  */
22        $\text{add element } (p_i, \text{Change}(s_i, X))$  to  $\gamma$ 
23    $G \leftarrow (\mathbf{T}^1, \mathbf{T}^2, \gamma)$ 
24  $G \leftarrow \text{Trim}(G)$ 
25  $\text{Cache}(F, x, \text{Change}) \leftarrow G$ 
26 return  $G$ 

```

s.t. $\|G\|_s = \text{Change}(\|F\|_s, x)$. Let $\mathbf{T}^3$ be the leaf vtree node with the label x . We first consider three special cases. If F denotes the combination set $\{\emptyset\}$, then the resulting STSDD G represents $\{\{x\}\}$, and vice versa (lines 2 & 3). If one of the following three cases hold: (1) $F = (0, 0, 0)$; or (2) $F = (\mathbf{T}^3, 0, \varepsilon)$; or (3) $\mathbf{T}^3 \prec \mathbf{T}^1$ and $\mathbf{T}^3$ is not a subtree of $\mathbf{T}^2$, then the change operation do not modify the input STSDD F (lines 4 & 5). In the case where $\mathbf{T}^1$ and $\mathbf{T}^3$ are incomparable, then the change of F by x is the orthogonal join of the two STSDDs F and $(\mathbf{T}^3, \mathbf{T}^3, \bar{\varepsilon})$. If none of the above cases holds, then $\mathbf{T}^3 \preceq \mathbf{T}^2$. We analyze the following two cases: $\mathbf{T}^3 = \mathbf{T}^2$ and $\mathbf{T}^3 \prec \mathbf{T}^2$. In the case where $\mathbf{T}^3 = \mathbf{T}^2$, we construct G as $(\mathbf{T}^1, \mathbf{T}_p^2, \{((0, 0, \varepsilon), ((\mathbf{T}_p^2)_r, 0, \varepsilon)), (H, (0, 0, 0))\})$ where H denotes the complement of $\{\emptyset\}$ if $\mathbf{T}^2$ is the left child of its parent $\mathbf{T}_p^2$ (lines 9 – 13); and as $(\mathbf{T}^1, \mathbf{T}_p^2, \{(((\mathbf{T}_p^2)_l, 0, \varepsilon), (0, 0, \varepsilon))\})$ if $\mathbf{T}^2$ is the right child of $\mathbf{T}_p^2$ (lines 14 & 15). In the case where $\mathbf{T}^3 \prec \mathbf{T}^2$, α must be a decomposition node. We recursively apply the change operation on the prime p_i of elements of α if $\mathbf{T}^3 \preceq \mathbf{T}_l^2$ (lines 19 & 20), and on the sub s_i if $\mathbf{T}^3 \preceq \mathbf{T}_r^2$ (lines 21 & 22). Finally, we use the trimming rules on G (line 24). The algorithm uses the cache table to avoid the recomputation on the same STSDD and variable (lines 6 & 25) and runs in linear time w.r.t. $|F|$.

V. EXPERIMENTAL RESULTS

In this section, we compare four variants of TSDDs against SDDs and ZSDDs with respect to their compactness in four categories of benchmarks: dictionary, n -queens problems, safe petri nets and digital circuits. For convenience, NSTSDD, NZTSDD, ESTSDD, and EZTSDD are the abbreviations for node-based

TSDD, node-based ZTSDD, edge-based STSDD, and edge-based ZTSDD, respectively. We implemented an efficient TSDD package with the proposed algorithms in C language. We have also devised minimization algorithm for TSDDs via so as to searching a good vtree and integrate this algorithm into our package because the size of TSDDs is sensitive to the vtree. Due to the space limit, we do not present minimization algorithm in this paper, which will be clarified in future work. All experiments were carried out on a machine equipped with an Intel Core i7-8086K 4GHz CPU and 64GB RAM. Table II shows the experimental results of 6 decision diagrams for 85 test cases across 4 benchmark categories on size and time. The columns “size” denotes the size of compiled decision diagrams and “time” the overall compilation runtime in seconds. The smallest sizes among six decision diagrams are highlighted in bold font. The last column shows the smallest size among 4 variants of TSDDs. The entry “-” denotes a failed compilation due to the timeout of 2 hours.

The dictionaries we use are the English words in file /usr/shar/dict/words on MacOS system with 235,886 words of length up to 24 from 54 symbols and the password list with 979,247 words of length up to 32 from 79 symbols [20]. A dictionary will be encoded in two ways: *binary* and *one-hot*. We consider two sets of symbols: the *compact form* consisting of the symbols only found in the dictionary, and the *ASCII form* consisting of all 128 characters. We can compile all of the 8 test cases of dictionaries in ZSDDs and 4 variants of TSDDs. However, SDD compilation fails in the password dictionary. In addition, ZSDDs and TSDDs have a significant size advantage over SDDs. Especially for words in one-hot encoding, NSTSDDs and ESTSDDs have over 96% fewer size than SDDs. This is because the zero-suppressed trimming rule has a tremendous benefit to reduce the size of decision diagrams for dictionaries. Finally, NSTSDD performs the best in 6 out of 8 test cases and hence being an efficient representation for dictionaries in terms of time.

The n -Queens problem aims to place n queens in such a manner on an $n \times n$ chessboard that no two queens can attack each other by being in the same row, column or diagonal. We also have two ways (one-hot and binary) to encode this problem. Firstly, one of the variants of TSDDs is the most compact representation in 13 out of 14 test cases. This indicates that combining two trimming rules can result in a smaller decision diagram than using one trimming rule. Secondly, compilations in SDDs, NZTSDDs and EZTSDDs are more effective than the other three. However, the size of SDD representation is obviously larger than others. Specifically, 13- and 14-Queens problems in one-hot encoding are of size 2,096,517 and 8,604,470 that are approximately 5.12 and 4.62 times larger than NSTSDDs, respectively. Finally, ZSDD compilation is time-consuming, especially, 13- and 14-Queens problem in binary encoding take 5,061 and 5,943 seconds longer than NZTSDDs by high factors of 40.50 and 13.20, respectively.

Petri nets are a popular graphical modeling tool for representing and analyzing concurrent systems. A Petri net is safe iff there is at most one token in each one of its places. We utilize decision diagrams to denote the set of reachable states of safe Petri net. The Petri net benchmark comes from the 2018 Model Checking Contest (<https://mcc.lip6.fr/2018/>). As for the size, one of the variants TSDDs performs the best in 20 out of 21 test cases. In particular, for the two large test cases: NQueens-PT-08 and ParamProductionCell-PT-4, the minimal size among TSDDs are 20.3% and 97.6% smaller than the size of SDDs. Moreover, ZSDDs fails in compilation of the above two test cases.

The final benchmark we consider is digital circuits. We use

TABLE II: The comparison among SDDs, ZSDDs, NSTSDDs, NZTSDDs, ESTSDDs and EZTSDDs over 4 categories of benchmarks

Benchmarks	SDD			ZSDD			NSTSDD			NZTSDD			ESTSDD			EZTSDD			best size
	size	time		size	time		size	time		size	time		size	time		size	time		
Words-Binary-Compact	1,562,787	100.735		922,405	34.884		935,605	23.289		636,393	41.198		937,770	41.463		1,005,408	60.207		636,393
Words-Binary-ASCII	2,356,875	110.015		1,610,952	64.302		1,602,317	38.075		802,562	65.870		1,683,324	78.939		1,780,959	98.862		802,562
Words-OneHot-Compact	19,205,734	755.775		593,624	26.218		593,624	18.800		706,927	68.202		593,766	28.809		706,968	86.394		593,624
Words-OneHot-ASCII	46,281,905	2,472.188		594,124	27.261		594,100	16.834		707,912	61.939		594,126	31.508		707,914	80.362		594,100
Passwords-Binary-Compact	-	-		3,488,141	339.656		3,440,014	369.770		3,464,514	437.667		3,488,592	755.580		3,834,894	767.570		3,440,014
Passwords-Binary-ASCII	-	-		5,664,643	764.867		5,579,382	711.813		5,733,694	721.128		5,732,059	1,337.874		6,061,272	1,336.517		5,579,382
Passwords-OneHot-Compact	-	-		2,243,018	266.236		2,243,016	265.143		2,663,610	1,620.616		2,243,018	457.473		2,663,610	1,965.936		2,243,016
Passwords-OneHot-ASCII	-	-		2,258,146	257.638		2,258,216	284.455		2,681,843	1,052.822		2,258,146	512.400		2,681,843	1,190.616		2,258,146
8-Queens-Binary	1,336	0.045		858	0.092		830	0.083		960	0.065		854	0.068		952	0.07		830
9-Queens-Binary	4,696	0.185		3,001	0.311		2,897	0.332		3,398	0.193		3,052	0.302		3,422	0.216		2,897
10-Queens-Binary	6,017	2.263		6,839	1.013		6,222	0.664		6,568	1.617		7,450	0.794		8,168	0.692		6,222
11-Queens-Binary	32,835	6.668		21,314	30.874		19,441	2.246		20,453	8.654		19,826	140.624		19,123	21.130		19,123
12-Queens-Binary	118,925	30.833		102,345	26.736		81,192	11.592		117,215	20.406		97,545	2,247.166		89,546	54.305		81,192
13-Queens-Binary	666,910	52.210		433,856	5,061.758		35,2637	160.228		511,928	124.993		466,128	1,171.961		48,6257	241.812		35,2637
14-Queens-Binary	2,610,502	556.376		1,871,878	5,943.739		1,554,825	682.726		2,340,878	450.193		2,134,480	511.551		2,371,137	366.242		1,554,825
8-Queens-OneHot	2,691	0.178		953	0.266		763	0.167		832	0.144		730	0.147		837	0.208		730
9-Queens-OneHot	12,526	0.359		3,213	0.549		2,616	0.344		3,170	0.301		2,604	0.349		3,171	0.338		2,604
10-Queens-OneHot	24,542	0.903		7,213	1.517		6,677	3.525		7,462	1.368		6,520	4.481		7,467	1.099		6,520
11-Queens-OneHot	95,390	4.049		22,729	15.659		21,340	19.492		25,780	4.614		21,585	138.004		26,015	40.781		21,340
12-Queens-OneHot	376,669	77.366		98,148	168.901		94,134	798.095		100,139	182.361		91,622	798.389		115,873	117.148		91,622
13-Queens-OneHot	2,096,517	231.321		431,436	92.528		409,525	677.2		491,912	279.847		416,729	974.899		523,479	278.24		409,525
14-Queens-OneHot	8,604,470	623.433		1,923,847	4,313.754		1,863,942	3,102.857		2,348,386	620.163		1,822,922	1,111.114		2,348,381	603.111		1,822,922
AutoFlight-PT-06a	2,276	167.958		-	-		855	5060.282		1,228	254.188		889	7,185.497		-	-		855
Dekker-PT-015	606	5.094		3,562	216.347		408	7.699		393	5.665		418	7.349		399	5.949		393
DiscoveryGPU-PT-14a	1,188	3,758.586		-	-		372	627.382		-	-		496	1,681.220		739	6,228.986		372
DLCround-PT-04a	1,016	49.017		734	742.783		463	157.861		547	16.599		514	335.422		571	16.350		463
FlexibleBarrier-PT-12a	-	-		-	-		445	2,447.236		732	2,924.438		474	2,282.550		-	-		445
IBM319-PT-none	3,013	3.606		726	16.185		837	19.177		810	11.799		873	9.773		-	-		810
LampportFastMutEx-PT-4	43,597	418.267		-	-		11,087	1,131.252		24,759	614.258		4,241	843.778		7,210	204.395		4,241
MAPKbis-PT-5320	558	64.483		771	1,997.135		-	-		415	86.565		-	-		372	66.46		372
NeoElection-PT-3	6,039	29.293		-	-		2,395	3,800.556		1,908	148.984		-	-		1,882	83.911		1,882
NQueens-PT-08	243,080	170.705		-	-		229,506	2,136.447		271,281	173.608		193,757	1,290.575		316,199	240.006		193,757
ParamProductionCell-PT-4	322,645	1,214.601		-	-		-	-		7,755	1,764.348		-	-		20,939	797.247		7,755
Peterson-PT-2	3,477	4.727		1,821	24.098		2,207	38.392		1,128	9.160		1,737	48.345		1,164	19.055		1,128
Philosophers-PT-000010	1,330	2,420		755	114.018		6,165	39.386		496	1.365		221	7.882		505	1.540		221
Raft-PT-03	375	1.955		432	2.423		234	1.314		292	1.937		239	1.462		314	1.508		234
Railroad-PT-010	2,462	6.619		16,033	694.707		1,191	20.191		1,249	8.802		1,352	8.949		-	-		1,191
ResAllocation-PT-R020C002	1,876	228.02		-	-		1,334	2,935.194		1,162	658.513		965	1,564.616		808	186.662		808
Referendum-PT-020	413	1.598		267	328.033		173	1.497		258	3.005		238	1.689		224	1.458		173
RwMutex-PT-0020w0010	829	0.918		797	2.007		455	4.042		586	1.377		52,032	5,796.452		465	1.831		455
SafeBus-PT-03	3,113	1.525		6,871	7.664		1,565	1.829		2,152	2.520		1,360	2.596		1,543	2.104		1,360
SharedMemory-PT-000010	2,339	3.768		1,746	11.448		809	3.592		1,414	6.219		680	3.438		739	10.231		680
SimpleLoadBal-PT-10	16,136	1,040.229		-	-		-	-		5,365	499.745		-	-		85,566	5,353.878		5,365
9symml	27,119	80.097		22,152	2,102.474		19,729	23.757		24,399	120.719		22,520	585.506		16,442	100.958		16,442
alu2	-	-		-	-		-	-		13,658	573.479		-	-		13,208	1,219.279		13,208
apex6	-	-		-	-		-	-		1,166,371	5,573.009		-	-		-	-		1,166,371
apex7	7,800	13.541		8,688	17.721		20,022	173.013		5,958	18.721		-	-		12,972	37.326		5,958
b9	8,096	9.849		19,066	121.499		24,115	696.112		18,192	18.052		69,642	210.751		12,719	20.033		12,719
C432	11,190	9.503		-	-		-	-		12,990	205.840		17,113	101.410		30,777	69.729		12,990
C499	294,686	1,560.059		-	-		-	-		-	-		-	-		-	-		-
c8	19,656	49.683		66,069	972.876		19,681	177.154		15,849	67.920		14,644	29.474		17,382	76.453		14,644
count	4,027	2.018		4,126	3.008		5,175	6.823		6,051	7.400		199,478	248.969		2,864	6.569		2,864
example2	9,246	18.161		9,925	54.859		8,597	204.963		11,172	31.125		19,650	27.535		18,079	64.686		8,597
frg1	137,905	60.219		-	-		-	-		200,455	1,501.116		190,457	3,129.121		121,959	638.711		121,959
lal	8,540	4.178		8,172	48.631		40,712	266.991		5,978	9.787		12,061	8.583		75,768	40.378		5,978
mux	2,071	0.884		8,588	6.445		5,272	3.578		1,718	1.701		1,788	0.895		3,339	1.995		1,718
sct	8,078	6.147		71,845	90.752		39,136	89.677		7,529	9.910		15,638	11.557		7,777	9.230		7,529
term1	100,589	371.290		-	-		-	-		126,324	554.553		-	-		3,500,493	4,524.447		126,324
tt2	24,619	53.182		-	-		29,290	526.672		17,382	123.076		19,868	201.311		32,908	224.050		17,382
unreg	3,973	2.256		4,178	2.297		8,188	7.457		3,782	3.724		24,445	11.645		3,850	4.620		3,782
vda	-	-		-	-		-	-		17,338	652.971		-	-		-	-		17,338
x4	36,364	272.750		-	-		-	-		24,667	316.670		-	-		25,358	439.805		24,667
c1908	85,725,495	5,324.091		-	-		-	-		-	-		-	-		-	-		-
c432	11,647	25.205		-	-		-	-		19,277	21.255		126,779	1516.911		117,939	117.537		19,277
c499	131,517	29.554		-	-		-	-		245,306	81.118		307,962	297.176		160,305	70.929		160,305
s1196	-	-		-	-		-	-		-	-		-	-		174,796	1631.269		174,796
s1238	96,122	1,169.499		-	-		-	-		-	-		-	-		-	-		-
s1494	-	-		-	-		-	-		-	-		-						

In summary, we can observe that no single decision diagram dominates all categories of benchmarks in terms of size and compilation time. TSDDs are a significant representation of Boolean functions as an important addition to SDDs and ZSDDs,

VI. CONCLUSION

In this paper, we design four variants of TSDDs that mixes standard and zero-suppressed trimming rules. We divide TSDDs into STSDDs and ZTSDDs according to the order of trimming rules. The former requires the standard trimming rules to be applied first, and the latter requires the zero-suppressed trimming rules to be utilized first. In addition, we provide two approaches to implementing TSDDs: node-based and edge-based. Node-based implementation stores both primary vtree and secondary vtree in a TSDD node, and edge-based implementation store primary vtree as an edge pointing to a TSDD node. Therefore, we obtain four different kinds of TSDDs: node-based STSDDs, node-based ZTSDDs, edge-based STSDDs and edge-based ZTSDDs. We design their syntax and semantics and provide design three algorithms: Apply, OrthogonalJoin and Change for STSDDs to implement the corresponding operations over combination sets. We finally conduct experiments on four benchmarks, which confirms that TSDDs are a more compact form compared to SDDs and ZSDDs.

REFERENCES

- [1] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 677–691, 1986.
- [2] S. Thijssen, S. K. Jha, and R. Ewetz, "COMPACT: Flow-Based Computing on Nanoscale Crossbars With Minimal Semiperimeter and Maximum Dimension," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4600–4611, 2022.
- [3] R. Matsuo and S. Minato, "Space and Power Reduction in BDD-based Optical Logic Circuits Exploiting Dual Ports," in *Proceedings of 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE-2022)*, 2022, pp. 1071–1076.
- [4] J. Zhang, W. Qi, T. Tian, and Z. Wang, "Further Results on the Decomposition of an NFSR Into the Cascade Connection of an NFSR into an LFSR," *IEEE Transactions on Information Theory*, vol. 61, no. 1, pp. 645–654, 2015.
- [5] D. Knichel, P. Sasdrich, and A. Moradi, "SILVER - Statistical Independence and Leakage Verification," in *Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT-2020)*, ser. Lecture Notes in Computer Science. Springer, 2020, vol. 12491, pp. 787–816.
- [6] A. Mahzoon and R. Drechsler, "Late Breaking Results: Polynomial Formal Verification of Fast Adders," in *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC-2021)*, 2021, pp. 1376–1377.
- [7] C. Wei, Y. Tsai, C. Jhang, and J. R. Jiang, "Accurate bdd-based unitary operator manipulation for scalable and robust quantum circuit verification," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC-2022)*, 2022, pp. 523–528.
- [8] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," in *Proceedings of the 30th International Design Automation Conference (DAC-1993)*, 1993, pp. 272–277.
- [9] T. van Dijk, R. Wille, and R. Meolic, "Tagged BDDs: Combining Reduction Rules from Different Decision Diagram Types," in *Proceedings of the 17th International Conference on Formal Methods in Computer-Aided Design (FMCAD-2017)*. IEEE, 2017, pp. 108–115.
- [10] R. E. Bryant, "Chain Reduction for Binary and Zero-Suppressed Decision Diagrams," *Journal of Automated Reasoning*, vol. 64, p. 1361–1391, 2020.
- [11] J. Babar, C. Jiang, G. Ciardo, and A. Miner, "CESRBDDs: binary decision diagrams with complemented edges and edge-specified reductions," *International Journal on Software Tools for Technology Transfer*, vol. 24, p. 89–109, 2022.
- [12] 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.
- [13] K. Pipatsrisawat and A. Darwiche, "New Compilation Languages Based on Structured Decomposability," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-2008)*, 2008, pp. 517–522.
- [14] A. Darwiche, "SDD: A New Canonical Representation of Propositional Knowledge Bases," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-2011)*, 2011, pp. 819–826.
- [15] S. Bova, "SDDs Are Exponentially More Succinct than OBDDs," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*, 2016, pp. 929–935.
- [16] A. Choi and A. Darwiche, "Dynamic Minimization of Sentential Decision Diagrams," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-2013)*, 2013, pp. 187–194.
- [17] M. Nishino, N. Yasuda, S. ichi Minato, and M. Nagata, "Zero-Suppressed Sentential Decision Diagrams," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*, 2016, pp. 1058–1066.
- [18] L. Fang, B. Fang, H. Wan, Z. Zheng, L. Chang, and Q. Yu, "Tagged Sentential Decision Diagrams: Combining Standard and Zero-suppressed Compression and Trimming Rules," in *Proceedings of the 38th IEEE/ACM International Conference on Computer-Aided Design (ICCAD-2019)*, 2019, pp. 1–8.
- [19] G. V. den Broeck and A. Darwiche, "On the Role of Canonicity in Knowledge Compilation," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-2015)*, 2015, pp. 1641–1648.
- [20] R. E. Bryant, "Supplementary Material on Chain Reduction for Binary and Zero-Suppressed Decision Diagrams," <http://www.cs.cmu.edu/~bryant/bdd-chaining.html>, 2020.
- [21] R. Wille and R. Drechsler, "BDD-Based Synthesis of Reversible Logic for Large Functions," in *Proceedings of the 46th Annual Design Automation Conference (DAC-2009)*, 2009, p. 270–275.
- [22] L. Amarú, P.-E. Gaillardon, and G. D. Micheli, "BDS-MAJ: A BDD-Based Logic Synthesis Tool Exploiting Majority Logic Decomposition," in *Proceedings of the 50th Annual Design Automation Conference (DAC-2013)*, 2013.