ISCAS-SKLCS-11-17 September, 2011

中国科学院软件研究所
计算机科学实验室报告

Complexity Issues of Ternary Boolean Diagrams

by

Wenhui Zhang

State key Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
Beijing 100190. China

Complexity Issues of Ternary Boolean Diagrams

Wenhui Zhang
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
P.O.Box 8718, Beijing 100190, China

21 September 2011

1 Introduction

Ternary Boolean diagrams (TBDs) were introduced in [5] for boolean function manipulation and model checking. The complexity issues of TBDs with respect to the representation succinctness is investigated in this paper. The representation succinctness is explained via a comparison of TBDs and binary decision diagrams (BDDs) [4, 1, 2]. The rest of his paper is organized as follows. The basic idea of TBDs is presented in section 2. In Section 3, two formulas are used to demonstrate different kinds of differences in the representation succinctness between TBDs and BDDs. Experimental data are presented in Section 4. Concluding remarks are presented in Section 5.

2 Ternary Boolean Diagrams

Let $\mathcal{L}$ be a set of labels. Let $\mathcal{L}^- = \{-x \mid x \in \mathcal{L}\}$. A boolean diagram over $\mathcal{L}$ is a graph with a root node and each node is assigned a label of $\mathcal{L} \cup \mathcal{L}^-$. A ternary boolean diagram is such a graph where the out degree of a node is either 3 or 0, and the out edges of a node is ordered such that the left, the middle and the right out edges of a node can be identified. Ternary boolean diagrams are defined as follows.

Definition 1 (Ternary Boolean Diagram). *Let $\mathcal{L}$ be a set of labels. A ternary boolean diagram (TBD) over $\mathcal{L}$ is a quadruple*

$$(N, n_0, E, L)$$

where N is a set of nodes, $n_0 \in N$ is the root node, $E : N \to N^3$ is a partial function that defines the three out edges of a node, $L : N \to \mathcal{L} \cup \mathcal{L}^-$ is a labeling function which assigns each node a label of $\mathcal{L}$ or its negation.

The set of ternary boolean diagrams over $\mathcal{L}$ is denoted $\mathcal{D}(\mathcal{L})$. For simplicity, we fix an $\mathcal{L}$ and write $\mathcal{D}$ for $\mathcal{D}(\mathcal{L})$.

Notations When $E(n)$ is not defined for a node n, the value of $E(n)$ is denoted by a special triple $(\epsilon, \epsilon, \epsilon)$. A TBD t with n_0 as the root node may be represented by (x, a, b, c) when $L(n_0) = x, E(n_0) = (a, b, c)$. For the clearness of the presentation, additional notations to be used are as follows.

notation	meaning	condition
$x > 0$ $x < 0$	$x \in \mathcal{L}$ $x \in \mathcal{L}^-$	
$L(t)$	x	where $t = (x, a, b, c)$
$--x$	x	where $x > 0$
$\|x\|$	x	where $x > 0$
$\|x\|$	$-x$	where $x < 0$
$-(x, a, b, c)$	$(-x, a, b, c)$	where $x > 0$ or $x < 0$
$x \cdot t$	t	where $x > 0$
$x \cdot t$	$-t$	where $x < 0$
τ_x	$(x, \epsilon, \epsilon, \epsilon)$	where $x > 0$
$\tau_x^\sim$	$(x, \epsilon, \epsilon, \epsilon)$	where $x > 0$ or $x < 0$

2.1 Language, Model and Equivalence

Let $\Sigma = (\mathcal{L} \cup \mathcal{L}^-)^*$.

Definition 2. *Let $r \in \mathcal{D}$ and $\sigma = x_1 \cdots x_k \in \Sigma$. σ is accepted by r, denoted $\sigma \models r$, iff there is an $x \in \mathcal{L}$ such that $r\lceil_{x_1} \cdots \lceil_{x_k} = \tau_x$ in which $r\lceil_w$ is defined as follows:*

	$r\lceil_w$ *when* $w < 0$	$r\lceil_w$ *when* $w > 0$
$r = \tau_x^\sim$	r	r
$r = (v, \tau_x^\sim, \tau_y^\sim, \tau_z) \wedge \|v\| = \|w\|$	$v \cdot \tau_x^\sim$	$v \cdot \tau_y^\sim$
$r = (v, \tau_x^\sim, \tau_y^\sim, -\tau_z) \wedge \|v\| = \|w\|$	$v \cdot -\tau_z$	$v \cdot -\tau_z$
$r = (v, s, t, u)$	$(v, s\lceil_w, t\lceil_w, u\lceil_w)$	$(v, s\lceil_w, t\lceil_w, u\lceil_w)$

such that the last rule is applied only when the other rules are not applicable.

Definition 3. *Let $r \in \mathcal{D}$ and $m \subseteq \mathcal{L}$. m is a model of r, denoted $m \models r$, iff there is a sequence $\sigma \in \Sigma$ such that $x \in \sigma$ implies $x > 0 \wedge x \in m$ or $x < 0 \wedge -x \notin m$, and $\sigma \models r$.*

Definition 4. *Let $s, t \in \mathcal{D}$ be TBDs. s is equivalent to t, denoted $s \equiv t$, iff for all $m \subseteq \mathcal{L}$, $m \models s$ iff $m \models t$.*

2.2 Boolean Formulas as TBDs

Let $\Phi(S)$ denote the set of boolean formulas with variables in S. Let $\mathcal{L}_I \subseteq \mathcal{L}$ be a subset of $\mathcal{L}$.

Definition 5. *Let $\varphi \in \Phi(\mathcal{L}_I)$ and $m \subseteq \mathcal{L}$. $m \models \varphi$ iff φ evaluates to true under the assignment such that $x_i = 1$ iff $x_i \in m$.*

Definition 6. *Let $\varphi \in \Phi(\mathcal{L}_I)$ and $r \in \mathcal{D}$. φ is equivalent to r, denoted $\varphi \equiv r$, iff for all $m \subseteq \mathcal{L}$, $m \models \varphi$ iff $m \models r$.*

2.3 Ordered TBDs and Operations on Ordered TBDs

Let $\leq$ be a linear order on $\mathcal{L}$. $x < y$ iff $x \leq y$ and $x \neq y$. Let $\mathcal{L} = \{x_1, ..., x_n, x_{n+1}\}$ such that $x_i < x_{i+1}$ for $i = 1, ..., n$. Let $\tau = \tau_{x_{n+1}} = (x_{n+1}, \epsilon, \epsilon, \epsilon)$.

Definition 7. *The set $\mathcal{D}^{\mathcal{O}} \subseteq \mathcal{D}$ of ordered TBDs is defined as follows. $s \in \mathcal{D}^{\mathcal{O}}$ iff $s = \tau$, $s = -\tau$, or $s = (x, a_0, a_1, a_2) \in \mathcal{D}$ where a_0, a_1, a_2 are ordered and for all $i \in \{0, 1, 2\}$, $|x| < |L(a_i)|$.*

Negation: Let $s = (x, a, b, c) \in \mathcal{D}^{\mathcal{O}}$. The negation of s, denoted $\neg s$, is computed by $\neg s = (-x, a, b, c)$.

Conjunction: Let $s = (x, a, b, c)$, $t = (x', a', b', c')$ be ordered TBDs (in which a, b, c, a', b', c' may be ϵ). The conjunction of s and t, a commutative operator, denoted $s \wedge t$, is computed as follows, in which τ^c denotes ϵ when c is ϵ and denotes τ otherwise, both $\epsilon \wedge \epsilon$ and $\neg\neg\epsilon$ denote ϵ, both $\neg\epsilon \wedge \neg\epsilon$ and $\epsilon \wedge \neg\epsilon$ denote $\neg\epsilon$, and $(x, \neg\epsilon, \neg\epsilon, \epsilon)$ denotes $(-x, \epsilon, \epsilon, \epsilon)$.

	$s \wedge t$
$x = x' \wedge x > 0$	$(x, a \wedge a', b \wedge b', c \wedge c')$
$x = x' \wedge x < 0$	$(x, \neg(\neg(a \wedge c) \wedge \neg(a' \wedge c')), \neg(\neg(b \wedge c) \wedge \neg(b' \wedge c')), \tau^c)$
$\|x\| = \|x'\| \wedge x > x'$	$(x, a \wedge \neg(a' \wedge c'), b \wedge \neg(b' \wedge c'), c)$
$\|x\| < \|x'\| \wedge x > 0$	$(x, a, b, c \wedge t)$
$\|x\| < \|x'\| \wedge x < 0$	$(x, \neg(\neg a \wedge t), \neg(\neg b \wedge t), \neg(\neg c \wedge t))$

The cases where $|x| > |x'|$ or $|x| = |x'| \wedge x < x'$ are computed by $s \wedge t = t \wedge s$, since $\wedge$ is defined to be a commutative operator. Following from the definition, we have $s \wedge \tau \equiv s$ and $s \wedge -\tau \equiv -\tau$.

2.4 Equivalences and Rewrite Rules

Let $x \in \mathcal{L} \cup \mathcal{L}^-$ and $y \in \mathcal{L}$. The following equivalences hold for ordered TBDs.

$(x, -\tau, -\tau, c)$	$\equiv (x \cdot -\tau)$	$(x, a, b, -\tau)$	$\equiv (x \cdot -\tau)$
(x, τ, τ, c)	$\equiv (x \cdot c)$	(x, a, a, τ)	$\equiv (x \cdot a)$
$(x, a, -\tau, \tau)$	$\equiv (x, \tau, -\tau, a)$	$(x, -\tau, b, \tau)$	$\equiv (x, -\tau, \tau, b)$
(x, a, c, c)	$\equiv (x, a, \tau, c)$	(x, c, b, c)	$\equiv (x, \tau, b, c)$
$(-y, \tau, b, \tau)$	$\equiv (y, -\tau, -b, \tau)$	$(-y, a, \tau, \tau)$	$\equiv (y, -a, -\tau, \tau)$
$(-y, \tau, -\tau, c)$	$\equiv (y, -c, \tau, \tau)$	$(-y, -\tau, \tau, c)$	$\equiv (y, \tau, -c, \tau)$

These equivalences may be used as rewrite rules (replacing terms in the left hand sides with those in the right hand side of the equivalences) for the simplification of ordered TBDs.

2.5 Boolean Formulas as Ordered TBDs

Let $\mathcal{L}_I$ defined in the previous section be restricted to a subset of $\mathcal{L} \setminus \{x_{n+1}\}$. Let $x, \varphi_0, \varphi_1 \in \Phi(\mathcal{L}_I)$ and $s, t \in \mathcal{D}^{\mathcal{O}}$. The following hold.

- If $x \in \mathcal{L}_I$ is an atomic formulas, then $x \equiv (x, -\tau, \tau, \tau)$;
- if $\varphi_0 \equiv s$ and $\varphi_1 \equiv t$, then $\neg\varphi_0 \equiv \neg s$ and $\varphi_0 \wedge \varphi_1 \equiv s \wedge t$.

Then Boolean formulas can be transformed into TBDs according to these equivalences and the operations on TBDs.

Example Let φ be $\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q)$. Let $\mathcal{L} = \{p, q, z\}$ with $p < q < z$ such that $\mathcal{L}_I = \{p, q\}$. Following the definitions of negation and conjunction, and the rewrite rules, φ can be transformed to a TBD as follows.

p	$\equiv (p, -\tau, \tau, \tau)$
$\neg p$	$\equiv (p, \tau, -\tau, \tau)$
q	$\equiv (q, -\tau, \tau, \tau)$
$\neg q$	$\equiv (q, \tau, -\tau, \tau)$
$p \wedge \neg q$	$\equiv (p, -\tau, \tau, (q, \tau, -\tau, \tau))$
$\neg p \wedge q$	$\equiv (p, \tau, -\tau, (q, -\tau, \tau, \tau))$
$\neg(p \wedge \neg q)$	$\equiv (p, \tau, (q, -\tau, \tau, \tau), \tau)$
$\neg(\neg p \wedge q)$	$\equiv (p, (q, \tau, -\tau, \tau), \tau, \tau)$
φ	$\equiv (p, (q, \tau, -\tau, \tau), (q, -\tau, \tau, \tau), \tau)$

The TBD for φ has 10 nodes when it is considered as a tree, and it has 5 nodes when it is considered as a graph with shared nodes (in this case, only the 7 terminal nodes are merged into 2 nodes).

3 Complexity Issues

The representation succinctness is explained via a comparison of TBDs and BDDs. The node size of t, denoted $|t|$, is the number of different nodes of t that may use shared nodes. The tree size of t, denoted $||t||$, is the number of nodes of t where t is expanded as a tree. For TBDs, a TBD tree t has $(2||t|| + 1)/3$ terminal nodes, and therefore $|t| \leq (||t|| - 1)/3 + 2$.

Variable Order Let $v = v_1 \cdots v_m$ denote the partial order $(\{v_1, ..., v_m, z\}, \leq)$ with $v_i < v_j$ iff $i < j$ and $v_i < z$ for all i. An ordering v of variables of ψ is such an ordering where $\{v_1, ..., v_m\}$ is a permutation of variables appearing in ψ (the special variable z is needed for TBDs, it is not necessary for BDDs, however adding this special variable to an ordering of variables would not cause any trouble for BDDs). Let $v[x \to 1]$ denote the ordering v' where the first element of v' is x, and the rest of the elements of v' is the same as v with x removed from v.

3.1 Bit Comparators

TBDs may not be as sensitive to variable orderings as BDDs do.

Definition 8. *Let $p \leftrightarrow q$ denote $\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge q)$. Let a_i, b_i with $i \in \{1, ..., n\}$ be propositional variables.*

$$\boxed{\begin{array}{l} \varphi_i = a_i \leftrightarrow b_i \\ \varphi = \bigwedge_{i=1}^{n} \varphi_i \end{array}}$$

Proposition 1. *There is some variable ordering, such that the reduced ordered BDD representation of φ has node size $\geq 2^n$, while for all variable orderings, we can construct a TBD representation for φ with tree size and node size linear in n.*

Proof: It is known that representations of φ by reduced ordered BDDs varies from $3 \cdot n + 2$ to $3 \cdot 2^n - 1$ depending on the ordering of the variables [3]. The second part of this proposition follows from Lemma 3, which is to be established as follows.

Positive TBDs A positive TBD is an ordered TBD where all non-terminal nodes are marked by positive labels.

Lemma 1. *Let s, t be positive TBDs. Then we can construct a positive TBD for $s \wedge t$ such that $||s \wedge t|| \leq ||s|| + ||t|| - 1$.*

Proof: Since $\tau \wedge u = u \wedge \tau = u$ and $-\tau \wedge u = u \wedge -\tau = -\tau$ for any TBD u, this lemma holds when one of s and t is τ or $-\tau$. Let $s = (x, a, b, c)$ and $t = (x', a', b', c')$. We have three cases $x < x'$, $x = x'$ or $x > x'$.

- If $x < x'$, we have $s \wedge t = (x, a, b, c \wedge t)$.
  Then $||s \wedge t|| \leq ((||s|| - ||a|| - ||b|| - 1) + ||t|| - 1) + ||a|| + ||b|| + 1 = ||s|| + ||t|| - 1$.
- If $x = x'$, we have $s \wedge t = (x, a \wedge a', b \wedge b', c \wedge c')$.
  Then $||s \wedge t|| \leq (||a|| + ||a'|| - 1) + (||b|| + ||b'|| - 1) + (||c|| + ||c'|| - 1) + 1 < ||s|| + ||t|| - 1$.
- If $x > x'$, we have $s \wedge t = (x', a', b', c' \wedge s)$.
  Similar to the first case, $||s \wedge t|| \leq ||s|| + ||t|| - 1$.

Lemma 2. *For any ordering v of variables of φ, we can construct a positive TBD for φ_i with tree size* 10.

Proof: This follows from the construction demonstrated in the example shown above in Section 2.

Lemma 3. *For any ordering v of variables of φ, we can construct a positive TBD t for φ with $||t|| \leq 9n + 1$ and $|t| \leq 3n + 2$.*

Proof: According to Lemma 2, for each φ_i, we construct a positive TBD with tree size 10. Then according to Lemma 1, for φ, we construct a positive TBD t with $||t|| \leq n \cdot 10 - (n-1) = 9n + 1$. This implies $|t| \leq 3n + 2$.

3.2 Modified Bit Comparators

TBDs may be more succinct than BDDs for representing a formula regardless of the ordering of variables.

Definition 9. *Let $p \vee q$ denote $\neg(\neg p \wedge \neg q)$. Let $a_{i,j}$ with $i, j \in \{1, ..., n\}$ and b be propositional variables.*

φ_i	$= a_{i,1} \leftrightarrow \cdots \leftrightarrow a_{i,n}$
φ'_j	$= a_{1,j} \leftrightarrow \cdots \leftrightarrow a_{n,j}$
φ	$= \bigwedge_{i=1}^{n} \varphi_i$
φ'	$= \bigwedge_{j=1}^{n} \varphi'_j$
ψ	$= (b \wedge \varphi) \vee (\neg b \wedge \varphi')$

Proposition 2. *For all variable orderings, the reduced ordered BDD representation of ψ has node size $\geq 2^n$, while we can construct a TBD representation for ψ with node size polynomial in n.*

Proof: This proposition follows from Proposition 3 and Proposition 4, which are to be established as follows.

BDD size of ψ

Let $bdd_v(\varphi)$ denote the reduced ordered BDD of φ with the variable order specified in v.

Lemma 4. *Let ϕ be a formula.*
Then $|bdd_v(\phi)| \geq max(|bdd_v(\phi|_{x=0})|, |bdd_v(\phi|_{x=1})|)$ for any x and any ordering v of variables.

Proof: Starting with $bdd_v(\phi)$, in order to create $bdd_v(\phi|_{x=0})$, all pointers to the node labeled with x in $bdd_v(\phi)$ are redirected to the left branch of the node (with possibly further reductions). This modification will not increase the number of nodes.

Lemma 5. *Let ϕ be $(a_1 \leftrightarrow b_1) \wedge \cdots \wedge (a_n \leftrightarrow b_n)$.*
Let $(x_1, ..., x_n)$ and $(y_1, ..., y_n)$ be permutations of respectively $\{a_1, ..., a_n\}$ and $\{b_1, ..., b_n\}$. Let $v = x_1...x_n y_1...y_n$. Then $|bdd_v(\phi)| = 3 \cdot 2^n - 1$.

Proof: A similar result is known when $v = a_1...a_n b_1...b_n$ [3]. For $v = x_1...x_n y_1...y_n$, the reasoning is as follows.

- Expanding $x_1, ..., x_n$ we have $2^n - 1$ nodes and 2^n different cases (of type 0). 2^{n-1} of the cases (where the variable corresponding with y_1 is assigned false) of type 0 point to false when y_1 assigns false, creating 2^{n-1} new y_1 nodes. 2^{n-1} of the cases (where the variable corresponding with y_1 is assigned true) of type 0 point to false when y_1 assigns true, creating another 2^{n-1} new y_1 nodes. There remains 2^n undecided cases. Half of the cases are the same as (or symmetric with respect to the truth assignments of y_1 and the corresponding x-variable) to the other half of the cases. Therefore there remain 2^{n-1} different cases (of type 1).
- 2^{n-2} of the cases (where the variable corresponding with y_2 is assigned false) of type 1 point to false when y_2 assigns false, creating 2^{n-2} new y_2 nodes. 2^{n-2} of the cases (where the variable corresponding with y_2 is assigned true) of type 1 point to false when y_2 assigns true, creating another 2^{n-2} new y_2 nodes. Similarly, there remain 2^{n-2} different cases (of type 2).
- Generally, at the i-th round, there remains 2^{n-i} different cases of types i. Half of the cases point to false when y_{i+1} assigns false, creating 2^{n-i} new y_{i+1} nodes, and half of the cases point to false when y_{i+1} assigns true, also creating 2^{n-i} new y_{i+1} nodes. This process continues until there remains 2 different cases, requiring two new y_n nodes.
- Therefore there are 2^{n-i+1} y_i nodes, by looking backward, we know that they must be different, and in total, there are $2^{n+1} - 2$ different nodes for $y_1, ..., y_n$. Summing up the number of nodes for $x_1, ..., x_n, y_1, ..., y_n$ and the two terminal nodes, we have $|bdd_v(\phi)| = 3 \cdot 2^n - 1$.

Proposition 3. *Let o be an ordering o of variables of ψ. Then $|bdd_o(\psi)| \geq 3 \cdot 2^{n-1} - 1$.*

Proof: Let $o = c_1 \cdots c_m$ such that $(c_1, ..., c_m)$ is a permutation of $\{a_{i,j} | 1 \leq i, j \leq n\} \cup \{b\}$. Let $v(\varphi)$ denote the set of variables appearing in φ. Let $o_i = \{c_1, ..., c_i\}$. Let k be the least number such that $|v(\varphi_i) \cap o_m| = n - 1$ or $|v(\varphi'_i) \cap o_m| = n - 1$ for some $0 \leq 1 \leq n$. The proof is as follows.

- Either $|v(\varphi_l) \cap o_k| = n - 1$ or $|v(\varphi'_l) \cap o_k| = n - 1$ for some l. Suppose that the former is the case (the latter is symmetric).
- Let $S = v(\varphi_l) \cap o_k = \{a_{1,l}, ..., a_{x-1,l}, a_{x+1,l}, ..., a_{i,l}\}$.
- For each $a_{i,j} \in S$, we select an $a_{i,y_i} \notin o_k$. Let S' denote the set of these variables, and φ'' denote φ' with every variable not in $S \cup S'$ replaced by 1.
- Then φ'' is a formula corresponding to the one in Lemma 5 with $2(n-1)$ variables.
- According to Lemma 4 and Lemma 5,
  $|bdd_o(\psi)| \geq |bdd_o(\psi|_{b=0})| = |bdd_o(\varphi')| \geq |bdd_o(\varphi'')| = 3 \cdot 2^{n-1} - 1$.

TBD size of ψ

Let $[t]$ denote the height of the TBD t with $[\tau] = [-\tau] = 0$.

Lemma 6. *Let s, t be positive TBDs. If s and t have no variables in common, then we can construct a positive TBD for $s \wedge t$ with at most $[s] + [t]$ new nodes.*

Proof: By induction. This lemma holds when one of s and t is τ or $-\tau$. Let $s = (x, a, b, c)$ and $t = (x', a', b', c')$. We have two cases $x < x'$ or $x > x'$.

- If $x < x'$, we have $s \wedge t = (x, a, b, c \wedge t)$.
  According to the induction hypothesis, $c \wedge t$ requires at most $[c] + [t]$ new nodes, therefore $s \wedge t$ requires at most $[c] + [t] + 1 \leq [s] + [t]$ new nodes.
- If $x > x'$, we have $s \wedge t = (x', a', b', c' \wedge s)$.
  Similarly, $s \wedge t$ requires at most $[c'] + [s] + 1 \leq [s] + [t]$ new nodes.

Lemma 7. *Let ϕ_i be $(a_1 \leftrightarrow \cdots \leftrightarrow a_i)$. We can construct a positive TBD t for ϕ_n with $|t| \leq 2(n+1)$ for any ordering of variables.*

Proof: In fact, we can create a positive TBD for ϕ_n and a positive TBD for $\neg\phi_n$ such that the total number of different nodes in the two TBDs is $2(n+1)$. The reasoning by induction is as follows. The statement is true when $n = 1$. Then we can use the TBD t_0 for ϕ_{n-1} and the TBD t_1 for $\neg\phi_{n-1}$ to build a positive TBD $t'_0 = (a_n, t_1, t_0, \tau)$ for ϕ_n and a positive TBD $t'_1 = (a_n, t_0, t_1, \tau)$ for $\neg\phi_n$ with only two additional new nodes. The construction assumes variable order $v = a_n \cdots a_1$. Since the variables in ϕ_i are all symmetric, this construction can be done for any variable order by rearranging the position of the variables in the formula.

Lemma 8. *For any ordering v of variables of ψ, we can construct a positive TBD t for ψ with $|t| \leq 4n^2(n+1)+1$ for the modified variable ordering $v[b \rightarrow 1]$.*

Proof: The proof is as follows.

- According to Lemma 7, for each φ_i, we can construct a corresponding TBD with $2(n+1)$ nodes. According to Lemma 6, for φ, we can construct a corresponding TBD t with $|t| \leq 2(n+1) \cdot n + (n-1) \cdot 2 \cdot n^2 = 2n^2(n+1)$.
- Similarly, for φ', we can construct a corresponding TBD t' with $|t'| \leq 2n^2(n+1)$.
- Then $t'' = (b, t, t', \tau)$ is a TBD for ψ with $|t''| \leq 4n^2(n+1)+1$ different nodes (where τ is not counted, since it must have appeared in t or t') with b as the label of the top node of the TBD compatible with the order $v[b \rightarrow 1]$.

Lemma 9. *Let $p > 0$ and $q > 0$. Each of the following pairs represent the same formula with different variable orderings on p and q.*

$(p, (q, a, b, c), c', c'')$ $(q, (p, a, \tau, \tau), (p, b, \tau, \tau), (p, c, c', c''))$
$(p, c, (q, a', b', c'), c'')$ $(q, (p, \tau, a', \tau), (p, \tau, b', \tau), (p, c, c', c''))$
$(p, c, c', (q, a'', b'', c''))$ $(q, (p, \tau, \tau, a''), (p, \tau, \tau, b''), (p, c, c', c''))$
$(p, (q, a, b, c), (q, a', b', c'), c'')$ $(q, (p, a, a', \tau), (p, b, b', \tau), (p, c, c', c''))$
$(p, (q, a, b, c), c', (q, a'', b'', c''))$ $(q, (p, a, \tau, a''), (p, b, \tau, b''), (p, c, c', c''))$
$(p, c, (q, a', b', c'), (q, a'', b'', c''))$ $(q, (p, \tau, a', a''), (p, \tau, b', b''), (p, c, c', c''))$
$(p, (q, a, b, c), (q, a', b', c'), (q, a'', b'', c''))$ $(q, (p, a, a', a''), (p, b, b', b''), (p, c, c', c''))$

Proof: These equivalences follow from the semantics of ordered TBDs.

Lemma 10. *Let v be a given ordering of variables, and t be a positive TBD with variable ordering $v[b \to 1]$. Then we can construct a TBD t' with variable ordering v such that t and t' represent the same formula and $|t'| \leq 4 \cdot |t|$.*

Proof: By moving b down 1 level in the reordering of the variables (applying the equivalences in Lemma 9 for reordering), the nodes at the current level is replaced, and for each such node, it may create 3 new nodes. In total, it may create at most $3 \cdot |t|$ new nodes for all of the levels of t.

Proposition 4. *For any ordering v of variables of ψ, we can construct a positive TBD t for ψ with $|t| \leq 16n^2(n+1) + 4$.*

Proof: This follows from Lemma 8 and Lemma 10.

4 Experimental Comparison of TBDs and BDDs

It is already shown that TBDs can be more succinct than BDDs for representation of some Boolean formulas. For practical application of TBDs, the average complexity is in many cases more important than best case complexity. The contents of this section are experimental data on the average complexity of TBD representation of Boolean formulas for selected types of random Boolean formulas.

4.1 Types of Boolean Formulas

Consider Boolean formulas in CNF (conjunctive normal form) and DNF (disjunctive normal form). The number of variables is set to 30. The length of clauses in $\{10, 20, 30\}$ and the number of clauses in $\{100, 200, 300\}$. This makes 9 types of CNF formulas and 9 types of DNF formulas. For each type, we randomly choose 20 formulas, and for each formula, we randomly choose 20 different variable orders.

4.2 Experimental Data

Experimental Data for CNF Formulas The experimental data for the size of TBDs are presented in Table 1, where *cll* is the clause length, *cln* is the number of clauses in a formula, *min* is the minimum size of the TBD obtained among the 400 TBDs (20 instances with 20 different variable orders), *max* is the maximum size of the TBDs, and *average* is the average size of the TBDs. The experimental data for the size of BDDs are presented in Table 2. In addition, the ratio between the average size of BDDs (presented in this table) and that of TBDs (presented in the previous table) is calculated for each case. These ratios show that TBDs has significant advantage over BDDs for representing some of the formulas.

cll	cln	min	max	average
10	100	709	756	732.59
	200	1321	1389	1354.85
	300	1893	1977	1934.85
20	100	1472	1560	1516.81
	200	2808	2921	2865.39
	300	4088	4223	4149.23
30	100	1743	1847	1794.53
	200	3104	3252	3187.11
	300	4333	4514	4430.60

Table 1. Size of TBDs for CNF formulas

cll	cln	min	max	average	ratio
10	100	82729	215897	136146.64	186.35
	200	577533	1035169	768347.00	567.95
	300	1516800	2512466	1973997.62	1021.29
20	100	2606	3848	3173.77	2.09
	200	6715	9036	7634.59	2.67
	300	11380	15282	12916.41	3.11
30	100	1743	1847	1794.53	1.00
	200	3104	3252	3187.11	1.00
	300	4333	4514	4430.60	1.00

Table 2. Size of BDDs for CNF formulas

Experimental Data for DNF Formulas The experimental data for the size of TBDs and BDDs for the types of DNF formulas are shown in Table 3 and Table 4. These data are similar to those presented in Table 1 and Table 2, and

the ratios between size of TBDs and size of BDDs also show that TBDs has significant advantage over BDDs for representing some of the formulas.

cll	cln	min	max	average
10	100	709	758	731.39
	200	1325	1391	1353.56
	300	1893	1964	1933.90
20	100	1478	1551	1518.05
	200	2812	2918	2864.66
	300	4061	4212	4148.40
30	100	1738	1842	1793.01
	200	3107	3267	3186.71
	300	4299	4543	4427.99

Table 3. Size of TBDs for DNF formulas

cll	cln	min	max	average	ratio
10	100	80947	223120	135831.05	186.22
	200	576519	1050933	765908.19	566.68
	300	1561373	2481051	1966855.62	1018.09
20	100	2743	3728	3171.91	2.09
	200	6386	9028	7596.83	2.65
	300	11067	14691	12866.37	3.10
30	100	1738	1842	1793.01	1.00
	200	3107	3267	3186.71	1.00
	300	4299	4543	4427.99	1.00

Table 4. Size of BDDs for DNF formulas

5 Concluding Remarks

Complexity analysis has shown that there are cases TBDs can be more compact than BDDs for representation of Boolean functions. In particular, the examples have demonstrated that TBDs may not be as sensitive to variable orderings as BDDs, and moreover TBDs may be more succinct than BDDs for representing a formula regardless of the ordering of variables. Experimental data also support this fact. Therefore TBDs may be considered as a data structure complementary to BDDs, and may be used in certain cases where BDDs are not sufficiently efficient.

Acknowledgement This work is supported by the National Natural Science Foundation of China under Grant No. 60573012, 60421001 and the CAS innovation program.

References

1. R. E. Bryant. Graph based algorithms for boolean function manipulation. IEEE Transaction on Computers 35(8):677-691. 1986.
2. R. E. Bryant: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. ACM Comput. Surv. 24(3): 293-318. 1992.
3. E. M. Clarke, O. Grumberg and D. Peled. Model Checking. The MIT Press. 1999.
4. C. Y. Lee. Representation of Switching Circuits by Binary-Decision Programs. Bell Systems Technical Journal 38: 985-999. 1959.
5. W. Zhang. Ternary Boolean Diagrams. Technical Report ISCAS-LCS-10-24, Institute of Software, Chinese Academy of Sciences. 2010.