Technical Report #99009

Ordered Binary Decision Diagrams as Knowledge-Bases

Takashi HORIYAMA[1] and Toshihide IBARAKI[2]

April 1999
Department of Applied Mathematics and Physics,
Kyoto University, Kyoto 606-8501, Japan

Abstract. We propose to make use of ordered binary decision diagrams (OBDDs) as a means of realizing knowledge-bases. We show that the OBDD-based representation is more efficient and suitable in some cases, compared with the traditional CNF-based and/or model-based representations from the view point of space requirement. We then consider two recognition problems of OBDDs, and present polynomial time algorithms for testing whether a given OBDD represents a unate Boolean function, and whether it represents a Horn function.

[1]Graduate School of Information Science, Nara Institute of Science and Technology, Nara 630-0101, Japan. e-mail: horiyama@is.aist-nara.ac.jp

[2]Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan. e-mail: ibaraki@kuamp.kyoto-u.ac.jp

1 Introduction

Logical formulae are one of the traditional means of representing knowledge in AI [11]. However, it is known that deduction from a set of propositional clauses is co-NP-complete and abduction is NP-complete [13]. Recently, an alternative way of representing knowledge, i.e., by a subset of its models, which are called characteristic models, has been proposed (see e.g., [6, 7, 8, 9]). Deduction from a knowledge-base in this model-based approach can be performed in linear time, and abduction is also performed in polynomial time [6].

In this paper, we propose yet another method of knowledge representation, i.e., the use of ordered binary decision diagrams (OBDDs) [1, 2, 12]. An OBDD is a directed acyclic graph representing a Boolean function, and can be considered as a variant of decision trees. By restricting the order of variable appearances and by sharing isomorphic subgraphs, OBDDs have the following useful properties: 1) When a variable ordering is given, an OBDD has a reduced canonical form for each Boolean function. 2) Many Boolean functions appearing in practice can be compactly represented. 3) There are efficient algorithms for many Boolean operations on OBDDs. As a result of these properties, OBDDs are widely used for various applications, especially in computer-aided design and verification of digital systems (see e.g., [4, 14]). The manipulation of knowledge-bases by OBDDs (e.g. deduction and abduction) was first discussed by Madre and Coudert [10].

We first compare the above three representations, i.e., formula-based, model-based, and OBDD-based, on the basis of their sizes. In particular, we show that, in some cases, OBDDs require exponentially smaller space than the other two representations, while there are also cases in which each of the other two requires exponentially smaller space. In other words, these three representations are mutually incomparable with respect to space requirement.

It is known that OBDDs are efficient for knowledge-base operations such as deduction and abduction [10]. We investigate two fundamental recognition problems of OBDDs, that is testing whether a given OBDD represents a unate Boolean function, and testing whether it represents a Horn function. We often encounter these recognition problems, since a knowledge-base representing some real phenomenon is sometimes required to be unate or Horn, from the hypothesis posed on the phenomenon and/or from the investigation of the mechanism causing the phenomenon. For example, if the knowledge-base represents the data set of test results on various physical measurements (e.g., body temperature, blood pressure, number of pulses and so on), it is often the case that the diagnosis of a certain disease is monotonically depending on each test result (we allow changing the polarities of variables if necessary). Also in artificial intelligence, it is common to consider Horn knowledge-bases as they can be processed efficiently in many respects (for example, deduction from a set of Horn clauses can be done in linear time [5]). We show that these recognition problems for OBDDs can be solved in polynomial time for both the unate and Horn cases.

The rest of this paper is organized as follows. The next section gives fundamental definitions and concepts. We compare the three representations in Section 3, and consider the problems of recognizing unate and Horn OBDDs in Sections 4 and 5, respectively.

2 Preliminaries

2.1 Notations and Fundamental Concepts

We consider a Boolean function $f : \{0,1\}^n \to \{0,1\}$. An *assignment* is a vector $a \in \{0,1\}^n$, whose i-th coordinate is denoted by a_i. A *model of* f is a satisfying assignment a of f, i.e. $f(a) = 1$, and the *theory* $\Sigma(f)$ *representing* f is the set of all models of f. Given $a, b \in \{0,1\}^n$, we denote by $a \leq b$ the usual bitwise (i.e., componentwise) ordering of assignments; $a_i \leq b_i$ for all $i = 1, 2, \ldots, n$, where $0 < 1$. Given a subset $E \subseteq \{1, 2, \ldots, n\}$, χ^E denotes the characteristic vector of E; the i-th coordinate χ^E_i equals 1 if $i \in E$ and 0 if $i \notin E$.

Let $x_1, x_2, \ldots, x_n$ be the n variables of f, where each x_i corresponds to the i-th coordinate of assignments and evaluates to either 0 or 1. Negation of a variable x_i is denoted by $\overline{x}_i$. Variables and their negations are called literals. A clause is a disjunction of some literals, and a conjunction of clauses is called a *conjunctive normal form* (CNF). We say that f *is represented by a CNF* φ, if $f(a) = \varphi(a)$ holds for all $a \in \{0,1\}^n$. Any Boolean function can be represented by some CNF, which may not be unique.

We sometimes do not make a distinction among a function f, its theory $\Sigma(f)$, and a CNF φ that represents f, unless confusion arises. We define a *restriction* of f by replacing a variable x_i by a constant $a_i \in \{0,1\}$, and denote it by $f|_{x_i=a_i}$. Namely, $f|_{x_i=a_i}(x_1, \ldots, x_n)$ $= f(x_1, \ldots, x_{i-1}, a_i, x_{i+1}, \ldots, x_n)$ holds. Restriction may be applied to many variables. We also define $f \leq g$ (resp., $f < g$) by $\Sigma(f) \subseteq \Sigma(g)$ (resp., $\Sigma(f) \subset \Sigma(g)$).

Lemma 2.1 *The relation* $\leq$ *has the following properties:*

(1) $f \leq g$ *holds if and only if* $f|_{x_i=a_i} \leq g|_{x_i=a_i}$ *holds for both* $a_i = 0$ *and* 1.

(2) $f \vee g \leq h$ *holds if and only if* $f \leq h$ *and* $g \leq h$ *hold.*

For an assignment $p \in \{0,1\}^n$, we define $a \leq_p b$ if $(a \oplus_{bit} p) \leq (b \oplus_{bit} p)$ holds, where $\oplus_{bit}$ denotes the bitwise (i.e., componentwise) exclusive-or operation. A Boolean function f is *unate* with polarity p if $f(a) \leq f(b)$ holds for all assignments a and b such that $a \leq_p b$. A theory Σ is *unate* if Σ represents a unate function. A clause is *unate* with polarity p if $p_i = 0$ for all positive literals x_i and $p_i = 1$ for all negative literals $\overline{x}_i$ in the clause. A CNF is *unate* with polarity p if it contains only unate clauses with polarity p. It is known that a theory Σ is unate if and only if Σ can be represented by some unate CNF. A unate function is *positive* (resp., *negative*) if its polarity is $(00 \cdots 0)$ (resp., $(11 \cdots 1)$).

A theory Σ is *Horn* if Σ is closed under operation $\wedge_{bit}$, where $a \wedge_{bit} b$ is bitwise AND of models a and b. For example, if $a = (0011)$ and $b = (0101)$, then $a \wedge_{bit} b = (0001)$. The *closure* of a theory Σ with respect to $\wedge_{bit}$, denoted by $Cl_{\wedge_{bit}}(\Sigma)$, is defined as the smallest set that contains Σ and is closed under $\wedge_{bit}$. We also use the operation $\wedge_{bit}$ as a set operation; $\Sigma(f) \wedge_{bit} \Sigma(g)$ $= \{a \mid a = b \wedge_{bit} c$ holds for some $b \in \Sigma(f)$ and $c \in \Sigma(g)\}$. We often denotes $\Sigma(f) \wedge_{bit} \Sigma(g)$ by $f \wedge_{bit} g$, for convenience. Note that the two functions $f \wedge g$ and $f \wedge_{bit} g$ are different.

A Boolean function f is *Horn* if $\Sigma(f)$ is Horn; equivalently if $f \wedge_{bit} f = f$ holds (as sets of models). A clause is *Horn* if the number of positive literals in it is at most one, and a CNF is

Horn if it contains only Horn clauses. It is known that a theory Σ is Horn if and only if Σ can be represented by some Horn CNF. By definition, a negative function is Horn, but not conversely.

For any Horn theory Σ, a model $a \in \Sigma$ is called *characteristic* if it cannot be produced by bitwise AND of other models in Σ; $a \notin Cl_{\wedge_{bit}}(\Sigma - \{a\})$. The set of all characteristic models of a Horn theory Σ, which we call the *characteristic set of* Σ, is denoted by $Char(\Sigma)$. Note that every Horn theory Σ has a unique characteristic set $Char(\Sigma)$, which satisfies $Cl_{\wedge_{bit}}(Char(\Sigma)) = \Sigma$. The set of *minimal models* of f with respect to $p \in \{0,1\}^n$ is defined as

$$min_p(f) = \{a \in \Sigma(f) \mid \text{ there exists no } b \in \Sigma(f) \text{ satisfying } b <_p a\},$$

where $b <_p a$ denotes that $b \leq_p a$ and $b \neq a$ hold. The following lemma gives an upper bound on the size (i.e., cardinality) of the characteristic set.

Lemma 2.2 [9] *Let f be a Horn function on n variables. Then, the characteristic set of f has size at most $\sum_{p \in B_n} |min_p(f)|$, where $B_n = \{\chi^{E_{n,i}} \mid i = 0, 1, \ldots, n\}$ and $\chi^{E_{n,i}}$ is the characteristic vector of the set $E_{n,i} \subseteq \{0, 1, \ldots, n\}$ given by*

$$\begin{cases} E_{n,0} = \{1, 2, \ldots, n\} & \text{for} \quad i = 0 \qquad (i.e., \chi^{E_{n,0}} = (11 \cdots 1)), \\ E_{n,i} = E_{n,0} - \{i\} & \text{for} \quad i = 1, 2, \ldots, n. \end{cases}$$

2.2 Ordered Binary Decision Diagrams

An *ordered binary decision diagram* (OBDD) is a directed acyclic graph that represents a Boolean function. It has two sink nodes 0 and 1, called the *0-node* and the *1-node*, respectively (which are together called the *constant nodes*). Other nodes are called *variable nodes*, and each variable node v is labeled by one of the variables $x_1, x_2, \ldots, x_n$. Let $var(v)$ denote the label of node v. Each variable node has exactly two outgoing edges, called a *0-edge* and a *1-edge*, respectively. One of the variable nodes becomes the unique source node, which is called the *root node*. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote the set of n variables. A *variable ordering* is a total ordering $(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$, associated with each OBDD, where π is a permutation $\{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$. The *level*[3] of a node v, denoted by $level(v)$, is defined by its label; if node v has label $x_{\pi(i)}$, $level(v)$ is defined to be $n - i + 1$. That is, the root node is in level n and has label $x_{\pi(1)}$, the nodes in level $n-1$ have label $x_{\pi(2)}$ and so on. The level of the constant nodes is defined to be 0. On every path from the root node to a constant node in an OBDD, each variable appears at most once in the decreasing order of their levels.

Every node v of an OBDD also represents a Boolean function f_v, defined by the subgraph consisting of those edges and nodes reachable from v. If node v is a constant node, f_v equals to its label. If node v is a variable node, f_v is defined as $\overline{var(v)}\, f_{0\text{-}succ(v)} \vee var(v)\, f_{1\text{-}succ(v)}$ by Shannon's expansion, where $0\text{-}succ(v)$ and $1\text{-}succ(v)$, respectively, denote the nodes pointed by the 0-edge and the 1-edge of node v. The function f represented by an OBDD is the one represented by the root node. Figure 1 illustrates three OBDDs representing $x_3 x_2 \vee x_1$ with a

[3]This definition of level may be different from its common use.

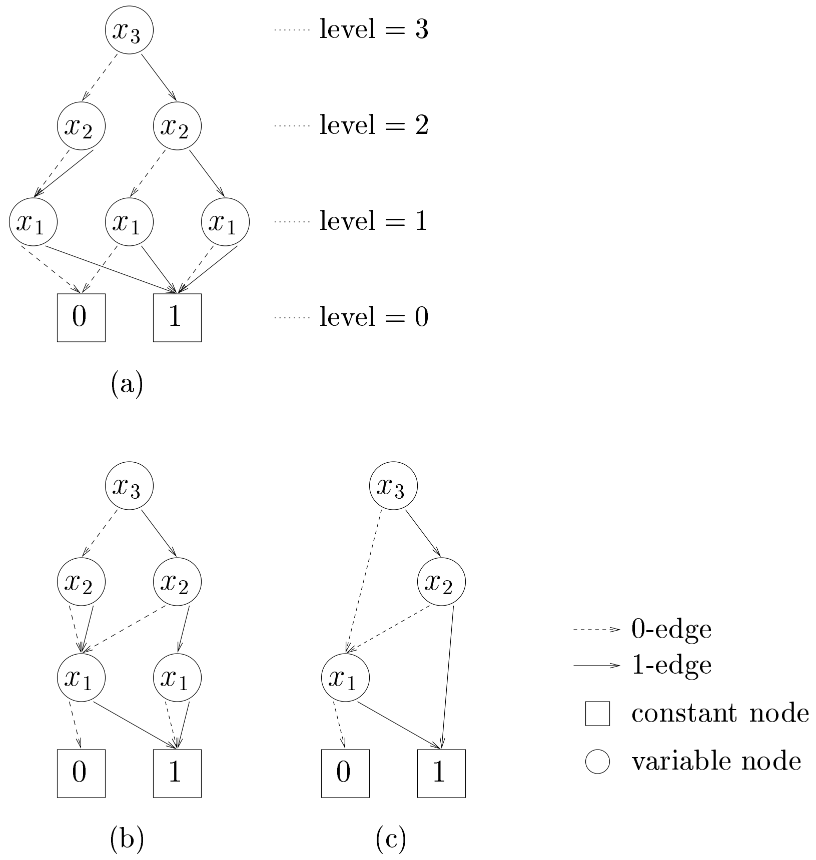

Figure 1: OBDDs representing $x_3x_2 \vee x_1$.

variable ordering (x_3, x_2, x_1). Given an assignment a, the value of $f(a)$ is determined by following the corresponding path from the root node to a constant node in the following manner: at a variable node v, one of the outgoing edges is selected according to the assignment $a_{var(v)}$ to the variable $var(v)$. The value of the function is the label of the final constant node.

When two nodes u and v in an OBDD represent the same function, and their levels are the same, they are called *equivalent*. A node whose 0-edge and 1-edge both point to the same node is called *redundant*. An OBDD is called *dense* if every variable node v satisfy $level(0\text{-}succ(v)) = level(1\text{-}succ(v)) = level(v) - 1$ (i.e., all paths from the root node to constant nodes visit $n + 1$ nodes). A dense OBDD which has no equivalent nodes is *quasi-reduced*. A quasi-reduced OBDD which has no redundant nodes is *reduced*. The OBDDs (a), (b) and (c) in Fig. 1 are dense, quasi-reduced and reduced, respectively. In the following, we assume that all OBDDs are reduced, unless otherwise stated. The *size* of an OBDD is the number of nodes in the OBDD. Given a function f and a variable ordering, its reduced OBDD is unique and has the minimum size among all OBDDs with the same variable ordering. The minimum sizes of OBDDs representing a given Boolean function depend on the variable orderings [2].

Given an OBDD that represents f, the OBDDs of $f|_{x_i=0}$ and $f|_{x_i=1}$ can be obtained in $O(|f| \log |f|)$ time, where $|f|$ denotes the size of the OBDD of f. The size does not increase by a restriction. Given two OBDDs representing f and g, applying fundamental logic operators, e.g., $f \wedge g$, $f \vee g$, $f \oplus g$ and $f \rightarrow g$, can be performed in $O(|f| \cdot |g|)$ time, and property $f \leq g$ can be also checked in $O(|f| \cdot |g|)$ time [2].

A *partition* for f is a pair of sets (L, R) satisfying $L, R \subseteq X = \{x_1, x_2, \ldots, x_n\}$, $L \cup R = X$ and $L \cap R = \emptyset$. L is called a *left partition* and R is called a *right partition*. Let l denote an assignment to the variables in L, and r denote an assignment to the variables in R. Then, $l \cdot r$ denotes the complete assignment obtained by combining l and r. Let X' be a subset of X, and ω be a positive number satisfying $0 < \omega < 1$. Then, a partition (L, R) is called *ω-balanced for X'*, if it satisfies $\lfloor \omega |X'| \rfloor \leq |X' \cap L| \leq \lceil \omega |X'| \rceil$. Given a partition (L, R), a set A of assignments l^i for L and r^i for R, $i = 1, 2, \ldots, h$, is called a *fooling set* if it satisfies

(1) $f(l^i \cdot r^i) = a$ for all i,

(2) $f(l^i \cdot r^j) \neq a$ or $f(l^j \cdot r^i) \neq a$ for all $i \neq j$,

for some $a \in \{0, 1\}$. The next lemma tells that the size h of a fooling set gives a lower bound on the size of an OBDD that represents f.

Lemma 2.3 [3] *Let f be a Boolean function on n variables, X' be a subset of the variables and ω be a positive number satisfying $0 < \omega < 1$. If f has a fooling set of size at least h for every ω-balanced partition (L, R) for X', then the size of OBDD representing f is at least h for any variable ordering.*

3 Three Approaches for Knowledge-Base Representation

In this section, we compare three knowledge-base representations: CNF-based, model-based, and OBDD-based. It is known that CNF-based and model-based representations play orthogonal roles with respect to space requirement. Namely, each of them sometimes allows exponentially smaller sizes than the other, depending on the functions. We show that OBDD-based representation is incomparable to the other two in the same sense.

We start with relations between OBDD and CNF representations.

Lemma 3.1 *There exists a negative theory on n variables, for which OBDD and CNF both require size $O(n)$, while its characteristic set requires size $\Omega(2^{n/2})$.*

Proof: Consider a function

$$f_A = \bigwedge_{i=1}^{m} (\overline{x}_{2i-1} \vee \overline{x}_{2i}),$$

where $n = 2m$. The size of this CNF is obviously $O(n)$. The characteristic set is given by $\{a \in \{0,1\}^{2m} \mid \text{exactly one of } a_{2i-1} \text{ or } a_{2i} \text{ is } 0 \text{ for all } i = 1, 2, \ldots, m\}$, whose size is $\Omega(2^{n/2})$ [6]. The OBDD representing f_A is illustrated in Fig. 2, with a variable ordering $(x_n, x_{n-1}, \ldots, x_1)$. The size of this OBDD is $O(n)$. □

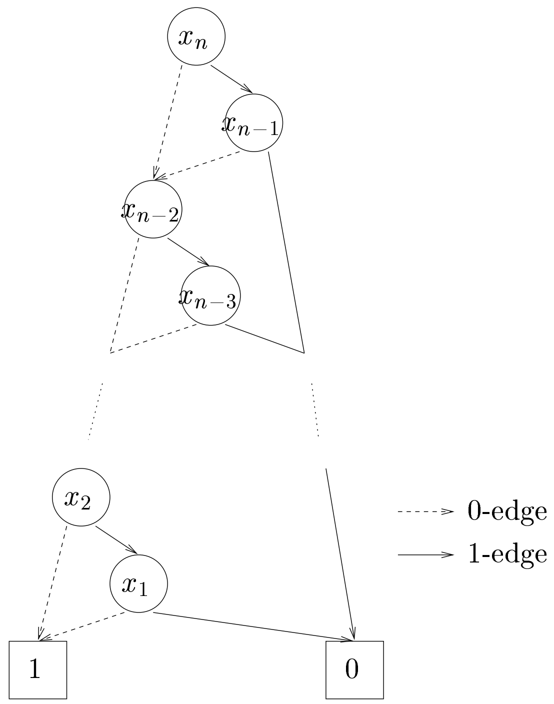

Figure 2: OBDD representing $f_A = \bigwedge_{i=1}^{m}(\overline{x}_{2i-1} \vee \overline{x}_{2i})$.

Lemma 3.2 *There exists a negative theory on n variables, for which OBDD requires size $O(n)$ and the characteristic set requires size $O(n^2)$, while CNF requires size $\Omega(2^{n/2})$.*

Proof: Consider a function

$$f_B = \bigvee_{i=1}^{m} (\overline{x}_{2i-1} \wedge \overline{x}_{2i}) = \bigwedge_{(r_1, r_2, \ldots, r_m) \in S_B} (\overline{r}_1 \vee \overline{r}_2 \vee \cdots \vee \overline{r}_m),$$

where $n = 2m$ and $S_B = \{(r_1, r_2, \ldots, r_m) \mid r_i \in \{x_{2i-1}, x_{2i}\}$ for all $i = 1, 2, \ldots, m\}$. f_B is dual to f_A. The smallest CNF representation of f_B, which is given above, has $\Omega(2^{n/2})$ clauses. The characteristic set is $\{\chi^{\{1,2,\ldots,2m\}-S} \in \{0,1\}^{2m} \mid S = \{2i-1, 2i\}$ or $S = \{2i-1, 2i, j\}$ for $i \in \{1, 2, \ldots, m\}$ and j $(\neq 2i-1, 2i) \in \{1, 2, \ldots, 2m\}\}$, whose size is $O(n^2)$ [6]. The OBDD representing f_B is illustrated in Fig. 3, with a variable ordering $(x_n, x_{n-1}, \ldots, x_1)$. Note that, as f_B is dual to f_A, this OBDD is obtained by negating input variables (i.e., exchanging the roles of 0-edges and 1-edges) and negating output (i.e., exchanging the roles of the 0-node and the 1-node) of the OBDD in Fig. 2. The size of this OBDD is $O(n)$. $\square$

By combining Lemmas 3.1 and 3.2, we show that, for some theory, OBDD can be exponentially smaller than its characteristic set and CNF representations.

Theorem 3.1 *There exists a negative theory on n variables, for which OBDD requires size $O(n)$, while both of the characteristic set and CNF require sizes $\Omega(2^{n/4})$.*

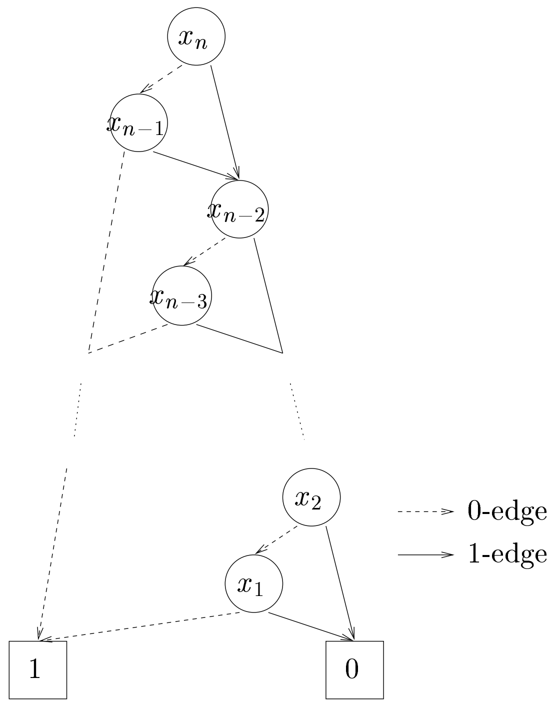

Figure 3: OBDD representing $f_B = \bigvee_{i=1}^{m}(\overline{x}_{2i-1} \wedge \overline{x}_{2i})$.

Proof: Consider a function

$$f_C = \left(\bigwedge_{i=1}^{m}(\overline{x}_{2i-1} \vee \overline{x}_{2i})\right) \wedge \left(\bigvee_{i=m+1}^{2m}(\overline{x}_{2i-1} \wedge \overline{x}_{2i})\right),$$

where $n = 4m$. As shown in Lemma 3.1, the characteristic set requires size $\Omega(2^{n/4})$ to represent the first half. Also by Lemma 3.2, CNF representation always requires size $\Omega(2^{n/4})$ to represent the second half. Note the first and second halves are independent since the variables in the first half do not appear in the second half and vice versa. Therefore, the above lower bounds of the characteristic set and CNF are valid also for f_C. An OBDD that represents f_C is illustrated in Fig. 4, with a variable ordering $(x_n, x_{n-1}, \ldots, x_1)$. The size of this OBDD is $O(n)$. $\square$

We now turn to the opposite direction, i.e., CNF and the characteristic set can be exponentially smaller than the size of OBDD.

Lemma 3.3 *The size of the characteristic set is $O(n)$ for the following Horn function on n variables $x_{i,j}$, $1 \leq i,j \leq m+1$, where $n = (m+1)^2$:*

$$f_D = \left(\bigwedge_{i=1}^{m+1}\left(x_{i,m+1} \vee \bigvee_{j=1}^{m}\overline{x}_{i,j}\right)\right) \wedge \left(\bigwedge_{j=1}^{m+1}\left(x_{m+1,j} \vee \bigvee_{i=1}^{m}\overline{x}_{i,j}\right)\right).$$

Proof: Consider $\{\chi^{E_{n,0}}\} \cup \{\chi^{E_{n,i,j}} \mid 1 \leq i,j \leq m+1\}$ as the set B_n defined in Lemma 2.2, for convenience, where $E_{n,0} = \{(i,j) \mid 1 \leq i,j \leq m+1\}$ and $E_{n,i,j}$ is the set $E_{n,0} - \{(i,j)\}$

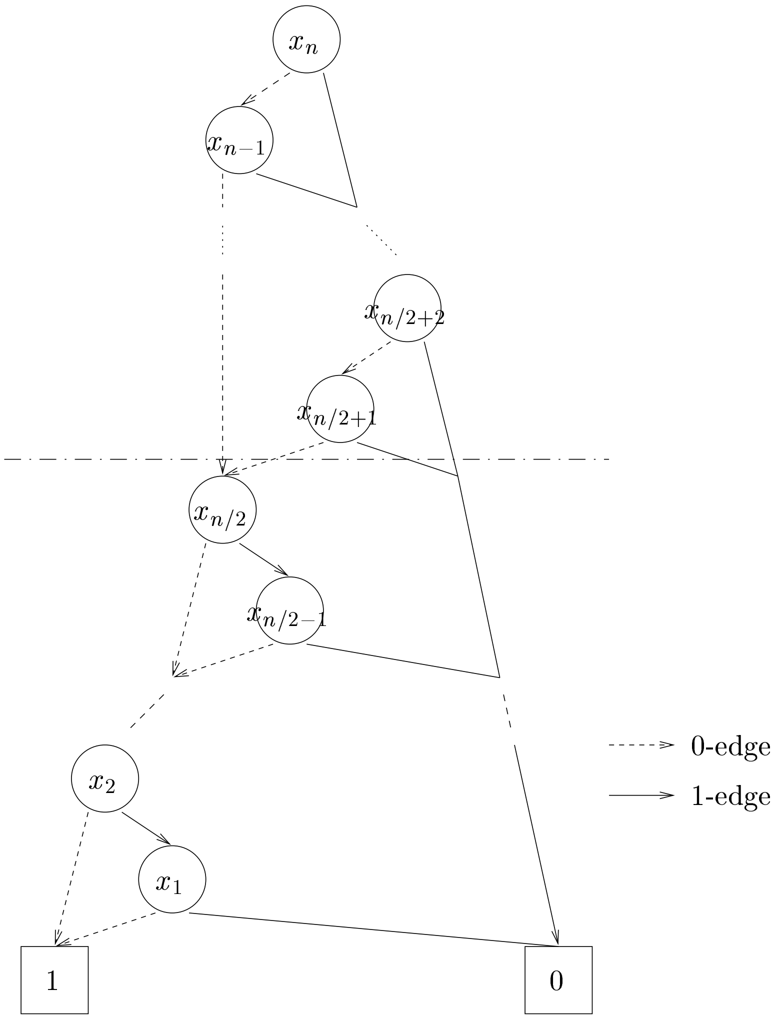

Figure 4: OBDD representing $f_C = (\bigwedge_{i=1}^{m}(\overline{x}_{2i-1} \vee \overline{x}_{2i})) \wedge \left(\bigvee_{i=m+1}^{2m}(\overline{x}_{2i-1} \wedge \overline{x}_{2i})\right)$.

corresponding to variable $x_{i,j}$. $f_D(\chi^{E_{n,0}}) = 1$ holds for the characteristic vector $\chi^{E_{n,0}}$. Thus, $|min_{\chi^{E_{n,0}}}(f_D)| = 1$. Similarly, $|min_{\chi^{E_{n,i,j}}}(f_D)| = 1$ holds for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$, since $f_D(\chi^{E_{n,i,j}}) = 1$.

Next, since $f_D(\chi^{E_{n,i,m+1}}) = 0$ is implied by $\left(x_{i,m+1} \vee \bigvee_{j=1}^{m} \overline{x}_{i,j}\right)(\chi^{E_{n,i,m+1}}) = 0$ for $i = 1, 2, \ldots, m$, we enumerate all minimal models for each $\chi^{E_{n,i,m+1}}$. By definition, we obtain $\chi^{E_{n,0}}$ by flipping the $(i, m+1)$-th coordinate of $\chi^{E_{n,i,m+1}}$. This $\chi^{E_{n,0}}$ is a minimal model for $\chi^{E_{n,i,m+1}}$ since $f_D(\chi^{E_{n,0}}) = 1$. When the $(i, m+1)$-th coordinate is fixed to 0, the clause $\left(x_{i,m+1} \vee \bigvee_{j=1}^{m} \overline{x}_{i,j}\right)$ is satisfied by flipping at least one of the (i, j)-th coordinates among $j = 1, 2, \ldots, m$. However, if two or more (i, j)-th coordinates are flipped, the corresponding vector is not minimal. Thus, we have $|min_{\chi^{E_{n,i,m+1}}}(f_D)| = m+1$ for $i = 1, 2, \ldots, m$. Similarly, we have $|min_{\chi^{E_{n,m+1,j}}}(f_D)| = m+1$ for $j = 1, 2, \ldots, m$.

We also enumerate all minimal models for $\chi^{E_{n,m+1,m+1}}$ since $f_D(\chi^{E_{n,m+1,m+1}}) = 0$. We obtain $\chi^{E_{n,0}}$ by flipping the $(m+1, m+1)$-th coordinate. When the $(m+1, m+1)$-th coordinate is fixed to 0, minimal models are obtained by flipping exactly one of the $(i, m+1)$-th coordinates among $i = 1, 2, \ldots, m$ and exactly one of the $(m+1, j)$-th coordinates among $j = 1, 2, \ldots, m$. Thus, we have $|min_{\chi^{E_{n,m+1,m+1}}}(f_D)| = m^2 + 1$. In total, we have $\sum_{a \in B_n} |min_a(f_D)| = O(m^2)$, i.e., $O(n)$. By Lemma 2.2, this means that the size of the characteristic set of f_D is $O(n)$. □

Lemma 3.4 [15] *Let f be a Boolean function on n variables $x_{i,j}$, $1 \leq i, j \leq m$, where $n = m^2$. Then, for any partition (L, R) satisfying $|L| = |R| = n/2$, either of the following properties holds:*

(1) *There are at least $m/\sqrt{2}$ different i's satisfying $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\} \cap L \neq \emptyset$ and $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\} \cap R \neq \emptyset$.*

(2) *There are at least $m/\sqrt{2}$ different j's satisfying $\{x_{1,j}, x_{2,j}, \ldots, x_{m,j}\} \cap L \neq \emptyset$ and $\{x_{1,j}, x_{2,j}, \ldots, x_{m,j}\} \cap R \neq \emptyset$.*

Lemma 3.5 *The size of OBDD representing the following negative function f_E on n variables $x_{i,j}$, $1 \leq i, j \leq m$, where $n = m^2$, is $\Omega(2^{m/\sqrt{2}})$ for any variable ordering:*

$$f_E = \left(\bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{m} \overline{x}_{i,j}\right)\right) \wedge \left(\bigwedge_{j=1}^{m} \left(\bigvee_{i=1}^{m} \overline{x}_{i,j}\right)\right).$$

Proof: We prove this by Lemma 2.3 in Section 2.2. Let us consider that the set X' in Lemma 2.3 is given by the set of all variables, and $\omega = 1/2$. Then, for every balanced partition (L, R), assuming case (1) of Lemma 3.4 without loss of generality, we have at least $m/\sqrt{2}$ different i's satisfying $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\} \cap L \neq \emptyset$ and $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\} \cap R \neq \emptyset$. We select $m/\sqrt{2}$ of these i's, $I = \{i_1, i_2, \ldots, i_{m/\sqrt{2}}\}$. For every $i_k \in I$, we can select two variables $x_{i_k, l_k} \in L$ and $x_{i_k, r_k} \in R$. We construct a set A of assignments such that each assignment satisfies the following restrictions:

(1) For every $i_k \in I$, $(x_{i_k, l_k}, x_{i_k, r_k})$ is assigned either $(0, 1)$ or $(1, 0)$.

(2) For every $i_k \in I$, all variables in $\{x_{i_k,1}, x_{i_k,2}, \ldots, x_{i_k,m}\} - \{x_{i_k,l_k}, x_{i_k,r_k}\}$ are assigned 1.

(3) Other variables are assigned 0.

The size of the set A is $2^{m/\sqrt{2}}$ since there are choices in restriction (1). Let $l^h \cdot r^h$ denote the assignment satisfying $h = \sum_{k=1}^{m/\sqrt{2}} a_k \cdot 2^{k-1}$, where $x_{i_k,l_k} = a_k \in \{0,1\}$ (and $x_{i_k,r_k} = \overline{a}_k$) for each $i_k \in I$. h satisfies $0 \leq h < 2^{m/\sqrt{2}}$.

Now, we prove that set A is a fooling set, defined just before Lemma 2.3. First, we show $f_E(l^h \cdot r^h) = 1$ for all h. Since one of x_{i_k,l_k} and x_{i_k,r_k} is assigned 0, $\bigvee_{j=1}^m \overline{x}_{i_k,j} = 1$ holds for all $i_k \in I$. For $i_k \notin I$, since $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ are assigned 0, we have $\bigvee_{j=1}^m \overline{x}_{i,j} = 1$. Also $\bigvee_{i=1}^m \overline{x}_{i,j} = 1$ holds for all $j \in \{1, 2, \ldots, m\}$. Thus, we have $f_E(l^h \cdot r^h) = 1$ for all h.

Next, we show that $f_E(l^h \cdot r^{h'}) = 0$ holds for $h > h'$. Since $h > h'$, there exists at least one variable x_{i_k,l_k} which is assigned 1 by $l^h \cdot r^h$ and 0 by $l^{h'} \cdot r^{h'}$. By restriction (1), x_{i_k,r_k} is then assigned 0 by $l^h \cdot r^h$ and 1 by $l^{h'} \cdot r^{h'}$. Therefore, x_{i_k,l_k} and x_{i_k,r_k} are assigned 1 by assignment $l^h \cdot r^{h'}$, implying that $\bigvee_{j=1}^m \overline{x}_{i_k,j} = 0$ holds. This proves that A is a fooling set. Since the size of this fooling set is at least $2^{m/\sqrt{2}}$ for any balanced partition, this lemma follows from Lemma 2.3.

□

Theorem 3.2 *There exists a Horn theory on n variables, for which CNF requires size $O(n)$ and the characteristic set requires size $O(n)$, while the size of the smallest OBDD representation is $\Omega(2^{\sqrt{n}/\sqrt{2}})$.*

Proof: Consider the function f_D in Lemma 3.3. As stated in Lemma 3.3, the size of its characteristic set is $O(n)$. Also the size of the CNF is obviously $O(n)$. The function f_E in Lemma 3.5 is obtained by restricting $x_{1,m+1}, \ldots, x_{m,m+1}$, $x_{m+1,1}, \ldots, x_{m+1,m}$ and $x_{m+1,m+1}$ of f_D to 0. Since the size of OBDD does not increase by a restriction, the size of the smallest OBDD of f_D is $\Omega(2^{\sqrt{n}/\sqrt{2}})$. □

The above results show that none of the three representations can always dominate the other two. OBDDs can find a place in knowledge-bases as they can represent some theories more efficiently than others.

Unfortunately, by combining Theorems 3.1 and 3.2, the following negative result is obtained.

Corollary 3.1 *There exists a Horn function on n variables, for which both of the characteristic set and CNF require sizes $\Omega(2^{n/8})$ and the size of the smallest OBDD representation is $\Omega(2^{\sqrt{n}/2})$.*

Proof: Consider a function which consists of two parts, where the first one (resp., second one) corresponds to f_C in Theorem 3.1 (resp., f_D in Theorem 3.2). Both have $n/2$ variables respectively, and share none of the variables. Similarly to the case of Theorem 3.1, the lower bounds for the three representations are easily obtained. □

4 Checking Unateness of OBDD

In this section, we discuss the problem of checking whether a given OBDD represents a unate function. We assume, without loss of generality, that the variable ordering is always $(x_n, x_{n-1}, \ldots, x_1)$. The following well-known property will show that this problem can be solved in polynomial time.

Property 4.1 *Let f be a Boolean function on n variables $x_1, x_2, \ldots, x_n$. Then, f is unate with polarity $p = (p_1, p_2, \ldots, p_n)$ if and only if $f|_{x_i=0} \leq f|_{x_i=1}$ for $p_i = 0$ (resp., $f|_{x_i=0} \geq f|_{x_i=1}$ for $p_i = 1$) holds for every $i = 1, 2, \ldots, n$.*

As noted in subsection 2.2, an OBDD representing $f|_{x_i=0}$ (resp., $f|_{x_i=1}$) can be obtained in $O(|f| \log |f|)$ time from the OBDD representing f, where $|f|$ denotes its size. The size does not increase by a restriction $f|_{x_i=0}$ or $f|_{x_i=1}$. Since the property $g \leq h$ can be checked in $O(|g| \cdot |h|)$ time, the unateness of f can be checked in $O(n|f|^2)$ time by checking the conditions $f|_{x_i=0} \leq f|_{x_i=1}$ and $f|_{x_i=0} \geq f|_{x_i=1}$ for all $i = 1, 2, \ldots, n$.

The following well-known property is useful to reduce the computation time.

Property 4.2 *Let f be a Boolean function on n variables $x_1, x_2, \ldots, x_n$. Then, f is unate with polarity $p = (p_1, p_2, \ldots, p_n)$ if and only if* (i) *both $f|_{x_n=0}$ and $f|_{x_n=1}$ are unate with polarity $(p_1, p_2, \ldots, p_{n-1})$, and* (ii) *$p_n = 0$ implies $f|_{x_n=0} \leq f|_{x_n=1}$ and $p_n = 1$ implies $f|_{x_n=0} \geq f|_{x_n=1}$.*

The unateness of functions $f|_{x_n=0}$ and $f|_{x_n=1}$ can be checked by applying Property 4.2 recursively, with an additional condition that $f|_{x_n=0}$ and $f|_{x_n=1}$ have the same polarity. Note that the property $f|_{x_n=0} \leq f|_{x_n=1}$ (resp., $f|_{x_n=0} \geq f|_{x_n=1}$) can be also checked recursively, since it holds if and only if $f|_{x_n=0, x_{n-1}=a_{n-1}} \leq f|_{x_n=1, x_{n-1}=a_{n-1}}$ (resp., $f|_{x_n=0, x_{n-1}=a_{n-1}} \geq f|_{x_n=1, x_{n-1}=a_{n-1}}$) holds for both $a_{n-1} = 0$ and 1.

Algorithm CHECK-UNATE in Fig. 5 checks the above conditions in the bottom-up manner (i.e., from level $\ell = 1$ to the root node). We use an array $p[\ell]$ to denote the polarity of f with respect to x_ℓ in level ℓ; each element stores 0, 1 or $*$ (not checked yet). We also use a two-dimensional array $imp[u, v]$ to denote whether $f_u \leq f_v$ holds or not; each element stores YES, NO or $*$ (not checked yet). In Step 2, the unateness with the polarity specified by array p is checked for the functions of the nodes in level ℓ. More precisely, the unateness of the functions is checked in Step 2-1, and the consistency of their polarities is checked in Step 2-2. In Step 3, $imp[u, v]$ are computed for the functions f_u and f_v in levels up to ℓ.

The unateness check of f_v in Step 2-1 can be easily done, since both $f_{0\text{-}succ(v)}$ (i.e., $f_v|_{x_\ell=0}$) and $f_{1\text{-}succ(v)}$ (i.e., $f_v|_{x_\ell=1}$) have already been checked to be unate with polarity $(p[1], p[2], \ldots, p[\ell-1])$, and $f_{0\text{-}succ(v)}$ and $f_{1\text{-}succ(v)}$ have been compared in Step 3 of the previous iteration. Note that constant functions 0 and 1 are considered to be unate. The polarity of f_v with respect to x_ℓ in level ℓ is temporarily stored in *pol* in Step 2-1.

In Step 2-2, the polarity consistency with respect to x_ℓ is checked by comparing the polarity of node v (which is *pol*) and $p[\ell]$. If $p[\ell] = *$ (i.e., v is the first node checked in level ℓ), we store

Algorithm CHECK-UNATE

Input: An OBDD representing f with a variable ordering $(x_n, x_{n-1}, \ldots, x_1)$.
Output: "yes" and its polarity if f is unate; otherwise, "no".

Step 1 (initialize). Set $p[\,i\,] := *$ for all $i = 1, 2, \ldots, n$;

$$imp[u,v] := \begin{cases} \text{NO} & \text{if } (f_u, f_v) = (1,0); \\ \text{YES} & \text{if } (f_u, f_v) = (0,0), (0,1), (1,1); \\ * & \text{otherwise}; \end{cases}$$

$\ell := 1$.

Step 2 (check unateness in level ℓ and compute $p[\,\ell\,]$). For each node v in level ℓ (i.e., labeled with x_ℓ), apply Steps 2-1 and 2-2.

> **Step 2-1.** Set $pol := 0$ if $imp[0\text{-}succ(v), 1\text{-}succ(v)] = \text{YES}$ holds; set $pol := 1$ if $imp[1\text{-}succ(v), 0\text{-}succ(v)] = \text{YES}$ holds; otherwise, output "no" and halt.
>
> **Step 2-2.** If $p[\,\ell\,] = *$, then set $p[\,\ell\,] := pol$. If $p[\,\ell\,] \neq *$ and $p[\,\ell\,] \neq pol$ hold, then output "no" and halt.

Step 3 (compute imp in level ℓ). For each pair of nodes (u,v) (where (u,v) and (v,u) are considered different) such that $level(u) \leq \ell$ and $level(v) \leq \ell$, and at least one of $level(u)$ and $level(v)$ is equal to ℓ, set $imp[u,v] := \text{YES}$ if both $imp[0\text{-}succ'(u), 0\text{-}succ'(v)]$ and $imp[1\text{-}succ'(u), 1\text{-}succ'(v)]$ are YES; otherwise, set $imp[u,v] := \text{NO}$.

Step 4 (iterate). If $\ell = n$, where n is the level of the root node, then output "yes" and polarity $p = (p[1], p[2], \ldots, p[n])$, and halt. Otherwise set $\ell := \ell + 1$ and return to Step 2.

Figure 5: Algorithm CHECK-UNATE to check the unateness of an OBDD.

pol in $p[\ell]$. Otherwise, *pol* is checked against $p[\ell]$ and "no" is output if they are not consistent. Note that CHECK-UNATE outputs $p[\ell] = *$ if there are no nodes in level ℓ (i.e., f does not depend on x_ℓ).

In Step 3, comparison between f_u and f_v is also performed easily, since the comparisons between $f_u|_{x_\ell=a_\ell}$ and $f_v|_{x_\ell=a_\ell}$ for both $a_\ell = 0$ and 1 have already been completed. Here we use the convention that 0-$succ'(v)$ (resp., 1-$succ'(v)$) denotes 0-$succ(v)$ (resp., 1-$succ(v)$) if $level(v) = \ell$, but denotes v itself if $level(v) < \ell$. This is because $f_v|_{x_\ell=0} = f_{0\text{-}succ(v)}$ and $f_v|_{x_\ell=1} = f_{1\text{-}succ(v)}$ hold if $level(v) = \ell$, and $f_v|_{x_\ell=0} = f_v|_{x_\ell=1} = f_v$ holds if $level(v) < \ell$. Note that $f_u = f_v$ holds if and only if u and v are the same node. After Step 3 is done for some ℓ, we know $imp[u,v]$ for all pairs of nodes u and v such that $level(u) \leq \ell$ and $level(v) \leq \ell$. We store all the results, although some of them may not be needed.

Next, we consider the computation time of this algorithm. In Step 2, the computation for each v is performed in constant time from the data already computed in the previous Step 3. Thus the total time of Step 2 is $O(|f|)$. In Step 3, the comparison between f_u and f_v for each pair (u, v) is performed in constant time. The number of pairs compared in Step 3 during the entire computation is $O\left(\binom{|f|}{2}\right) = O(|f|^2)$, which requires $O(|f|^2)$ time.

Theorem 4.1 *Given an OBDD representing a Boolean function f, checking whether f is unate can be done in $O(|f|^2)$ time, where $|f|$ is the size of the given OBDD.*

If we start Algorithm CHECK-UNATE with initial condition $p[i] := 0$ (resp., $p[i] := 1$) for all $i = 1, 2, \ldots, n$, we can check the positivity (resp. negativity) of f. This is because f is positive (resp., negative) if and only if the polarities of all nodes are 0 (resp., 1).

Corollary 4.1 *Given an OBDD representing a Boolean function f, checking whether f is positive (resp., negative) can be done in $O(|f|^2)$ time, where $|f|$ is the size of the given OBDD.*

5 Checking Horness of OBDD

In this section, we discuss the problem of checking whether a given OBDD represents a Horn function. After examining the condition for Horness in the next subsection, an algorithm will be given in subsection 5.2.

5.1 Conditions for Horness

We assume, without loss of generality, that the variable ordering is always $(x_n, x_{n-1}, \ldots, x_1)$. Denoting $f|_{x_n=0}$ and $f|_{x_n=1}$ by f_0 and f_1 for simplicity, f is given by

$$f = \overline{x}_n f_0 \vee x_n f_1,$$

where f_0 and f_1 are Boolean functions on $n-1$ variables $x_1, x_2, \ldots, x_{n-1}$. Similarly to the case of unateness, we check the Horness of f in the bottom-up manner.

Lemma 5.1 *Let f be a Boolean function on n variables $x_1, x_2, \ldots, x_n$, which is expanded as $f = \overline{x}_n f_0 \vee x_n f_1$. Then, f is Horn if and only if both f_0 and f_1 are Horn and $f_0 \wedge_{bit} f_1 \leq f_0$ holds.*

Proof: We first prove the identity

$$f \wedge_{bit} f = \overline{x}_n \ ((f_0 \wedge_{bit} f_0) \vee (f_0 \wedge_{bit} f_1)) \vee x_n \ (f_1 \wedge_{bit} f_1) \tag{1}$$

by considering all models:

$$\begin{aligned}
&\{a \mid a \in \Sigma(f)\} \wedge_{bit} \{b \mid b \in \Sigma(f)\} \\
&\quad = (\{(a', 0) \mid a' \in \Sigma(f_0)\} \cup \{(a', 1) \mid a' \in \Sigma(f_1)\}) \\
&\qquad \wedge_{bit} (\{(b', 0) \mid b' \in \Sigma(f_0)\} \cup \{(b', 1) \mid b' \in \Sigma(f_1)\}) \\
&\quad = \{c \mid c = (a', 0) \wedge_{bit} (b', 0),\ a' \in \Sigma(f_0) \text{ and } b' \in \Sigma(f_0)\} \\
&\qquad \cup \{c \mid c = (a', 0) \wedge_{bit} (b', 1),\ a' \in \Sigma(f_0) \text{ and } b' \in \Sigma(f_1)\} \\
&\qquad \cup \{c \mid c = (a', 1) \wedge_{bit} (b', 0),\ a' \in \Sigma(f_1) \text{ and } b' \in \Sigma(f_0)\} \\
&\qquad \cup \{c \mid c = (a', 1) \wedge_{bit} (b', 1),\ a' \in \Sigma(f_1) \text{ and } b' \in \Sigma(f_1)\} \\
&\quad = \{(c', 0) \mid c' = a' \wedge_{bit} b',\ a' \in \Sigma(f_0) \text{ and } b' \in \Sigma(f_0)\} \\
&\qquad \cup \{(c', 0) \mid c' = a' \wedge_{bit} b',\ a' \in \Sigma(f_0) \text{ and } b' \in \Sigma(f_1)\} \\
&\qquad \cup \{(c', 1) \mid c' = a' \wedge_{bit} b',\ a' \in \Sigma(f_1) \text{ and } b' \in \Sigma(f_1)\}.
\end{aligned}$$

Now, the if-part of the lemma is immediate from (1), because the Horness of f_0 and f_1 (i.e., $f_0 \wedge_{bit} f_0 = f_0$ and $f_1 \wedge_{bit} f_1 = f_1$) and the property $f_0 \wedge_{bit} f_1 \leq f_0$ imply

$$\begin{aligned}
f \wedge_{bit} f &= \overline{x}_n \ ((f_0 \wedge_{bit} f_0) \vee (f_0 \wedge_{bit} f_1)) \vee x_n \ (f_1 \wedge_{bit} f_1) \\
&= \overline{x}_n \ (f_0 \vee (f_0 \wedge_{bit} f_1)) \vee x_n \, f_1 \\
&= \overline{x}_n \, f_0 \vee x_n \, f_1 \\
&= f.
\end{aligned}$$

Next, we consider the only-if-part. $f \wedge_{bit} f = f = \overline{x}_n f_0 \vee x_n f_1$ and equality (1) imply

$$f_1 \wedge_{bit} f_1 = f_1 \tag{2}$$

$$(f_0 \wedge_{bit} f_0) \vee (f_0 \wedge_{bit} f_1) = f_0. \tag{3}$$

Equality (2) implies that f_1 is Horn. Also $f_0 \wedge_{bit} f_0 \geq f_0$ always holds since $a \wedge_{bit} a = a$ holds for any model a in $\Sigma(f_0)$. Thus, we have

$$(f_0 \wedge_{bit} f_0) \vee (f_0 \wedge_{bit} f_1) \geq f_0 \wedge_{bit} f_0 \geq f_0. \tag{4}$$

By combining (3) and (4), we have $f_0 \wedge_{bit} f_0 = f_0$, i.e., f_0 is Horn. Since $f \vee g = f$ holds if and only if $g \leq f$ holds, we also have $f_0 \wedge_{bit} f_1 \leq f_0$. □

The Horness of f_0 and f_1 can be checked by applying Lemma 5.1 recursively. The following lemma says that the condition $f_0 \wedge_{bit} f_1 \leq f_0$ in Lemma 5.1 can be also checked in the bottom-up manner.

Lemma 5.2 *Let f, g and h be Boolean functions on n variables, which are expanded as $f = \overline{x}_n f_0 \vee x_n f_1$, $g = \overline{x}_n g_0 \vee x_n g_1$ and $h = \overline{x}_n h_0 \vee x_n h_1$, respectively. Then, property $f \wedge_{bit} g \leq h$ holds if and only if $f_0 \wedge_{bit} g_0 \leq h_0$, $f_0 \wedge_{bit} g_1 \leq h_0$, $f_1 \wedge_{bit} g_0 \leq h_0$ and $f_1 \wedge_{bit} g_1 \leq h_1$ hold.*

Proof: The identity

$$f \wedge_{bit} g = \overline{x}_n \ ((f_0 \wedge_{bit} g_0) \vee (f_0 \wedge_{bit} g_1) \vee (f_1 \wedge_{bit} g_0)) \vee x_n \ (f_1 \wedge_{bit} g_1) \tag{5}$$

can be proved in a manner similar to (1) by considering all models. Then, since $f \wedge_{bit} g \leq h$ holds if and only if $(f \wedge_{bit} g)|_{x_n=0} = (f_0 \wedge_{bit} g_0) \vee (f_0 \wedge_{bit} g_1) \vee (f_1 \wedge_{bit} g_0) \leq h|_{x_n=0} = h_0$ and $(f \wedge_{bit} g)|_{x_n=1} = f_1 \wedge_{bit} g_1 \leq h|_{x_n=1} = h_1$ hold, we can prove this lemma by Lemma 2.1(2). □

Note that the condition of type $f \wedge_{bit} g \leq f$ in Lemma 5.1 requires to check the condition of type $f_1 \wedge_{bit} g_0 \leq f_0$ (i.e., checking of type $f \wedge_{bit} g \leq h$ for three functions f, g and h). The last condition can be checked recursively by Lemma 5.2.

5.2 Algorithm to Check Horness

Applying Lemmas 5.1 and 5.2 recursively, the Horness of a Boolean function f can be checked by scanning all nodes in a given OBDD in the bottom-up manner. Namely, for each node v in level ℓ, we check the condition of Lemma 5.1, i.e., whether both $f_v|_{x_\ell=0}$ and $f_v|_{x_\ell=1}$ are Horn and $f_v|_{x_\ell=0} \wedge_{bit} f_v|_{x_\ell=1} \leq f_v|_{x_\ell=0}$ holds. Lemma 5.2 gives the condition how $f_v|_{x_\ell=0} \wedge_{bit} f_v|_{x_\ell=1} \leq f_v|_{x_\ell=0}$ can be checked in the bottom-up manner.

Algorithm CHECK-HORN in Fig. 6 checks the Horness of a given OBDD in the above manner. We use an array $horn[v]$ to denote whether each node v represents a Horn function or not, and a three-dimensional array $bit\text{-}imp[u,v,w]$ to denote whether $f_u \wedge_{bit} f_v \leq f_w$ holds or not. Each element of the arrays stores YES, NO or $*$ (not checked yet); $horn[v]$ = YES says that f_v is Horn and $bit\text{-}imp[u,v,w]$ = YES says that $f_u \wedge_{bit} f_v \leq f_w$ holds. We note here that, since the OBDD is reduced, the condition $f_u \wedge_{bit} f_v \leq f_w$ may be checked for functions in different levels; in such case, all functions are considered to have l_{max} variables by adding dummy variables, where l_{max} denotes the maximum level of the nodes u, v and w.

In Step 2 of Algorithm CHECK-HORN, $horn[v]$ for each v can be computed in constant time by Fig. 7, which corresponds to Lemma 5.1, since $f_v|_{x_{level(v)}=0} = f_{0\text{-}succ(v)}$ and $f_v|_{x_{level(v)}=1} = f_{1\text{-}succ(v)}$ hold, and $horn[0\text{-}succ(v)]$, $horn[1\text{-}succ(v)]$ and $bit\text{-}imp[0\text{-}succ(v), 1\text{-}succ(v), 0\text{-}succ(v)]$ in Fig. 7 have already been computed in the previous iterations.

Similarly, $bit\text{-}imp[u,v,w]$ in Step 3 for each triple (u,v,w) can be computed in constant time by Fig. 8, which corresponds to Lemma 5.2. As in the case of Algorithm CHECK-UNATE, $0\text{-}succ'(v)$ (resp., $1\text{-}succ'(v)$) denotes $0\text{-}succ(v)$ (resp., $1\text{-}succ(v)$) if $level(v) = \ell$, but denotes v itself if $level(v) < \ell$. Upon completing Step 3 for ℓ, we have the results $bit\text{-}imp[u,v,w]$ for all triples (u,v,w) such that $level(u) \leq \ell$, $level(v) \leq \ell$ and $level(w) \leq \ell$. These contain all the information required in the next iteration, although some of them may not be needed.

Now, we consider the computation time of Algorithm CHECK-HORN. In Step 2, since $horn[v]$ for each node v is computed in constant time, $O(|f|)$ time is required for checking all

Algorithm CHECK-HORN

Input: An OBDD representing f with a variable ordering $(x_n, x_{n-1}, \dots, x_1)$.
Output: "yes" if f is Horn; otherwise, "no".

Step 1 (initialize). Set

$$horn[v] := \begin{cases} \text{YES} & \text{if } v \text{ is a constant node 0 or 1;} \\ * & \text{otherwise;} \end{cases}$$

$$\mathit{bit\text{-}imp}[u,v,w] := \begin{cases} \text{NO} & \text{if } (f_u, f_v, f_w) = (1,1,0); \\ \text{YES} & \text{if } f_u, f_v, f_w \in \{0,1\} \text{ and } (f_u, f_v, f_w) \neq (1,1,0); \\ * & \text{otherwise;} \end{cases}$$

$\ell := 1$.

Step 2 (check Horness in level ℓ). For each node v in level ℓ (i.e., labeled with x_ℓ), check whether the function f_v is Horn according to Fig. 7, and set its result YES or NO to $horn[v]$. If there exists at least one node in level ℓ which is not Horn, output "no" and halt.

Step 3 (compute *bit-imp* in level ℓ). For each triple (u, v, w) of nodes such that $level(u) \leq \ell$, $level(v) \leq \ell$ and $level(w) \leq \ell$, and at least one of $level(u)$, $level(v)$ and $level(w)$ is equal to ℓ, check whether $f_u \wedge_{bit} f_v \leq f_w$ holds according to Fig. 8, and set its result YES or NO to $\mathit{bit\text{-}imp}[u, v, w]$.

Step 4 (iterate). If $\ell = n$ then output "yes" and halt. Otherwise set $\ell := \ell + 1$ and return to Step 2.

Figure 6: Algorithm CHECK-HORN to check the Horneness of an OBDD.

YES	if	all of $horn[0\text{-}succ(v)]$, $horn[1\text{-}succ(v)]$ and $bit\text{-}imp[0\text{-}succ(v), 1\text{-}succ(v), 0\text{-}succ(v)]$ are YES.
NO	otherwise.	

Figure 7: Checking $horn[v]$ for a node v in Step 2.

YES	if	all of $bit\text{-}imp[1\text{-}succ'(u), 1\text{-}succ'(v), 1\text{-}succ'(w)]$, $bit\text{-}imp[0\text{-}succ'(u), 0\text{-}succ'(v), 0\text{-}succ'(w)]$, $bit\text{-}imp[0\text{-}succ'(u), 1\text{-}succ'(v), 0\text{-}succ'(w)]$ and $bit\text{-}imp[1\text{-}succ'(u), 0\text{-}succ'(v), 0\text{-}succ'(w)]$ are YES.
NO	otherwise.	

Figure 8: Checking $bit\text{-}imp[u, v, w]$ (i.e., $f_u \wedge_{bit} f_v \leq f_w$)
for a triple of nodes (u, v, w) in Step 3.

nodes in the OBDD. In Step 3, $bit\text{-}imp[u, v, w]$ for each triple (u, v, w) is also computed in constant time. The number of triples to be checked in Step 3 during the entire computation is $O(|f|^3)$. The time for the rest of computation is minor.

Theorem 5.1 *Given an OBDD representing a Boolean function f, checking whether f is Horn can be done in $O(|f|^3)$ time, where $|f|$ is the size of the given OBDD.*

6 Conclusion

In this paper, we considered to use OBDDs to represent knowledge-bases. We have shown that the conventional CNF-based and model-based representations, and the new OBDD representation are mutually incomparable with respect to space requirement. Thus, OBDDs can find their place in knowledge-bases, as they can represent some theories more efficiently than others.

We then considered the problem of recognizing whether a given OBDD represents a unate Boolean function, and whether it represents a Horn function. It turned out that checking unateness can be done in quadratic time of the size of OBDD, while checking Horness can be done in cubic time.

OBDDs are dominatingly used in the field of computer-aided design and verification of digital systems. The reason for this is that many Boolean functions which we encounter in practice can be compactly represented, and that many operations on OBDDs can be efficiently performed. We believe that OBDDs are also useful for manipulating knowledge-bases. Developing efficient algorithms for knowledge-base operations such as deduction and abduction should be addressed in the further work.

Acknowledgement

The authors would like to thank Professor Endre Boros of Rutgers University for his valuable comments. This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan.

References

[1] S.B. Akers, "Binary Decision Diagrams," *IEEE Trans. Comput.*, C-27, no.6, pp.509–516, 1978.

[2] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Comput.*, C-35, no.8, pp.677–691, 1986.

[3] R.E. Bryant, "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication," *IEEE Trans. Comput.*, 40, no.2, pp.205–213, 1991.

[4] J.R. Burch, E.M. Clarke, K.L. McMillan, and D.L. Dill, "Sequential Circuit Verification Using Symbolic Model Checking," *in Proc. of 27th ACM/IEEE DAC*, pp.46–51, 1990.

[5] W.F. Dowling and J.H. Gallier, "Linear Time Algorithms for Testing the Satisfiability of Horn Formula," *J. Logic Programm.*, 3, pp.267–284, 1984.

[6] H.A. Kautz, M.J. Kearns, and B. Selman, "Reasoning with Characteristic Models," *in Proc. of AAAI-93*, pp.34–39, 1993.

[7] H.A. Kautz, M.J. Kearns, and B. Selman, "Horn Approximations of Empirical Data," *Artificial Intelligence*, 74, pp.129–245, 1995.

[8] D. Kavvadias, C. Papadimitriou, and M. Sideri, "On Horn Envelopes and Hypergraph Transversals," *in Proc. of 4th International Symposium on Algorithms and Computation ISAAC-93*, eds. K.W. Ng et al., LNCS 762, Springer, pp. 399–405, 1993.

[9] R. Khardon and D. Roth, "Reasoning with Models," *Artificial Intelligence*, 87, pp.187–213, 1996.

[10] J.C. Madre and O. Coudert, "A Logically Complete Reasoning Maintenance System Based on a Logical Constraint Solver," *in Proc. of IJCAI-91*, pp.294–299, 1991.

[11] J. McCarthy and P.J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," *in Machine Intelligence 4*, ed. D. Michie, Edinburgh University Press, 1969.

[12] S. Minato, N. Ishiura and S. Yajima, "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation," *in Proc. of 27th ACM/IEEE DAC*, pp.52–57, 1990.

[13] B. Selman and H.J. Levesque, "Abductive and Default Reasoning: A Computational Core," *in Proc. of AAAI-90*, pp.343–348, 1990.

[14] N. Takahashi, N. Ishiura, and S. Yajima, "Fault Simulation for Multiple Faults Using BDD Representation of Fault Sets," *in Proc. of IEEE/ACM ICCAD-91*, pp.550–553, 1991.

[15] Y. Takenaga, Theoretical Studies on Memory-Based Parallel Computation and Ordered Binary Decision Diagrams, Ph.D. Thesis, Department of Information Science, Graduate School of Engineering, Kyoto University, 1994.