SDDs Are Exponentially More Succinct than OBDDs

Simone Bova
Technische Universität Wien
Favoritenstraße 9–11
1040 Wien (Austria)

Abstract

Introduced by Darwiche (2011), sentential decision diagrams (SDDs) are essentially as tractable as ordered binary decision diagrams (OBDDs), but tend to be more succinct *in practice*. This makes SDDs a prominent representation language, with many applications in artificial intelligence and knowledge compilation.

We prove that SDDs are more succinct than OBDDs also *in theory*, by constructing a family of boolean functions where each member has polynomial SDD size but exponential OBDD size. This exponential separation improves a quasipolynomial separation recently established by Razgon (2014a), and settles an open problem in knowledge compilation (Darwiche 2011).

1 Introduction

The idea of *knowledge compilation* is to deal with the intractability of certain computational tasks on a knowledge base by compiling it into a different data structure where the tasks are feasible. The choice of the target data structure involves an unavoidable trade-off between succinctness and tractability.

Darwiche and Marquis (2002) systematically investigated this trade-off in the fundamental case where the knowledge bases are boolean functions and the data structures are classes of boolean circuits (*representation languages*).

In their setting, *decomposable negation normal forms (DNNFs)* and *ordered binary decision diagrams (OBDDs)* arise as benchmark languages for succinctness and tractability respectively (Darwiche 2001; Darwiche and Marquis 2002). On the one hand, DNNFs are exponentially more succinct than OBDDs; moreover, in contrast to OBDDs, they implement efficiently conjunctive normal forms of small treewidth (Darwiche 2001; Razgon and Petke 2013; Oztok and Darwiche 2014; Razgon 2014b). On the other hand, the vast applicability of OBDDs in verification and synthesis relies on the tractability of equivalence testing (speeded up by canonicity) and boolean combinations, which DNNFs lack (Darwiche and Marquis 2002).

This gap between DNNFs (succinct but hard) and OBDDs (verbose but tractable) led to the quest for intermediate languages exponentially more succinct than, but essentially as tractable as, OBDDs.

Introduced by Darwiche (2011), *sentential decision diagrams (SDDs)* are a most prominent candidate to narrow the gap between DNNFs and OBDDs. They are designed by strengthening the decomposability property (Pipatsrisawat and Darwiche 2008) and further imposing a very strong form of determinism (Pipatsrisawat and Darwiche 2010). The resulting language can implement decisions of the form

$$\bigvee_{i=1}^{m} P_i(X) \wedge S_i(Y), \qquad (1)$$

where X and Y are disjoint sets of variables nicely structured by an underlying *variable tree*, and the subcircuits $P_1, \ldots, P_m$, called *primes*,[1] implement an exhaustive case distinction into exclusive and consistent cases.[2] Binary (or Shannon) decisions in OBDDs boil down to very special sentential decisions having the form

$$(\neg x \wedge S_1(Y)) \vee (x \wedge S_2(Y)),$$

where the variable x is not in the variable set Y.

Indeed, SDDs properly contain OBDDs, and hence are at least as succinct as OBDDs, while preserving tractability of all key tasks that are tractable on OBDDs. For this reason, they have been used in a variety of applications in artificial intelligence and probabilistic reasoning, as reported, for instance, by (Van den Broek and Darwiche 2015; Oztok and Darwiche 2015).

Not only SDDs are as tractable as OBDDs, but they also tend to be more succinct than OBDDs in practice; in fact, knowledge compilers often produce much smaller SDDs than OBDDs by heuristically leveraging the additional flexibility of variable trees in SDDs with respect to variable orderings in OBDDs (Choi and Darwiche 2013; Oztok and Darwiche 2015).

Nonetheless, the basic theoretical question about the relative succinctness of OBDDs and SDDs has been open since Darwiche introduced SDDs (Darwiche 2011; Razgon 2014a):

Are SDDs exponentially more succinct than OBDDs?

The results in the literature did not even exclude the possibility for OBDDs to polynomially simulate SDDs (Xue,

---

[1] The S_i's are called *subs*.

[2] Formally, the models of $P_1, \ldots, P_m$ partition the set of assignments of X to $\{0, 1\}$ into m nonempty blocks; see Section 2.

Choi, and Darwiche 2012), until recently Razgon proved a quasipolynomial separation (Razgon 2014a). The above question stands, though, as for instance OBDDs could still quasipolynomially simulate SDDs.

Contribution. We prove in this article that SDDs are exponentially more succinct than OBDDs. Thus, in particular, OBDDs cannot quasipolynomially simulate SDDs.

More precisely, *we construct an infinite family of boolean functions such that every member of the family has polynomial compressed SDD size but exponential OBDD size* (Theorem 4).

Compressed SDDs contain OBDDs,[3] and are regarded as a natural SDD class because of their *canonicity*: two compressed SDDs computing the same function are syntactically equal up to syntactic manipulations preserving polynomial size (Darwiche 2011). The restriction to compressed SDDs makes our result stronger, because general SDDs are believed (despite not known) to be exponentially more succinct than compressed SDDs (Van den Broek and Darwiche 2015).

We separate compressed SDDs and OBDDs by a function, which we call the *generalized hidden weighted bit* function because, indeed, it contains the *hidden weighted bit* function (HWB) as a subfunction. HWB is perhaps the simplest function known to be hard on OBDDs (Bryant 1986): it computes the subsets of $\{1, \ldots, n\}$ having size i and containing the number i, for $i = 1, \ldots, n$.

It turns out that HWB itself has small (uncompressed) SDDs (Theorem 3), which immediately separates SDDs and OBDDs. The construction, a slight variation of which gives the compressed case (Lemma 1 and Lemma 2), is based on the following two observations.

The first observation is that HWB can be expressed as a sentential decision of the form (1) by distinguishing the following primes:

- for $i = 1, \ldots, n$, the subsets of size i containing the number i (each of these n primes is taken by HWB, so their subs will be equivalent to $\top$);

- the empty subset, and the subsets of size i not containing the number i for $i = 1, \ldots, n-1$ (none of these n primes is taken by HWB, so their subs will be equivalent to $\bot$).

The second observation is that each of the above primes has small OBDD size under any variable ordering (Proposition 2). With these two observations it is fairly straightforward to implement the hidden weighted bit function by a small (uncompressed) SDD (Theorem 3).

A direct inspection of our construction allows to straightforwardly derive some facts about compression previously observed in the literature (Van den Broek and Darwiche 2015), namely that the SDD size may increase exponentially either by compressing SDDs over fixed variable trees, or by conditioning (unboundedly many variables) over fixed variable trees (see Section 4).

_____

[3] More precisely, compressed SDDs contain reduced OBDDs; see (Wegener 2000, Definition 1.3.2).

Organization. The article is organized as follows. In Section 2 we present the technical background, culminating in the quasipolynomial separation of SDDs and OBDDs proved by Razgon (Theorem 1). In Section 3, we separate (uncompressed) SDDs and OBDDs by the hidden weighted bit function (Theorem 3) and then modify the construction to separate compressed SDDs and OBDDs (Theorem 4). We discuss our results in Section 4.

2 Background

We collect background notions and facts from the literature (Darwiche and Marquis 2002; Pipatsrisawat and Darwiche 2008; Darwiche 2011; Razgon 2014a).

Structured Deterministic NNFs. Let X be a finite set of variables. Let C be a boolean circuit on input variables X, built using fanin 0 constant gates (labelled by $\bot$ or $\top$), fanin 1 negation gates (labelled by $\neg$), and unbounded fanin disjunction and conjunction gates (labelled by $\vee$ and $\wedge$). The unique sink node (outdegree 0) in the underlying directed acyclic graph (DAG) of C is called the output gate of C; source nodes (indegree 0) are called input gates, and are labelled by constants or variables in X; in particular, C is allowed to not read some of the variables in X, see Figure 1 (left).

A boolean circuit C on variables X is in *negation normal form*, in short an *NNF*, if the gates labelled by $\neg$ have wires only from input gates. Without loss of generality we assume that NNFs have input gates labelled by constants or literals on variables in X (and no internal gates labelled by $\neg$).

As usual, an NNF C on input variables X computes a boolean function $f: \{0, 1\}^X \rightarrow \{0, 1\}$; in this case we also write $C \equiv f$. Two NNFs C and C' on the same input variables are equivalent if they compute the same boolean function; again we write $C \equiv C'$.

The *size* of an NNF C, in symbols $\text{size}(C)$, is the number of arcs in its underlying DAG. Let f be a boolean function and let $\mathcal{L}$ be a class of NNFs. The *size of f relative to $\mathcal{L}$* (or, in short, the $\mathcal{L}$ size of f), denoted by $\mathcal{L}(f)$, is equal to the minimum over the sizes of all circuits in $\mathcal{L}$ computing f:

$$\mathcal{L}(f) = \min\{\text{size}(C): C \in \mathcal{L}, C \equiv f\}.$$

Let C be an NNF on input variables X, and let g be a gate of C. We denote by C_g the subcircuit of C having g as its output gate, that is, the circuit whose underlying DAG is the subgraph of the underlying DAG of C induced by the nodes having a directed path to g (labelled as in C).

An NNF C on input variables X is *deterministic* if, for every $\vee$-gate g in C, say of the form $\bigvee_{i=1}^{m} g_i$, it holds that

$$C_{g_i} \wedge C_{g_j} \equiv \bot$$

for all $1 \leq i < j \leq m$, where we formally regard C_{g_i}, C_{g_j}, and $\bot$ as NNFs on input variables X. We denote by $\mathcal{NNF}_d$ the class of all deterministic NNFs.

Let Y be a finite nonempty set of variables. A *variable tree* (in short, a *vtree*) for the variable set Y is a rooted, full, ordered, binary tree T whose leaves correspond bijectively to Y; indeed, we identify each leaf in T with the variable in Y it corresponds to.
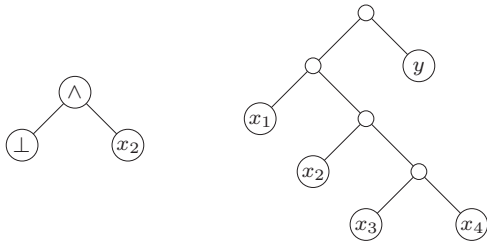
Figure 1: A circuit on input variables $\{x_2, x_4\}$ on the left (in the underlying DAG, the edges are oriented upwards), respecting the vtree for the variable set $\{x_1, x_2, x_3, x_4, y\}$ on the right. The left subtree is a vtree for the variable set $\{x_1, x_2, x_3, x_4\}$, and the right subtree is a vtree for the variable set $\{y\}$. The $\wedge$-gate in the circuit respects the root of the vtree.

Let v be an internal node of the vtree T. We let v_l and v_r denote respectively the left and right child of v, and T_v denote the subtree of T rooted at v. We also let $Y_v \subseteq Y$ denote (the variables corresponding to) the leaves of T_v; clearly T_v is a vtree for the variable set Y_v.

Let C be an NNF on input variables X, and let T be a vtree for the variable set Y.

We say that C *respects* T if the following holds. First, every $\wedge$-gate g in C has fanin exactly 2. Second, let g be an $\wedge$-gate in C having wires from gates h_1 and h_2. Then there exists an internal node v in T such that the input gates of the subcircuit C_{h_1} mention only variables in T_{v_l} and the input gates of the subcircuit C_{h_2} mention only variables in T_{v_r}. In this case, we also say that g respects v.

Note that, in particular, the sets of variables mentioned by C_{h_1} and C_{h_2} are disjoint; it follows that C is decomposable (Darwiche 2001). Also note that, by definition, if an NNF reading all the variables in a set X is structured by a vtree for the variable set Y, then $X \subseteq Y$ and the inclusion can be strict; see Figure 1. This feature is crucial in our construction (see, for instance, the proof of Theorem 3).

A *structured NNF* is an NNF respecting some vtree. See Figure 1. We denote by $\mathcal{NNF}_s$ the class of all structured NNFs.

SDDs and OBDDs. A *sentential decision diagram (SDD)* C *respecting a vtree* T is defined inductively as follows.

- C is a single gate labelled by a literal on a variable x, and x is in the variable set of T.

- C is a single gate labelled by a constant, and T is any vtree.

- C is formed by an output gate g labelled by $\vee$, with $m \geq 2$ wires from gates $g_1, \ldots, g_m$ labelled by $\wedge$, where each g_i has wires from two gates p_i and s_i, that is,

$$C = \bigvee_{i=1}^{m} C_{p_i} \wedge C_{s_i}, \qquad (2)$$

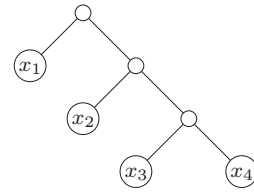such that for some internal node v of T the following holds $(i = 1, \ldots, m)$:

(S1) C_{p_i} is an SDD respecting a subtree of T_{v_l}.
(S2) C_{s_i} is an SDD respecting a subtree of T_{v_r}.
(S3) $C_{p_i} \not\equiv \bot$.
(S4) $C_{p_i} \wedge C_{p_j} \equiv \bot \ (1 \leq i < j \leq m)$.
(S5) $\bigvee_{i=1}^{m} C_{p_i} \equiv \top$.

In the equivalences in (S3)-(S5), we formally regard the C_{p_i}'s, $\bot$ and $\top$ as NNFs on variables Y_{v_l}. In words, conditions (S3)-(S5) say that the C_{p_i}'s define a partition of $\{0, 1\}^{Y_{v_l}}$ into m nonempty blocks, where the ith block contains exactly the models of C_{p_i} $(i = 1, \ldots, m)$.

An SDD is an SDD respecting some vtree. We let $\mathcal{SDD}$ denote the class of all SDDs.

An SDD C is called *compressed* if the following holds. Let h be an $\vee$-gate of C, so that $h = \bigvee_{i=1}^{m'} C_{p_i'} \wedge C_{s_i'}$ specified as in (2) relative to some node v' in T. Then

(C) $C_{s_i'} \not\equiv C_{s_j'} \ (1 \leq i < j \leq m')$,

where we formally regard $C_{s_i'}$ as an NNF on variables $Y_{v_r'}$ for $i = 1, \ldots, m'$. We let $\mathcal{SDD}_c$ denote the class of all compressed SDDs.

An *ordered binary decision diagram (OBDD)* is a compressed SDD respecting a *right-linear* vtree T (that is, where each left child is a leaf); see Figure 2. We let $\mathcal{OBDD}$ denote the class of all OBDDs.[4]

Let C be an OBDD respecting a vtree T, and let $\sigma = x_1 < \cdots < x_n$ be the variable ordering induced by a left first traversal of T; in this case, we also say that C respects σ. For an ordering σ of a set of variables, we let $\mathcal{OBDD}_\sigma$ denote the class of all OBDDs respecting σ.

Quasipolynomial Separation. It follows from the definitions that

$$\mathcal{OBDD} \subseteq \mathcal{SDD}_c \subseteq \mathcal{SDD} \subseteq \mathcal{NNF}_s \cap \mathcal{NNF}_d \qquad (3)$$

which raises the natural question how OBDDs and SDDs are related in succinctness; indeed, the quest for the relative succinctness of OBDDs and SDDs has been an open problem in knowledge compilation since Darwiche introduced SDDs (Darwiche 2011).

Recently, Razgon (2014a, Corollary 3) has established a *quasipolynomial separation* of OBDDs from compressed SDDs.

Figure 2: A right-linear vtree; its left first traversal induces the variable ordering $x_1 < x_2 < x_3 < x_4$.

---

[4]Reduced OBDDs as usually defined in the literature (Wegener 2000, Definition 1.3.2) are indeed compressed SDD respecting right-linear vtrees (Darwiche 2011, Section 6).

Theorem 1 (Razgon). *There exists an unbounded arity class of boolean functions $\mathcal{F}$ such that every arity n function $f \in \mathcal{F}$ has $\mathcal{SDD}_c$ size in $O(n^3)$ and $\mathcal{OBDD}$ size in $n^{\Omega(\log n)}$.*

We remark that the restriction to compressed SDDs in the above statement is nontrivial; to the best of our knowledge, compressed SDDs might be exponentially more succinct than uncompressed SDDs (Van den Broek and Darwiche 2015); see also the discussion in Section 4.

3   Exponential Separation

The quasipolynomial separation stated in Theorem 1 implies that OBDDs do not simulate SDDs in polynomial size, but leaves open the possibility for OBDDs to simulate SDDs in quasipolynomial size. In this section we exclude this possibility by establishing an *exponential separation* of OBDDs from compressed SDDs.

Hidden Weighted Bit.   The separation is obtained by (a variant of) the *hidden weighted bit* function

$$\text{HWB}_n(x_1, \ldots, x_n),$$

that is the boolean function on n inputs $x_1, \ldots, x_n$ such that, for all assignments $f \colon \{x_1, \ldots, x_n\} \to \{0, 1\}$, it holds that f is a model of HWB_n if and only if $f(x_1) + \cdots + f(x_n) = i$ and $f(x_i) = 1$ $(i \geq 1)$.

It is well known that the hidden weighted bit function has exponential OBDD size (Bryant 1986).

Theorem 2 (Bryant). *The $\mathcal{OBDD}$ size of HWB_n is $2^{\Omega(n)}$.*

Intuitively, a model of HWB_n is a subsets of $\{1, \ldots, n\}$ of size i containing the number i, for $i = 1, \ldots, n$. For instance, $\text{HWB}_2(1, 0) = 1$, because the set $\{1\}$ has size 1 and contains the number 1, and $\text{HWB}_2(0, 1) = 0$, because the set $\{2\}$ has size 1 but does not contain the number 1.

The simple but crucial observation underlying our construction is that the models of HWB_n can be decided arguing by cases, as follows: If S is a subset of $\{1, \ldots, n\}$ of size i, then S is a model of HWB_n if and only if $i \in S$ $(i = 1, \ldots, n)$. With this insight it is not hard to setup an exhaustive and exclusive case distinction equivalent to HWB_n; the key observation is that each individual case in the distinction is computable by a small OBDD with respect to any variable ordering.

We formalize the above intuition. For $i \in \{0, 1, \ldots, n\}$, let

$$E_n^i(x_1, \ldots, x_n)$$

be the boolean function on n inputs $x_1, \ldots, x_n$ such that, for all assignments $f \colon \{x_1, \ldots, x_n\} \to \{0, 1\}$, it holds that f is a model of E_n^i if and only if $f(x_1) + \cdots + f(x_n) = i$. Hence E_n^i computes the subsets of $\{1, \ldots, n\}$ of size i $(i \geq 0)$. Let now

$$\mathcal{P}_n = \{P_0, P_n\} \cup \{P_{i,0}, P_{i,1} \colon i = 1, \ldots, n-1\} \quad (4)$$

be the family of $2n$ boolean functions, each over the variables $\{x_1, \ldots, x_n\}$, defined as follows:

- $P_0 \equiv E_n^0$
- $P_n \equiv E_n^n$
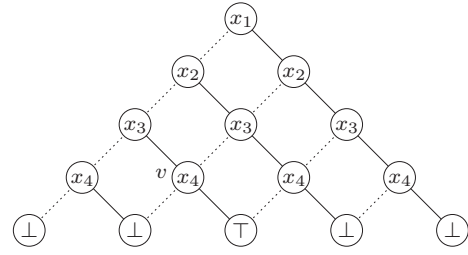
Figure 3: An OBDD for the boolean function E_4^2 respecting the variable ordering $x_1 < x_2 < x_3 < x_4$, drawn (in an unreduced form) using the graphical conventions for decision diagrams (Wegener 2000). Each decision node generates 6 wires in the circuit; for instance, the decision node v generates a 6-wire subcircuit isomorphic to $(\neg x_4 \wedge \bot) \vee (x_4 \wedge \top)$.

and for $i = 1, \ldots, n-1$ let

- $P_{i,0} \equiv E_n^i \wedge \neg x_i$
- $P_{i,1} \equiv E_n^i \wedge x_i$

See Figure 4 for an illustration.

Each function in $\mathcal{P}_n$ computes a family of subsets of $\{1, \ldots, n\}$. Namely, P_0 computes the empty subset, P_n computes $\{1, \ldots, n\}$, $P_{i,0}$ computes the subsets of $\{1, \ldots, n\}$ of size i not containing the number i, and $P_{i,1}$ computes the subsets of $\{1, \ldots, n\}$ of size i containing the number i $(i = 1, \ldots, n-1)$.

It is readily observed that the members of $\mathcal{P}_n$ partition the powerset of $\{1, \ldots, n\}$ in nonempty blocks. Formally,

Fact 1. *Let $\mathcal{P}_n$ be as in (4), and let $P, P' \in \mathcal{P}_n$ with $P \neq P'$.*

- $P \not\equiv \bot$.
- $P \wedge P' \equiv \bot$.
- $\bigvee_{P \in \mathcal{P}_n} P \equiv \top$.

We now establish the key property, that each member of $\mathcal{P}_n$ is computable by a small OBDD with respect to any variable ordering.

First consider the functions E_n^i. An OBDD computing E_n^i with respect to the variable ordering $\sigma = x_1 < \cdots < x_n$ is displayed in Figure 3 for the case $n = 4$ and $i = 2$. Generalizing the construction, we have that an OBDD C computing E_n^i and respecting σ has at most $1 + 2 + \cdots + n = n(n+1)/2$ decision nodes, each contributing 6 wires in the circuit; hence C has size $O(n^2)$.

Since E_n^i is symmetric (Wegener 2000, Definition 2.3.2 and Lemma 4.7.1), the following holds.

Proposition 1. *Let σ be an ordering of $x_1, \ldots, x_n$. The $\mathcal{OBDD}_\sigma$ size of E_n^i is $O(n^2)$.*

It follows that every $P \in \mathcal{P}_n$ has a small OBDD with respect to every variable ordering.

Proposition 2. *Let σ be an ordering of $x_1, \ldots, x_n$ and let $P \in \mathcal{P}_n$, where $\mathcal{P}_n$ is as in (4). The $\mathcal{OBDD}_\sigma$ size of P is $O(n^2)$.*

Proof. For P_0 and P_n the statement follows directly from Proposition 1. For $i = 1, \ldots, n-1$ we have that $P_{i,0} \equiv E_n^i \wedge \neg x_i$ and $P_{i,1} \equiv E_n^i \wedge x_i$.
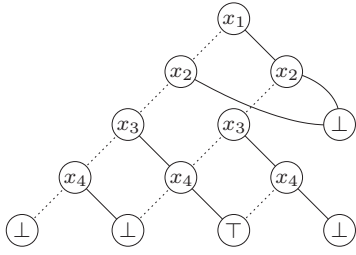
Figure 4: An OBDD for the boolean function $E_4^2 \wedge \neg x_2$ respecting the variable ordering $x_1 < x_2 < x_3 < x_4$.

Recall that if f and f' are boolean functions on X, and ρ is any ordering of X, then (Wegener 2000, Theorem 3.3.6):

$$\mathcal{OBDD}_\rho(f \wedge f') \leq \mathcal{OBDD}_\rho(f) \cdot \mathcal{OBDD}_\rho(f'). \quad (5)$$

Regarding the literals $\neg x_i$ and x_i as boolean functions on $\{x_1, \ldots, x_n\}$ whose $\mathcal{OBDD}_\sigma$ size is constant (6 wires), the statement follows from (5) and Proposition 1. $\quad\square$

SDDs vs OBDDs. We now prove that the hidden weighted bit function has small (uncompressed) SDD size; a slight modification of the construction, described later, gives the compressed case.

The key observation is that, by the definition of $\mathcal{P}_n$, the hidden weighted bit function HWB_n is equivalent to

$$(P_0 \wedge \bot) \vee (P_n \wedge \top) \vee \bigvee_{i=1}^{n-1} ((P_{i,0} \wedge \bot) \vee (P_{i,1} \wedge \top)) \quad (6)$$

because the latter is equivalent to

$$(E_n^1 \wedge x_1) \vee \cdots \vee (E_n^n \wedge x_n)$$

which is in turn equivalent to HWB_n. Using the form (6), it is easy to build an SDD computing HWB_n and respecting a vtree for $\{x_1, \ldots, x_n, y\}$ like the one on the right in Figure 1; upon implementing the P_i's and $P_{i,j}$'s by OBDDs, the construction has polynomial size by Proposition 2. Note that the SDD is not compressed because $\bot$ and $\top$ are reused n times. The details follow.

Theorem 3. *The $\mathcal{SDD}$ size of* HWB_n *is* $O(n^3)$.

Proof. We first define an NNF C on input variables $X = \{x_1, \ldots, x_n\}$ computing (6) as follows. The output gate of C is a fanin $2n$ $\vee$-gate, with wires from $2n$ fanin 2 $\wedge$-gates g_0, g_n, and $g_{i,j}$ for $i = 1, \ldots, n-1$ and $j = 0, 1$.

Let p_0 and s_0 be the two gates wiring g_0, let p_n and s_n be the two gates wiring g_n, and for $i = 1, \ldots, n-1$ and $j = 0, 1$ let $p_{i,j}$ and $s_{i,j}$ be the two gates wiring $g_{i,j}$.

Let σ be any ordering of $x_1, \ldots, x_n$. All the subcircuits of C rooted at $p_0, s_0, p_n, s_n, p_{i,j}$, and $s_{i,j}$ ($i = 1, \ldots, n-1$, $j = 0, 1$) are OBDDs respecting the ordering σ. Moreover:

- C_{p_i} computes P_i for $i \in \{1, n\}$;
- $C_{p_{i,j}}$ computes $P_{i,j}$ for $i = 1, \ldots, n-1, j = 0, 1$;
- C_{s_0} and $C_{s_{i,0}}$ compute $\bot$ for $i = 1, \ldots, n-1$;

- C_{s_n} and $C_{s_{i,1}}$ compute $\top$ for $i = 1, \ldots, n-1$.

We prove that C is an SDD respecting a suitable vtree T for the variable set $X \cup \{y\}$. Roughly, T is a right-linear vtree with the exception of the variable y; see the diagram on the right in Figure 1 for the case $n = 4$ and $\sigma = x_1 < x_2 < x_3 < x_4$. Formally, T is defined as follows. Let v be the root of T. The left subtree $T_l = T_{v_l}$ of T is a right-linear vtree for $\{x_1, \ldots, x_n\}$ such that the variable ordering induced by its left first traversal is σ. Similarly, the right subtree $T_r = T_{v_r}$ of T is a vtree for $\{y\}$.

We check that C is an SDD respecting T.

- The subcircuits C_{p_0}, C_{p_n}, and $C_{p_{i,j}}$ are OBDDs respecting σ, and hence SDDs respecting T_l ($i = 1, \ldots, n-1, j = 0, 1$). This settles (S1).
- The subcircuits C_{s_0}, C_{s_n}, and $C_{s_{i,j}}$ are input gates labelled by a constant, and hence SDDs respecting T_r ($i = 1, \ldots, n-1, j = 0, 1$). This settles (S2).

Note how the construction crucially exploits the special position of y in the vtree T, while the circuit C does not even read y.

The partitioning properties (S3)-(S5) follow by construction and Fact 1. Therefore, C is an SDD respecting T. It remains to check that C has size cubic in n.

By construction, C contains the $2n$ subcircuits C_{p_0}, C_{p_n}, and $C_{p_{i,j}}$ for $i = 1, \ldots, n-1$ and $j = 0, 1$; each has size $O(n^2)$ by Proposition 2 hence, altogether, they contribute $O(n^3)$ wires in C. There remain $O(n)$ wires entering the output gate and the gates $g_0, g_1, \ldots, g_m$. $\quad\square$

Combining Theorem 2 and Theorem 3, we conclude that OBDDs and SDDs are exponentially separated by the hidden weighted bit function.

Compressed SDDs vs OBDDs. A slight variant of the previous construction gives an exponential separation of OBDDs and compressed SDDs.

Let $y_0, y_1, \ldots, y_n$ be fresh variables. The boolean function F_n of the variables $x_1, \ldots, x_n, y_0, y_1, \ldots, y_n$, called *generalized hidden weighted bit* function, is defined by

$$(P_0 \wedge \neg y_0) \vee (P_n \wedge y_n) \vee \bigvee_{i=1}^{n-1} ((P_{i,0} \wedge \neg y_i) \vee (P_{i,1} \wedge y_i)). \quad (7)$$

Notice that the form (7) is exactly as the form (6), except that the n copies of $\bot$ and the n copies of $\top$ are replaced by the $2n$ pairwise nonequivalent formulas $\neg y_0, y_n, y_i$, and $\neg y_i$ ($i = 1, \ldots, n-1$), so that (7) has indeed a compressed SDD implementation. The details follow.

Lemma 1. *The $\mathcal{SDD}_c$ size of* F_n *is* $O(n^3)$.

Proof. We construct an NNF C on input variables $X = \{x_1, \ldots, x_n, y_0, y_1, \ldots, y_n\}$ computing (7) along the lines of Theorem 3. The only modification is that C_{s_0} is an input gate labelled $\neg y_0$, C_{s_n} is an input gate labelled y_n, $C_{s_{i,0}}$ is an input gate labelled $\neg y_i$, and $C_{s_{i,1}}$ is an input gate labelled y_i ($i = 1, \ldots, n-1$).

Figure 5: The vtree for F_4 in the proof of Lemma 1.

We claim that C is a compressed SDD respecting a vtree T for the variable set X built exactly as in Theorem 3 except that the right subtree $T_r = T_{v_r}$ of T is a right-linear vtree for $\{y_0, y_1, \ldots, y_n\}$ such that the variable ordering induced by its left first traversal is ρ. See Figure 5 for the case $n = 4$, $\sigma = x_1 < \cdots < x_4$, and $\rho = y_0 < y_1 < \cdots < y_4$.

To check that C is a compressed SDD respecting T, notice that the subcircuits C_{p_0} and $C_{p_{i,j}}$ are OBDDs respecting σ, and hence compressed SDDs respecting T_l ($i = 1, \ldots, n$, $j = 0, 1$), and the subcircuits C_{s_0} and $C_{s_{i,j}}$ are OBDDs respecting ρ, and hence compressed SDDs respecting T_r ($i = 1, \ldots, n$, $j = 0, 1$). Moreover, it is easily verified that the output gate of C is compressed as by condition (C). Hence C is compressed. The rest of the proof is identical to that of Theorem 3. □

We now prove that the generalized hidden weighted bit function F_n needs large OBDDs.

Lemma 2. *The $\mathcal{OBDD}$ size of F_n is $2^{\Omega(n)}$.*

Proof. Let N be the size of a smallest $\mathcal{OBDD}$ on variables $X = \{x_1, \ldots, x_n, y_0, y_1, \ldots, y_n\}$ computing F_n, and let ρ be any ordering of X such that $\mathcal{OBDD}_\rho(F_n) = N$.

Let $G_n(x_1, \ldots, x_n)$ be the subfunction of F_n where $y_0, y_1, \ldots, y_n$ are replaced by 1, in symbols:

$$G_n \equiv F_n(x_1, \ldots, x_n, 1, 1, \ldots, 1). \quad (8)$$

Since conditioning (unboundedly many variables of) an OBDD does not increase its size (Wegener 2000, Theorem 2.4.1), we have that

$$\mathcal{OBDD}_\rho(G_n) \leq \mathcal{OBDD}_\rho(F_n). \quad (9)$$

We now claim that G_n is the hidden weighted bit function on n variables. Indeed, by construction,

$$G_n \equiv F_n(x_1, \ldots, x_n, 1, 1, \ldots, 1)$$
$$\equiv P_n \vee \bigvee_{i=1}^{n-1} P_{i,1}$$

which we already observed being equivalent to HWB_n. Therefore $\mathcal{OBDD}(G_n) = 2^{\Omega(n)}$ by Theorem 2, and in particular $\mathcal{OBDD}_\rho(G_n) \geq 2^{\Omega(n)}$. By (9), we are done. □

An exponential separation of OBDDs and compressed SDDs follows.

Theorem 4. *There exists an unbounded arity class of boolean functions $\mathcal{F}$ such that every arity n function $f \in \mathcal{F}$ has $\mathcal{SDD}_c$ size in $O(n^3)$ and $\mathcal{OBDD}$ size in $2^{\Omega(n)}$.*

Proof. Take $\mathcal{F} = \{F_m : m \in \mathbb{N}\}$, where F_m is as in (7). Then F_m has compressed $\mathcal{SDD}$ size $O(m^3)$ by Lemma 1 and $\mathcal{OBDD}$ size $2^{\Omega(m)}$ by Lemma 2. Since F_m has $n = 2m + 1$ variables, it follows that F_m has $\mathcal{SDD}_c$ size in $O(n^3)$ and $\mathcal{OBDD}$ size in $2^{\Omega(n)}$. □

Notably, the function class giving the exponential separation is as hard on compressed SDDs as the function class giving the quasipolynomial separation (cubic in both cases, see Theorem 1).

4 Discussion

We have shown that OBDDs and SDDs are exponentially separated by the hidden weighted bit function, while OBDDs and compressed SDDs are exponentially separated by the generalized hidden weighted bit function, F_n in (7), that contains the hidden weighted bit function as a subfunction:

$$F_n(x_1, \ldots, x_n, 1, 1, \ldots, 1) = \mathrm{HWB}_n(x_1, \ldots, x_n). \quad (10)$$

Separating OBDDs and SDDs by the hidden weighted bit function, instead of by a function designed adhoc, further corroborates the theoretical quality of SDDs. As articulated by Bollig et al. (1999), any useful extension of OBDDs is expected to implement the hidden weighted bit function efficiently.

The SDD C described in the proof of Theorem 3 is not compressed, because $\bot$ and $\top$ are reused n times. In view of the canonical construction of an SDD over a vtree (Darwiche 2011, Theorem 3), it is readily observed that compressing C with respect to the vtree T in the proof of Theorem 3 implies finding a small SDD for HWB_n with respect to the left subtree of T, that is, a small OBDD for HWB_n; but this is impossible by Theorem 2. The fact that compressing an SDD over its vtree may increase the size exponentially has been observed already (Van den Broek and Darwiche 2015, Theorem 1). We reiterate the observation here only because our argument is significantly shorter.

We conclude mentioning a nonobvious, and perhaps even unexpected, aspect of our separation result. An inspection of our construction shows that SDDs are already exponentially more succinct than *general OBDDs* even allowing only *one sentential decision* (and possibly many Shannon decisions); recall (6) and (7). The construction by Xue et al. (2012) already uses *nested sentential decisions* even to separate *OBDDs over a fixed variable ordering* from SDDs!

Questions. We do not know whether the hidden weighted bit function has superpolynomial compressed SDD size for all vtrees; a positive answer would separate compressed and uncompressed SDDs in succinctness and, in view of Lemma 1 and (10), would prove that compressed SDDs do not support conditioning (of unboundedly many variables) in polynomial size.

In view of Theorem 1, it is natural to ask which SDDs are quasipolynomially simulated by OBDDs. Our separating

family shows that SDDs with unbounded fanin disjunctions cannot be quasipolynomially simulated by OBDDs. On the other hand, recent work by Darwiche and Oztok essentially shows that SDDs over binary disjunctions (fanin 2) admit a quasipolynomial simulation by OBDDs (Oztok and Darwiche 2015, Theorem 1). In this light, it is tempting to conjecture that the above criterion is exact, that is, every SDD class over bounded fanin disjunctions does indeed admit a quasipolynomial simulation by OBDDs.

Finally, a natural question arising in the context of the present work is about the relative succinctness of SDDs and structured deterministic NNFs (see (3)); to the best of our knowledge, the question is open. By Theorem 3, at least we now know that the hidden weighted bit function is not a candidate to separate the two classes.

Acknowledgments

References

Bollig, B.; Löbbing, M.; Sauerhoff, M.; and Wegener, I. 1999. On the Complexity of the Hidden Weighted Bit Function for Various BDD Models. *Theoretical Informatics and Applications* 33(2):103–116.

van den Broek, G., and Darwiche, A. 2015. On the Role of Canonicity in Knowledge Compilation. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 1641–1648. AAAI Press.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Choi, A., and Darwiche, A. 2013. Dynamic Minimization of Sentential Decision Diagrams. In desJardins, M., and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.* AAAI Press.

Darwiche, A., and Marquis, P. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17:229–264.

Darwiche, A. 2001. Decomposable Negation Normal Form. *Journal of the ACM* 48(4):608–647.

Darwiche, A. 2011. SDD: A New Canonical Representation of Propositional Knowledge Bases. In Walsh, T., ed., *IJCAI 2011, Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 819–826. IJCAI/AAAI.

Oztok, U., and Darwiche, A. 2014. CV-Width: A New Complexity Parameter for CNFs. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *ECAI 2014, Proceedings of the Twenty-First European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic, August 18-22, 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 675–680. IOS Press.

Oztok, U., and Darwiche, A. 2015. A Top-Down Compiler for Sentential Decision Diagrams. In Yang, Q., and Wooldridge, M., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3141–3148. AAAI Press.

Pipatsrisawat, K., and Darwiche, A. 2008. New Compilation Languages Based on Structured Decomposability. In Fox, D., and Gomes, C. P., eds., *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 517–522. AAAI Press.

Pipatsrisawat, T., and Darwiche, A. 2010. A Lower Bound on the Size of Decomposable Negation Normal Form. In Fox, M., and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Razgon, I., and Petke, J. 2013. Cliquewidth and Knowledge Compilation. In Järvisalo, M., and Gelder, A. V., eds., *SAT 2015, Proceedings of the Sixteenth International Conference on Theory and Applications of Satisfiability Testing, Helsinki, Finland, July 8-12, 2013*, volume 7962 of *Lecture Notes in Computer Science*, 335–350. Springer.

Razgon, I. 2014a. On OBDDs for CNFs of Bounded Treewidth. *CoRR* abs/1308.3829v3.

Razgon, I. 2014b. On OBDDs for CNFs of bounded treewidth. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *KR 2014, Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Wegener, I. 2000. *Branching Programs and Binary Decision Diagrams*. SIAM.

Xue, Y.; Choi, A.; and Darwiche, A. 2012. Basing Decisions on Sentences in Decision Diagrams. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press.