# A Lower Bound on the Size of Decomposable Negation Normal Form

**Knot Pipatsrisawat** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095 USA
{thammakn,darwiche}@cs.ucla.edu

## Abstract

We consider in this paper the size of a Decomposable Negation Normal Form (DNNF) that respects a given *vtree* (known as *structured* DNNF). This representation of propositional knowledge bases was introduced recently and shown to include OBDD as a special case (an OBDD variable ordering is a special type of vtree). We provide a lower bound on the size of any structured DNNF and discuss three particular instances of this bound, which correspond to three distinct subsets of structured DNNF (including OBDD). We show that our lower bound subsumes the influential Sieling and Wegener's lower bound for OBDDs, which is typically used for identifying variable orderings that will cause a blowup in the OBDD size. We show that our lower bound allows for similar usage but with respect to vtrees, which provide structure for DNNFs in the same way that variable orderings provide structure for OBDDs. We finally discuss some of the theoretical and practical implications of our lower bound.

## Introduction

An Ordered Binary Decision Diagram (OBDD) is a canonical representation of a propositional knowledge base (also known as a Boolean function). Because of its compactness and efficient support for many Boolean operations, OBDD has become an indispensable tool in many research areas such as diagnosis, verification, system design, and planning. A fundamental concept that underlies the OBDD representation is the notion of *decomposability*.

The most general language satisfying decomposability is known as Decomposable Negation Normal Form (DNNF) (Darwiche 2001a). Imposing other properties on top of decomposability yields various languages such as deterministic DNNF (Darwiche 2001b), structured DNNF (SDNNF) (Pipatsrisawat and Darwiche 2008), AOMDD (Boolean case) (Mateescu and Dechter 2006), and OBDD (Bryant 1986). These languages differ on their supports for various operations and on their abilities to compactly represent Boolean functions (Darwiche and Marquis 2002; Pipatsrisawat and Darwiche 2008).

Among the subsets of DNNF, OBDD is by far the most studied language. Much effort has been dedicated to studying the size of OBDD for various Boolean functions. As a result, lower and upper bounds have been presented on the size of OBDD for various Boolean functions (e.g., see (Breitbart, Hunt, and Rosenkrantz 1995; Ponzio 1995; Wegener 2000)). A key result that underlies much work in this direction is the one by Sieling and Wegener (Sieling and Wegener 1993), who showed that the size of an OBDD representation of any Boolean function is closely related to the number of sub-functions obtained by conditioning the Boolean function on partial variable instantiations.

In this work, we consider the problem of providing a lower bound on the size of structured DNNF, which include OBDD as a special case. In particular, we introduce the general notion of a *decomposition,* which allows one to write a Boolean function in terms of *independent functions* (i.e., ones that do not share variables). We introduce several types of decompositions and show that each type characterizes one of the key subsets of structured DNNF. For example, we show that the Shannon decomposition, which underlies OBDD, is an instance of our notion of decompositions. We then provide a general lower bound on the size of structured DNNF and show that it subsumes the Sieling and Wegener's bound when applied to OBDD. Theoretically, our result unveils the fundamental notion of a decomposition, which generalizes the influential Shannon decomposition, and shows how it can be used to characterize different subsets of structured DNNF. Practically, it allows us to show that some vtrees are guaranteed to lead to exponential structured DNNF representations. This result allows us to leverage knowledge about the decomposition of Boolean functions in providing guarantees on the size of their structured DNNF representations in the same way knowledge about sub-functions has been used in the context of OBDD.

The rest of the paper is structured as follows. We first present basic definitions used in the paper. We then state the OBDD lower bound by Sieling and Wegener, followed by an overview of structured DNNF. The fundamental notion of a decomposition is introduced next, followed by the lower bound result. We finally discuss the relationship between our lower bound and the OBDD lower bound, and close with some concluding remarks.

## Basic Definitions

We use the standard notation for variables and their instantiations. In particular, we use an upper case letter to denote

a variable (e.g., $X$) and a lower case letter (e.g., $x$) to denote its instantiation (assignment). Moreover, we use a bold upper case letter to denote a set of variables (e.g., $\mathbf{X}$) and a bold lower case letter (e.g., $\mathbf{x}$) to denote their instantiations.

A *Boolean function* (or simply function) over a set of variables $\mathbf{Z}$ is a function that maps each complete assignment of variables $\mathbf{Z}$ to either *true* or *false*. The *conditioning* of function $f$ on variable assignment $\mathbf{x}$ is defined as $f|\mathbf{x} = \exists \mathbf{X}(f \wedge \mathbf{x})$. If $f$ is represented by a propositional formula, we can obtain $f|\mathbf{x}$ by replacing each occurrence of variable $X \in \mathbf{X}$ by its value in $\mathbf{x}$.

A function $f$ *depends only* on variables $\mathbf{Z}$ iff for any variable $X \notin \mathbf{Z}$, we have $f|X = f|\neg X$. We will write $f(\mathbf{Z})$ to indicate that $f$ depends only on variables $\mathbf{Z}$. Note that $f(\mathbf{Z})$ does not necessarily depend on every variable in $\mathbf{Z}$.

A conjunction is *decomposable* if each pair of its conjuncts share no variables. A disjunction is *deterministic* if each pair of its disjuncts is mutually exclusive. A *negation normal form* (NNF) is a DAG whose internal nodes are labelled with disjunctions and conjunctions, and whose leaf nodes are labeled with literals or the constants *true* and *false*; see Figure 4.[1] An NNF is decomposable (called a DNNF) iff each of its conjunctions is decomposable; see Figure 4(a). A DNNF is deterministic (called a d-DNNF) iff each of its disjunctions is deterministic; see Figure 4(b). We use $vars(N)$ to denote the set of variables in the sub-NNF rooted at $N$.

## OBDDs and the Sieling and Wegener's Bound

An OBDD is a DAG whose non-leaf nodes are labeled with variables, and whose leaf nodes are labeled with Boolean constants; see Figure 1(a). Every non-leaf node in an OBDD has exactly one high child (pointed to by a solid edge) and one low child (pointed to by a dotted edge). An OBDD respects a total variable ordering if the order of variables on any path from the root to a leaf is consistent with the given order. An OBDD is interpreted as a representation of a Boolean function over the variables labeling its nodes.

One key concept that underlies OBDD is the *Shannon decomposition*, which states that every function $f$ can always be written as $f = (X \wedge f|X) \vee (\neg X \wedge f|\neg X)$, where $X$ is any variable. Since $f|X$ and $f|\neg X$ no longer depend on $X$, the conjunctions in the Shannon decomposition are clearly decomposable. Any OBDD can be viewed as the result of applying this decomposition recursively to some Boolean function according to a given variable ordering.

An OBDD can also be interpreted as an NNF as shown in Figure 1(b). Every OBDD node labeled with $X$ is expanded into $(X \wedge h) \vee (\neg X \wedge l)$, where $h$ and $l$ are the results of expanding the high and low children of the node. The resulting NNF is decomposable and deterministic by construction (see (Darwiche and Marquis 2002) for more details).

An OBDD is an attractive representation of Boolean functions because of its ability to represent some functions compactly and its polytime support for many logical operations (Bryant 1986). Since most applications of OBDDs rely on their compactness, much work exists on bounding the size of OBDDs for various Boolean functions. The following (paraphrased) result by Sieling and Wegener provides a basis for numerous work in this direction. Central to this result is the notion of a *sub-function* $f|\mathbf{x}$, which is a function obtained by conditioning $f$ on a variable instantiation $\mathbf{x}$.

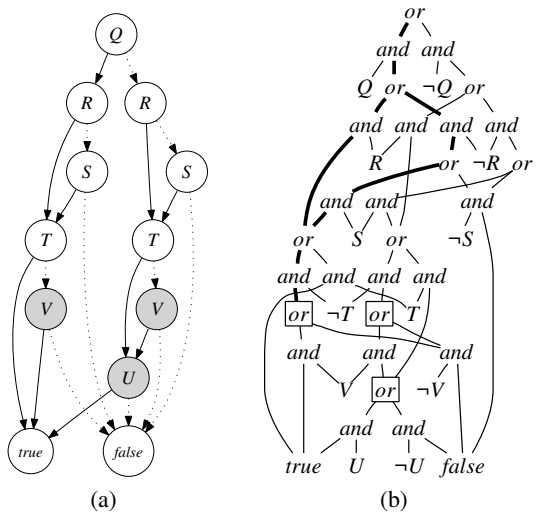


Figure 1: (a) an OBDD in the conventional representation (b) the same OBDD in the NNF representation.

**Theorem 1 ((Sieling and Wegener 1993))** *Let $f$ be a function over $X_1, \ldots, X_n$ and $m$ be the number of distinct sub-functions of $f$ obtained by conditioning on $X_1, \ldots, X_{i-1}$ that depend on $X_i$. A reduced[2] OBDD for $f$ using variable ordering $X_1, \ldots, X_n$ contains exactly $m$ nodes labeled with $X_i$.*

The main use of the Sieling and Wegener's result is in showing which variable orderings lead to efficient OBDD representations (upper bound), and which ones lead to exponential OBDDs (lower bound). Consider the Boolean function $f = (X_1 \wedge Y_1) \vee \ldots \vee (X_n \wedge Y_n)$, for example. If we use variable ordering $X_1, Y_1, \ldots, X_n, Y_n$, we find that the number of distinct sub-functions that depend on $X_i$ or $Y_i$ (by conditioning $f$ on all preceding variables) is no more than 2 for any $i$. Hence, this variable order will lead to an efficient OBDD representation. However, if we use the variable ordering $X_1, \ldots, X_n, Y_1, \ldots, Y_n$, we find that we have $\Omega(2^n)$ distinct sub-functions that depend on variable $Y_1$, when we condition $f$ on $X_1, \ldots, X_n$. Hence, this variable ordering will lead to an exponentially-sized OBDD.

## Structured DNNF

Our goal is to derive an analogue of the lower bound by Sieling and Wegener, but for a more general class of representations that subsume OBDDs. In particular, our focus is on *structured DNNFs.* A structured DNNF is a DNNF that respects a given *vtree* (Pipatsrisawat and Darwiche 2008).

[1] The circles, boxes, and dashed/bold edges in this figure have no special meaning; they are visual aids that will be used later.

[2] An OBDD is reduced iff no two distinct nodes in the OBDD represent the same Boolean function.

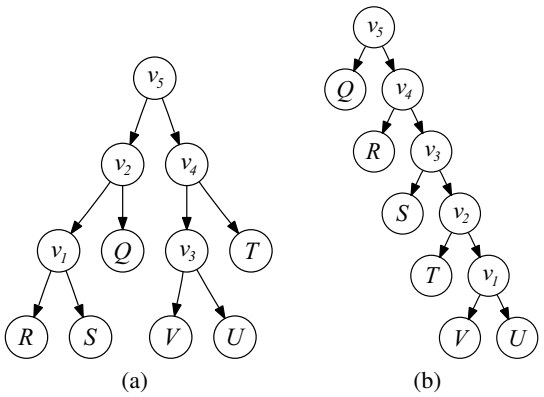


Figure 2: A vtree in (a) and a linear vtree in (b). Their internal nodes are labeled to support the discussion.

**Definition 1 (Vtree)** *A vtree for a set of variables* $\mathbf{Z}$ *is a full, rooted binary tree whose leaves are in one-to-one correspondence with the variables in* $\mathbf{Z}$.

Figure 2 depicts two example vtrees of $\{Q, R, S, T, U, V\}$. Given an internal node $v$ in a vtree, we use $v^l$ and $v^r$ to refer to its left and right children, and use $vars(v)$ to denote the set of variables at or below $v$ in the tree. We can now define what it means for a DNNF to respect a vtree.

**Definition 2** *A DNNF respects a vtree iff each and-node has exactly two children* $N^l$ *and* $N^r$*, and* $vars(N^l) \subseteq vars(v^l)$ *and* $vars(N^r) \subseteq vars(v^r)$ *for some vtree node* $v$.

The DNNF in Figure 4(a) and the d-DNNF in Figure 4(b) respect the vtree in Figure 2(a). The language of structured DNNF simply contains all DNNFs that respects some vtree.[3]

Note that a variable ordering corresponds to a *linear vtree* as shown in Figure 2(b). Moreover, every OBDD is a DNNF that respects the corresponding linear vtree. Figure 1(b) shows a DNNF representation of the OBDD in Figure 1(a), which respects the vtree in Figure 2(b). The result we shall present later will allow us to show that any DNNF that respects a given vtree must have at least a certain number of nodes (lower bound). When applied to an OBDD, our bound will be shown to subsume the Sieling and Wegener's bound. We will later discuss the applicability of our result to some key subsets of structured DNNF.

## Decompositions of Boolean Functions

We now introduce the fundamental notion behind our lower bound on the size of structured DNNF. In the rest of the paper, we will assume that variable sets $\mathbf{X}$ and $\mathbf{Y}$ form a partition of variable set $\mathbf{Z}$.

**Definition 3** *An* $\mathbf{X}$*-decomposition of function* $f(\mathbf{Z})$ *is a collection of functions (a.k.a. elements)* $f^1(\mathbf{Z}), \ldots, f^m(\mathbf{Z})$ *such that* $f(\mathbf{Z}) = \bigvee_{i=1}^{m} f^i(\mathbf{Z})$ *and each* $f^i(\mathbf{Z})$ *can be expressed as follows:*

$$f^i(\mathbf{Z}) = g^i(\mathbf{X}) \wedge h^i(\mathbf{Y}).$$

*The number* $m$ *is called the size of the decomposition in this case. A decomposition is minimal if no other decomposition has a smaller size. A decomposition is deterministic if* $f^i \wedge f^j$ *is inconsistent for all* $i \neq j$.

Note that an $\mathbf{X}$-decomposition for $f(\mathbf{Z})$ is also a $\mathbf{Y}$-decomposition for $f(\mathbf{Z})$. We will typically just say "decomposition" when $\mathbf{X}$ and $\mathbf{Y}$ are clear from the context.

Consider the Boolean function $f = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee (X_2 \wedge Y_3)$ and the partition $\mathbf{X} = \{X_1, X_2\}, \mathbf{Y} = \{Y_1, Y_2, Y_3\}$. The following are two decompositions of this function:

| $g^i(\mathbf{X})$ | $h^i(\mathbf{Y})$ |
|---|---|
| $X_1$ | $Y_1$ |
| $X_2$ | $Y_2 \vee Y_3$ |

| $g^i(\mathbf{X})$ | $h^i(\mathbf{Y})$ |
|---|---|
| $X_1$ | $Y_1$ |
| $\neg X_1 \wedge X_2$ | $Y_2 \vee Y_3$ |
| $X_1 \wedge X_2$ | $\neg Y_1 \wedge (Y_2 \vee Y_3)$ |

Each row corresponds to an element of the decomposition. Moreover, we present each element in terms of its $\mathbf{X}$ and $\mathbf{Y}$ components, where the element is simply the conjunction of these components. Note that the left decomposition is non-deterministic, while the right decomposition is deterministic. One can always find a decomposition for any function $f(\mathbf{Z})$ if one is not concerned about the size of the decomposition. In particular, the models of $f(\mathbf{Z})$ can be the basis for a trivial decomposition for any partition of $\mathbf{Z}$.

The notion of a decomposition generalizes the Shannon decomposition as used in the OBDD literature. According to the Shannon decomposition, each function $f(\mathbf{Z})$ can be expressed as $f = (X \wedge f|X) \vee (\neg X \wedge f|\neg X)$. If we let $\mathbf{X} = \{X\}, \mathbf{Y} = \mathbf{Z} \setminus \mathbf{X}$, then this can be thought of as the following decomposition:

| $g^i(\mathbf{X})$ | $h^i(\mathbf{Y})$ |
|---|---|
| $X$ | $f\|X$ |
| $\neg X$ | $f\|\neg X$ |

The Shannon decomposition is always of size two, deterministic, and is completely determined by the choice of variable $X$.

## A Lower Bound

We are now ready to state our main result, which is a lower bound on the size of structured DNNFs. This result is a corollary of a more specific result that we discuss later.

**Theorem 2 (Lower bound)** *Suppose we are given a (resp. deterministic) DNNF for function* $f(\mathbf{Z})$ *and the DNNF respects a given vtree, which has a node with variables* $\mathbf{X}$*. The DNNF must have at least* $k$ *nodes, where* $k$ *is the size of a minimal (resp. deterministic)* $\mathbf{X}$*-decomposition of* $f(\mathbf{Z})$.

One application of the above result is in demonstrating that, for certain Boolean functions, any DNNF representation that respects certain vtrees will be exponentially-sized. This approach could form a basis of both theoretical and practical work. Theoretically, our lower bound can be used to show the separation in terms of succinctness (Darwiche and Marquis 2002) between different languages.[4] Practically, the lower bound can be used to rule out bad vtrees

[3] Some DNNFs, such as $(((a \wedge b) \wedge (\neg c \wedge d)) \vee ((\neg a \wedge c) \wedge (b \wedge \neg d)))$, do not respect any vtree.

[4] We will provide a brief example of this usage later.

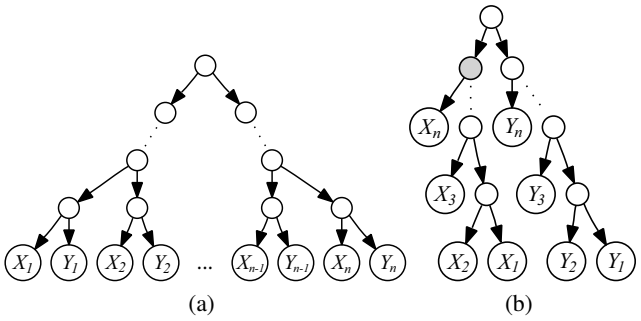


Figure 3: Two vtrees for $\{X_1, \ldots, X_n, Y_1, \ldots, Y_n\}$.

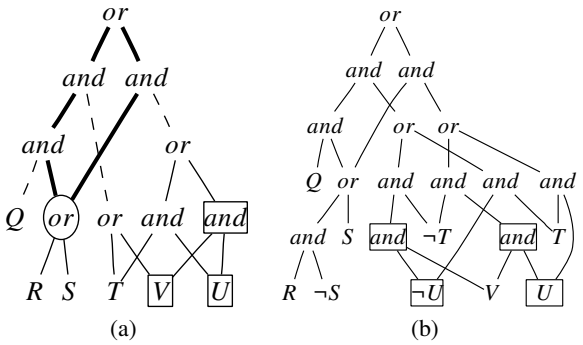


Figure 4: (a) A DNNF depicting a node and its paths (bold lines). The edges from the paths to their branches are dashed. (b) a d-DNNF of the same Boolean function.

when we construct structured DNNFs. In what follows, we demonstrate the potentials of our result by providing examples of how it can be used to prove (exponential) lower bounds for some functions (with respect to some vtrees). We omit the proofs of these claims due to space limitations.

Consider the function $F_1 = (X_1 \oplus Y_1) \wedge \ldots \wedge (X_n \oplus Y_n)$, where $\oplus$ is the exclusive-or operator. This function has a compact DNNF representation that respects the vtree in Figure 3(a).[5] However, any DNNF of $F_1$ that respects the vtree in Figure 3(b) must have an exponential size. This is because this vtree has a node (shaded) with variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. The function $F_1$ has $2^n$ models and one can show that no two of these models can reside in the same element of any $\mathbf{X}$-decomposition (proof in the full paper). Thus, any optimal $\mathbf{X}$-decomposition of $F_1$ has size $\Omega(2^n)$. Hence, by Theorem 2, any DNNF that respects this vtree (or any vtree with a node containing just the variables $\mathbf{X}$) must contain an exponential number of nodes.

Note the simplicity of the proof as our lower bound removes the burden of analyzing the structure of the considered DNNF. In this approach, one only needs to show that some variable partition induced by the vtree will always induce an exponentially-sized decomposition. This is analogous to how the Sieling and Wegener's result is typically used. The following list gives several examples of functions whose exponential lower bound (with respect to certain vtrees) can be easily established using the same approach.

1. $F_2 = (X_1 \vee Y_1) \wedge \ldots \wedge (X_n \vee Y_n)$.
2. $F_3 = T_k(\mathbf{X}_0, \mathbf{Y}_0) \wedge \ldots \wedge T_k(\mathbf{X}_{(n-1)}, \mathbf{Y}_{(n-1)})$, where $\mathbf{X}_i = X_{ik+1}, \ldots, X_{ik+k}$ ($\mathbf{Y}_i$ is defined similarly) and $T_k$ evaluates to *true* iff at least $k$ (a constant) of its inputs are *true* (threshold function).
3. $F_4 = diff(\mathbf{X}, \mathbf{Y}, \mathbf{N})$: the difference function, which evaluates to *true* iff the difference between the values of $\mathbf{X}$ and $\mathbf{Y}$ is $\mathbf{N}$ (all interpreted as binary numbers).
4. $F_5 = CBS(\mathbf{X}, \mathbf{Y}, \mathbf{N})$: the circular bit shift function (Fortune, Hopcroft, and Schmidt 1978). This function evaluates to *true* iff $\mathbf{X}$ is equal to $\mathbf{Y}$ after being circularly shifted by $\mathbf{N}$ (binary number) positions.

[5] Simply create a tree of and-nodes with the same structure as the vtree and place a DNNF for each $(X_i \oplus Y_i)$ at each leaf.

In all these cases, we can easily use our lower bound result to show that any DNNF that respects a vtree having a node $v$ with $vars(v) = \mathbf{X}$ or $vars(v) = \mathbf{Y}$ cannot have a polynomial number of nodes. Note that one can construct a vtree that yields polysize DNNFs for $F_2 - F_4$. The key idea is to position $X_i$ close to $Y_i$ in the tree. However, for $F_5$, we can actually use our lower bound as a basis of a proof to show that $F_5$ does not admit a polysize structured DNNF representation *for any vtree*. Since $F_5$ has a polysize representation as a free binary decision diagram (FBDD) (see (Fortune, Hopcroft, and Schmidt 1978)), this proof provides an exponential separation between structured DNNF and DNNF.[6] This claim was previously made in (Pipatsrisawat and Darwiche 2008) based on a different proof. The lower bound presented here allows for a more direct and shorter proof (to be presented in the full paper).

If the considered DNNF is deterministic, Theorem 2 tells us that its size must be lower bounded by the size of a minimal deterministic decomposition. The insistence on determinism may lead to stronger lower bounds. Consider for example the function $f = (X_1 \wedge Y_1) \vee \ldots \vee (X_n \wedge Y_n)$. In this form, this formula, which is polysized, is already a (nondeterministic) DNNF that respects the vtree in Figure 3(b). However, for $\mathbf{X} = \{X_1, \ldots, X_n\}$, we conjecture that any deterministic $\mathbf{X}$-decomposition of this function must have an exponential size. The truth of this conjecture would imply that any deterministic DNNF of $f$ that respects the vtree in Figure 3(b) must have an exponential size, even though we have a polysized DNNF that respects this vtree.

We will later show that OBDDs are governed by an even stronger type of deterministic decompositions, leading to an even stronger lower bound. This, however, requires a more specific result, which we present next.

## Decompositions Induced by Structured DNNF

We will now present a result that explicates the particular decompositions induced by structured DNNF. This result re-

[6] According to the notation in (Darwiche and Marquis 2002), this shows that DNNF is *strictly more succinct* than structured DNNF.

veals the main idea that underlies Theorem 2. To state this result, we need a few definitions.

**Definition 4** *In a DNNF, a path $p$ for node $N$ is a path from the DNNF root to node $N$. If an and-node on the path (excluding $N$) has a child $L$ that is not on the path, then node $L$ is called a branch of the path. The path function of $p$ is the conjunction of functions corresponding to its branches.*

Consider the DNNF in Figure 4(a). The node $(R \vee S)$ (circled) has two paths from the root (bold edges). The left path has two branches that correspond to $Q$ and $(T \vee V)$. Hence, its path function is $Q \wedge (T \vee V)$. The right path has only one branch with function $((T \wedge U) \vee (V \wedge U))$.

For OBDDs, path functions are always terms (conjunctions of literals). Consider the OBDD in Figure 1(b) for an example. The top-left boxed node has two paths (bold edges). The corresponding path functions are $Q \wedge R \wedge \neg T$ and $Q \wedge \neg R \wedge S \wedge \neg T$.

**Definition 5** *Consider a DNNF that respects a vtree having node $v$. A node $N$ in the DNNF is said to be a $v$-node if it is not true or false, its variables are all in $vars(v)$, and has at least one $v$-path. A $v$-path for node $N$ is a path from the root to node $N$ with no other node whose variables are also all in $vars(v)$. The $v$-paths function of a $v$-node is the disjunction of the path functions of all its $v$-paths.*

Consider the node $v_3$ in Figure 2(a). All $v$-nodes (with $v = v_3$) of the structured DNNFs in Figures 4(a) and 4(b) are shown in boxes. Node $U$ in Figure 4(a) is a $v$-node with $v$-paths function $T \wedge (R \vee S)$. Note that the rightmost path is not a $v$-path of $U$ as it contains another node $(V \wedge U)$ with variables in $vars(v)$. We are now ready to present the result.

**Theorem 3** *Consider a (resp. deterministic) DNNF for function $f(\mathbf{Z})$ that respects a given vtree. Let $v$ be a vtree node with $vars(v) = \mathbf{X}$. If the (resp. deterministic) DNNF has $v$-nodes $1, \ldots, m$, then function $f$ must have the following (resp. deterministic) $\mathbf{X}$-decomposition:*

$$g^1(\mathbf{X}) \wedge h^1(\mathbf{Y}), \ldots, g^m(\mathbf{X}) \wedge h^m(\mathbf{Y}), H(\mathbf{Y})$$

*where $g^i(\mathbf{X})$ is the function of $v$-node $i$, $h^i(\mathbf{Y})$ is the $v$-paths function of $v$-node $i$, and $H(\mathbf{Y})$ is equivalent to the DNNF after having replaced all $v$-nodes by false.*

We will next illustrate this theorem with some examples. First, consider the DNNF in Figure 4(a), which respects the vtree in Figure 2(a). Consider vtree node $v_3$ which has variables $\mathbf{X} = \{U, V\}$. The corresponding $v$-nodes of this DNNF are boxed in Figure 4(a). These nodes induce the following (non-deterministic) decomposition.

| $v$-node function, $g^i(\mathbf{X})$ | $v$-paths function, $h^i(\mathbf{Y})$ |
|---|---|
| $V \wedge U$ | $R \vee S$ |
| $U$ | $T \wedge (R \vee S)$ |
| $V$ | $Q \wedge (R \vee S)$ |
| $H(\mathbf{Y}) = Q \wedge T \wedge (R \vee S)$ | |

Consider now the deterministic DNNF in Figure 4(b) and the same vtree node $v_3$ in Figure 2(a). The corresponding $v$-nodes of this DNNF are boxed in Figure 4(b) and induce the following deterministic decomposition.

| $v$-node function, $g^i(\mathbf{X})$ | $v$-paths function, $h^i(\mathbf{Y})$ |
|---|---|
| $\neg U \wedge V$ | $\neg T \wedge Q \wedge (S \vee (R \wedge \neg S))$ |
| $V \wedge U$ | $\neg T \wedge (S \vee (R \wedge \neg S))$ |
| $U$ | $T \wedge (S \vee (R \wedge \neg S))$ |
| $\neg U$ | $T \wedge Q \wedge (S \vee (R \wedge \neg S))$ |
| $H(\mathbf{Y}) = false$ | |

Finally, consider the OBDD in Figure 1(b). This OBDD respects the linear vtree in Figure 2(b). Consider vtree node $v_1$ which has variables $\mathbf{X} = \{U, V\}$. The corresponding $v$-nodes are boxed in Figure 1(b) and induce the following decomposition.

| $v$-node function, $g^i(\mathbf{X})$ | $v$-paths function, $h^i(\mathbf{Y})$ |
|---|---|
| $(V \wedge U)$ | $\neg T \wedge (R \vee (S \wedge \neg R)) \wedge \neg Q$ |
| $V$ | $\neg T \wedge (R \vee (S \wedge \neg R)) \wedge Q$ |
| $U$ | $T \wedge (R \vee (S \wedge \neg R)) \wedge \neg Q$ |
| $H(\mathbf{Y}) = T \wedge (R \vee (S \wedge \neg R)) \wedge Q$ | |

If we carefully examine this decomposition, we find that it is not just deterministic, but deterministic in a strong way. In particular, not only does every pair of elements contradict each other, but the contradiction can be established by considering only the $v$-paths function of each element. This is an example of the following type of decompositions.

**Definition 6** *A decomposition $g^1(\mathbf{X}) \wedge h^1(\mathbf{Y}), \ldots, g^m(\mathbf{X}) \wedge h^m(\mathbf{Y})$ is strongly deterministic on $\mathbf{X}$ iff $g^i(\mathbf{X}) \wedge g^j(\mathbf{X})$ is inconsistent for all $i \neq j$.*

We will later show that OBDDs induce strongly deterministic decompositions in general. This has implications as we have the following lower bound on the size of strongly deterministic decompositions.

**Proposition 1** *Let $g^1(\mathbf{X}) \wedge h^1(\mathbf{Y}), \ldots, g^m(\mathbf{X}) \wedge h^m(\mathbf{Y})$ be a decomposition of function $f$ that is strongly deterministic on $\mathbf{X}$. If the sub-function $f|\mathbf{x}$ is consistent, then $f|\mathbf{x} = h^i(\mathbf{Y})$ for some $i$ (satisfying $\mathbf{x} \models g^i(\mathbf{X})$). Hence, the size of the decomposition, $m$, cannot be less than the number of distinct (consistent) sub-functions $f|\mathbf{x}$.*

Hence, the elements of a decomposition that is strongly deterministic on $\mathbf{X}$ provide an encoding of all consistent sub-functions $f|\mathbf{x}$. In fact, if the decomposition is minimal, we have a one-to-one correspondence between these elements and consistent sub-functions.

## Relationship to the Sieling and Wegener's Lower Bound

We will now provide another version of Theorem 3, which is now specialized for OBDDs using their standard representations (instead of NNF representations). Stating the result in this form will help us draw connections between our lower bound and the one given in Theorem 1. We begin with some definitions needed to state the result.

**Definition 7** *Let $p$ be a path from an OBDD root to some node. The path function of $p$ is a term that includes literal $X$ iff $X$ and one of its high children appear on the path, and includes literal $\neg X$ iff $X$ and one of its low children appear on the path.*

**Definition 8** *Let $X_1, \ldots, X_n$ be a variable ordering for an OBDD. An OBDD node $N$ is said to be a k-node if it is labeled with a variable in $\mathbf{X} = \{X_k, \ldots, X_n\}$ and has at least one $k$-path. A $k$-path for node $N$ is a path from the OBDD root to node $N$ that has no other node whose label is in $\mathbf{X}$. The k-paths function of a $k$-node is the disjunction of the path functions of all its $k$-paths.*

The set of $k$-nodes of any OBDD must contain all the nodes labeled with $X_k$ and potentially some other nodes. For example, consider the OBDD in Figure 1(a) and let $k = 5$ (the $5^{th}$ and $6^{th}$ variables in the order are $V$ and $U$, respectively). The three $k$-nodes in the OBDD, which are shown in gray, include both nodes labeled with $V$ and one node labeled with $U$.

**Corollary 1** *Consider an OBDD with variable ordering $X_1, \ldots, X_n$ that represents function $f(\mathbf{Z})$. Let $1 \ldots, m$ be all $k$-nodes in the OBDD. The following must be an $\mathbf{X}$-decomposition for function $f$:*

$$g^1(\mathbf{X}) \wedge h^1(\mathbf{Y}), \ldots, g^m(\mathbf{X}) \wedge h^m(\mathbf{Y}), H(\mathbf{Y}).$$

*Here, $\mathbf{X} = \{X_k, \ldots, X_n\}$, $g^i(\mathbf{X})$ is the function of $k$-node $i$, $h^i(\mathbf{Y})$ is the $k$-paths function of $k$-node $i$, and $H(\mathbf{Y})$ is equivalent to the OBDD after replacing all $k$-nodes with false. Moreover, the decomposition is strongly deterministic on $\mathbf{Y}$.*

We will now contrast this result with the Sieling and Wegener's lower bound. Consider again the $k$-nodes ($k = 5$) of the OBDD in Figure 1(a). In this case, we have $\mathbf{X} = \{V, U\}$. These nodes lead to the following decomposition.

| $k$-node label | $k$-node function ($\mathbf{X}$) | $k$-paths function ($\mathbf{Y}$) |
|---|---|---|
| $V$ | $V \wedge U$ | $\neg T \wedge (R \vee (S \wedge \neg R)) \wedge \neg Q$ |
| $V$ | $V$ | $\neg T \wedge (R \vee (S \wedge \neg R)) \wedge Q$ |
| $U$ | $U$ | $T \wedge (R \vee (S \wedge \neg R)) \wedge \neg Q$ |
| $H(\mathbf{Y}) = T \wedge (R \vee (S \wedge \neg R)) \wedge Q$ | | |

Since this decomposition is strongly deterministic on $\mathbf{Y} = \{Q, R, S, T\}$, every consistent sub-function $f|\mathbf{y}$ must appear in the second column of the above table by Proposition 1. Note that only two of these sub-functions depend on the $k^{th}$ variable in the ordering, $V$. Hence, the Sieling and Wegener's lower bound says that we must have at least two OBDD nodes labeled with variable $V$. Corollary 1, together with Proposition 1, is saying this and more. In particular, the proposition implies that every $f|\mathbf{y}$ that depend on $X_k$ must appear in any decomposition, including the one given by the corollary. Each of these functions (that depends on $X_k$) must correspond to the function of a node labeled with $X_k$ in the OBDD. As a result, we can conclude that the number of nodes labeled with $X_k$ must be lower bounded by the number of distinct functions $f|\mathbf{y}$ that depend on $X_k$. Moreover, the corollary is saying that we must have one distinct OBDD node for each distinct and consistent sub-function $f|\mathbf{y}$ (regardless of its dependence on $X_k$). These distinct nodes are labeled with variables in $\mathbf{X}$ or *true*.

## Conclusions

We presented in this paper a lower bound on the size of structured DNNF. The result, which relies on the general notion of decompositions, enables us to transform knowledge about Boolean functions into a size guarantee for DNNFs that respect certain vtrees. We discussed our result in the context of a few distinguished subsets of DNNF: deterministic DNNF and OBDD. We showed that our lower bound, when applied to OBDD, subsumes the well-known OBDD lower bound given by Sieling and Wegener. Finally, we demonstrated some usages of our result in proving lower bounds for DNNF representations.

Unlike the Sieling and Wegener's result, our result in this paper does not provide an upper bound on the size of structured DNNF. While we always insist on using the Shannon decompositions in the construction of every OBDD, we do not make any assumption about the type of decompositions used to construct structured DNNF in general. As a result, coming up with a useful upper bound requires some assumptions about the decompositions used. This is a topic of ongoing research.

## References


Breitbart, Y.; Hunt, III, H.; and Rosenkrantz, D. 1995. On the size of binary decision diagrams representing boolean functions. *Theor. Comput. Sci.* 145(1-2):45–69.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Tran. Com.* C-35:677–691.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *JAIR* 17:229–264.

Darwiche, A. 2001a. Decomposable negation normal form. *Journal of the ACM* 48(4):608–647.

Darwiche, A. 2001b. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics* 11(1-2):11–34.

Fortune, S.; Hopcroft, J. E.; and Schmidt, E. M. 1978. The complexity of equivalence and containment for free single variable program schemes. In *Proc. of 5th Colloquium on Automata, Languages and Programming*, 227–240.

Mateescu, R., and Dechter, R. 2006. Compiling constraint networks into and/or multi-valued decision diagrams (AOMDDs). In *Proc. of CP-06*, 329–343.

Pipatsrisawat, K., and Darwiche, A. 2008. New compilation languages based on structured decomposability. In *Proc. of AAAI-08*, 517–522.

Ponzio, S. 1995. A lower bound for integer multiplication with read-once branching programs. In *Proc. of STOC '95*, 130–139.

Sieling, D., and Wegener, I. 1993. Nc-algorithms for operations on binary decision diagrams. *Parallel Processing Letters* 3:3–12.

Wegener, I. 2000. *Branching programs and binary decision diagrams: theory and applications*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.