Structured d-DNNF Is Not Closed under Negation

Harry Vinall-Smeeth
Technische Universität Ilmenau, Germany
harry.vinall-smeeth@tu-ilmenau.de

Abstract

Both structured d-DNNF and SDD can be exponentially more succinct than OBDD. Moreover, SDD is essentially as tractable as OBDD. But this leaves two important open questions. Firstly, does OBDD support more tractable transformations than structured d-DNNF? And secondly, is structured d-DNNF more succinct than SDD? In this paper, we answer both questions in the affirmative. For the first question we show that, unlike OBDD, structured d-DNNF does not support polytime negation, disjunction, or existential quantification operations. As a corollary, we deduce that there are functions with an equivalent polynomial-sized structured d-DNNF but with no such representation as an SDD, thus answering the second question. We also lift this second result to *arithmetic circuits* (AC) to show a succinctness gap between PSDD and the positive AC analogue to structured d-DNNF.

1 Introduction

Knowledge compilation aims to provide useful representations of Boolean functions (propositional knowledge bases). What 'useful' means has, broadly speaking, three aspects. The first is succinctness: how big is our representation? The second is transformations. For instance, given a representation for f and a representation for g can we form a representation for $f \wedge g$ in polynomial time? The third is queries: given our representations what can we (efficiently) determine about our function? For example, given a representation for f can we compute $|f^{-1}(1)|$ in polynomial time? These aspects may be in tension with one another; to get a representation which supports more queries or transformations we may have to accept increased size. A key task in knowledge compilation is to map out the trade-offs of using different representations.

In the landmark paper [Darwiche and Marquis, 2002], it is shown that many well-studied representation formats are subsets of Boolean circuits in Negation Normal Form (NNF). Consequently, over the past two decades, research on representations within the AI community has focused on classes which arise from imposing syntactic restrictions on NNF. Two influential restrictions are *decomposability* and *determinism*, an NNF with both properties is called a d-DNNF. Such circuits support a large range of polynomial time queries such as clausal entailment and model enumeration.

An older representation format is the Ordered Binary Decision Diagram (OBDD) [Bryant, 1986]. In fact, OBDD is a subset of d-DNNF [Darwiche and Marquis, 2002]. While a Boolean function may have an equivalent d-DNNF that is exponentially smaller than any equivalent OBDD, in many practical settings the latter is preferred. There are two crucial reasons for this. Firstly, OBDDs that use a common variable order are closed under Boolean operations; this is useful for instance in *bottom-up* approaches to knowledge compilation, see e.g. [Somenzi, 2009]. Secondly, OBDDs are *canonical* which simplifies the task of finding an optimal compilation; one just needs to find an optimal variable order.

A natural question is whether there are compilation languages that are more succinct than OBDD but have nicer properties than d-DNNF? This paper will be concerned with two such languages: *structured* d-DNNF [Pipatsrisawat and Darwiche, 2008] and Sequential Decision Diagram (SDD) [Darwiche, 2011]. SDD has become a popular representation format since they may be exponentially more succinct than OBDD yet still support all Boolean operations in polynomial time. Moreover, *compressed* SDDs are canonical [Darwiche, 2011; Van den Broeck and Darwiche, 2015]. Structured d-DNNF, on the other hand, contains SDD as a subset and supports a polynomial time conjoin operation. They may, however, be exponentially more verbose than d-DNNF.

One may then wonder: is there any advantage to using structured d-DNNF over SDD? To be precise are there functions which have polynomially sized representations in structured d-DNNF but do not have such SDD representations? This is a question which has been raised since at least 2015 [Beame and Liew, 2015] and has received substantial interest, see e.g. [Bova and Szeider, 2017; Bollig and Farenholtz, 2021], but has remained open until now. In this paper, we answer this question in the affirmative.

Theorem 1. *For every $s \in \mathbb{N}$, there exists a function f with an equivalent structured d-DNNF of size s such that any SDD equivalent to f has size $s^{\tilde{\Omega}(\log(s))}$.*

Here $\tilde{\Omega}$ is the variant of Ω notation which ignores polylogarithmic factors; we analogously define $\tilde{O}$. We prove Theorem 1 by showing that structured d-DNNF is *not* closed under negation which has been an open question in its own right

since [Pipatsrisawat and Darwiche, 2008].

Theorem 2. *For every $s \in \mathbb{N}$, there exists a Boolean function f with an equivalent structured* d-DNNF *of size s and such that any structured* DNNF *equivalent to $\neg f$ has size $s^{\tilde{\Omega}(\log s)}$.*

Thus, we simultaneously show that there is an advantage to using SDD over structured d-DNNF. We similarly show that structured d-DNNF is not closed under disjunction or existential quantification thus completing the 'knowledge compilation map' for structured d-DNNF.

Arithmetic circuits (AC) also play an important role in AI, particularly in *probabilistic reasoning*. Here one prominent circuit type is PSDD [Kisa *et al.*, 2014]. As the name suggests, these are the AC analogue of SDD. PSDDs have several nice properties making them ripe for applications. For example, they support a polynomial time multiplication operation (analogous to the polynomial time conjoin operation for SDDs), which is useful when compiling probabilistic graphical models [Shen *et al.*, 2016]. de Colnet and Mengel observed that in some cases separations between representations of Boolean functions can be extended to positive AC in a straightforward manner [de Colnet and Mengel, 2021]. We exploit this to show a succinctness gap between PSDD and the positive AC analogue to structured d-DNNF.

Our proof of Theorem 2 exploits a connection between knowledge compilation and *communication complexity* which has been widely deployed in recent years, see e.g. [Beame and Liew, 2015; Bova *et al.*, 2016; Amarilli *et al.*, 2020]. We start from the same piece of communication complexity as [Göös *et al.*, 2022], where an analogous result for unambiguous finite automata (UFA) is obtained. However, while the size of UFAs is related to the *fixed partition* communication complexity model the size of structured d-DNNF is related to another model: the *best partition* communication complexity. We therefore adapt an ingenious construction from [Knop, 2017], which allows one to lift results from the fixed partition model to the best partition model.[1]

The rest of the paper is structured as follows. In Section 2 we define our main objects of study. In Section 3 we introduce the communication complexity we need. Following this, in Section 4 we prove our main theorem. Finally, in Section 5 we show how our results extend to ACs.

2 Formulas, NNF, Structured d-DNNF and SDD

2.1 Formulas and Boolean Functions

Recall that a propositional formula is a DNF if it is a disjunction of conjunctions. We call each disjunct a *term*. A DNF ψ is a k-DNF if each term contains at most k-literals and *unambiguous* if every assignment $\alpha : \mathsf{var}(\psi) \to \{0,1\}$ satisfies at most one term of ψ. Let $\mathsf{sat}(\psi)$ denote the set of satisfying assignments for a propositional formula ψ. We identify each such ψ with a Boolean function with domain $\{0,1\}^{\mathsf{var}(\psi)}$ in the standard way, i.e., the function evaluates to 1 on input $\underline{x}$ iff $\underline{x} \in \mathsf{sat}(\psi)$.

For $f : \{0,1\}^X \to \{0,1\}$ a Boolean function, we write $\mathsf{sat}(f) := f^{-1}(1)$. We may view $\underline{x}$, an input to f, as a tuple with one coordinate for each element of X; we write $\underline{x}(x)$ to denote the coordinate of $\underline{x}$ corresponding to variable $x \in X$. For $Y \subseteq X$, we say that the projection of $\underline{x}$ to Y—denoted $\pi_Y(\underline{x})$—is the $\underline{y} \in \{0,1\}^Y$ with $\underline{y}(y) = \underline{x}(y)$ for all $y \in Y$ and extend this notion to sets in the natural way. We will be interested in the following transformations.

Definition 1. *Let $f, g : \{0,1\}^X \to \{0,1\}$ be Boolean functions and $x \in X$. Then we write:*

1. *(negation) $\neg f$ to denote the Boolean function with $\mathsf{sat}(\neg f) = f^{-1}(0)$;*
2. *(existential quantification) $\exists x f$ to denote the Boolean function with $\mathsf{sat}(\exists x f)$ equal to the projection of $\mathsf{sat}(f)$ to $\{0,1\}^{X \setminus \{x\}}$;*
3. *(disjunction) $f \vee g$ to denote the Boolean function with $\mathsf{sat}(f \vee g) = \mathsf{sat}(f) \cup \mathsf{sat}(g)$ and*
4. *(conjunction) $f \wedge g$ to denote the Boolean function with $\mathsf{sat}(f \wedge g) = \mathsf{sat}(f) \cap \mathsf{sat}(g)$.*

2.2 Negation Normal Form

Definition 2. *A Boolean circuit in* Negation Normal Form (NNF) *is a vertex-labelled directed acyclic graph with a unique source such that every internal node is a fan-in two $\wedge$- or $\vee$-node and whose leaves are each labelled by 0, 1, a variable x or a negated variable $\neg x$.*

We define the size of $\mathcal{C}$, an NNF, to be the number of vertices in the underlying graph and denote this by $|\mathcal{C}|$. By expanding out an NNF circuit $\mathcal{C}$ we get a unique propositional formula which we denote by $\langle \mathcal{C} \rangle$. Further, we write $\mathsf{var}(\mathcal{C})$ for the set of variables occurring in $\mathcal{C}$. It will be convenient to associate a set $\mathsf{dom}(\mathcal{C})$ to $\mathcal{C}$ which contains $\mathsf{var}(\mathcal{C})$ (together with possibly other variables). Unless otherwise stated, we assume that $\mathsf{dom}(\mathcal{C}) = \mathsf{var}(C)$. We write $f_C : \{0,1\}^{\mathsf{dom}(\mathcal{C})} \to \{0,1\}$ to denote the Boolean function computed by $\mathcal{C}$ in the obvious way. We say that $\mathcal{C}$ is *equivalent* to $f_{\mathcal{C}}$ and define $\mathsf{sat}(\mathcal{C}) := \mathsf{sat}(f_{\mathcal{C}})$. If for some $\mathsf{C} \subseteq$ NNF, a Boolean function f is equivalent to $\mathcal{C} \in \mathsf{C}$ of size s then we say that f *admits* a C of size s.

2.3 Decomposability, Determinism and Structuredness

For a node g of $\mathcal{C}$, we write $\mathcal{C}(g)$ for the subcircuit rooted at g; for conciseness we write $\mathsf{var}(g)$ to mean $\mathsf{var}(\mathcal{C}(g))$. If g is not a leaf we write g_ℓ (resp. g_r) for its left (resp. right) child. An NNF, $\mathcal{C}$, is *decomposable* if for every $\wedge$-node $g \in \mathcal{C}$, $\mathsf{var}(g_\ell) \cap \mathsf{var}(g_r) = \emptyset$ [Darwiche, 2001a]. $\mathcal{C}$ is *deterministic* if for every $\vee$-node $g \in \mathcal{C}$, $\mathsf{sat}(\mathcal{C}(g_\ell)) \cap \mathsf{sat}(\mathcal{C}(g_r)) = \emptyset$, where we set $\mathsf{dom}(\mathcal{C}(g_\ell)) = \mathsf{dom}(\mathcal{C}(g_r)) = \mathsf{dom}(\mathcal{C})$ [Darwiche, 2001b]. The set of decomposable NNF is denoted by DNNF and the set of deterministic DNNF by d-DNNF.

We now only need one ingredient to get to structured d-DNNF; for this, we need the notion of a *v-tree*.

[1] There is an older construction from [Lam and Ruzzo, 1992] which also allows one to lift communication complexity results to the best partition model. However, this construction requires that the functions involved are *paddable*; as far as we can tell, this is not the case for the functions we use.

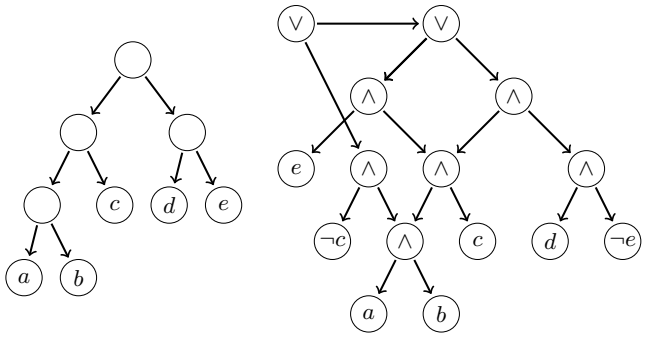

Figure 1: (left) A v-tree T. (right) A structured d-DNNF $\mathcal{C}$ that respects T. $\langle \mathcal{C} \rangle = (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c \wedge e) \vee (a \wedge b \wedge c \wedge d \wedge \neg e))$.

Definition 3. *A* v-tree *over variables X is a full, rooted, binary tree whose leaves are in 1-1 correspondence with the elements of X.*

For a non-leaf node t of a v-tree T, we write t_ℓ for its left child, t_r for its right child and $\mathsf{var}(t)$ for the variables appearing in the subtree rooted at t. A DNNF $\mathcal{C}$ *respects* a v-tree T, if for every $\wedge$-node $g \in \mathcal{C}$, there is a node t of T such that $\mathsf{var}(g_\ell) \subseteq \mathsf{var}(t_\ell)$ and $\mathsf{var}(g_r) \subseteq \mathsf{var}(t_r)$; see Figure 1.

Definition 4 ([Pipatsrisawat and Darwiche, 2008])**.** *A (d)-DNNF $\mathcal{C}$ is* structured *if it respects some v-tree. We denote the set of structured (d)-DNNF by (d)-SDNNF.*

2.4 SDDs

SDD is a subset of d-SDNNF which arises from imposing a stricter form of determinism and structuredness called *strong determinism*. The idea is that SDDs respect a certain type of decomposition which generalises the well-known Shannon decomposition on which OBDDs are based.

Definition 5. *Let $f : \{0,1\}^Z \to \{0,1\}$ be a Boolean function and $X, Y \subseteq Z$ be disjoint sets of variables. Then if*

$$f = \bigvee_{i=1}^{n} p_i(X) \wedge s_i(Y)$$

then $\{(p_1, s_1), \ldots, (p_n, s_n)\}$ is an X-decomposition for f if $\vee_{i=1}^n p_i \equiv 1$, $p_i \wedge p_j \equiv 0$ for all $i \neq j$ and $p_i \not\equiv 0$ for all i.

The idea is that the $\{p_i\}_{i \in [n]}$ form a partition. We can now define SDDs.

Definition 6. *Let T be a v-tree over variables Z. An SDD respecting T is a DNNF $\mathcal{C}$ with one of the following forms.*

- *$\mathcal{C}$ consists of a single node labelled by 0, 1, x or $\neg x$, where $x \in Z$.*
- *The source of $\mathcal{C}$ is a $\vee$ node g and there exists some t, an internal node of T, such that:*
  1. *$\langle \mathcal{C} \rangle = \bigvee_{i=1}^n p_i(X) \wedge s_i(Y)$ where $\{(p_i, s_i)\}_{i \in [n]}$ is an X decomposition for $f_{\mathcal{C}}$,*
  2. *$X \subseteq \mathsf{var}(t_\ell)$, $Y \subseteq \mathsf{var}(t_r)$ and*
  3. *if $h \in \mathcal{C}$ with $\langle \mathcal{C}(h) \rangle = p_i(X)$ (resp. $s_i(Y)$) for some i then $\mathcal{C}(h)$ is an SDD that respects the subtree of T rooted at t_ℓ (resp. t_r).*

An SDD is an SDD that respects some v-tree.

It follows from the definition that SDDs are deterministic and structured. One can further show that SDDs admit conjunction, disjunction and complementation in polynomial time [Darwiche, 2011]. These are the main facts we need; we include the full definition for context and because it is needed for the connection to arithmetic circuits,[2] see [Darwiche, 2011; Bollig and Farenholtz, 2021] for a more thorough introduction to SDDs.

2.5 Succinctness

Since we want to compare the succinctness of different representations we need the following notion.

Definition 7. *[Gogic* et al.*, 1995] Let $\mathsf{C}_1, \mathsf{C}_2 \subseteq$ NNF. Then C_1 is* at least as succinct *as C_2 if there is a polynomial p, such that for every $\mathcal{C} \in \mathsf{C}_2$ there is an equivalent $\mathcal{C}' \in \mathsf{C}_1$ with $|\mathcal{C}'| \leq p(|\mathcal{C}|)$. We write $\mathsf{C}_1 \leq \mathsf{C}_2$. C_1 is* more succinct *than C_2, denoted $\mathsf{C}_1 < \mathsf{C}_2$, when $\mathsf{C}_1 \leq \mathsf{C}_2$ and $\mathsf{C}_2 \not\leq \mathsf{C}_1$.*

3 Knowledge Compilation and Communication Complexity

We will next introduce all of the background from communication complexity we will use in Section 4, see [Kushilevitz and Nisan, 1997] for an introduction to the subject.

Let $f \colon \{0,1\}^Z \to \{0,1\}$, for some set Z, and $\Pi = (X, Y)$ be a partition of Z. We will only consider *balanced* partitions, i.e. partitions with $|Z|/3 \leq \min\{|X|, |Y|\}$. Then a Π-rectangle is a set $A \times B \subseteq \{0,1\}^Z$, with $A \subseteq \{0,1\}^X$ and $B \subseteq \{0,1\}^Y$. We say that Π-rectangles $R_1, \ldots, R_k$ *cover* a set $S \subseteq \{0,1\}^Z$, if $\bigcup_i R_i = S$. We write $\mathsf{Cov}_b^\Pi(f)$ to denote the minimum size of a set of Π-rectangles that cover $f^{-1}(b)$. This is a communication complexity measure in the *fixed partition* model; we are given Π and then find a cover of minimal size using Π-rectangles.

In order to get a connection to SDNNF we need to instead look at the *best partition* model: here we may choose Π. Formally, we define $\mathsf{Cov}_b(f) := \min_\Pi \mathsf{Cov}_b(f)$, where the minimum is over all balanced partitions. The following lemma shows that if there is no small rectangular cover of $\mathsf{sat}(f)$ then f does not admit a small SDNNF.

Lemma 1 ([Pipatsrisawat and Darwiche, 2010; Bova *et al.*, 2016].)**.** *If f admits an SDNNF of size s then $\mathsf{Cov}_1(f) \leq s$.*

We end this section by stating some simple but useful combinatorial properties of rectangles. Let $R \subseteq \{0,1\}^Z$ be a (X, Y)-rectangle, $W \subseteq Z$ and $\underline{a} \in \{0,1\}^W$. Then we define a selection operation by removing all tuples from R which do not agree with $\underline{a}$. Formally, we define $\mathsf{sel}_{\underline{a}}(R) := \{\underline{x} \in R \mid \underline{x}(w) = \underline{a}(w) \text{ for all } w \in W\}$. We also define a operation allowing us to rename variables. Formally, for a set Z' and a bijection $\sigma \colon Z \to Z'$, $\underline{b} \in \{0,1\}^Z$ we define $\underline{b}^\sigma \in \{0,1\}^{Z'}$ to be the tuple with $\underline{b}^\sigma(\sigma(z)) = \underline{b}(z)$ for all $z \in Z$ and $R^\sigma = \{\underline{b}^\sigma \mid \underline{b} \in R\}$. The following is immediate.

[2] The definition is somewhat cumbersome as all nodes have fan-in two. We enforce this to make the overall presentation cleaner. Note, that circuits with unbounded fan-in can be rewritten as fan-in two circuits with only a quadratic size blow-up.

Lemma 2. *Let $R \subseteq \{0,1\}^Z$ be a (X,Y)-rectangle, $W \subseteq Z$, $\underline{a} \in \{0,1\}^W$ and σ a bijection with domain Z. Then the projection of R to W is a $(X \cap W, Y \cap W)$-rectangle, $\mathsf{sel}_{\underline{a}}(R)$ is a (X,Y)-rectangle and R^σ is a $(\sigma(X), \sigma(Y))$-rectangle.*

4 Proof of Theorems 1 and 2

4.1 Proof Outline

In [Göös *et al.*, 2022] an analogue of Theorem 2 is proved but for UFA. Their approach goes via communication complexity in the fixed partition model. Our proof starts from this same piece of communication complexity but we need to work in the best-partition model. It turns out we can make this jump by adapting an ingenious construction from [Knop, 2017] based on the work of [Segerlind, 2008].

Our starting point is the following result shown in the proof of [Göös *et al.*, 2022, Theorem 1] building on results from [Göös *et al.*, 2016] and [Balodis *et al.*, 2021].

Theorem 3. *For every $k \in \mathbb{N}$, there exists an integer $n = k^{O(1)}$, a Boolean function $g : \{0,1\}^n \to \{0,1\}$ and Π, a balanced partition, such that the following properties hold.*

1. *g is equivalent to an unambiguous k-DNF ψ with $2^{\tilde{O}(k)}$ terms.*
2. *$\mathsf{Cov}_0^\Pi(g) = 2^{\tilde{\Omega}(k^2)}$.*

One can show that g admits a $\mathsf{d\text{-}SDNNF}$ of size $2^{\tilde{O}(k)}$. We would therefore like to show a lower bound on the size of an SDNNF equivalent to $\neg g$. However, we cannot use Lemma 1 since Theorem 3 only gives lower bounds in the fixed-partition model. To get around this we transform g into a new function f which still admits a $\mathsf{d\text{-}SDNNF}$ of size $2^{\tilde{O}(k)}$ and with $\mathsf{Cov}_0(f) \geq \mathsf{Cov}_0^\Pi(g) = 2^{\tilde{\Omega}(k^2)}$.

4.2 From Fixed Partition to Best Partition

We now present the construction which allows us to build f. This is almost the same as that given in [Knop, 2017]. The following things are different (1) we are now working with formulas in DNF rather than CNF, (2) we now want to ensure that the construction transforms an unambiguous DNF into an unambiguous DNF and (3) the notion of balancedness we use is different. We end up with the following result.

Theorem 4. *Let ψ be an unambiguous n-variable k-DNF with ℓ terms. Then there exists an unambiguous $O(n^2)$ variable $O(kn)$-DNF ψ' with $O(\ell n^{k+4})$ terms such that for $\delta \in \{0,1\}$ and any balanced partition Π of the variables of ψ, $\mathsf{Cov}_\delta(\psi') \geq \mathsf{Cov}_\delta^\Pi(\psi)$.*

We next show how this implies Theorem 2.

Proof of Theorem 2. Fix some $k \in \mathbb{N}$ and let $g : \{0,1\}^n \to \{0,1\}$ be the function from Theorem 3. Then we know there is some equivalent unambiguous k-DNF ψ with $2^{\tilde{O}(k)}$ terms. Let $f \equiv \psi'$. By Theorem 4 and since $n = k^{O(1)}$, ψ' has $2^{\tilde{O}(k)}$ terms. Then we can form a $\mathsf{d\text{-}SDNNF}$ equivalent to f as follows. Fix any v-tree T over $\mathsf{var}(\psi')$. Then since every term of ψ' is a conjunction of $O(kn)$ literals, every term admits a $\mathsf{d\text{-}DNNF}$ respecting T of size $O(kn)$. By taking the disjunction of all such $\mathsf{d\text{-}DNNF}$ we get a $\mathsf{d\text{-}DNNF}$ for ψ' respecting T of size $2^{\tilde{O}(k)} := s$. Here determinism follows as ψ' is unambiguous.

But also by Theorem 3, $\mathsf{Cov}_0^\Pi(g) = 2^{\tilde{\Omega}(k^2)}$. So applying Theorem 4 we obtain that $\mathsf{Cov}_0(f) = 2^{\tilde{\Omega}(k^2)}$. Therefore, by Lemma 1, any SDNNF for $\neg f$ has size at least $2^{\tilde{\Omega}(k^2)} = s^{\tilde{\Omega}(\log s)}$. □

It only remains to prove Theorem 4.

4.3 Idea of the Construction

One way to try and transfer bounds from the fixed partition model to the best partition model is to extend a function to include a permutation as part of the input. A naive way of doing this is as follows. Let $\psi(x_1, \ldots, x_n)$ be a propositional formula. Associate a binary string of length $m := \log_2(n!)$ to each of the permutations of these variables and for each such permutation σ write $\mathsf{rep}(\sigma)_i$ for the ith bit of the string associated with σ. We define

$$\mathsf{perm}(\psi)(z_1, \ldots, z_m, x_1, \ldots, x_n) \equiv \bigwedge_{\sigma \in S_n} \left(\bigwedge_{i=1}^{m} (z_i = \mathsf{rep}(\sigma)_i) \to \psi\left(\sigma(x_1), \ldots, \sigma(x_n)\right) \right).$$

What is the idea? Let $\Pi = (\Pi_1, \Pi_2)$ be a balanced partition of $\mathsf{var}(\psi)$ and $\Gamma = (\Gamma_1, \Gamma_2)$ be any balanced partition of $\mathsf{var}(\mathsf{perm}(\psi))$ such that $|\Pi_1| = |\{x_i \mid x_i \in \Gamma_1\}|$. Further, let $R_1, \ldots, R_\alpha$ be a set of Γ-rectangles covering $\mathsf{sat}(\psi')$ and σ be the permutation such that $\Pi_1 = \{\sigma(x_i) \mid x_i \in \Gamma_1\}$. We use this to define a set of Π-rectangles covering $\mathsf{sat}(\psi)$.

To do this let $\underline{a} \in \{0,1\}^{\{z_1, \ldots, z_n\}}$ be the tuple with $\underline{a}(z_i) = \mathsf{rep}(\sigma)_i$ and set $R'_i = (\pi_{\{x_1, \ldots, x_n\}}(\mathsf{sel}_{\underline{a}}(R_i)))^\sigma$. That is we first select from R_i according to $\underline{a}$, then project to $\{x_1, \ldots, x_n\}$ and finally rename the variables according to σ. By Lemma 2 and the choice of σ, R'_i is a Π-rectangle. Moreover, it is easy to see that $\mathsf{sat}(\psi) = \bigcup_{i=1}^{\alpha} R'_i$.

But there is an issue: the size of $\mathsf{perm}(\psi)$ is exponential in n. So our proof doesn't go through with this construction because there is no reason to think that ψ admitting a small $\mathsf{d\text{-}SDNNF}$ implies that $\mathsf{perm}(\psi)$ admits such a representation.[3]

The obvious solution is to consider a smaller set of permutations. But then the σ such that $\Pi_1 = \{\sigma(x_i) \mid x_i \in \Gamma_1\}$ might not be in our set. To get around this we, following [Knop, 2017], *first* add copies of variables to our original formula and *then* permute these new variables using a carefully chosen set of permutations. This effectively increases the number of permutations we can reach without excessively blowing up the size of our formula.

4.4 The Construction

Step 1: Making Copies of Variables

Let ψ be an unambiguous DNF formula on n variables. We replace every occurrence of variable x_i by a disjunction of m fresh variables $\bigvee_{j \in [m]} y_{i,j}$, where $m = cn$, for some sufficiently large constant c. Denote the subformula obtained

[3] Another issue is that this argument only gives lower bounds for partitions where $|X| = |\{x_i \mid x_i \in X'\}|$.

from a term C of ψ by $C^\vee$. The resulting formula is *not* a DNF. To change this first we use distributivity to expand out our formula into a DNF ϕ. If a term of ϕ is the result of expanding out $C^\vee$ we say it is *derived* from C. As this DNF may not be unambiguous we need an extra step not in [Knop, 2017]. For each term C of ϕ and every positive literal $y_{i,j}$ in C add conjuncts $\neg y_{i,j'}$ for every $j' \neq j$ to get a term C^u. If C is derived from D we also say that C^u is derived from D. We denote the resulting DNF by $\psi^\vee$.

Lemma 3. *If ψ is an unambiguous n-variable k-DNF with ℓ terms then $\psi^\vee$ is an $O(n^2)$-variable unambiguous $O(kn)$-DNF with $O(\ell n^k)$ terms.*

Proof. To see that $\psi^\vee$ is unambiguous suppose some assignment α of the variables of $\psi^\vee$ satisfies a term C. Define the assignment $\beta = \beta(\alpha)$ on the variables of ψ by $\beta(x_i) = 1$ if and only if $\alpha(\bigvee_{j\in[m]} y_{i,j}) = 1$. We know that C was derived from some D. Then clearly β satisfies D. Since ψ is unambiguous this is the unique term satisfied by β. Therefore, if α satisfies some C' then C' must also be derived from D. We have that there are $I_p, I_n \subseteq [n]$ with $|I_p \cup I_n| = O(k)$ such that $D^\vee$ is equal to

$$\bigwedge_{i\in I_p}\bigvee_{j=1}^{m} y_{i,j} \wedge \bigwedge_{i\in I_n} \neg \bigvee_{j=1}^{m} y_{i,j} \equiv \bigwedge_{i\in I_p}\bigvee_{j=1}^{m} y_{i,j} \wedge \bigwedge_{i\in I_n}\bigwedge_{j=1}^{m} \neg y_{i,j}$$

It follows that each term of $\psi^\vee$ deriving from D is of the form

$$\bigwedge_{i\in I_p} y_{i,j_i} \wedge \bigwedge_{j\neq j_i} \neg y_{i,j} \wedge \bigwedge_{i\in I_n}\bigwedge_{j=1}^{m} \neg y_{i,j}$$

where $(j_i)_{i\in I_p}$ is some sequence of elements of $[m]$. It is therefore easy to see that $C = C'$ and so $\psi^\vee$ is unambiguous. Moreover, by the above each term contains $O(km) = O(kn)$ variables. Finally, by construction, there are at most m^k terms of $\phi^\vee$ derived from each term of ϕ. □

Step 2: Adding Permutations

Let $|\mathsf{var}(\psi^\vee)| = n'$. For notational convenience we write v_{im+j} for $y_{i,j}$ and V for $\mathsf{var}(\psi^\vee)$. For simplicity[4] assume $n' = 2^t$ for some integer t. Let $\mathbb{F}$ be the unique field of order n' and $\mathcal{P}$ be the set of mapping on $\mathbb{F}$ with $x \to ax + b$, $a, b \in \mathbb{F}$ and $a \neq 0$. The reason for using $\mathcal{P}$ is that it is a set of independent permutations in the following sense.

Lemma 4. *[Wegman and Carter, 1981] Every mapping in $\mathcal{P}$ is a permutation and $|\mathcal{P}| = n' \cdot (n' - 1)$. Moreover, for any $a, b, c, d \in [n']$ with $a \neq b$ and $c \neq d$,*

$$\Pr_{\sigma\in\mathcal{P}}[\sigma(a) = c, \sigma(b) = d] = \frac{1}{|\mathcal{P}|}$$

Elements of $\mathcal{P}$ may be represented by binary strings of length $2t$ such that the first t bits are not all zero; we denote the ith bit of the representation of $\sigma \in \mathcal{P}$ by $\mathsf{rep}(\sigma)_i$.

[4]The general case is not much more difficult: we just add extra 'dummy' variables until we reach a power of two. This does not change anything, other than making the notation slightly messier, see [Knop, 2017, Theorem 4.2].

Let C be a term of $\psi^\vee$ with $C = \bigwedge_{i\in I} a_i$, for some $a_i \in \{v_i, \neg v_i\}$. Then for each $\sigma \in \mathcal{P}$ and C a term of $\psi^\vee$ we define a term

$$\mathsf{perm}_\pi(C) := \bigwedge_{i=1}^{2t} (z_i = \mathsf{rep}(\sigma)_i) \wedge \bigwedge_{i\in I} a_{\sigma(i)}$$

To form ψ' we take the disjunction of every $\mathsf{perm}_\pi(C)$ with $\pi \in \mathcal{P}$, C a term of $\psi^\vee$. Note, that if $\phi^\vee$ is unambiguous so is ψ' and that $|\mathcal{P}| = O(n^4)$. The following is immediate.

Lemma 5. *If ψ is a n-variable unambiguous k-DNF with ℓ terms then ψ' is a $O(n^2)$-variable unambiguous $O(kn)$-DNF with $O(\ell n^{k+4})$ terms.*

4.5 Proof of Theorem 4

Proof. Fix two arbitrary balanced partitions $\Pi = (\Pi_1, \Pi_2)$ and $\Gamma = (\Gamma_1, \Gamma_2)$ of the variables of ψ and ψ' respectively. It is enough to show that $\mathsf{Cov}_b^\Gamma(\psi') \geq \mathsf{Cov}_b^\Pi(\psi)$. So let $C = \{R_i\}_{i\in[\alpha]}$ be a set of Γ-rectangles which cover $\mathsf{sat}(\phi')$; we will form a set of Π-rectangles which cover $\mathsf{sat}(\psi)$ of size at most α. The key is the following.

Claim 1. *There is a permutation $\sigma \in \mathcal{P}$, such that for any $i \in [n]$ and $k \in \{0, 1\}$, there is a j such that $y_{i,j}$ is mapped to a variable from Γ_k by σ.*

Assume the claim and let σ be such a permutation. Let $v_{r(i,k)}$ denote some $y_{i,j}$ that is mapped to an element of Γ_k by the permutation σ. Let $V_1 = \{v_{r(i,k)} \mid x_i \in \Pi_k\}$ and $V_2 = V \setminus V_1$. Define the bijection $\tau : V_1 \to \mathsf{var}(\psi)$ by $\tau(v_{r(i,k)}) = x_i$. The key observation is that for every $\underline{x} \in \{0, 1\}^{\mathsf{var}(\phi)}$, $\psi'(\underline{x}') = \psi(\underline{x})$ where $\underline{x}' \in \{0, 1\}^{\mathsf{var}(\phi')}$ is the unique tuple with:

$$\underline{x}'(x) = \begin{cases} \mathsf{rep}(\sigma)_i & x = z_i \\ \underline{x}(x_i) & x = v_{r(i,k)} \text{ and } x \in V_1 \\ 0 & \text{otherwise.} \end{cases}$$

Note that the projection of $\underline{x}'$ to $\mathsf{var}(\phi') \setminus V_1$ is the same for every $\underline{x}$, call the resulting tuple $\underline{a}$. Then, for each R_i we form R_i' by selecting only the tuples agreeing with $\underline{a}$ then projecting to V_1 before renaming variables via τ, i.e. $R_i' := (\pi_{V_1}(\mathsf{sel}_{\underline{a}} R_i))^\tau$. By Lemma 2 and the choice of σ, every R_i is a Π-rectangle and it is easy to verify that $\mathsf{sat}(\psi) = \bigcup_{i=1}^{\alpha} R_i'$. It follows that $\mathsf{Cov}_1^\Gamma(\psi') \geq \mathsf{Cov}_1^\Pi(\psi)$. The case where $b = 0$ is identical except now $\neg\psi'$ plays the role of ψ'.

Claim 1 follows by a relatively simple probabilistic argument: the two key tools are Chebyshev's inequality and Lemma 4. The proof is almost identical to [Knop, 2017, Theorem 4.2.]. The only difference is that we need to use our more relaxed notion of balancedness but an inspection of the proof shows that everything goes through. □

Theorem 1 follows almost immediately.

Proof of Theorem 1. Fix $s \in \mathbb{N}$, let f be the function given by Theorem 2 and let $\mathcal{C}$ be an SDD equivalent to f. Then we may complement this SDD to get $\mathcal{C}'$ an SDD for $\neg f$ of size polynomial in $|\mathcal{C}|$. Since SDD is a subset of d-SDNNF by Theorem 2, $|\mathcal{C}'| = s^{\tilde{\Omega}(\log(s))}$. The result follows. □

4.6 Disjunction and Existential Quantification

In the appendix, we also prove the following.

Theorem 5 (Disjunction)**.** *For every $s \in \mathbb{N}$, there exists Boolean functions f, g sharing a common domain with the following properties.*

1. *There is a v-tree T such that f and g both admit d-DNNFs of size s respecting T.*
2. *Any d-SDNNF equivalent to $f \vee g$ has size $s^{\tilde{\Omega}(\log s)}$.*

The proof follows the same structure as that given above. In fact, Theorem 5 almost implies[5] Theorem 2; however we focused our expositional on negation for pedagogical reasons and since this is related to a long-standing open question. Namely, does (unstructured) d-DNNF admit polynomial time negation [Darwiche and Marquis, 2002]? We discuss this question in more detail in the conclusion. We also obtain the following corollary.

Corollary 1 (Existential Quantification)**.** *For every $s \in \mathbb{N}$, there exists a set X, a Boolean function $f : \{0,1\}^X \to \{0,1\}$ and a variable $x \in X$, such that*

1. *f admits a d-SDNNF of size s and*
2. *any d-SDNNF equivalent to $\exists x f$ has size $s^{\tilde{\Omega}(\log s)}$.*

Proof. Let f, g, T be as in the statement of Theorem 5. Then there are d-DNNFs $\mathcal{C}_f$ and $\mathcal{C}_g$, for f and g respectively, which both have size at most s and respect T. Let x be a fresh variable and form a new circuit $\mathcal{C}$ with

$$\langle \mathcal{C} \rangle = (x \wedge \langle \mathcal{C}_f \rangle) \vee (\neg x \wedge \langle \mathcal{C}_g \rangle)$$

The source is a $\vee$-node v which is deterministic since any element of $\mathsf{sat}(v_\ell)$ must map $x \to 1$ and any element of $\mathsf{sat}(v_r)$ must map $x \to 0$. Take T and form a v-tree T' by adding two fresh nodes r, t such that r has children t and the root of T. Then $\mathcal{C}$ is a d-DNNF of size $O(s)$ respecting T'. Moreover, $\exists x f_C \equiv f \vee g$. The result follows by Theorem 5. □

5 Lifting Results to Positive AC

So far we have been focussed on representations of Boolean functions; now we switch gears and look at representations of polynomials with real coefficients: arithmetic circuits (AC). These are defined similarly to NNF except now internal nodes are labelled by $+$ and $\times$ and any real number may be a constant.

Definition 8. *An arithmetic circuit (AC) is a vertex-labelled directed acyclic graph with a unique source such that every internal node is a fan-in two $+$- or $\times$-node and whose leaves are each labelled by a real number, a variable x or a negated variable $\neg x$.*

[5]Note that in Theorem 2 we prove lower bounds on the size of SDNNF equivalent to $\neg f$. Using Theorem 5 we could only get a lower bound on d-SDNNF. Still, this is, arguably, the most important part of the result. To go from Theorem 5 to the result on negation one uses the equivalence $f \vee g \equiv \neg(\neg f \wedge \neg g)$ and the tractability of $\wedge$ for d-SDNNF [Pipatsrisawat and Darwiche, 2008].

As in the case of NNFs, we associate a set of variables $\mathsf{dom}(\mathcal{C})$ to each arithmetic circuit which contains every variable occurring in the circuit. Then we associate to $\mathcal{C}$ a function $f_{\mathcal{C}} : \{0,1\}^{\mathsf{dom}(\mathcal{C})} \to \mathbb{R}$ as follows. On input $\underline{x}$, replace each variable x occurring positively in $\mathcal{C}$ with $\underline{x}(x)$ and each variable occurring negatively with $1 - \underline{x}(x)$. Then evaluate the circuit bottom up in the obvious way: the output is $f_{\mathcal{C}}(\underline{x})$. If we expand out the circuit we get a formula in $\mathsf{var}(\mathcal{C})$ (with possible negations). We denote this by $\langle \mathcal{C} \rangle$ and identify this expression with the function $f_{\mathcal{C}}$.

Since in many cases, ACs are used in the context of probabilistic reasoning it often makes sense to restrict our attention to ACs which output non-negative polynomials; call this fragment *positive* AC, denoted AC_p. This can be enforced syntactically by insisting that every constant is non-negative, giving the *monotone* fragment, denoted AC_m. This includes many well-studied classes such as PSDD [Kisa *et al.*, 2014] and Sum Product Networks (SPN) [Poon and Domingos, 2011]. Moreover, [de Colnet and Mengel, 2021] made the following observations connecting AC with NNF.

Let $\mathcal{C} \in \mathsf{AC}$. We form an NNF circuit $\phi(\mathcal{C})$ with the same underlying directed graph as $\mathcal{C}$ by relabelling each node as follows:

- **Leaves:** nodes labelled by a variable, a negated variable or the constant 0 are unchanged. Otherwise, change the label to the constant 1.
- **Internal node:** change $+$-nodes to $\vee$-nodes and $\times$-nodes to $\wedge$-nodes.

We will use the following result.[6]

Lemma 6 ([de Colnet and Mengel, 2021, Proposition 2])**.** *Let $\mathsf{C}_1, \mathsf{C}_2 \subseteq \mathsf{AC}_m$. Then $\phi(\mathsf{C}_1) < \phi(\mathsf{C}_2)$ implies that $\mathsf{C}_1 < \mathsf{C}_2$.*

Here $\leq$ is defined exactly as for subsets of NNF. Write $\mathsf{supp}(\mathcal{C})$ to denote the support of $\mathcal{C}$, i.e., the set of inputs for which $f_{\mathcal{C}}$ is non-zero. We can also lift the definitions of decomposability, determinism and structuredness to AC, by replacing the role of $\wedge$ with $\times$, $\vee$ with $+$ and sat with supp in the definitions. Let dSD-$\{\mathsf{AC}_\mathrm{m}, \mathsf{AC}_\mathrm{p}\}$ denote the deterministic, structured, decomposable, {monotone, positive} ACs. We next define the AC_m analogue to SDD.

Definition 9. *Let $f : \{0,1\}^X \to \mathbb{R}^+$ be a positive polynomial and $X, Y \subseteq Z$ be disjoint sets of variables. Suppose*

$$f = \sum_{i=1}^{n} \alpha_i \times p_i(X) \times s_i(Y)$$

where each $\alpha_i > 0$ and $\sum_{i=1}^n \alpha_i = 1$. Then $\{(p_1, s_1, \alpha_1), \ldots, (p_n, s_n, \alpha_n)\}$ is an X p-decomposition for f if $\sum_{i=1}^n p_i \equiv 1$, $p_i \times p_j \equiv 0$ for all $i \neq j$ and $p_i \not\equiv 0$ for all i.

The idea is that we take an X decomposition, put a distribution on the disjuncts and then replace $\vee$ with $+$ and $\wedge$

[6]We should note that although the original paper states the proposition as an if and only if, in fact only one of the directions holds. Luckily, this is the direction used in the rest of the paper and which we need to translate our results to ACs.

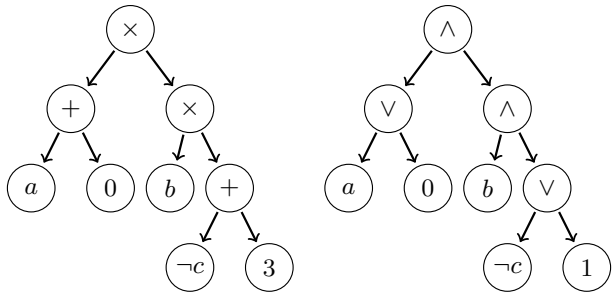

Figure 2: (left) A monotone arithmetic circuit, $\mathcal{C}$. Note that $\langle \mathcal{C} \rangle = (a+0) \times (b \times (\neg c + 3))$. To evaluate $f_{\mathcal{C}}$ for $a = 1, b = 1, c = 0$ we compute $(1+0) \times (1 \times ((1-0)+3)) = 4$. (right) $\phi(\mathcal{C})$, an NNF. Note that $\mathsf{supp}(\mathcal{C}) = \mathsf{sat}(\phi(\mathcal{C}))$.

with $\times$. We call the α_i *parameters*. A PSDD is defined analogously to an SDD but with $+$, $\times$ and X p-decomposition replacing the roles of $\vee, \wedge$ and X decomposition.

Definition 10. *Let T be a v-tree over variables Z. A PSDD respecting T is a AC_m $\mathcal{C}$ with one of the following forms.*

- *$\mathcal{C}$ consists of a single node labelled by a constant, x or $\neg x$, where $x \in Z$.*
- *The source of $\mathcal{C}$ is a $+$ node g and there exists some t, an internal node of T, such that:*
  1. *$\langle \mathcal{C} \rangle = \sum_{i=1}^{n} \alpha_i \times p_i(X) \times s_i(Y)$ where $\{(p_i, s_i, \alpha_i)\}_{i \in [n]}$ is an X p-decomposition for $f_{\mathcal{C}}$,*
  2. *$X \subseteq \mathsf{var}(t_\ell)$, $Y \subseteq \mathsf{var}(t_r)$ and*
  3. *if $h \in \mathcal{C}$ with $\langle \mathcal{C}(h) \rangle = p_i(X)$ (resp. $s_i(Y)$) for some i then $\mathcal{C}(h)$ is a PSDD that respects the subtree of T rooted at t_ℓ (resp. t_r).*

A PSDD is a PSDD respecting some v-tree.

The following is now almost immediate.

Corollary 2. *dSD-AC$_p$ < PSDD.*

Proof. First, observe that $\phi(\mathsf{dSD\text{-}AC_m}) = \mathsf{d\text{-}SDNNF}$. It is not quite true that $\phi(\mathsf{PSDD}) = \mathsf{SDD}$. However, for any $\mathcal{C} \in \phi(\mathsf{PSDD})$, if we propagate away constants corresponding to parameters from $\phi(\mathcal{C})$ we get an equivalent SDD of smaller size. Therefore, $\phi(\mathsf{PSDD}) \geq \mathsf{SDD}$. By Theorem 1, $\phi(\mathsf{dSD\text{-}AC_m}) = \mathsf{d\text{-}SDNNF} < \mathsf{SDD} \leq \phi(\mathsf{PSDD})$ and so by Lemma 6 $\mathsf{dSD\text{-}AC_m} < \mathsf{PSDD}$. Finally, by [de Colnet and Mengel, 2021, Lemma 10] if we take a $\mathsf{dSD\text{-}AC_p}$ and switch the sign of every negative constant we get an equivalent $\mathsf{dSD\text{-}AC_m}$. Therefore, $\mathsf{dSD\text{-}AC_p}$ and $\mathsf{dSD\text{-}AC_m}$ are equally succinct and the result follows. □

Since adding two positive functions yields a positive function we also lift Theorem 5.

Corollary 3. *For every $s \in \mathbb{N}$, there exists positive polynomials f and g which both admit a dSD-AC$_p$ of size s and such that any dSD-AC$_p$ equivalent to $f + g$ has size $s^{\tilde{\Omega}(\log s)}$.*

Proof. Let f and g be as in the statement of Theorem 5. Take any d-SDNNF equivalent to f. Then if we change every $\vee$ to a $+$ and every $\wedge$ to a $\times$ we get a $\mathsf{dSD\text{-}AC_m}$ of the same size which is equivalent to f viewed as a positive polynomial. The same is true for g. Let $\mathcal{C}$ be a $\mathsf{dSD\text{-}AC_m}$ equivalent to $f + g$, again viewed as a positive polynomial. Then $\phi(\mathcal{C})$ is a d-SDNNF for $f \cup g$. Therefore, by Theorem 5, $|\mathcal{C}| = s^{\tilde{\Omega}(\log s)}$. By again applying [de Colnet and Mengel, 2021, Lemma 10] the result follows. □

6 Conclusion and Open Problems

We have shown that d-SDNNF does not admit polynomial time complementation, disjunction or existential quantification and that it is more succinct than SDD. Therefore, there is a trade-off between succinctness and supported transformations in choosing one representation over the other. We have shown a quasi-polynomial separation but have not ruled out the possibility that the gap is exponential.

A tantalising open problem, first raised over twenty years ago [Darwiche and Marquis, 2002], is whether d-DNNF is closed under complementation. We have solved a restricted form of this problem and one could attempt the general case using similar methods. Just as the size of d-SDNNF is related to the best partition communication complexity, the size of d-DNNF is related to *multi-partition* communication complexity [Bova, 2016]. However, adapting the methods from this paper to this setting still appears to be a daunting task.

A Proof of Theorem 5

We will follow a similar strategy to in the proof of Theorem 2 but this time we need to start from a different piece of communication complexity and use *disjoint rectangular covers*.

Formally, if Π is a partition of Z, we say that Π-rectangles $R_1, \ldots, R_k$ are a *disjoint cover* of $S \subseteq \{0,1\}^Z$, if $\bigcup_i R_i = S$ and $R_i \cap R_j = \emptyset$ for all $i \neq j$. For $b \in \{0,1\}$, we define $\mathsf{DCov}_b^{\Pi}(f)$ to be the minimum number of Π-rectangles that partition $f^{-1}(b)$ and $\mathsf{DCov}_b(f) := \min_{\Pi} \mathsf{DCov}_b^{\Pi}$, where the minimum is over all balanced partitions. The following lemma shows that if there is no small disjoint rectangular cover of $\mathsf{sat}(f)$ then f does not admit a small d-SDNNF.

Lemma 7 ([Pipatsrisawat and Darwiche, 2010; Bova *et al.*, 2016].)**.** *If $f\colon \{0,1\}^Z \to \{0,1\}$ admits a d-SDNNF of size s, then $\mathsf{DCov}_1(f) \leq s$.*

The key is the following result which is shown in the proof of [Göös *et al.*, 2022, Theorem 2].

Theorem 6. *For every $k \in \mathbb{N}$, there exists $n = k^{O(1)}$, Boolean function $f, g : \{0,1\}^n \to \{0,1\}$ and a balanced partition Π such that the following properties hold.*

1. *f, g have equivalent unambiguous k-DNFs ψ, ϕ respectively with $2^{\tilde{O}(k)}$ terms.*
2. *$\mathsf{DCov}_1^{\Pi}(f \cup g) = 2^{\tilde{\Omega}(k^2)}$.*

So let f, g, ψ, ϕ be as above. Then, by Theorem 4 and the same argument as in Theorem 2, ψ' and ϕ' have equivalent d-SDNNFs of size $2^{\tilde{O}(k)}$. Therefore, by Lemma 7, it is enough to show that $\mathsf{DCov}_1(\psi' \cup \phi') = 2^{\tilde{\Omega}(k^2)}$. But this follows by essentially the same argument as in the proof of Theorem 2 by applying Claim 1. The only extra thing we need to observe is that the mapping between rectangles we defined in the proof of Theorem 2 preserves disjointness.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 414325841. The author would like to thank Stefan Mengel—for his many helpful suggestions pertaining to the topics of this paper—and the anonymous reviewers—for their suggestions which helped to significantly improve the presentation of the paper.

References

[Amarilli *et al.*, 2020] Antoine Amarilli, Florent Capelli, Mikaël Monet, and Pierre Senellart. Connecting Knowledge Compilation Classes and Width Parameters. *Theory Comput. Syst.*, 64(5):861–914, 2020.

[Balodis *et al.*, 2021] Kaspars Balodis, Shalev Ben-David, Mika Göös, Siddhartha Jain, and Robin Kothari. Unambiguous DNFs and Alon-Saks-Seymour. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 116–124. IEEE, 2021.

[Beame and Liew, 2015] Paul Beame and Vincent Liew. New Limits for Knowledge Compilation and Applications to Exact Model Counting. In Marina Meila and Tom Heskes, editors, *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, The Netherlands*, pages 131–140. AUAI Press, 2015.

[Bollig and Farenholtz, 2021] Beate Bollig and Martin Farenholtz. On the Relation Between Structured d-DNNFs and SDDs. *Theory Comput. Syst.*, 65(2):274–295, 2021.

[Bova and Szeider, 2017] Simone Bova and Stefan Szeider. Circuit Treewidth, Sentential Decision, and Query Compilation. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 233–246. ACM, 2017.

[Bova *et al.*, 2016] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge Compilation Meets Communication Complexity. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1008–1014. IJCAI/AAAI Press, 2016.

[Bova, 2016] Simone Bova. SDDs Are Exponentially More Succinct than OBDDs. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 929–935. AAAI Press, 2016.

[Bryant, 1986] Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A Knowledge Compilation Map. *J. Artif. Intell. Res.*, 17:229–264, 2002.

[Darwiche, 2001a] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.

[Darwiche, 2001b] Adnan Darwiche. On the Tractable Counting of Theory Models and its Application to Truth Maintenance and Belief Revision. *J. Appl. Non Class. Logics*, 11(1-2):11–34, 2001.

[Darwiche, 2011] Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 819–826. IJCAI/AAAI, 2011.

[de Colnet and Mengel, 2021] Alexis de Colnet and Stefan Mengel. A Compilation of Succinctness Results for Arithmetic Circuits. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 205–215, 2021.

[Gogic *et al.*, 1995] Goran Gogic, Henry A. Kautz, Christos H. Papadimitriou, and Bart Selman. The Comparative Linguistics of Knowledge Representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 862–869. Morgan Kaufmann, 1995.

[Göös *et al.*, 2016] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles Are Nonnegative Juntas. *SIAM J. Comput.*, 45(5):1835–1869, 2016.

[Göös *et al.*, 2022] Mika Göös, Stefan Kiefer, and Weiqiang Yuan. Lower Bounds for Unambiguous Automata via Communication Complexity. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 126:1–126:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Kisa *et al.*, 2014] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic Sentential Decision Diagrams. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014.

[Knop, 2017] Alexander Knop. IPS-like Proof Systems Based on Binary Decision Diagrams. *Electron. Colloquium Comput. Complex.*, TR17-179, 2017.

[Kushilevitz and Nisan, 1997] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[Lam and Ruzzo, 1992] Tak Wah Lam and Walter L. Ruzzo. Results on Communication Complexity Classes. *J. Comput. Syst. Sci.*, 44(2):324–342, 1992.

[Pipatsrisawat and Darwiche, 2008] Knot Pipatsrisawat and Adnan Darwiche. New Compilation Languages Based on Structured Decomposability. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 517–522. AAAI Press, 2008.

[Pipatsrisawat and Darwiche, 2010] Thammanit Pipatsrisawat and Adnan Darwiche. A Lower Bound on the Size of Decomposable Negation Normal Form. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 345–350. AAAI Press, 2010.

[Poon and Domingos, 2011] Hoifung Poon and Pedro M. Domingos. Sum-product networks: A new deep architecture. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 689–690. IEEE Computer Society, 2011.

[Segerlind, 2008] Nathan Segerlind. On the Relative Efficiency of Resolution-Like Proofs and Ordered Binary Decision Diagram Proofs. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 100–111. IEEE Computer Society, 2008.

[Shen *et al.*, 2016] Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable Operations for Arithmetic Circuits of Probabilistic Models. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3936–3944, 2016.

[Somenzi, 2009] Fabio Somenzi. CUDD: CU decision diagram package release 2.4. 2. *University of Colorado at Boulder*, 2009.

[Van den Broeck and Darwiche, 2015] Guy Van den Broeck and Adnan Darwiche. On the Role of Canonicity in Knowledge Compilation. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 1641–1648. AAAI Press, 2015.

[Wegman and Carter, 1981] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.