Proof Systems Based on Structured Circuits

Christoph Berkholz
Technische Universität Ilmenau, Germany

Matthäus Micun
Technische Universität Ilmenau, Germany

Abstract

Since their introduction by Atserias, Kolaitis, and Vardi in 2004, proof systems where each line is represented by an ordered binary decision diagram (OBDD) have been intensively studied as they allow to compactly represent Boolean functions. We extend this line of work by considering representation formats that can be even more succinct than OBDDs and have gained a lot of attention in the area of knowledge compilation: sentential decision diagrams (SDDs) and deterministic structured DNNF circuits (d-SDNNFs).

We show that both variants can provide strictly smaller refutations of unsatisfiable CNFs than their OBDD counterparts. Furthermore, we investigate the relative strength of these systems depending on which of the three fundamental derivation rules join, reordering, and weakening are allowed. Here we obtain several separations and identify interesting open problems. To streamline our proofs we establish a sat-to-unsat lifting theorem that might be of independent interest: it turns satisfiable CNFs that are hard to represent by SDDs and d-SDNNFs into unsatisfiable CNFs that are hard to refute in the corresponding proof system.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Proof Complexity, Sentential Decision Diagram, DNNF, OBDD

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.6

Related Version *Full Version*: `https://arxiv.org/abs/2605.12378`

Funding *Christoph Berkholz*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 414325841.
Matthäus Micun: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 414325841.

1 Introduction

Propositional proof systems where every line is represented by an Ordered Binary Decision Diagram (OBDD) have been introduced in 2004 [2] and have been intensively studied since then [7, 19, 11, 10, 13, 6, 12] (see [6] for an overview). One main feature of the approach to take OBDDs instead of simpler objects such as disjunctive clauses is that even complex Boolean functions can be succinctly represented as OBDDs, which then results in more succinct proofs. At the same time, important operations such as equivalence, entailment, and satisfiability tests can be implemented in polynomial time on OBDDs, which is a prerequisite for designing useful and efficiently verifiable derivation rules.

OBDDs also play a fundamental role in *knowledge compilation*, an area pioneered in [9] that studies different representation formats for Boolean functions with a particular focus on the trade-off between *succinctness* and *usefulness*. Motivated by recent advances in that field, the central question we want to answer with this paper is whether there are more succinct representation formats that are still useful enough to serve as a basis for a proof system and whether these systems lead to strictly smaller refutations of unsatisfiable CNFs than OBDD-based systems. We identify two such representation formats.

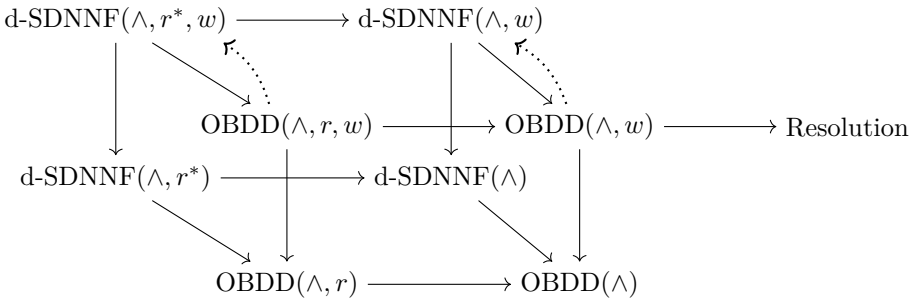

Figure 1 The various OBDD- and d-SDNNF-based proof systems. A solid edge from A to B implies that proof system A polynomially simulates proof system B, that is, A is at least as succinct as B. A dotted edge from A to B indicates that no superpolynomial separation between A and B is known. If there is no path from A to B using solid or dotted edges, then A does not polynomially simulate B. Dotted edges from OBDD$(\wedge, w)$ to d-SDNNF$(\wedge)$ as well as from OBDD$(\wedge, r, w)$ to every d-SDNNF proof system are implied, but left out for legibility.

The first one are *Sentential Decision Diagrams* (SDDs) introduced in [8]. SDDs generalise OBDDs by branching not only on a single variable, but on several variables at the same time. Moreover, while each OBDD respects some linear order on the variables, structure is enforced in SDDs by a *variable tree* (vtree). Similarly to OBDDs, they also have several nice algorithmic properties, such as polynomial time conjunction and negation. On the other hand, SDDs can be strictly more succinct than OBDDs [18, 3]. The second representation format we consider are *deterministic structured circuits in Decomposable Negation Normal Form* (d-SDNNF), which have been introduced in [16]. Here, variables are also structured along a *vtree* and d-SDNNFs generalise SDDs. In fact, it has recently be shown that d-SDNNFs can be strictly more succinct that SDDs [20]. While being more succinct, they also lose some important algorithmic properties; luckily not the ones we need to use as a basis for a proof system.

The main contribution of this paper is to introduce proof systems based on these representation formats and to analyse the relative succinctness compared to their OBDD counterparts and to each other. The derivation rules of the new systems mimic the ones that have been studied for OBDD-based systems: if we have derived D', D'', the join $(\wedge)$ rule allows us to infer the conjunction $D = D' \wedge D''$ provided that D' and D'' respect the same order/vtree; the reordering (r) rule allows us to infer D from an equivalent D' with a different order/vtree; the weakening (w) rule allows us to infer D from D' over the same order/vtree if D' entails D. Since in some cases reordering of vtrees is not known to be polynomial time verifiable, we introduce a *small restructuring rule* (r^*) that is verifiable and modifies the vtree only locally. However, even with this restricted rule, the SDD/d-SDNNF-systems still polynomially simulate their OBDD counterparts with one-step reordering (r). Figure 1 depicts the obtained simulations and separations between OBDD and d-SDNNF-based proof systems. The proof systems based on SDDs are not included for simplicity, as all separations and simulations agree with the d-SDNNF case. In fact, we leave it as open problem, whether d-SDNNF circuits (that can be more succinct than SDDs) prove a strictly more powerful propositional proof system.

Technical contributions. Our first technical contribution is a method for lifting lower bounds on the size of OBDDs/SDDs/d-SDNNFs that represent the models of a satisfiable CNF φ to a lower bound on the proof size for refuting an unsatisfiable CNF $\mathcal{Z}(\varphi)$ in a

corresponding proof system without weakening (Theorem 13). This method can be applied to a broad class of $(k, \log n)$-CNFs, where every clause has at most k literals and every variable appears in at most $\log n$ clauses.

We then use this method to obtain several separations in Section 5. In particular, we show that in the absence of weakening, SDD and d-SDNNF proof systems cannot be polynomially simulated by their OBDD-counterparts and are indeed strictly stronger (Section 5.1). We also apply the lifting method to obtain separations between different variants of these proof systems. These separations have been known for OBDD proof system (see [6] for an overview) and the challenge is to show that the lower bound constructions are also hard for the stronger SDD/d-SDNNF format. Besides applying the lifting method (Theorem 13), we also use the connection between structured circuits and rectangle covers in communication complexity (see [5]) to establish the bounds. In particular we show:

- d-SDNNF$(\wedge, r)$ does not polynomially simulate OBDD$(\wedge, w)$ (Section 5.2)
- d-SDNNF$(\wedge, w)$ does not polynomially simulate OBDD$(\wedge, r)$ (Section 5.3)
- d-SDNNF$(\wedge)$ does not subexponentially simulate OBDD$(\wedge, r)$ (Section 5.4)

The possibility of even stronger DNNF proof systems. SDDs and d-SDNNF can be generalised even further. In particular, the structured decomposable negation normal form (SDNNF) generalises d-SDNNF by dropping the *determinism* and has the potential to be more compact. In that sense, it might seem useful to also define proof systems using this stronger class. In fact, there has already been some related work on derivation systems using SDNNFs [10] in the context of bottom-up knowledge compilation. However, neither weakening nor reordering are polynomial-time verifiable for SDNNF, even if both SDNNF respect the same vtree [16] and even worse, as every DNF is a SDNNF it is coNP-hard to decide whether they are equivalent and hence $(\wedge)$ isn't polynomial time verifiable as well. Therefore, systems based on SDNNF (with some of the derivation steps $\wedge, r, w$) are not propositional proof systems and a similar reasoning applies to d-DNNF (omitting the *structure*, see Section 3). In that sense, d-SDNNF is the most general DNNF-circuit variant that is suitable for defining a propositional proof system.

2 Preliminaries

Boolean functions. A *Boolean function* is a function $f\colon \{0,1\}^X \to \{0,1\}$ for some set of *variables* X. An *assignment* for f is a function $\mathcal{I} : X \to \{0,1\}$. A literal γ is either a variable x, or its negation $\neg x$. A *clause* C is a disjunction of literals (also considered as set of literals), and a formula φ in *conjunctive normal form* (CNF) is a conjunction of clauses, where no clause contains a variable x and its negation. A (k, ℓ)-*CNF* is a CNF where every clause has at most k literals, and every variable appears in at most ℓ clauses. For a CNF φ, we let clause(φ) be the set of clauses in φ, var(φ) be the set of variables mentioned in φ, and $f_\varphi : \{0,1\}^{\mathrm{var}(\varphi)} \to \{0,1\}$ be the Boolean function associated with φ.

We write $\mathcal{I} \models \varphi$ to denote that an assignment $\mathcal{I}$ satisfies a CNF φ. A *partial assignment* for a Boolean function $f\colon \{0,1\}^X \to \{0,1\}$ is a function $\alpha : S \to \{0,1\}$ for some $S \subset X$. We also call S the *support* of α and denote it with supp(α). For two partial assignments α, β with disjoint supports S, T we let $\alpha \cup \beta : S \cup T \to \{0,1\}$ be the partial assignment that agrees with α on S and with β on T. If α is a partial assignment of f, then the *restriction* of f to α is Boolean function $f|_\alpha : \{0,1\}^{X \setminus S} \to \{0,1\}$ with $f|_\alpha(\beta) := f(\alpha \cup \beta)$. Observe that for any assignment β of $f|_\alpha$ it holds that $f|_\alpha(\beta) = f|_\beta(\alpha)$.

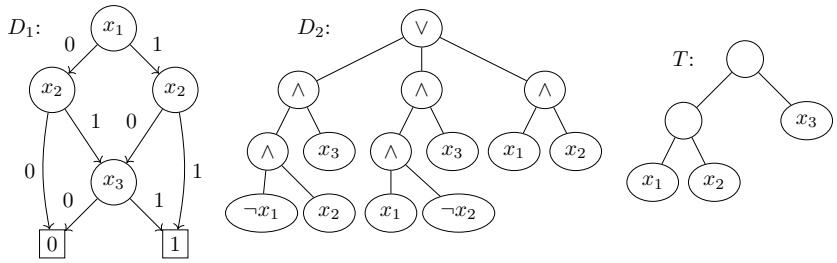

Figure 2 An OBDD D_1, and a d-SDNNF D_2 computing the majority function on variables $\{x_1, x_2, x_3\}$. The d-SDNNF D_2 respects the vtree T.

Graphs. A graph $G = (V, E)$ consists of a set V of *vertices* and a set E of undirected *edges*. We use $G[V']$ to denote the *induced subgraph* $G' = (V', (E \cap \binom{V'}{2}))$. For a graph $H = (V', E')$, we use $H \subseteq G$ to denote that H is a subgraph of G. For a given $v \in V$ we let the *neighbourhood* $N(v)$ of v be the set of vertices adjacent to v, and $E(v)$ be the set of edges incident to v. For a set $S \subseteq V$, we let $N(S)$ be the set of vertices $v \in V \setminus S$ that are adjacent to some $w \in S$, $E(S)$ be the set of edges in $G[S]$ and $\partial(S)$ be the set of edges that are incident to *exactly one* $v \in S$. We may also use $V(G)$ and $E(G)$ to describe the vertex set and edge set of G. A *tree decomposition* of a graph G is a pair (T, β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$ a function such that

1. For all $e \in E(G)$ there is a $t \in V(T)$ such that $e \in \beta(t)$
2. For all $v \in V(G)$, $\beta^{-1}(v)$ is connected.

The *width* of a tree decomposition (T, β) is $w = \max_{t \in V(T)}(|\beta(t)| - 1)$ and the *treewidth* $tw(G)$ of G is the minimum width over all tree decompositions of G. The *treewidth of a CNF* φ, denoted by $tw(\varphi)$ is the treewidth of the primal graph G_φ, where $V(G_\varphi)$ is the set of variables in φ, and $\{u, v\} \in E(G_\varphi)$, if the variables associated with u, and v share a clause.

Expander graphs are a very useful ingredient in lower bound constructions: for $d \geq 3$ and $0 < c < 1$ an (n, d, c)-*expander* is a d-regular n-vertex graph $G = (V, E)$ such that $|N(S)| \geq c \cdot |S|$ for every $S \subset V$ with $|S| \leq \frac{n}{2}$. A family $(G_n)_{n \in \mathbb{N}}$ is called a *family of* (d, c)-*expanders*, if every G_n is an (n, d, c)-expander. It is well established that for $d \geq 3$ and $0 < c < 1$ there are families of (d, c)-expanders (see e. g. [1]).

2.1 Knowledge Compilation

One of the main aims of *knowledge compilation* is to provide succinct representation formats for Boolean functions that allow efficient operations ranging from counting and enumerating satisfying assignments to testing equivalence and entailment [9]. In this paper we focus on *structured* representation formats, in particular ordered binary decision diagrams (OBDDs), sentential decision diagrams (SDDs) and structured deterministic decomposable circuits (d-SDNNF). In the following paragraphs we provide a brief introduction.

Ordered Binary Decision Diagrams. A *binary decision diagram* (BDD) is a directed acyclic graph D with exactly one source and two sinks, labelled 0-sink, and 1-sink, respectively. Each internal vertex v of D has exactly two children. The outgoing edges of v are labelled 0-edge and 1-edge, and the children of v are referred to as 0-child, and 1-child, respectively. Furthermore, D is equipped with a labelling function $\ell : V(D) \to X$, where $V(D)$ is the set of internal vertices of D, and X some set of variables. The size $|D|$ of a BDD D is the number of vertices in $V(D)$. A BDD represents a Boolean function $f_D : \{0,1\}^X \to \{0,1\}$

such that for any assignment $\mathcal{I} : X \to \{0,1\}$, we can construct a path from the source s to a sink t, which respects the assignment $\mathcal{I}$: if an internal vertex v along the path is labelled with variable a, then the next vertex in the path will be the $\mathcal{I}(a)$-child of v. Then $f_D(\mathcal{I}) = 1$, if the path defined by $\mathcal{I}$ on D ends in the 1-sink.

Let $\leq$ be a linear order on variables X. We say, a BDD D *respects* the order $\leq$, if for every internal edge (v, w) it holds that $\ell(v) < \ell(w)$. An *ordered binary decision diagram* (OBDD) is a BDD that respects some order $\leq$. We may also use the term $\leq$-OBDD, if the order is relevant. For any order $\leq$ and Boolean function f, there is a canonical OBDD D respecting $\leq$ such that $f = f_D$. For an OBDD D and partial assignment α, we let the *partial restriction* $D|_\alpha$ of D to α be the OBDD that results from restricting decisions according to α and that computes $f_D|_\alpha$.

Throughout this paper, we use the following well-known properties of OBDDs:

▶ **Lemma 1** ([21]). *The following properties hold for* OBDD*s:*

1. *There is a polynomial-time algorithm that computes for two* OBDD*s D_1 and D_2 respecting the same linear order $\leq$ an* OBDD *D respecting $\leq$ such that $D \equiv D_1 \wedge D_2$.*
2. *There exists a polynomial-time algorithm that computes for an* OBDD *D, whether $f_D \equiv \bot$*
3. *Given two* OBDD*s D_1 and D_2 respecting the same linear order $\leq$, it can be checked in polynomial time whether $D_1 \models D_2$.*

Variable trees. The other two representation formats are not structured along an order as OBDDs, but along a variable tree (vtree). A *vtree* over a variable set X is a pair (T, b), where T is a rooted full binary tree and b is a bijection from the leaves of T to X. Furthermore, the two children of each internal vertex are labelled as left and right child. For an internal vertex v, we let T_v be the subtree rooted at v and $T_{v,l}$, $T_{v,r}$ be the subtrees rooted at the left and right children of v, respectively. For a subtree T' we denote by $b(T')$ be the image of b restricted to the leaves of T'.

Sentential Decision Diagrams. The central idea of sentential decision diagrams is that they do not branch over a single variable $x \in X$, but on multiple variables $Y \subseteq X$ at the same time while grouping some of the $2^{|Y|}$ assignments together. To define this formally, let an *Y-partition* be a set $\{f_1, \ldots, f_k\}$ of Boolean functions on Y such that

- All f_i are satisfiable ($f_i^{-1}(1) \neq \emptyset$).
- The set $\{f_i^{-1}(1) \mid i \in [k]\}$ is a partition of $\{0,1\}^Y$.

A *Sentential Decision Diagram (SDD) respecting a vtree* (T, b) on variables X is defined recursively as follows:

- $\top$ and $\bot$ are SDDs with $f_\top = \top$ and $f_\bot = \bot$.
- x and $\neg x$ are SDDs that respect the vtree consisting of a single leaf v with $b(v) = x$. They compute $f_x = x$ and $f_{\neg x} = \neg x$, respectively.
- Let (T, b) be a vtree with root v. Suppose that $F_1, \ldots, F_k$ are SDDs each respecting some subtree of $T_{v,l}$ such that $\{f_{F_1}, \ldots, f_{F_k}\}$ is a $b(T_{v,l})$-partition. Further suppose that $G_1, \ldots, G_k$ are SDDs each respecting some subtree of $T_{v,r}$. Then $\alpha := \{(F_1, G_1), \ldots, (F_k, G_k)\}$ is an SDD that respects (T, b). Intuitively, the SDD branches over the assignment groups specified by the F_i. Formally, $f_\alpha := \bigvee_{i \in [k]} (f_{F_i} \wedge f_{G_i})$.

The size $|D|$ of an SDD D is the number of gates in D, if D is viewed as a *circuit* (which is the number of distinct sub-SDD used in the recursive definition).

Let D be an SDD respecting a vtree (T, b) rooted at a node v. D is called *normalised*, if either v is a leaf, or if $D = \{(F_1, G_1), \ldots, (F_k, G_k)\}$, where each F_i is normalised and respects the vtree $T_{v,l}$, and each G_i is normalised and respects the vtree $T_{v,r}$.

As OBDDs, SDDs have desirable closure properties:

▶ **Theorem 2** ([8]). *Let D_1, D_2 be normalised* SDD*s respecting the same vtree (T, b). Then there exist polynomial-time algorithms that compute $D_1 \wedge D_2$, $D_1 \vee D_2$, and $\neg D_1$. Furthermore, it is possible to verify in polynomial time whether $D_1 \equiv D_2$.*

It follows directly that if D_1, D_2 are SDDs respecting the same vtree (T, b), then verifying whether $f_{D_1}^{-1}(1) \subseteq f_{D_2}^{-1}(1)$, that is, whether D_1 implies D_2, can be done in polynomial time. Note that every SDD D can be transformed into a normalised SDD D' in polynomial time [8]. Thus, we can also omit the condition that D_1 and D_2 are normalised from Theorem 2.

Structured DNNFs. Structured circuits generalise SDDs. They allow even more succinct representations, but lose some desirable closure properties. A *Boolean circuit* C over a variable set X is a directed, acyclic graph with a unique source v_s. The internal vertices v of C are labelled by some gate $g_v \in \{\wedge, \vee, \neg\}$; if $g_v \in \{\vee, \wedge\}$, then v has exactly two children, and if $g_v = \neg$, then v has exactly one child. The leaves of C are marked with variables in X. For a vertex v of C we let $\mathrm{var}(v)$ be the set of variables mentioned at leaves below v and $f_v \colon \{0,1\}^{\mathrm{var}(v)} \to \{0,1\}$ be the Boolean function computed at v. The size $|C|$ of a circuit C is the number of gates (inner vertices) in C. A circuit C is in *negation normal form* (NNF), if all $\neg$-gates of C appear directly above leaves. Furthermore, an NNF C is called *decomposable* (DNNF), if for all $\wedge$-gates v with children u_1, u_2 it holds that $\mathrm{var}(u_1) \cap \mathrm{var}(u_2) = \emptyset$. As SDDs, DNNFs can also be structured along a vtree. A DNNF C is a *structured* DNNF (SDNNF) if it respects a vtree (T, b) in the following way:

- For every $\wedge$-gate v with children u_1, u_2, there is an $s \in V(T)$ such that $\mathrm{var}(u_1) \subseteq b(T_{s,l})$ and $\mathrm{var}(u_2) \subseteq b(T_{s,r})$.
- For every $\vee$-gate v with children u_1, u_2, there is an $s \in V(T)$, such that $\mathrm{var}(s) = \mathrm{var}(u_1) = \mathrm{var}(u_2) = b(T_s)$.

Finally, a DNNF is called a *deterministic* DNNF (d-DNNF), if for every $\vee$-gate with children u, w no assignment satisfies f_u and f_w at the same time. If we require determinism for structured DNNF we get d-SDNNFs, which in turn generalise SDDs:

▶ **Observation 3.** *Let S be an* SDD *that respects a vtree (T, b). Then there exists a* d-SDNNF *D computing $f_D = f_S$ that respects (T, b) and has size $\mathcal{O}(|S|)$.*

In the other direction, it has recently be shown that d-SDNNFs can be strictly more succinct than SDDs, but as a downside they lose polynomial-time disjunction and negation [20]. However, we still have the following:

▶ **Lemma 4.** *[16] There is an algorithm that computes for two* d-SDNNF*s D_1, D_2, that respect the same vtree (T, b), a* d-SDNNF *for $f_{D_1} \wedge f_{D_2}$ in time $\mathcal{O}(|D_1| \cdot |D_2|)$.*

Moreover, for every deterministic DNNF (hence also d-SDNNFs and SDDs) we can count the number of models in polynomial time (see [9]). We will use this at several places and write $\#D$ as an abbreviation for $|f_D^{-1}(1)|$.

3 Proof system based on structured representations

An OBDD-based proof system over a variable set X is a derivation system where each line consists of an order $\leq$ over X and an OBDD D that respects $\leq$. An OBDD$(\wedge, r, w)$-refutation of an unsatisfiable CNF $\varphi = \bigwedge_{j \in [m]} C_j$ is a derivation $((D_1, \leq_1), \ldots, (D_\ell, \leq_\ell))$ of an OBDD

D_ℓ that represents $f_{D_\ell} = \bot$ where each line $(D_i, \leq_i)$ is derived according to one of the following rules:

init	$f_{D_i} = f_{C_j}$ for some clause C_j in φ.
join ($\wedge$)	There are $j, j' < i$ such that $\leq_i = \leq_j = \leq_{j'}$ and $f_{D_i} = f_{D_j} \wedge f_{D_{j'}}$.
reordering (r)	There is a $j < i$ such that $f_{D_i} = f_{D_j}$ (but $\leq_i \neq \leq_j$).
weakening (w)	There is a $j < i$ such that $\leq_i = \leq_j$ and $f_{D_j}^{-1}(1) \subseteq f_{D_i}^{-1}(1)$.

It is well known that all derivation rules can be verified in polynomial time (see [21]). We use the common convention and denote by OBDD($\wedge, r$), OBDD($\wedge, w$), and OBDD($\wedge$) the subsystem that use only the stated derivation rules. Note that OBDD($\wedge$) is already a sound and complete proof system for UNSAT.

To define the new proof systems SDD($\wedge, r, w$) and d-SDNNF($\wedge, r, w$) as well as their restrictions, we literally only have to replace "OBDD" by "SDD" or "d-SDNNF" and "order $\leq$" by "vtree (T, b)" in the definition above. However, we need to carefully check whether they actually produce polynomially verifiable proofs! In particular, to the best of our knowledge it is not known whether the reordering rule stated as above is polynomially verifiable for SDDs or d-SDNNFs. To be more precise, given two SDDs D_1, D_2 with vtrees $(T_1, b_1) \neq (T_2, b_2)$ it is unknown whether there exists a polynomial-time algorithm verifying whether $f_{D_1} = f_{D_2}$ and the same holds for d-SDNNFs. Before discussing this further, let's verify that the all other properties are fulfilled.

▶ **Lemma 5.** *The following properties hold for* d-SDNNF*s and hence, for* SDD*s.*

1. *Given a circuit D and a vtree (T, b) it can be verified in polynomial time whether D is an* d-SDNNF *that respects (T, b).*
2. *Given* d-SDNNF*s D_1, D_2, D_3 that respect the same vtree, it can be checked in polynomial time whether $f_{D_1} \wedge f_{D_2} = f_{D_3}$.*
3. *Given* d-SDNNF*s D_1, D_2 that respect the same vtree, it can be checked in polynomial time whether $f_{D_1}^{-1}(1) \subseteq f_{D_2}^{-1}(1)$.*
4. *Given a* d-SDNNF*s D, it can be checked in polynomial time whether $f_D = \bot$.*

A proof for Lemma 5 can be found in the full version.

▶ **Corollary 6.** SDD($\wedge$), SDD($\wedge, w$), d-SDNNF($\wedge$), *and* d-SDNNF($\wedge, w$) *are sound and complete propositional proof systems.*

It turns out that without weakening, reordering steps can be verified in polynomial time.[1]

▶ **Lemma 7.** SDD($\wedge, r$) *and* d-SDNNF($\wedge, r$) *are sound and complete propositional proof systems.*

Proof. We prove the lemma for d-SDNNF($\wedge, r$), the statement for SDDs follows immediately. The *clausal entailment problem* asks whether a representation D entails a clause C ($D \models C$), i. e., whether $f_D^{-1}(1) \subseteq f_C^{-1}(1)$. The key of the proof is that by [16] we know clausal entailment for d-SDNNFs is decidable in polynomial time.

Let φ be a CNF and consider a d-SDNNF($\wedge, r$)-derivation over φ. Since reordering does not change the function computed by a d-SDNNF, it follows by induction that each d-SDNNF D in the derivation computes a CNF $f_D = \varphi_D$ for some $\varphi_D \subseteq \varphi$. To prove the lemma, we only need to show that reordering steps can be verified in polynomial time. If $(D', (T', b'))$ has been derived from $(D, (T, b))$ by a reordering step, we can verify that $f_{D'} = f_D = \varphi_D$ as follows.

[1] We thank the anonymous reviewer of an earlier version for pointing this out.

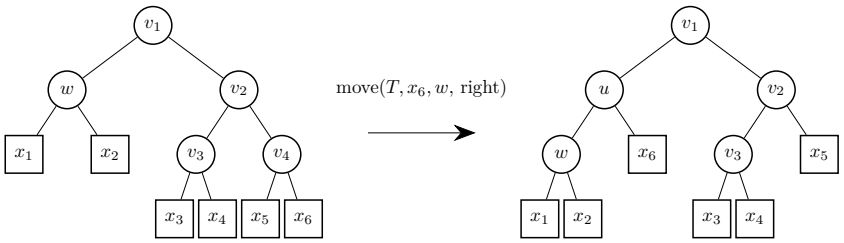

Figure 3 The result of using $\texttt{move}(T, x_6, w, \text{right})$ on the left vtree.

1. Check, whether D and D' have the same number of models.
2. Check, whether $D' \models C$ for all clauses $C \in \varphi_D$.

Since both tests can be implemented in polynomial time and together they are equivalent to the test whether $f_{D'} = \varphi_D$, the lemma follows. ◀

Unfortunately, this argument does not work for d-SDNNF$(\wedge, r, w)$, as weakening prevents us from constructing φ_D. To circumvent this issue, we introduce a *small restructuring rule* that allows to locally modify the vtree. Interestingly, it follows from known results that OBDD proof system that only locally modify the order are equivalent to the ones with unrestricted reordering. For completeness, we provide a proof in the full version.

▶ **Lemma 8.** *Let D and D' be two equivalent* OBDD*s on n variables that respect the orders $\leq$ and $\leq'$, respectively. There is a sequence $(D, \leq) = (D_1, \leq_1), \ldots, (D_\ell, \leq_\ell) = (D', \leq')$ with $\ell \leq n$ and $f_{D_1} = \cdots = f_{D_\ell}$ such that*

1. *For each $2 \leq i \leq \ell$ the order $\leq_i$ is obtained from $\leq_{i-1}$ by moving a single variable to a different position.*
2. *For each $1 \leq i \leq \ell$: $|D_i| \leq |D| \cdot |D'|$.*

We extend the notion of swapping two positions in an order to locally restructuring a vtree as follows. Let (T, b) be a vtree, $x \in X$ some variable, $v \in V(T)$ the unique leaf node such that $b(v) = x$, $w \in V(T) \setminus \{v\}$ another node in T, and $d \in \{\text{left}, \text{right}\}$. We define the operation $\texttt{move}(T, x, w, d)$ as follows:

1. Remove the edge connecting v to its parent p_v:
   - If p_v is the root of T, remove it.
   - If p_v is not a root, then the vertex has a unique parent s_1, and a second child s_2. Replace p_v with the edge $\{s_1, s_2\}$.
2. Add the vertex v back to the tree:
   - If w is a root, add a new root u with children v and w. The vertex v becomes the d-child of u.
   - Otherwise, let p_w be the parent of w. Subdivide the edge $\{p_w, w\}$. The new vertex is called u. Add an edge $\{u, v\}$ such that v is the d-child of u.

▶ **Definition 9** (small restructuring rule r^*)**.** *In* SDD- *and* d-SDNNF*-proofs the* small restructuring rule r^* *allows to derive $(D', (T', b'))$ from $(D, (T, b))$ if $f_{D'} = f_D$ and (T', b') has been obtained from (T, b) by an instantiation of a* $\texttt{move}(T, x, w, d)$ *operation.*

We say that a proof system A *polynomially simulates* (p-simulates) a proof system B, if there is a polynomial p such that for every unsatisfiable CNF φ and B-refutation of φ of size T there is an A-refutation of φ of size $p(T)$.

▶ **Theorem 10.** SDD$(\wedge, w, r^*)$ *and* d-SDNNF$(\wedge, w, r^*)$ *are sound and complete propositional proof systems that polynomially simulate* OBDD$(\wedge, w, r)$.

Proof. First observe that OBDDs can be viewed as SDDs respecting *right-linear* vtrees, which are vtrees where each left child is a leaf, and that moving a single variable in an order can be simulated by one `move`-operation in the right-linear vtree. Thus, by Lemma 8, SDD$(\wedge, w, r^*)$ and hence d-SDNNF$(\wedge, w, r^*)$ p-simulate OBDD$(\wedge, w, r)$.

It remains to show that small restructuring for SDDs (d-SDNNFs, resp.) can be verified in polynomial time. Assume that (T', b') is obtained from (T, b) by the operation $\texttt{move}(T, x, w, d)$, and let (T'', b'') be obtained by adding a dummy variable d to (T', b') at the old position of x in (T, b). First, we construct two SDDs (d-SDNNFs) $D_0 \coloneqq D|_{x=0}$ and $D_1 \coloneqq D|_{x=1}$. We observe that D_0 and D_1 respect *both* (T, b) and (T'', b''), as the two vtrees only differ in the position of x, and the presence of d, neither of which are mentioned by either representation. We further observe that D' respects both (T', b') and (T'', b'') for similar reasons, and that D_0 and D_1 each have size at most S.

We then construct $D'_0 \coloneqq D_0 \wedge (\neg x)$ as well as $D'_1 \coloneqq D_1 \wedge (x)$ that respect (T'', b'') by Theorem 2 (Lemma 4). Moreover, again by applying Lemma 2 (Lemma 4) we construct $D''_0 \coloneqq D'_0 \wedge D'$ and $D''_1 \coloneqq D'_1 \wedge D'$. Since the models of D are the disjoint union of the models of D'_0 and D'_1, testing whether $f_D = f_{D'}$ is equivalent to testing whether $\#D'_0 = \#D''_0$, $\#D_1 = \#D''_1$, and $\#D'_0 + \#D'_1 = \#D'$. ◀

Finally, let us briefly discuss the possibility of proof systems based on even stronger representation formats. The first to consider are generalisations of d-SDNNF, while the second are so-called *free binary decision diagrams* (FBDDs), which generalise OBDDs by dropping the requirement that edges need to respect a linear order. While these representation formats may be more compact, it turns out that verifying even the join rule is NP-hard for both d-DNNF [16], as well as FBDD [9]. It turns out that SDNNF$(\wedge)$ is not really suited as a proof system either: while it is possible, given two SDNNF D_1, D_2 respecting the same vtree, to compute in polynomial time a third D such that $D \equiv D_1 \wedge D_2$, SDNNF does not in general support polynomial-time equivalence checking, unless P=NP [16]. This result can easily be further extended to verification of the join rule, that is *verifying* whether $D \equiv D_1 \wedge D_2$ for some SDNNFs D_1, D_2, D respecting the same vtree. Therefore, it is not useful to define proof systems using these stronger representation formats.

4 Lifting from satisfiable CNFs

Our main goal is to separate proof systems relying on different representation formats. While the relative succinctness of different structured circuits that represent Boolean functions (with many models) is well-studied, the main obstacle is that we need to work with unsatisfiable CNFs that have a trivial representation and at the same time are hard to refute.

To deal with this, we establish in this section a "lifting method" and show that certain satisfiable CNFs φ can be transformed into an unsatisfiable $\mathcal{Z}(\varphi)$ such that superpolynomial lower bounds on $\mathfrak{C}$-representations for φ imply superpolynomial lower bounds for $\mathfrak{C}(\wedge, r)$-refutations of $\mathcal{Z}(\varphi)$ for $\mathfrak{C} \in \{\text{OBDD}, \text{SDD}, \text{d-SDNNF}\}$. This method is inspired by and generalises a related approach for OBDD proof systems from [11]. In the next section we then focus on applying this lifting method to obtain our separations.

We call an unsatisfiable CNF $\varphi = \bigwedge_{i\in[m]} C_i$ *minimally unsatisfiable* if it does not have unnecessary clauses, i. e. $\varphi = \bigwedge_{i\in[m]\setminus\{j\}} C_i$ is satisfiable for every $j \in [m]$.

Before we define the lifting operation properly, we first give a brief intuition: let $\varphi \coloneqq \bigwedge_{i\in[m]} C_i$ be a CNF. We start by adding to each clause C_i a fresh *control variable* y_i. This allows us to "turn off" C_i by assigning y_i positively, whenever necessary. Finally, we add new

conditions of the form $(C_i \vee y_i) \to (z_i \to z_{i+1})$ along with z_1 and $\neg z_{m+1}$. This means that, if all clauses $C_i \vee y_i$ hold, then the implication chain simplifies to a contradiction. However, as all clauses need to be satisfied, the transformed CNF is unsatisfiable. Formally, the lifting operation transforms φ to $\mathcal{Z}(\varphi)$ as follows:

▶ **Definition 11.** *Let $\varphi = \bigwedge_{i\in[m]} C_i$ be a* CNF *over $X = \{x_1, \ldots, x_n\}$. We define $\mathcal{Z}(\varphi)$ as*

$$\mathcal{Z}(\varphi) \coloneqq \left(\bigwedge_{i\in[m]} (C_i \vee y_i)\right) \wedge \left(\bigwedge_{i\in[m]} \bigwedge_{\lambda\in C_i\cup\{y_i\}} (\neg\lambda \vee \neg z_i \vee z_{i+1})\right) \wedge z_1 \wedge \neg z_{m+1}, \quad (1)$$

where $Y = \{y_1, \ldots, y_m\}$ and $Z = \{z_1, \ldots, z_{m+1}\}$ are fresh variables.

Note that $\bigwedge_{\lambda_j\in C_i\cup\{y_i\}} (\neg\lambda_j \vee \neg z_i \vee z_{i+1})$ is equivalent to $(C_i \vee y_i \to (z_i \to z_{i+1}))$. It is therefore easy to see that $\mathcal{Z}(\varphi)$ is always unsatisfiable.

▶ **Lemma 12.** *Let $\varphi = \bigwedge_{i\in[m]} C_i$ be a* CNF*. Then $\mathcal{Z}(\varphi)$ is minimally unsatisfiable.*

Proof. Let $C \in \mathcal{Z}(\varphi)$ be some clause in $\mathcal{Z}(\varphi)$ and $\Phi' \coloneqq \mathcal{Z}(\varphi) \setminus \{C\}$ the remaining CNF. We have to show that Φ' is satisfiable and consider the following cases:

1. $C = C_\ell \vee y_\ell$ for some $\ell \in [m]$. Any assignment $\mathcal{I}$ with $\mathcal{I}(y_\ell) = 0$, $\mathcal{I}(y_i) = 1$ for $i \neq \ell$, $\mathcal{I}(z_i) = 1$ for $i \leq \ell$, and $\mathcal{I}(z_i) = 0$ for $i > \ell$ satisfies Φ'.
2. $C = (\neg y_\ell \vee \neg z_\ell \vee z_{\ell+1})$. Any assignment $\mathcal{I}$ that falsifies all literals from C_ℓ and sets $\mathcal{I}(y_i) = 1$ for $i \in [m]$, $\mathcal{I}(z_i) = 1$ for $i \leq \ell$, and $\mathcal{I}(z_i) = 0$ for $i > \ell$ satisfies Φ'.
3. $C = (\neg\lambda \vee \neg z_\ell \vee z_{\ell+1})$ for $\lambda \in C_\ell$. Let $\mathcal{I}$ be an assignment that satisfies λ, falsifies all literals from $C_\ell \setminus \{\lambda\}$ and sets $\mathcal{I}(y_\ell) = 0$, $\mathcal{I}(y_i) = 1$ for $i \in [m] \setminus \{\ell\}$, $\mathcal{I}(z_i) = 1$ for $i \leq \ell$, and $\mathcal{I}(z_i) = 0$ for $i > \ell$. Then $\mathcal{I}$ satisfies Φ'.
4. $C = z_1$. Any $\mathcal{I}$ with $\mathcal{I}(y_i) = 1$ and $\mathcal{I}(z_i) = 0$ for all i satisfies Φ'.
5. $C = \neg z_m$. Any $\mathcal{I}$ with $\mathcal{I}(y_i) = 1$ and $\mathcal{I}(z_i) = 1$ for all i satisfies Φ'. ◀

Recall from the preliminaries that in an (k, ℓ)-CNF every clause contains at most k variables and every variable appears in at most ℓ clauses.

▶ **Theorem 13.** *Let $\mathfrak{C} \in \{\text{OBDD}, \text{SDD}, \text{d-SDNNF}\}$, and φ be an n-variable $(k, \log n)$-CNF. If there is a $\mathfrak{C}(\wedge, r)$-refutation of $\mathcal{Z}(\varphi)$ of size $t(n)$, then there is a $\mathfrak{C}$-representation of φ of size $\mathcal{O}(t(n)^2 \cdot n^{2k^2})$.*

Proof. To avoid notational clutter we focus on $\mathfrak{C} = \text{d-SDNNF}$, the proof for SDDs and OBDDs (with *order* instead of *vtree*) is similar. Consider a d-SDNNF-refutation of $\mathcal{Z}(\varphi)$ and w. l. o. g. assume that in the last step $(D_\ell, (T, b))$ with $D_\ell \equiv \bot$ has been derived from $(D_p, (T, b))$ and $(D_q, (T, b))$ with $D_p \not\equiv \bot$ and $f_{D_q} \not\equiv \bot$ via the $\wedge$-rule. To prove the theorem we show that there are two partial assignments α_p, α_q and a d-SDNNF D^* of size $\mathcal{O}(n^{2k^2})$ that respects (T, b) such that

$$\varphi \equiv D_p|_{\alpha_p} \wedge D_q|_{\alpha_q} \wedge D^* \quad (2)$$

The theorem then follows from Lemma 4 (Lemma 2 for SDD and Lemma 1 for OBDD).

As in the proof of Lemma 7 we associate to each d-SDNNF D in the derivation the CNF $\psi_D \subseteq \mathcal{Z}(\varphi)$ with $D \equiv \psi_D$. Since $D_p \not\equiv \bot$ and $D_q \not\equiv \bot$, let $C_p \in \mathcal{Z}(\varphi) \setminus \psi_{D_p}$ and $C_q \in \mathcal{Z}(\varphi) \setminus \psi_{D_q}$ be two missing clauses.

Our goal is to construct partial assignments α_p, α_q such that we can satisfy all implication clauses without removing too many of the original clauses. We now define α_p by a case distinction on C_p; α_q is obtained analogously. While parsing the definition, the reader is

asked to verify that $(\mathcal{Z}(\varphi) \setminus C_p)|_{\alpha_p} \subseteq \varphi$ as an important feature of the construction (besides removing the auxiliary variables) is, that the restriction satisfies some clauses of φ, but does not produce sub-clauses.

1. $C_p = C_\ell \vee y_\ell$ for some $\ell \in [m]$. We let α_p falsify all literals in C_ℓ and leave the remaining x_i unassigned. For the control variables y_i we set $\alpha_p(y_i) := 1$ if $i \in \{j \mid \mathrm{var}(C_j) \cap \mathrm{var}(C_\ell) \neq \emptyset\} \setminus \{\ell\}$, and $\alpha_p(y_i) := 0$ in all other cases. Furthermore, set $\alpha_p(z_i) := 1$ for $i \leq \ell$, and $\alpha_p(z_i) := 0$ for $i > \ell$.
2. $C_p = (\neg\lambda \vee \neg z_\ell \vee z_{\ell+1})$ for $\lambda \in C_\ell$. We let α_p satisfy λ, falsify all other literals in C_ℓ, and leave the remaining x_i unassigned. As in case 1, we set $\alpha_p(y_i) := 1$ if $i \in \{j \mid \mathrm{var}(C_j) \cap \mathrm{var}(C_\ell) \neq \emptyset\} \setminus \{\ell\}$, and $\alpha_p(y_i) := 0$ in all other cases. Again, we set $\alpha_p(z_i) := 1$ for $i \leq \ell$, and $\alpha_p(z_i) := 0$ for $i > \ell$.
3. $C_p = (\neg y_\ell \vee \neg z_\ell \vee z_{\ell+1})$. We leave all x_i unassigned and set $\alpha_p(y_i) := 0$ for all $i \in [m]$, $\alpha_p(z_i) := 1$ for $i \leq \ell$, and $\alpha_p(z_i) := 0$ for $i > \ell$.
4. $C_p = z_1$. Set all $\alpha_p(y_i) := 0$, all $\alpha_p(z_i) := 0$, and leave all x_i unassigned.
5. $C_p = \neg z_m$. Set all $\alpha_p(y_i) := 0$, all $\alpha_p(z_i) := 1$, and leave all x_i unassigned.

Since by Lemma 12 $\mathcal{Z}(\varphi)$ is minimally unsatisfiable and $D_p \wedge D_q \equiv \bot$ we have that every clause of $\mathcal{Z}(\varphi)$ is contained in the clause set $\psi_{D_p} \cup \psi_{D_q}$. Moreover, by construction α_p (α_q) falsifies no clause of ψ_{D_p} (ψ_{D_q}). Let us estimate how many clauses of the form $C_i \vee y_i$ are satisfied. In cases 3-5 none of them is satisfied. In cases 1 and 2 we satisfy all clauses $C_j \vee y_j$ (via the control variable y_j) that share a variable with C_ℓ. Since φ is a $(k, \log n)$-CNF, each of the at most k variables in C_ℓ appears in at most $\log n$ clauses. Hence, at most $k \cdot \log n$ such clauses are satisfied by α_p and let $I_p \subseteq [m]$ be the set of their indices. As the same holds for q, it follows that $\varphi \equiv D_p|_{\alpha_p} \wedge D_q|_{\alpha_q} \wedge \bigwedge_{i \in I_p \cup I_q} C_i$. To finalise the proof of (2) and hence of Theorem 13, observe that $\bigwedge_{i \in I_p \cup I_q} C_i$ mentions at most $2k^2 \log n$ variables. As d-SDNNFs (OBDDs, SDDs, ...) can represent any Boolean function over N variables with any vtree in size 2^N, the existence of the desired d-SDNNF D^* of size at most $2^{2k^2 \log n} = n^{2k^2}$ follows. ◀

It is also possible to extend this theorem to further compilation classes $\mathfrak{C}'$, as long as $\mathfrak{C}'$ supports polynomial conjunction, restriction to partial assignments, and can efficiently represent D^*.

We can also use lower bounds on $\mathfrak{C}(\wedge)$-compilations for φ to obtain a lower bound for $\mathfrak{C}(\wedge)$-refutations of $\mathcal{Z}(\varphi)$. A $\mathfrak{C}(\wedge)$-compilation of a satisfiable CNF φ is defined analogously to a $\mathfrak{C}(\wedge)$-refutation, however, instead of obtaining a contradiction, the goal is to obtain a $\mathfrak{C}$-representation of φ.

▶ **Theorem 14.** *Let $\varphi = \bigwedge_{i \in [m]} C_i$ be a* CNF*, and $\mathfrak{C} \in \{\mathrm{OBDD}, \mathrm{SDD}, \text{d-SDNNF}\}$. If there exists a $\mathfrak{C}(\wedge)$-compilation of $\mathcal{C}(\varphi) := \bigwedge_{i \in [m]} C_i \vee y_i$ of size S, then there exists a $\mathfrak{C}(\wedge)$-refutation of $\mathcal{Z}(\varphi)$ of size $\mathcal{O}(m \cdot S)$.*

Proof. Let $\mathcal{D} := D_1, \ldots, D_\ell$ be such a compilation. Notably, D_ℓ is equivalent to $f_\varphi = \bigwedge_{i \in [m]} C_i \vee y_i$. We can extend $\mathcal{D}$ into a $\mathfrak{C}(\wedge)$-refutation for $\mathcal{Z}(\varphi)$ as follows: first, we compute $\widehat{D^0} = D_\ell \wedge z_1$. This trivially has size $\mathcal{O}(S)$. Next, we compute for all $i \in [m]$ a $\mathfrak{C}$-representation D^i for the ith implication condition as

$$D^i := \bigwedge_{\lambda \in C_i \cup \{y_i\}} (\lambda \to (z_i \to z_{i+1})) .$$

Finally, we iteratively compute $\widehat{D^i} = \widehat{D^{i-1}} \wedge D^i$. We observe that for all $i \in [m]$

$$\widehat{D^i} \equiv \mathcal{C}(\varphi) \wedge \bigwedge_{j \in [i]} z_j .$$

Because each z_j is a fresh variable that does not otherwise appear in $\mathcal{C}(\varphi)$, each $\widehat{D^i}$ has size $\mathcal{O}(S)$ as well. Finally, if $D^{m+1} = \neg z_{m+1}$, then $D^* = \widehat{D^{m+1}} \wedge D^{m+1} = \bot$ is unsatisfiable, and $\mathcal{D}' = D_1, \ldots, D_\ell, \widehat{D^0}, D^1, \widehat{D^1} \ldots, D^{m+1}, \widehat{D^{m+1}}$ is a $\mathfrak{C}(\wedge)$-refutation of $\mathcal{Z}(\varphi)$ of size $\mathcal{O}(m \cdot S)$. ◀

5 Separations

Now we are able to utilize the lifting theorem from the last section to obtain separations between different proof systems. We start by showing in Section 5.1 that in the absence of weakening, proof systems based on structured circuits can provide strictly shorter proofs than their OBDD counterparts. Moreover, this holds even if the OBDD-system has reordering available and the SDD- or d-SDNNF-system does not.

Second, we show that separations that have been obtained for different variants of OBDD-systems can also be obtained for proof systems based on structured circuits. In particular, we prove the following for all $\mathfrak{C}, \mathfrak{D} \in \{\text{OBDD}, \text{SDD}, \text{d-SDNNF}\}$, where the case $\mathfrak{C} = \mathfrak{D} = \text{OBDD}$ was already known:

- $\mathfrak{D}(\wedge, r)$ does not p-simulate $\mathfrak{C}(\wedge, w)$ (Section 5.2)
- $\mathfrak{C}(\wedge, w)$ does not p-simulate $\mathfrak{D}(\wedge, r)$ (Section 5.3)
- An exponential separation between $\mathfrak{C}(\wedge)$ and $\mathfrak{D}(\wedge, r)$ (Section 5.4)

5.1 OBDD($\wedge$, r) does not p-simulate SDD($\wedge$)

To obtain the separation, we lift from *vertex cover formulas*. For a graph $G = (V, E)$ we define VC_G as

$$\text{VC}_G \coloneqq \bigwedge_{\{u,v\} \in E} (u \vee v). \tag{3}$$

Note that if G has maximum degree Δ, then VC_G is a $(2, \Delta)$-CNF. To define a suitable graph underlying this formula, we use the following product construction: for two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$ we define $G \times H = (V_1 \times V_2, E')$ with

$$\begin{aligned} E' \quad = \quad & \{\{(v_1, w_1), (v_2, w_2)\} \mid v_1 = v_2 \text{ and } \{w_1, w_2\} \in E_2\} \\ & \cup \{\{(v_1, w_1), (v_2, w_2)\} \mid w_1 = w_2 \text{ and } \{v_1, v_2\} \in E_1\}. \end{aligned}$$

Essentially, $G \times H$ is obtained by replacing every $v \in V_1$ with a copy H_v of H, and connecting $w_1 \in V(H_v), w_2 \in V(H_w)$, if the corresponding v and w are adjacent in G and $w_1 = w_2$. Here we consider the case where G is a tree, and H is a path: we define $G[h, \ell] \coloneqq T_h \times P_\ell$, where T_h is a complete binary tree of height h and P_ℓ is a path of length ℓ.

▶ **Observation 15.** *For all $h, \ell \geq 1$, $VC_{G[h,\ell]}$ is a $(2, 5)$-CNF of treewidth ℓ.*

We now lift from the following lower bound of representing $\text{VC}_{G[h,\ell]}$ by OBDDs.

▶ **Theorem 16** ([17])**.** *Let $\ell > 50$ and $h > \log(\ell)$. Any* OBDD*-representation of $VC_{G[h,\ell]}$ has size at least $n^{\Omega(\ell)}$.*

From this lower bound and Theorem 13 we directly get:

▶ **Lemma 17.** *Let $\ell > 50, h > \log(\ell)$. Every* OBDD$(\wedge, r)$*-refutation of $\mathcal{Z}(VC_{G[h,\ell]})$ has size at least $n^{\Omega(\ell)}$.*

Note that if we let $\ell = \log n$, then we obtain a quasipolynomial lower bound for OBDD$(\wedge, r)$-refutations as $n^{\Omega(\ell)} = n^{\Omega(\log n)}$.

On the other hand, we can obtain an upper bound on an SDD$(\wedge)$-refutation using the following lemma:

▶ **Lemma 18.** *The following hold: Let $\varphi = \bigwedge_{i\in[m]} C_i$ be a* CNF *with n variables and $tw(\varphi) = w$. Then there exists a vtree T_φ such that for every subformula $\psi \subseteq \varphi$ there is an* SDD *D_ψ respecting T_φ of size at most $\mathcal{O}(n2^w)$.*

Furthermore, there exists an SDD$(\wedge)$*-compilation of φ of size $\mathcal{O}(m \cdot n2^w)$.*

Proof. It was shown in [8] that there is a vtree T_φ and an SDD D respecting T_φ, which is equivalent to φ and has size $\mathcal{O}(n2^w)$. Since the treewidth of every $\psi \subseteq \varphi$ is *at most* w, it follows that there is a vtree T_ψ and an equivalent vtree D_ψ of size $\mathcal{O}(n2^w)$ as well. It turns out that we can simply chose the same vtree T_φ for all sub-CNFs, as T_φ is considered nice for every $\psi \subseteq \varphi$, as defined in [8]. Using the first two statements, it follows immediately that there is an SDD$(\wedge)$-compilation of φ with the desired upper bound. ◀

Furthermore, we observe that, if the CNF VC_G has treewidth w, then the transformed CNF $\mathcal{C}(\mathrm{VC}_G) \coloneqq \bigwedge_{\{u,v\}\in E}(u \vee v \vee y_e)$ has treewidth at most $w + 1$.

As the treewidth of $\mathrm{VC}_{G[h,\ell]}$ is ℓ, by Lemma 18 we get that $\mathrm{VC}_{G[h,\ell]}$ has a SDD$(\wedge)$-compilation of size $\mathcal{O}(n^2 \cdot 2^\ell)$. By applying our refutation-to-compilation lifting (Theorem 14) we obtain that

▶ **Lemma 19.** *Let $\ell > 50, h > \log(\ell)$. Then there is a* SDD$(\wedge)$*-refutation of $\mathcal{Z}(VC_{G[h,\ell]})$ of size $n^3 \cdot 2^\ell$, where n is the number of variables.*

Combining Lemma 17 and Lemma 19 we obtain the desired separation.

▶ **Theorem 20.** *There is a family (φ_n) of unsatisfiable n-variable* CNF*s of size $O(n^2)$ that have* SDD$(\wedge)$*-refutations of size $n^{O(1)}$ but where every* OBDD$(\wedge, r)$*-refutation has size $n^{\Omega(\log n)}$.*

5.2 $\mathfrak{D}(\wedge, r)$ does not p-simulate $\mathfrak{C}(\wedge, w)$

The vertex cover formulas defined in the previous paragraph can also be used to prove lower bounds for d-SDNNFs and SDDs when applied to graphs of high treewidth.

▶ **Theorem 21** ([4])**.** *Let $d \geq 3, n \in \mathbb{N}$, $0 < c < 1$, and G an (n, d, c)-expander. Every* d-SDNNF *representing VC_G has size $2^{\Omega(n)}$.*

Note that this lower bound also holds for SDDs and OBDDs. Again by applying Theorem 13, we can lift this to a proof size lower bound:

▶ **Lemma 22.** *Let $d \geq 3, n \in \mathbb{N}$, $0 < c < 1$, and G an (n, d, c)-expander. Then, every* d-SDNNF$(\wedge, r)$*-refutation of $\mathcal{Z}(VC_G)$ has size $2^{\Omega(n)}$.*

To obtain the separation we need to show that $\mathcal{Z}(\varphi)$ is easily refutable in $\mathfrak{C}(\wedge, w)$ for $\mathfrak{C} \in \{\text{OBDD}, \text{SDD}, \text{d-SDNNF}\}$. We prove the even stronger result that resolution can efficiently refute every lifted $\mathcal{Z}(\varphi)$. Since OBDD$(\wedge, w)$ is known to p-simulate resolution [2], the result follows.

▶ **Lemma 23.** *Let $\varphi = \bigwedge_{i\in[m]} C_i$ be a* CNF*. Then, there exists a polynomial resolution-refutation of $\mathcal{Z}(\varphi)$.*

Proof. We construct the resolution refutation inductively as follows: Let $C_i = \bigvee_{j\in[\ell]} \lambda_j$. Consider the clauses $\bigwedge_{j\in[\ell]} K_j$, where $K_j \coloneqq (\neg\lambda_j \vee \neg z_i \vee z_{i+1})$. We first compute $C_j \wedge K_1 = C_j \setminus \{\lambda_1\} \vee (\neg z_i \vee z_{i+1})$ and proceed iteratively through all K_j. At each step s we obtain clauses representing $C_j \setminus \{\lambda_1, \dots, \lambda_s\} \vee (\neg z_i \vee z_{i+1})$ until we finally have the clause representing $(\neg z_i \vee z_{i+1}) = (z_i \to z_{i+1})$. Note that each intermediate step of this construction has size at most $|C_i|$. With at most $|C_i|$ steps, one inductive step has size at most $|C_i|^2$. If we continue for all $i \in [m]$, we are finally left with clauses representing $(z_i \to z_{i+1})$ for all $i \in [m]$ as well as (z_1) and $(\neg z_{m+1})$. We can trivially join those clauses to obtain a contradiction. The resulting resolution refutation has size $\mathcal{O}(m \cdot M^2)$, where M is the size of the largest clause in φ. Thus, there exists a polynomial resolution refutation of $\mathcal{Z}(\varphi)$. ◀

Combining the upper and the lower bound yields the desired separation.

▶ **Theorem 24.** *There is a family (φ_n) of unsatisfiable n-variable* CNF*s of size $O(n^2)$ that have resolution- and $\mathfrak{C}(\wedge, w)$-refutations of size $n^{O(1)}$, but where every $\mathfrak{D}(\wedge, r)$-refutation has size $2^{\Omega(n)}$ for all $\mathfrak{C}, \mathfrak{D} \in \{\text{OBDD}, \text{SDD}, \text{d-SDNNF}\}$.*

5.3 $\mathfrak{C}(\wedge, w)$ does not p-simulate $\mathfrak{D}(\wedge, r)$

The above separation between systems without reordering and without weakening also holds in the converse direction, that is, d-SDNNF$(\wedge, w)$ does not p-simulate OBDD$(\wedge, r)$. Here it turns out that the same CNF-family that separates OBDD$(\wedge, w)$ from OBDD$(\wedge, r)$ [7] also separates the stronger d-SDNNF$(\wedge, w)$ system from OBDD$(\wedge, r)$. To establish this, one basically needs to verify, that the communication complexity arguments used in [7] also imply lower bounds on d-SDNNF$(\wedge, w)$-refutations. As this is technically tedious, we only state the result here and provide some more details in the full version. Note that by Theorem 10 this also implies a quasi-polynomial separation between d-SDNNF$(\wedge, w)$ and d-SDNNF$(\wedge, w, r^*)$.

▶ **Theorem 25.** *There is a family (φ_n) of* CNF*s such that:*
- *φ_n has $n^{\mathcal{O}(1)}$ variables and $n^{\mathcal{O}(n \log n)}$ clauses.*
- *φ_n has an* OBDD$(\wedge, r)$*-refutation of size $n^{\mathcal{O}(n \log n)} = |\varphi_n|^{\mathcal{O}(1)}$.*
- *Every* d-SDNNF$(\wedge, w)$*-refutation of φ_n has size at least $n^{\Omega(n^2)} = |\varphi_n|^{\Omega(n/\log n)}$.*

5.4 An exponential separation between $\mathfrak{C}(\wedge)$ and $\mathfrak{D}(\wedge, r)$

While Theorem 25 implies a quasipolynomial separation between $\mathfrak{C}(\wedge)$ and $\mathfrak{D}(\wedge, r)$, the goal of this section is to obtain an exponential separation (and showcase another application of the lifting theorem).

▶ **Theorem 26.** *There exists a family (φ_n) of unsatisfiable* CNF*s such that:*
- *φ_n has $\mathcal{O}(n)$ variables and size $n^{\mathcal{O}(1)}$.*
- *φ_n has an* OBDD$(\wedge, r)$*-refutation of size $|\varphi_n|^{\mathcal{O}(1)}$*
- *Every* d-SDNNF$(\wedge)$*-refutation of φ_n has size $2^{\Omega(n)}$.*

To prove the theorem we follow the same route as for the known exponential separation of OBDD$(\wedge)$ from OBDD$(\wedge, r)$ [11]: we start with *shifted equality* φ [15] and transform it to an unsatisfiable CNF by adding a suitable implication chain. However, since our $\mathcal{Z}(\varphi)$ lifting for SDD/d-SDNNF is slightly different as the one used for OBDDs in [11] we need to adapt the proof accordingly.

Let X be a set of variables. We define $T_{\mathcal{I}}$ as the unique conjunction over X such that for all assignments $\mathcal{I}'$ over X it holds that $\mathcal{I}' \models T_{\mathcal{I}}$ exactly if $\mathcal{I}' = \mathcal{I}$. We observe that for all $\mathcal{I}, \mathcal{I}'$, which disagree on at least one variable, $T_{\mathcal{I}} \wedge T_{\mathcal{I}'}$ is unsatisfiable. We can use these terms to define permutation formulas.

▶ **Definition 27.** *Let φ be a* CNF *over variables X, and let $S_X : \{\pi \mid \pi : X \to X\}$ be the set of all permutations on X. Furthermore, let $\Pi \subseteq S_X$ be some set of permutations on X, let $\sigma_\Pi : \Pi \to \{0,1\}^{W_\Pi}$ be an injective function for some fresh variable set W_Π with $|W_\Pi| = \lceil \log(|\Pi|) \rceil$, Then we define:*

$$perm_\Pi(\varphi) \coloneqq \bigwedge_{\pi \in \Pi} \left((T_{\sigma_\Pi(\pi)}) \to \varphi_\pi \right) \wedge \bigwedge_{\alpha \in \{0,1\}^{W_\Pi} \setminus \sigma(\Pi)} \neg \left(T_\alpha \right),$$

where φ_π is constructed by replacing all variables x in φ by $\pi(x)$ and $\{0,1\}^{W_\Pi} \setminus \sigma(\Pi)$ is the set of all assignments to W_Π, which are not mentioned by some $\sigma(\pi)$.

Note that, as T_α is a term, $\neg T_\alpha$ is a clause, and all T_α only appear negatively. We are now ready to describe our separation:

▶ **Lemma 28.** *Let φ be an unsatisfiable* CNF *such that the following holds:*

- *There exists an order $\leq$ and a polynomial size* OBDD($\wedge$)*-refutation of φ respecting $\leq$*
- *There exists a set $\Pi \subseteq S_X$ of permutations of size $|\Pi| = |X|^{\mathcal{O}(1)}$ such that for every vtree (T, b) there exists a $\pi \in \Pi$ such that any* d-SDNNF($\wedge$)*-refutation of φ_π respecting (T, b) has size $2^{\Omega(n)}$.*

Then there exists a polynomial size OBDD($\wedge, r$) *refutation of $perm_\Pi(\varphi)$, but every* d-SDNNF($\wedge$) *refutation of $perm_\Pi(\varphi)$ has exponential size.*

This lemma is very similar to Theorem 4.1 in [11] and its proof is deferred to the full version. In order to apply Lemma 28, we use *shifted equality* [15].

▶ **Definition 29.** *Let $n = 2^m$, for some $m \in \mathbb{N}$, $X = \{x_0, \ldots, x_{n-1}\}$, $Y = \{y_0, \ldots, y_{n-1}\}$ and $\ell \in \{0, \ldots, n-1\}$. We define EQ_n^ℓ as:*

$$EQ_n^\ell \coloneqq \bigwedge_{i=0}^{n-1} \left(x_i \vee \neg y_{\ell+i \,(\mathrm{mod}\ n)} \right) \wedge \left(\neg x_i \vee y_{\ell+i \,(\mathrm{mod}\ n)} \right).$$

For $W = \{w_1, \ldots, w_m\}$ and a bijection $\sigma : [n] \to \{0,1\}^W$, shifted equality *is defined as*

$$SEQ_n \coloneqq \bigwedge_{\ell=0}^{n-1} \left(T_{\sigma(\ell)} \to \mathcal{Z}(EQ_n^\ell) \right) = perm_\Pi \left(\mathcal{Z}(EQ_n^0) \right),$$

where $\Pi \coloneqq \{\pi_\ell \mid 0 \leq \ell \leq n-1\}$ with $\pi_\ell(y_i) \coloneqq y_{i+\ell \ (\mathrm{mod}\ n)}$ for all $y_i \in Y$ and $\pi_\ell(v) \coloneqq v$ for all $v \in \mathrm{vars}(\mathcal{Z}(EQ_n^0)) \setminus Y$.

As $\mathcal{Z}(\mathrm{EQ}_n^\ell)$ is unsatisfiable for all ℓ, SEQ_n is unsatisfiable as well. The proof of our separations (Theorem 26) follows from the following key lemma. While its first claim is very similar to Proposition 4.5 in [11], the second claim requires a proof that utilizes known communication complexity lower bounds. The proof of the lemma is provided in the full version.

▶ **Lemma 30.** *The following hold:*

1. *For every $\ell \in \{0, \ldots, n-1\}$ there exists a polynomial size* OBDD($\wedge$)*-refutation of $\mathcal{Z}(EQ_n^\ell)$.*
2. *For every vtree (T, b) there exists an $\ell \in \{0, \ldots, n-1\}$ such that every* d-SDNNF*-representation of EQ_n^ℓ has size $2^{\Omega(n)}$.*

Proof of Theorem 26. We let $\varphi_n \coloneqq \mathrm{SEQ}_n$ and the size bound follows from the definition. All we need to do is to apply Lemma 28 to $\mathcal{Z}(\mathrm{EQ}_n^0)$; for this we need to verify the two prerequisites. The first one (existence of OBDD($\wedge$)-refutations) follows from the first part of Lemma 30. Combining the second claim of Lemma 30 with Theorem 13 yields that for every (T, b) there is an ℓ such that every d-SDNNF($\wedge$)-refutation of $\mathcal{Z}(\mathrm{EQ}_n^\ell)$ respecting (T, b) has size $2^{\Omega(n)}$. By using the set Π from Definition 29 this implies the second prerequisite from Lemma 28. ◀

5.5 Other separations

Finally, let us briefly discuss seperations between our proof systems and some well-known further proof systems. In [6] it was shown that Cutting Planes does not p-simulate OBDD($\wedge$), and by extension all OBDD-based proof systems. This clearly also holds for all of our proof systems. Furthermore, in [2] it was shown that Cutting Planes *with unary coefficients* is simulated by OBDD($\wedge, w$), though to our knowledge it is unknown whether any OBDD-based proof system simulates the general version of Cutting Planes. All of these separations can be easily extended to our SDD/d-SDNNF-based proof systems without further issues.

Frege systems are also often considered in relation to OBDD-based systems. It is well-known that OBDD($\wedge, w$) is incomparable to *bounded-depth* Frege [14]. However, we leave it is an open question, whether the methods developed in [14] can be extended to SDDs and d-SDNNFs as well. Surprisingly, it has not been studied to our knowledge whether (unbounded depth) Frege simulates OBDD($\wedge, w$). We also leave this question, as well as its extension to SDDs and d-SDNNFs open.

6 Conclusion

Despite its versatility, the lifting method introduced in this paper is not strong enough for some purposes. Notably, even though an *exponential* separation between OBDDs and SDDs exists in the satisfiable setting [3], we are not aware of an exponential separation that can be encoded as a $(k, \log n)$-CNF. Therefore, a proper exponential separation between OBDD($\wedge, r$) and SDD($\wedge$) would require more sophisticated methods. The same holds for any separations between proof systems using SDDs and those using d-SDNNFs, as even though there are known superpolynomial separations between the two representation formats in the satisfiable setting [20], it is unclear, whether these separations can be extended to the unsatisfiable setting as well. Furthermore, as a result of Lemma 23, our lifting method is completely defeated by resolution, and therefore by any proof systems using weakening. Thus it remains open, whether OBDD($\wedge, w$) p-simulates SDD($\wedge, w$) or even SDD($\wedge$).

Another open problem concerns the one-shot restructuring rule (r) for SDDs and d-SDNNFs: it is currently open whether SDD($\wedge, w, r$) and d-SDNNF($\wedge, w, r$) are polynomially verifiable proof systems, as it is open whether the equivalence problem for SDDs/d-SDNNFs respecting different vtrees is polynomial-time verifiable, which is a problem that might also be of independent interest.

Finally, we reiterate the long standing open problem of superpolynomial lower bounds for OBDD($\wedge, w, r$), which are a necessary requirement for separating OBDD($\wedge, w, r$) from d-SDNNF($\wedge, w, r$).

References

1 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Second Edition*. John Wiley, 2000. doi:10.1002/0471722154.

2 Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2004. doi:10.1007/978-3-540-30201-8_9.

3 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. doi:10.1609/AAAI.V30I1.10107.

4 Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. A strongly exponential separation of dnnfs from cnf formulas, 2015. arXiv:1411.1995.

5 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. URL: http://www.ijcai.org/Abstract/16/147.

6 Sam Buss, Dmitry Itsykson, Alexander Knop, Artur Riazanov, and Dmitry Sokolov. Lower bounds on OBDD proofs with several orders. *ACM Trans. Comput. Log.*, 22(4):26:1–26:30, 2021. doi:10.1145/3468855.

7 Sam Buss, Dmitry Itsykson, Alexander Knop, and Dmitry Sokolov. Reordering rule makes OBDD proof systems stronger. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 16:1–16:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.16.

8 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. doi:10.5591/978-1-57735-516-8/IJCAI11-143.

9 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *CoRR*, abs/1106.1819, 2011. arXiv:1106.1819.

10 Alexis de Colnet and Stefan Mengel. Lower bounds on intermediate results in bottom-up knowledge compilation. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 5564–5572. AAAI Press, 2022. doi:10.1609/AAAI.V36I5.20496.

11 Dmitry Itsykson, Alexander Knop, Andrei E. Romashchenko, and Dmitry Sokolov. On obdd-based algorithms and proof systems that dynamically change the order of variables. *J. Symb. Log.*, 85(2):632–670, 2020. doi:10.1017/JSL.2019.53.

12 Dmitry Itsykson and Artur Riazanov. Automating OBDD proofs is np-hard. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 59:1–59:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.59.

13 Dmitry Itsykson, Artur Riazanov, and Petr Smirnov. Tight bounds for tseitin formulas. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.6.

14 Jan Krajícek. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *Electron. Colloquium Comput. Complex.*, TR07, 2007. URL: https://eccc.weizmann.ac.il/eccc-reports/2007/TR07-007/index.html.

15 Eyal Kushilevitz and Noam Nisan. *Variable Partition Models*, pages 97–104. Cambridge University Press, 1996.

16 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. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-082.php.

17 Igor Razgon. No small nondeterministic read-once branching programs for cnfs of bounded treewidth. In Marek Cygan and Pinar Heggernes, editors, *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, volume 8894 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2014. doi:10.1007/978-3-319-13524-3_27.

18 Igor Razgon. On obdds for cnfs of bounded treewidth. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7982.

19 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. doi:10.1109/CCC.2008.34.

20 Harry Vinall-Smeeth. Structured d-dnnf is not closed under negation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 3593–3601. ijcai.org, 2024. URL: https://www.ijcai.org/proceedings/2024/398.

21 Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000. URL: http://ls2-www.cs.uni-dortmund.de/monographs/bdd/.