A Characterization of Efficiently Compilable Constraint Languages

Christoph Berkholz
Technische Universität Ilmenau, Germany

Stefan Mengel
Univ. Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), France

Hermann Wilhelm
Technische Universität Ilmenau, Germany

Abstract

A central task in *knowledge compilation* is to compile a CNF-SAT instance into a succinct representation format that allows efficient operations such as testing satisfiability, counting, or enumerating all solutions. Useful representation formats studied in this area range from ordered binary decision diagrams (OBDDs) to circuits in decomposable negation normal form (DNNFs).

While it is known that there exist CNF formulas that require exponential size representations, the situation is less well studied for other types of constraints than Boolean disjunctive clauses. The *constraint satisfaction problem* (CSP) is a powerful framework that generalizes CNF-SAT by allowing arbitrary sets of constraints over any finite domain. The main goal of our work is to understand for which type of constraints (also called the *constraint language*) it is possible to efficiently compute representations of polynomial size. We answer this question completely and prove two tight characterizations of efficiently compilable constraint languages, depending on whether target format is structured.

We first identify the combinatorial property of "strong blockwise decomposability" and show that if a constraint language has this property, we can compute DNNF representations of linear size. For all other constraint languages we construct families of CSP-instances that provably require DNNFs of exponential size. For a subclass of "strong *uniformly* blockwise decomposable" constraint languages we obtain a similar dichotomy for *structured* DNNFs. In fact, strong (uniform) blockwise decomposability even allows efficient compilation into multi-valued analogs of OBDDs and FBDDs, respectively. Thus, we get complete characterizations for all knowledge compilation classes between O(B)DDs and DNNFs.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Complexity theory and logic

Keywords and phrases constraint satisfaction, knowledge compilation, dichotomy, DNNF

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.11

Related Version *Full Version*: https://arxiv.org/abs/2311.10040 [5]

Funding *Christoph Berkholz*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 414325841.
Stefan Mengel: This work has been partially supported by the ANR project EQUUS ANR-19-CE48-0019.
Hermann Wilhelm: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 414325841.

1 Introduction

One of the main aims of *knowledge compilation* is to encode solution sets of computational problems into a succinct but usable form [25]. Typical target formats for this compilation process are different forms of decision diagrams [44] or restricted classes of Boolean circuits. One of the most general representation formats are circuits in *decomposable negation normal form* (DNNF), which have been introduced in [23] as a compilation target for Boolean functions. Related notions, which also rely on the central decomposability property, have been independently considered in databases [38, 37] and circuit complexity [41, 1]. Besides these, DNNF circuits and related compilation classes have also been proven useful in areas like probabilistic inference [26], constraint satisfaction [31, 6, 4, 34], MSO evaluation [2], QBF solving [15], to name a few. The DNNF representation format has become a popular data structure because it has a particularly good balance between generality and usefulness [25]. There has also been a large amount of practical work on compiling solution sets into DNNF or its fragments, see e.g. [33, 24, 36, 17, 39, 31]. In all these works, it assumed that the solutions to be compiled are given as a system of constraints, often as a set of disjunctive Boolean clauses, i. e., in conjunctive normal form (CNF).

In this setting, however, strong lower bounds are known: it was shown that there are CNF-formulas whose representation as DNNF requires exponential size [7, 8, 14, 3]. The constraints out of which the hard instances in [7, 3, 14] are constructed are very easy – they are only monotone clauses of two variables. Faced with these negative results, it is natural to ask if there are any classes of constraints that guarantee efficient compilability into DNNF.

We answer this question completely and prove a tight characterization for every constraint language Γ. We first examine the combinatorial property of *strong blockwise decomposability* and show that if a constraint language Γ has this property, any system of constraints over Γ can be compiled into a DNNF representation of linear size within polynomial time. Otherwise, there are systems of constraints that require exponential size DNNF representations. In the tractable case, one can even compile to the restricted fragment of *free decision diagrams* (FDD) that are in general known to be exponentially less succinct than DNNF [44, 25].

We also consider the important special case of so-called structured DNNF [40] which are a generalization of the well-known ordered binary decision diagrams [9]. We show that there is a restriction of strong blockwise decomposability that determines if systems of constraints over a set Γ can be compiled into structured DNNF in polynomial time. In the tractable case, we can in fact again compile into a restricted fragment, this time ordered decision diagrams (ODD).

Let us stress that all our lower bounds are unconditional and do not depend on any unproven complexity assumptions.

Further related work. Our work is part of a long line of work in constraint satisfaction, where the goal is to precisely characterize those constraint languages that are "tractable". Starting with the groundbreaking work of Schaefer [42] which showed a dichotomy for deciding consistency of systems of *Boolean* constraints, there has been much work culminating in the dichotomy for general constraint languages [11, 45]. Beyond decision, there are dichotomies for counting [19, 10, 27], enumeration [21], optimization [18, 30, 20] and in the context of valued CSPs [43, 13]. Note that the hardness part (showing that certain constraint languages do not admit efficient algorithms) always relies on some complexity theoretic assumption.

The complexity of constraint satisfaction has also been studied "from the other side", where the constraint language is unrestricted and the structure of the *constraint network* (i. e. how the constraints are arranged) has been analyzed. In this setting, characterizations

of (bounded arity) constraint networks have been obtained for the deciding the existence of solutions [29] and counting solutions [22] (again, under some complexity theoretic assumption), while for enumerating solutions only partial results exist [12]. Tractability classifications of the constraint network have also been obtained in the context of valued CSPs [16]. Recently, an unconditional characterization of (bounded arity) constraint networks that allow efficient compilation into DNNFs has been proven [6].

Out of these works, our results are most closely related to the counting dichotomy of [27]. The tractable classes we obtain in our dichotomies are also tractable for counting, and in fact we use the known counting algorithm in our compilation algorithms as a subroutine. Also, the tractability criterion has a similar flavor, but wherever in [27] the count of elements in certain relations is important, for our setting one actually has to understand their structure and how exactly they decompose. That said, while the tractability criterion is related to the one in [27], the techniques to show our results are very different. In particular, where [27] use reductions from #P-complete problems to show hardness, we make an explicit construction and then use communication complexity to get strong, unconditional lower bounds. Also, our two algorithms for construction ODDs and FDDs are quite different to the counting algorithm.

Outline of the paper. After some preliminaries in Section 2, we introduce the decomposability notions that we need to formulate our results in Section 3. We also give the formal formulation of our main results there. In Section 4, we show some of the properties of the constraints we defined in the section before. These properties will be useful throughout the rest of the paper. In Section 5, we present the algorithms for the positive cases of our dichotomies, then, in Section 6, we show the corresponding lower bounds. We specialize our results to the case of Boolean relations in Section 7, showing that the tractable cases are essentially only equalities and disequalities. Finally, we conclude in Section 8.

2 Preliminaries

Constraints. Throughout the paper we let D be a finite *domain* and X a finite set of *variables* that can take values over D. We typically denote variables by u, v, w, x, y, z and domain elements by a, b, c, d. A k-tuple $(x_1, \dots, x_k) \in X^k$ of variables is also denoted by $\vec{x} = x_1 x_2 \cdots x_k$ and we let $\tilde{x} := \{x_1, \dots, x_k\}$ be the set of variables occurring in $\vec{x}$. We use the same notation for tuples of domain elements. A *k-ary relation* R (over D) is a set $R \subseteq D^k$. A *k-ary constraint* (over X and D), denoted by $R(\vec{x})$, consists of a k-tuple of (not necessarily distinct) variables $\vec{x} \in X^k$ and a k-ary relation $R \subseteq D^k$. For a constraint $R(\vec{x})$ we call $\tilde{x}$ the *scope* of the constraint and R the *constraint relation*. The *solution set* of a constraint $R(x_1, \dots, x_k)$ is defined by $\mathrm{sol}(R(x_1, \dots, x_k)) := \{\beta \mid \beta \colon \{x_1, \dots, x_k\} \to D;\ (\beta(x_1), \dots, \beta(x_k)) \in R\}$. Since the order of the columns in a constraint relation is not of great importance for us, we often identify constraints with their solution set and treat sets $\mathcal{S}$ of mappings from $Y \subseteq X$ to D as a constraint with scope Y and solution set $\mathcal{S}$. Moreover, for readability we will often write, e. g., "a constraint $R(x, y, \vec{w})$, such that x and y ..." when we do not strictly require x and y to be at the first and second position and actually refer to any constraint having x and y in its scope.

CSP-instance and constraint language. A *constraint satisfaction instance* $I = (X, D, C)$ consists of a finite set of variables X, a finite domain D and a finite set C of constraints. The solution set $\mathrm{sol}(I) := \{\alpha \mid \alpha \colon X \to D;\ \alpha|_{\tilde{x}} \in \mathrm{sol}(R(\vec{x})) \text{ for all } R(\vec{x}) \in C\}$ of an instance I is

the set of mappings from X to D that satisfies all constraints. A *constraint language* Γ is a finite set of relations (over some finite domain D). A constraint satisfaction instance I is a CSP(Γ)-instance if every constraint relation R occurring in I is contained in the constraint language Γ.

Conjunction and projection. A *conjunction* $R(\vec{u}) = S(\vec{v}) \wedge T(\vec{w})$ of two constraints $S(\vec{v})$ and $T(\vec{w})$ defines a constraint $R(\vec{u})$ over $\vec{u} = u_1 \cdots u_\ell$ with scope $\tilde{u} = \tilde{v} \cup \tilde{w}$ and constraint relation $R := \{(\beta(u_1), \ldots, \beta(u_\ell)) \mid \beta\colon \tilde{u} \to D;\ \beta|_{\tilde{v}} \in \mathrm{sol}(S(\vec{v}))$ and $\beta|_{\tilde{w}} \in \mathrm{sol}(T(\vec{w}))\}$.[1] If $\tilde{v} \cap \tilde{w} = \emptyset$, then a conjunction is called a *(Cartesian) product* and written $R(\vec{u}) = S(\vec{v}) \times T(\vec{w})$. For a constraint $R(\vec{x})$ and a set $Y \subseteq \tilde{x}$ we let the *projection* $\pi_Y R(\vec{x})$ be the constraint $S(\vec{y})$ with scope $\tilde{y} = Y$ obtained by projecting the constraint relation to corresponding coordinates, i. e., for $\vec{x} = x_1 \cdots x_k$ let $1 \leq i_1 < \cdots < i_\ell \leq k$ s. t. $\{i_1, \ldots, i_\ell\} = \{i \mid x_i \in Y\}$, $\vec{y} := x_{i_1} \cdots x_{i_\ell}$, and $S := \{(a_{i_1}, \ldots, a_{i_\ell}) \mid (a_1, \ldots, a_k) \in R\}$.

Formulas and pp-definability. A Γ-*formula* $F(\vec{x}) = \bigwedge_i S_i(\vec{x}_i)$ is a conjunction over several constraints whose constraint relations are in Γ. To avoid notational clutter, we use "$F(\vec{x})$" for both, the conjunction as syntactic expression *and* the constraint defined by this formula. Note that any CSP(Γ)-instance $I = (X, D, C)$ corresponds to a Γ-formula $F(\vec{x}) = \bigwedge_{R_i(\vec{x}_i) \in C} R_i(\vec{x}_i)$ with $\tilde{x} = X$ and $\mathrm{sol}(F(\vec{x})) = \mathrm{sol}(I)$. Thus, we can treat CSP(Γ)-instances as Γ-formulas and vice versa. A *primitive positive (pp)* formula over Γ is an expression $F(\vec{y}) = \pi_{\tilde{y}}(\bigwedge_i S_i(\vec{x}_i) \wedge \bigwedge_{(j,k)} x_j = x_k)$ consisting of a projection applied to a $\Gamma \cup \big\{\{(a,a) \mid a \in D\}\big\}$-formula, which uses constraint relations from Γ and the equality constraint. Note that conjunctions of pp-formulas can be written as pp-formula by putting one projection at the front and renaming variables. A constraint is *pp-definable* over Γ, if it can be defined by a pp-formula over Γ. Moreover, a relation $R \subseteq D^k$ is *pp-definable* over Γ if it is the constraint relation of a pp-definable constraint $R(x_1, \ldots, x_k)$ for pairwise distinct $x_1, \ldots, x_k$. The co-clone $\langle\Gamma\rangle$ of Γ is the set of all pp-definable relations over Γ.

Selection. It is often helpful to use additional unary relations $S \subseteq D$ and to write $U_S(x)$ for the constraint $S(x)$ with scope $\{x\}$ and constraint relation S. We use $U_a(x)$ as an abbreviation for $U_{\{a\}}(x)$. For a constraint $R(\vec{u})$, a variable $x \in \tilde{u}$, and a domain element $a \in D$ we define the *selection* $R(\vec{u})|_{x=a}$ be the constraint obtained by forcing x to take value a, that is, $R(\vec{u})|_{x=a} := R(\vec{u}) \wedge U_a(x)$. Similarly, for a set $S \subseteq D$ we write $R(\vec{u})|_{x \in S} := R(\vec{u}) \wedge U_S(x)$.

DNNF. We will be interested in circuits representing assignments of variables to a finite set of values. To this end, we introduce a multi-valued variant of DNNF; we remark that usually DNNF are only defined over the Boolean domain $\{0,1\}$ [23], but the extension we make here is straightforward and restricted variants have been studied e.g. in [31, 4, 35, 28] under different names.

Let X be a set of variables and D be a finite set of values. A circuit over the operations $\times$ and $\cup$ is a directed acyclic graph with a single sink, called output gate, and whose inner nodes, called gates, are labeled with $\times$ or $\cup$. The source-nodes, called inputs of the circuit, are labeled by expressions of the form $x \mapsto a$ where $x \in X$ and $a \in D$. We say that $\times$-gate v is *decomposable* if there are no two inputs labeled with $x \mapsto a$ and $x \mapsto b$ (with possibly

[1] Again, we may just write "$S(\vec{v}) \wedge T(\vec{w})$" instead of "$R(\vec{u}) = S(\vec{v}) \wedge T(\vec{w})$" if the order or multiple occurrences of variables in $\vec{u}$ does not matter (the solution set $\mathrm{sol}(R(\vec{u}))$ is always the same).

$a = b$) that have a path to v going through different children of v. A DNNF is a circuit in which all $\times$-gates are decomposable. Note that in a DNNF, for every $\times$-gate v, every variable $x \in X$ can only appear below one child of v.

For every DNNF O we define the set $S(O)$ of assignments *captured* by O inductively:

- The set captured by an input v with label $x \mapsto d$ is the single element set $S(v) = \{x \mapsto d\}$.
- For a $\cup$-gate with children v_1, v_2, we set $S(v) := S(v_1) \cup S(v_2)$.
- For a $\times$-gate with children v_1, v_2, we set $S(v) := S(v_1) \times S(v_2)$ where for two assignments $a : X_1 \to D$ and $b : X_2 \to D$, we interpret (a, b) as the joined assignment $c : X_1 \cup X_2 \to D$ with $c(x) := \begin{cases} a(x), & \text{if } x \in X_1 \\ b(x), & \text{if } x \in X_2 \end{cases}$.

We define $S(O) := S(v_o)$ where v_o is the output gate of O. Note that $S(O)$ is well-defined in the case of $\times$, since X_1 and X_2 are disjoint because of decomposability. Finally, we say that O accepts an assignment α to X if there is an assignment $\beta \in S(O)$ such that α is an extension of β to all variables in X. We say that O represents a constraint $C(\vec{x})$ if O accepts exactly the assignments in $\mathrm{sol}(C(\vec{x}))$.

A v-tree of a variable set X is a rooted binary tree whose leaves are in bijection to X. For a v-tree T of X and a node t of T we define $\mathrm{var}(t)$ to be the variables in X that appear as labels in the subtree of T rooted in t. We say that a DNNF O over X is *structured by* T if for every sub-representation O' of O there is a node t in T such that O' is exactly over the variables $\mathrm{var}(t)$ [40]. We say that O is *structured* if there is a v-tree T of X such that O is structured by T.

We will use the following basic results on DNNF which correspond to projection and selection on constraints.

▶ **Lemma 1.** *Let $C(\vec{u})$ be a constraint for which there exists a* DNNF *of size s, $x \in \tilde{u}$ and $A \subseteq D$. Then there exists a* DNNF *for the selection $C(\vec{u})|_{x \in A}$ of size $\leq s$.*

▶ **Lemma 2.** *Let $C(\vec{u})$ be a constraint for which there exists a* DNNF *of size s and $\tilde{v} \in \tilde{u}$. Then there exists a* DNNF *for the projection $\pi_{\tilde{v}}(C(\vec{u}))$ of size $\leq s$.*

Decision Diagrams. A decision diagram O over a variable set X is a directed acyclic graph with a single source and two sinks and in which all non-sinks have $|D|$ outgoing edges. The sinks are labeled with 0 and 1, respectively, while all other nodes are labeled with variables from x. For every non-sink, the $|D|$ outgoing edges are labeled in such a way that every value in D appears in exactly one label. Given an assignment α to X, the value computed by O is defined as follows: we start in the source and iteratively follow the edge labeled by $\alpha(x)$ where x is the label of the current node. We continue this process until we end up in a leaf whose label then gives the value of O on α. Clearly, this way O computes a constraint over X with relation $\{\alpha \mid O \text{ computes 1 on input } \alpha\}$.

We are interested in decision diagrams in which on every source-sink-path every variable appears at most once as a label. We call these diagrams *free decision diagrams (FDD)*. An FDD for which there is an order π such that when a variable x appears before y on a path then x also appears before y in π is called *ordered decision diagrams*. We remark that FDD and ODD are in the literature mostly studied for the domain $\{0, 1\}$ in which case they are called FBDD and OBDD, respectively, where the "B" stands for binary. Note also that there is an easy linear time translation of FDD into DNNF and ODD into structured DNNF, see e.g. [25]. In the other direction there are no efficient translations, see again [25].

Rectangles. Let $\vec{u}$ be a variable vector, $\tilde{u} = \tilde{x} \cup \tilde{y}$ and $\tilde{x} \cap \tilde{y} = \emptyset$. Then we say that the constraint $R(\vec{u})$ is a *(combinatorial) rectangle* with respect to the partition $(\tilde{x}, \tilde{y})$ if and only if $R(\vec{u}) = \pi_{\tilde{x}}(R(\vec{u})) \times \pi_{\tilde{y}}(R(\vec{u}))$. Let $Z \subseteq \tilde{u}$. Then we call the partition $(\tilde{x}, \tilde{y})$ *Z-balanced* if $\frac{|Z|}{3} \leq |\tilde{x} \cap Z| \leq \frac{2|Z|}{3}$. A constraint $R(\vec{u})$ is called a *Z-balanced rectangle* if it is a rectangle with respect to a Z-balanced partition. A *Z-balanced rectangle cover* $\mathcal{R}$ of a constraint $R(\vec{u})$ is defined to be a set of Z-balanced rectangles such that $\mathrm{sol}(R(\vec{u})) := \bigcup_{\mathfrak{r} \in \mathcal{R}} \mathrm{sol}(\mathfrak{r}(\vec{u}))$. The size of $\mathcal{R}$ is the number of rectangles in it.

3 Blockwise decomposability

In this section we introduce our central notion of blockwise and uniformly blockwise decomposable constraints and formulate our main theorems that lead to a characterization of efficiently representable constraint languages.

The first simple insight is the following. Suppose two constraints $S(\vec{v})$ and $T(\vec{w})$ with disjoint scopes are efficiently representable, e. g., by a small ODD. Then their Cartesian product $R(\vec{v}, \vec{w}) = S(\vec{v}) \times T(\vec{w})$ also has a small ODD: given an assignment $(\vec{a}, \vec{b})$, we just need to check independently whether $\vec{a} \in S$ and $\vec{b} \in T$, for example, by first using the ODD for $S(\vec{v})$ and then using the ODD for $T(\vec{w})$. Thus, if a constraint can be expressed as a Cartesian product of two constraints, we only have to investigate whether the two parts are easy to represent. This brings us to our first definition.

▶ **Definition 3.** *Let $R(\vec{u})$ be a constraint and $(V_1, \ldots, V_\ell)$ be a partition of its scope. We call* $R(\vec{u})$ decomposable w.r.t. $(V_1, \ldots, V_\ell)$ *if $R(\vec{u}) = \pi_{V_1}(R(\vec{u})) \times \cdots \times \pi_{V_\ell}(R(\vec{u}))$. A constraint $R(\vec{u})$ is* indecomposable *if it is only decomposable w.r.t. trivial partitions $(V_1, \ldots, V_\ell)$ where $V_i = \emptyset$ or $V_i = \tilde{u}$ for $i \in [\ell]$.*

Next, we want to relax this notion to constraints that are "almost" decomposable. Suppose we have four relations S_1, S_2 of arity s and T_1, T_2 of arity t and let a, b be two distinct domain elements. Let

$$R := (\{(a,a)\} \times S_1 \times T_1) \cup (\{(b,b)\} \times S_2 \times T_2). \tag{1}$$

The constraint $R(x, y, \vec{v}, \vec{w})$ may now not be decomposable in any non-trivial variable partition. However, after fixing values for x and y the remaining selection $R(x, y, \vec{v}, \vec{w})|_{x=c, y=d}$ is decomposable in $(\tilde{v}, \tilde{w})$ for any pair $(c, d) \in D^2$. Thus, an ODD could first read values for x, y and then use ODDs for $S_1(\vec{v})$ and $T_1(\vec{w})$ if $x = y = a$, ODDs for $S_2(\vec{v})$ and $T_2(\vec{w})$ if $x = y = b$, or reject otherwise. This requires, of course, that $S_1(\vec{v})$ and $S_2(\vec{v})$, as well as $T_1(\vec{w})$ and $T_2(\vec{w})$, have small ODDs over the *same* variable order. For FDDs and DNNFs, however, we would not need this requirement on the variable orders.

To reason about the remaining constraints after two variables have been fixed, it is helpful to use the following matrix notation. Let $R(\vec{u})$ be a constraint and $x, y \in \tilde{u}$ be two variables in its scope. The *selection matrix* $M^R_{x,y}$ is the $|D| \times |D|$ matrix where the rows and columns are indexed by domain elements $a_i, a_j \in D$ and the entries are the constraints

$$M^R_{x,y}[a_i, a_j] := \pi_{\tilde{u} \setminus \{x,y\}}(R(\vec{u})|_{x=a_i, y=b_j}). \tag{2}$$

▶ **Example 4.** Let $D = \{a, b, c\}$ and $R(x, y, z, v)$ a constraint with constraint relation $R = \{(a,a,a,a), (b,b,a,b), (b,b,a,c), (b,b,c,c), (c,b,c,a)\}$. The selection matrix in x and y is depicted below, where the first line and column are the indices from D and the matrix entries contain the constraint relations of the corresponding constraints $M^R_{x,y}[a_i, a_j](z, v)$:

$$\left(\begin{array}{c|ccc} x\backslash y & a & b & c \\ \hline a & \{(a,a)\} & \emptyset & \emptyset \\ b & \emptyset & \{(a,b),(a,c),(c,c)\} & \emptyset \\ c & \emptyset & \{(c,a)\} & \emptyset \end{array}\right) \tag{3}$$

A *block* in the selection matrix is a subset of rows $A \subseteq D$ and columns $B \subseteq D$. We also associate with a block (A, B) the corresponding constraint $R(\vec{u})|_{x\in A, y\in B}$. A selection matrix is a *proper block matrix*, if there exist pairwise disjoint $A_1, \dots, A_k \subseteq D$ and pairwise disjoint $B_1, \dots, B_k \subseteq D$ such that for all $a_i, a_j \in D$:

$$\mathrm{sol}(M^R_{x,y}[a_i, a_j]) \neq \emptyset \iff \text{there is } \ell \in [k] \text{ such that } a_i \in A_\ell \text{ and } a_j \in B_\ell. \tag{4}$$

The selection matrix in Example 4 is a proper block matrix with $A_1 = \{a\}$, $A_2 = \{b, c\}$, $B_1 = \{a\}$, $B_2 = \{b\}$. We will make use of the following alternative characterization of proper block matrices. The simple proof is similar to [27, Lemma 1].

▶ **Lemma 5.** *A selection matrix $M^R_{x,y}$ is a proper block matrix if and only if it has no 2×2-submatrix with exactly one empty entry.*

Now we can define our central tractability criterion for constraints that have small ODDs, namely that any selection matrix is a proper block matrix whose blocks are decomposable over the same variable partition that separates x and y.

▶ **Definition 6** (Uniform blockwise decomposability)**.** *A constraint $R(\vec{u})$ is* uniformly blockwise decomposable in x, y *if $M^R_{x,y}$ is a proper block matrix with partitions $(A_1, \dots, A_k)$, $(B_1, \dots, B_k)$ and there is a partition (V, W) of $\tilde{u}$ with $x \in V$ and $y \in W$ such that each block $R(\vec{u})|_{x\in A_i, y\in B_i}$ is decomposable in (V, W). A constraint $R(\vec{u})$ is* uniformly blockwise decomposable *if it is uniformly decomposable in any pair $x, y \in \tilde{u}$.*

In the non-uniform version of blockwise decomposability, it is allowed that the blocks are decomposable over different partitions. This property will be used to characterize constraints having small FDD representations.

▶ **Definition 7** (Blockwise decomposability)**.** *A constraint $R(\vec{u})$ is* blockwise decomposable in x, y *if $M^R_{x,y}$ is a proper block matrix with partitions $(A_1, \dots, A_k)$, $(B_1, \dots, B_k)$ and for each $i \in [k]$ there are $V_i, W_i \subseteq \tilde{u}$ with $x \in V_i$ and $y \in V_j$ such that each block $R(\vec{u})|_{x\in A_i, y\in B_i}$ is decomposable in (V_i, W_j). A constraint $R(\vec{u})$ is* blockwise decomposable *if it is decomposable in any pair $x, y \in \tilde{u}$.*

Note that every uniformly blockwise decomposable relation is also blockwise decomposable. The next example illustrates that the converse does not hold.

▶ **Example 8.** Consider the 4-ary constraint relations

$$R_1 := \{(a,a,a,a), (a,b,a,b), (b,a,b,a), (b,b,b,b)\} \tag{5}$$

$$R_2 := \{(c,c,c,c), (c,d,d,c), (d,c,c,d), (d,d,d,d)\} \tag{6}$$

$$R := R_1 \cup R_2 \tag{7}$$

Then the selection matrix $M^R_{x,y}$ of the constraint $R(x, y, u, v)$ has two non-empty blocks:

$$M^R_{x,y} = \left(\begin{array}{c|cccc} x\backslash y & a & b & c & d \\ \hline a & \{(a,a)\} & \{(a,b)\} & \emptyset & \emptyset \\ b & \{(b,a)\} & \{(b,b)\} & \emptyset & \emptyset \\ c & \emptyset & \emptyset & \{(c,c)\} & \{(d,c)\} \\ d & \emptyset & \emptyset & \{(d,c)\} & \{(d,d)\} \end{array}\right) \tag{8}$$

The first block $R(x,y,u,v)|_{x\in\{a,b\},y\in\{a,b\}} = R_1(x,y,u,v)$ is decomposable in $\{x,u\}$ and $\{y,v\}$, while the second block $R(x,y,u,v)|_{x\in\{c,d\},y\in\{c,d\}} = R_2(x,y,u,v)$ is decomposable in $\{x,v\}$ and $\{y,u\}$. Thus, the constraint is blockwise decomposable in x,y, but not uniformly blockwise decomposable.

Finally, we transfer these characterizations to relations and constraint languages: A k-ary relation R is (uniformly) blockwise decomposable, if the constraint $R(x_1,\dots,x_k)$ is (uniformly) blockwise decomposable for pairwise distinct variables $x_1,\dots,x_k$. A constraint language Γ is (uniformly) blockwise decomposable if every relation in Γ is (uniformly) blockwise decomposable. A constraint language Γ is *strongly (uniformly) blockwise decomposable* if its co-clone $\langle\Gamma\rangle$ is (uniformly) blockwise decomposable.

Now we are ready to formulate our main theorems. The first one states that the strongly uniformly blockwise decomposable constraint languages are precisely those that can be efficiently compiled to a structured representation format (anything between ODDs and structured DNNFs).

▶ **Theorem 9.** *Let Γ be a constraint language.*

1. *If Γ is strongly uniformly blockwise decomposable, then there is a polynomial time algorithm that constructs an* ODD *for a given* CSP(Γ)*-instance.*
2. *If Γ is not strongly uniformly blockwise decomposable, then there is a family (I_n) of* CSP(Γ)*-instances such that any structured* DNNF *for I_n has size $2^{\Omega(\|I_n\|)}$.*

Our second main theorem states that the larger class of strongly blockwise decomposable constraint languages captures CSPs that can be efficiently compiled in an unstructured format between FDDs and DNNFs.

▶ **Theorem 10.** *Let Γ be a constraint language.*

1. *If Γ is strongly blockwise decomposable, then there is a polynomial time algorithm that constructs an* FDD *for a given* CSP(Γ)*-instance.*
2. *If Γ is not strongly blockwise decomposable, then there is a family (I_n) of* CSP(Γ)*-instances such that any* DNNF *for I_n has size $2^{\Omega(\|I_n\|)}$.*

4 Properties of the Decomposability Notions

In this section we state important properties about (uniform) blockwise decomposability – all omitted proofs are contained in the full version [5]. We start by observing that these notions are closed under projection and selection.

▶ **Lemma 11.** *Let $R(\vec{u})$ be a blockwise decomposable, resp. uniformly blockwise decomposable, constraint and let $Y \subseteq \tilde{u}$, $z \in \tilde{u}$, and $S \subseteq D$. Then the projection $\pi_Y R(\vec{u})$ as well as the selection $R(\vec{u})|_{z\in S}$ are also blockwise decomposable, resp. uniformly blockwise decomposable.*

▶ **Corollary 12.** *If a constraint language Γ is strongly (uniformly) blockwise decomposable, then its individualization $\Gamma^\bullet := \Gamma \cup \{\{a\} : a \in D\}$ is also strongly (uniformly) blockwise decomposable.*

Next, we will show that blockwise decomposable relations allow for efficient counting of solutions by making a connection to the work of Dyer and Richerby [27]. To state their dichotomy theorem, we need the following definitions. A constraint $R(\vec{x},\vec{y},\vec{z})$ is *balanced* (w.r.t. $\vec{x};\vec{y};\vec{z}$), if the $|D|^{|\vec{x}|} \times |D|^{|\vec{y}|}$ matrix $M^{\#}_{\vec{x},\vec{y}}$ defined by $M^{\#}_{\vec{x},\vec{y}}[\vec{a},\vec{b}] = \left|\operatorname{sol}(R(\vec{x},\vec{y},\vec{z})|_{\vec{x}=\vec{a},\vec{y}=\vec{b}})\right|$ is a block diagonal matrix (after permuting rows/columns), where each block has rank one. A constraint language Γ is *strongly balanced* if every at least ternary pp-definable constraint is balanced.

▶ **Theorem 13** (Effective counting dichotomy [27]). *If Γ is strongly balanced, then there is a polynomial time algorithm that computes $|\mathrm{sol}(I)|$ for a given $\mathrm{CSP}(\Gamma)$-instance I. If Γ is not strongly balanced, then counting solutions for $\mathrm{CSP}(\Gamma)$-instances is #P-complete. Moreover, there is a polynomial time algorithm that decides if a given constraint language Γ is strongly balanced.*

Our next lemma connects blockwise decomposability with strong balance and leads to a number of useful corollaries. We sketch the proof for the case when $R(\vec{x}, \vec{y}, \vec{z})$ is ternary, a full proof can be found in [5].

▶ **Lemma 14.** *Every strongly blockwise decomposable constraint language is strongly balanced.*

Proof sketch. Let Γ be strongly blockwise decomposable and $R(x, y, z)$ a pp-defined ternary constraint. Then the selection matrix $M_{x,y}^{R(x,y,z)}$ is a block matrix, where each block (A, B) is decomposable in some (V, W), either $(\{x, z\}, \{y\})$ or $(\{x\}, \{y, z\})$. In any case, for the corresponding block (A, B) in $M_{x,y}^{\#}$ we have $M_{x,y}^{\#}[a, b] = s_a \cdot t_b$ where $s_a = |\mathrm{sol}(\pi_V(R(x, y, z)|_{x=a, y\in B}))|$ and $t_b = |\mathrm{sol}(\pi_W(R(x, y, z)|_{x\in A, y=b}))|$. Thus, the block has rank 1. ◀

▶ **Corollary 15.** *Let Γ be a strongly blockwise decomposable constraint language.*

1. *Given a Γ-formula $F(\vec{u})$ and a (possibly empty) partial assignment α, the number $\left|\mathrm{sol}(F(\vec{u})|_\alpha)\right|$ of solutions that extend α can be computed in polynomial time.*
2. *Given a pp-formula $F(\vec{u})$ over Γ and $x, y \in \tilde{u}$, the blocks of $M_{x,y}^{F(\vec{u})}$ can be computed in polynomial time.*
3. *Given a pp-formula $F(\vec{u})$ over Γ, the indecomposable factors of $F(\vec{u})$ can be computed in polynomial time.*

Proof sketch. Claim 1 follows immediately from the combination of Lemma 14 with Theorem 13 and the fact that strongly blockwise decomposable constraint languages are closed under selection (Corollary 12). For Claim 2 let $F(\vec{u}) = \pi_{\tilde{u}}(F'(\vec{v}))$ for some Γ-formula $F'(\vec{v})$. To compute the blocks of $M_{x,y}^{F(\vec{u})}$, we can use Claim 1 to compute for every $x = a$ and $y = b$ whether $|\mathrm{sol}(M_{x,y}^{F'(\vec{v})}[a, b])| > 0$ and hence $M_{x,y}^{F(\vec{u})} \neq \emptyset$. Claim 3 is a bit more subtle and deferred to the full version [5]. ◀

We close this section by stating the following property that applies only to *uniformly* decomposable constraints.

▶ **Lemma 16.** *Let $R(\vec{u})$ be a constraint that is uniformly blockwise decomposable in x and y. Then there exist $\vec{v}$ and $\vec{w}$ with $\tilde{u} = \{x, y\} \dot\cup \tilde{v} \dot\cup \tilde{w}$ such that*

$$R(\vec{u}) = \pi_{x,\tilde{v}} R(\vec{u}) \wedge \pi_{y,\tilde{w}} R(\vec{u}) \wedge \pi_{x,y} R(\vec{u}). \tag{9}$$

Furthermore, if R is defined by a pp-formula F over a strongly uniformly blockwise decomposable Γ, then $\vec{v}$ and $\vec{w}$ can be computed from F in polynomial time.

5 Algorithms

5.1 Polynomial time construction of ODDs for strongly uniformly blockwise decomposable constraint languages

The key to the efficient construction of ODD for uniformly blockwise decomposable constraints is the following lemma, which states that any such constraint is equivalent to a treelike conjunction of binary projections of itself.

▶ **Lemma 17** (Tree structure lemma). *Let $R(\vec{u})$ be a constraint that is uniformly blockwise decomposable. Then there is an undirected tree T with vertex set $V(T) = \tilde{u}$ such that*

$$R(\vec{u}) = \bigwedge_{\{p,q\} \in E(T)} \pi_{\{p,q\}}(R(\vec{u})).$$

Furthermore, T can be calculated in polynomial time in $\|F\|$, if R is uniformly blockwise decomposable and given as pp-formula F.

Note that from Lemma 17 is follows in particular that any uniformly decomposable constraint is a conjunction of binary constraints. Thus, it follows from Theorem 9, our main result for ODD-representations, that any constraint language that allows efficient ODD-representations can essentially only consist of binary constraints. Hence, for example affine constraints, for which it is known that they allow efficient counting [19], are hard in our setting.

Proof of Lemma 17. We first fix x and y arbitrarily and apply Lemma 16 to obtain a tri-partition $(\tilde{v}, \{x, y\}, \tilde{w})$ of $\tilde{u}$ such that $R(\vec{u}) = \pi_{x,\tilde{v}}(R(\vec{u})) \wedge \pi_{x,y}(R(\vec{u})) \wedge \pi_{y,\tilde{w}}(R(\vec{u}))$. We add the edge $\{x, y\}$ to T. By Lemma 11, $\pi_{x,\tilde{v}}(R(\vec{u}))$ and $\pi_{y,\tilde{w}}(R(\vec{u}))$ are uniformly blockwise decomposable, so Lemma 16 can be recursively applied on both projections. For (say) $\pi_{x,\vec{v}}(R(\vec{u}))$ we fix x, choose an arbitrary $z \in \tilde{v}$, apply Lemma 16, and add the edge $\{x, z\}$ to T. Continuing this construction recursively until no projections with more than two variables are left yields the desired result. ◀

From this tree structure we will construct small ODDs by starting with a centroid, i. e., a variable whose removal splits the tree into connected components of at most $n/2$ vertices each. From the tree structure lemma it follows that we can handle the (projection on the) subtrees independently. A recursive application of this idea leads to an ODD of size $O(n^{\log|D|+1})$.

Proof of part 1 in Theorem 9. Let I be a CSP(Γ)-instance and $F_I(\vec{u})$ the corresponding Γ-formula. By Lemma 17 we can compute a tree T such that $F_I(\vec{u}) = \bigwedge_{\{p,q\} \in E(T)} \pi_{\{p,q\}}(F_I(\vec{u}))$. By Corollary 15.1 we can explicitly compute, for each $\{p, q\} \in E(T)$, a binary relation $R_{\{p,q\}} \subseteq D^2$ such that $R_{\{p,q\}}(p, q) = \pi_{\{p,q\}}(F_I(\vec{u}))$. Now we define the formula $F_T(\vec{u}) = \bigwedge_{\{p,q\} \in E(T)} R_{\{p,q\}}(p, q)$ and note that $\mathrm{sol}(F_T(\vec{u})) = \mathrm{sol}(F_I(\vec{u}))$. It remains to show that such tree-CSP instances can be efficiently compiled to ODDs. This follows from the following inductive claim, where for technical reasons we also add unary constraints $U_{D_v}(v)$ for each vertex v (setting $D_v := D$ implies the theorem).

▷ Claim. Let T be a tree on n vertices and $F_T(\vec{u}) = \bigwedge_{\{v,w\} \in E(T)} R_{\{v,w\}}(v, w) \wedge \bigwedge_{v \in \tilde{u}} U_{D_v}(v)$ be a formula. Then there is an order $<$, depending only on T, such that an $\mathrm{ODD}_<$ of size at most $f(n) := n|D|^{\log(n)}$ deciding $F_T(\vec{u})$ can be computed in $n^{O(1)}$.

We proof the claim by induction on n and the start $n = 1$ is trivial. If $n \geq 2$ let z be a centroid in this tree, that is a node whose removal splits the tree into $\ell \geq 1$ connected components $T_1, \ldots, T_\ell$ of at most $n/2$ vertices each. Let $\vec{v}_1, \ldots, \vec{v}_\ell$ be vectors of the variables in these components, so $(\{z\}, \tilde{v}_1, \ldots, \tilde{v}_\ell)$ partitions $\tilde{u}$. Let $x_i \in V(T_i)$ be the neighbors of z in T. We want to branch on z and recurse on the connected components T_i. To this end, for each assignment $z \mapsto a$ we remove for each neighbor x_i those values that cannot be extended to $z \mapsto a$. That is, $D^a_{x_i} := \{b \colon \{x_i \mapsto b, z \mapsto a\} \in \mathrm{sol}(U_{D_{x_i}}(x_i) \wedge R_{\{x_i,z\}}(x_i, z) \wedge U_{D_z}(z))\}$. Now we let $F^a_i(\vec{v}_i) := \bigwedge_{\{v,w\} \in E(T_i)} R_{\{v,w\}}(v, w) \wedge \bigwedge_{v \in \tilde{v}_i \setminus \{x_i\}} U_{D_v}(v) \wedge U_{D^a_{x_i}}(x_i)$ and observe that

$$\mathrm{sol}(F_T(\vec{u})) = \dot{\bigcup}_{a \in D} \mathrm{sol}(U_a(z) \times F^a_1(\vec{v}_1) \times \cdots \times F^a_\ell(\vec{v}_\ell)). \tag{10}$$

By induction assumption, for each $i \in [\ell]$ there is an order $<_i$ of $\tilde{v}_i$ such that each $F_i^a(\vec{v}_i)$ has an $\mathrm{ODD}^a_{<_i}$ of size $f(n_i)$ for $n_i := |\tilde{v}_i|$. Now we start our ODD for $F_T(\vec{u})$ with branching on z followed by the sequential combination of $\mathrm{ODD}^a_{<_1}, \ldots, \mathrm{ODD}^a_{<_\ell}$ for each assignment $a \in D$ to z. This completes the inductive construction. Since its size is bounded by $1 + |D| \sum_{i\in[\ell]} f(n_i)$, the following easy estimations finish the proof of the claim (recall that $\sum_{i\in[\ell]} n_i = n - 1$): $1 + |D| \sum_{i\in[\ell]} f(n_i) \;\leq\; 1 + |D| \sum_{i\in[\ell]} n_i |D|^{\log(n/2)} = 1 + |D|^{\log(n)}(n-1) \;\leq\; n|D|^{\log(n)}$ ◀

5.2 Polynomial size FDDs for strongly blockwise decomposable constraint languages

For blockwise decomposable constraints that are *not* uniformly blockwise decomposable, a good variable order may depend on the values assigned to variables that are already chosen, so it is not surprising that the tree approach for ODDs does not work in this setting.

For the construction of the FDD, we first compute the indecomposable factors (this can be done by Corollary 15.3 and treat them independently. This, of course, could have also been done for the ODD construction. The key point now is how we treat the indecomposable factors: every selection matrix $M^R_{x,y}$ for a (blockwise decomposable) indecomposable constraint necessarily has two non-empty blocks. But then every row $x = a$ must have at least one empty entry $\mathrm{sol}(M^R_{x,y}[a, b]) = \emptyset$. This in turn implies that, once we have chosen $x = a$, we can exclude b as a possible value for y! As we have chosen y arbitrarily, this also applies to any other variable (maybe with a different domain element b). So the set of possible values for every variable shrinks by one and since the domain is finite, this cannot happen too often. Algorithm 1 formalizes this recursive idea. To bound the runtime of this algorithm, we analyze the size of the recursion tree; the details are found in [5].

6 Lower Bounds

In this section, we will prove the lower bounds of Theorem 9 and Theorem 10. In the proofs, we will use the approach developed in [8] that makes a connection between DNNF size and rectangle covers. We will use the following variant:

▶ **Lemma 18.** *Let O be a DNNF of size s representing a constraint $R(\vec{x})$ and let $Z \subseteq \tilde{x}$. Then there is a Z-balanced rectangle cover of f is size s. Moreover, if O is structured, then the rectangles in the cover are all with respect to the same variable partition.*

The proof of Lemma 18 is very similar to existing proofs in [8] and given in the full version [5].

6.1 Lower Bound for DNNF

In this Section, we show the lower bound for Theorem 10 which we reformulate here.

▶ **Proposition 19.** *Let Γ be a constraint language that is not strongly blockwise decomposable. Then there Γ-formulas F_n of size $\Theta(n)$ and $\varepsilon > 0$ such that any DNNF for F_n has size at least $2^{\varepsilon\|F_n\|}$.*

In the remainder of this section, we show Proposition 19, splitting the proof into two cases. First, we consider the case where $M^R_{x,y}$ is not a proper block matrix.

▶ **Lemma 20.** *Let $R(x, y, \vec{z})$ be a constraint such that $M^R_{x,y}$ is not a proper block matrix. Then there is a family of $\{R\}$-formulas F_n and $\varepsilon > 0$ such that any DNNF for F_n has size at least $2^{\varepsilon\|F_n\|}$.*

Algorithm 1 FDD construction algorithm.

Input: Γ-formula $F(x_1, \dots, x_n)$ for strongly blockwise decomposable Γ over domain D.
Output: An FDD deciding $F(x_1, \dots, x_n)$.

Initialize variable domains $D_{x_i} \leftarrow D$ for $i = 1, \dots, n$.
return ConstructFDD($F(x_1, \dots, x_n)$; $D_{x_1}, \dots, D_{x_n}$)

procedure ConstructFDD($R(x_1, \dots, x_n)$; $D_{x_1}, \dots, D_{x_n}$)
    $R(x_1, \dots, x_n) \leftarrow R(x_1, \dots, x_n) \wedge \bigwedge_{i \in [n]} U_{D_{x_i}}(x_i)$
    if $n = 1$ **then**
        return 1-node FDD deciding $R(x_1)$, branching on all values D_{x_1}.
    Compute the indecomposable factors $R_1(\vec{u}_1)$, …, $R_m(\vec{u}_m)$ of $R(x_1, \dots, x_n)$.
    if $m \geq 2$ **then**
        for $i = 1 \dots m$ **do** $\mathrm{FDD}_i \leftarrow$ ConstructFDD($R_1(\vec{u}_1)$; D_y for $y \in \tilde{u}_i$)
        return Sequential composition of FDD_1, …, FDD_m
    else
        Introduce branching node for x_1.
        for $a \in D_{x_1}$ **do**
            for $i = 2, \dots, n$ **do**
                for $b \in D_{x_i}$ **do**
                    if $\mathrm{sol}(M^R_{x_1,x_i}[a, b]) = \emptyset$ **then**
                        $D_{x_i} \leftarrow D_{x_i} \setminus \{b\}$ ▷ This happens for at least one $b \in D_{x_i}$.
            $S_a(x_2, \dots, x_n) \leftarrow \pi_{x_2,\dots,x_n}(R(x_1, \dots, x_n)|_{x_1=a})$
            $\mathrm{FDD}^a \leftarrow$ ConstructFDD($S_a(x_2, \dots, x_n)$; $D_{x_2}, \dots, D_{x_n}$)
        Connect $x \xrightarrow{a} \mathrm{FDD}^a$ for all $a \in D_{x_1}$ and **return** resulting FDD

In the proof of Lemma 20, we will use a specific family of graphs. We remind the reader that a matching is a set of edges in a graph that do not share any end-points. The matching is induced if the graph induced by the end-points of the matching contains exactly the edges of the matching.

▶ **Lemma 21.** *There is an integer d and a constant $\varepsilon > 0$ such that there is an infinite family (G_n) of d-regular, bipartite graphs such that for each set $X \subseteq V(G_n)$ with $|X| \leq n = |V(G)|/2$ there is an induced matching of size at least $\varepsilon|X|$ in which each edge has exactly one endpoint in X.*

Proof of Lemma 20. If $M^R_{x,y}$ is not a proper block matrix, then, by Lemma 5, the matrix $M^R_{x,y}$ has a 2×2-submatrix with exactly three non-empty entries. So let $a, b, c, d \in D$ such that $M^R_{x,y}(b, d) = \emptyset$ and $M^R_{x,y}(a, c)$, $M^R_{x,y}(a, d)$ and $M^R_{x,y}(b, c)$ are all non-empty.

We describe a construction that to every bipartite graph $G = (A, B, E)$ gives a formula $F(G)$ as follows: for every vertex $u \in A$, we introduce a variable x_u and for every vertex $v \in B$ we introduce a variable x_v. Then, for every edge $e = uv \in E$ where $u \in A$ and $v \in V$, we add a constraint $R(x_u, x_v, \vec{z}_e)$ where $\vec{z}_e$ consists of variables only used in this constraint. We fix the notation $X_A := \{x_v \mid v \in A\}$, $X_B := \{x_v \mid v \in B\}$ and $X := X_A \cup X_B$.

Let (F_n) be the family of formulas defined by $F_n = F(G_n)$ where (G_n) is the family from Lemma 21. Clearly, $\|F_n\| = \Theta(|E(G_n)|) = \Theta(n)$, as required. Fix n for the remainder of the proof. Let F'_n be the formula we get from F_n by restricting all variables $x_u \in X_A$ to $\{a, b\}$ and all variables $x_v \in X_B$ to $\{c, d\}$ by adding some unary constraints. Let $\mathcal{R}$ be an X-balanced rectangle cover of F'_n. We claim that the size of $\mathcal{R}$ is at least $2^{\varepsilon' n}$, where

$$\varepsilon' = \frac{1}{3} \cdot \varepsilon \cdot \log_2 \left(1 + \frac{1}{2^{d+1}|R|^{d^2}}\right)$$

and d is the degree of G_n. To prove this, we first show that for every $\mathfrak{r} \in \mathcal{R}$, $2^{\varepsilon'|X|} \cdot |\operatorname{sol}(\mathfrak{r})| \leq |\operatorname{sol}(F'_n)|$. So let $\mathfrak{r}(\vec{v},\vec{w}) = \mathfrak{r}_1(\vec{v}) \times \mathfrak{r}_2(\vec{w}) \in \mathcal{R}$. Since $\mathfrak{r}$ is an X-balanced rectangle, we may assume $|X|/3 \leq |\vec{v} \cap V| \leq 2|X|/3$. By choice of G_n, we have that there is an induced matching $\mathcal{M}$ in G_n of size at least $\varepsilon|X|/3$ consisting of edges that have one endpoint corresponding to a variable in $\tilde{v}$ and one endpoint corresponding to a variable in $\tilde{w}$. Consider an edge $e = uv \in \mathcal{M}$. Assume that $x_u \in \tilde{v}$ and $x_v \in \tilde{w}$. Since we have $\mathfrak{r}(\vec{v},\vec{w}) = \mathfrak{r}(\vec{v}) \times \mathfrak{r}(\vec{w})$, we get

$$\pi_e \mathfrak{r}(\vec{v},\vec{w}) = \pi_{\tilde{v}\cap e}(r(\vec{v})) \times \pi_{\tilde{w}\cap e}(r(\vec{w})).$$

By construction $\pi_e\mathfrak{r}(\vec{v},\vec{w}) \subseteq \{(a,c),(a,d),(b,c)\}$, so it follows that either $\pi_e\mathfrak{r}(\vec{v},\vec{w}) \subseteq \{(a,c),(a,d)\} = \{a\}\times\{c,d\}$ or $\pi_e\mathfrak{r}(\vec{v},\vec{w}) \subseteq \{(a,c),(b,c)\} = \{a,b\}\times\{c\}$. Assume w.l.o.g. that $\{b,c\} \notin \pi_e r(\vec{v},\vec{w})$ (the other case can be treated analogously). It follows that for each solution $\beta \in \operatorname{sol}(r)$, we get a solution $q(\beta) \in \operatorname{sol}(F_n) \setminus \operatorname{sol}(\mathfrak{r}(\vec{v},\vec{w}))$ by setting

- $q(\beta)(x_u) := b$,
- For all $x_\ell \in N(x_u)$, we set $q(\beta)(x_\ell) := c$,
- For all $x_\ell \in N(x_u)$ and all $x_m \in N(y_\ell)$ we set $q(\beta)(\vec{z}_{m\ell}) := \vec{g}$ where $\vec{g}$ is such that $(b,c,\vec{g}) \in R$.

Note that values $\vec{g}$ exist because $M^R_{x,y}(b,c)$ is non-empty. Observe that for two different solutions β and β' the solutions $q(\beta)$ and $q(\beta')$ may be the same. However, we can bound the number $\left|q^{-1}(q(\beta))\right|$, giving a lower bound on the set of solutions not in r. To this end, suppose that $q(\beta) = q(\beta')$. Since q only changes the values of x_u, exactly d x_ℓ-variables and at most d^2 vectors of $\vec{z}$-variables (the two latter bounds come from the degree bounds on G_n), $q(\beta) = q(\beta')$ implies that β and β' coincide on all other variables. This implies

$$\left|q^{-1}(q(\beta))\right| \leq 2^{d+1}|R|^{d^2},$$

because there are only that many possibilities for the variables that q might change. By considering $\{x_u, y_v\}$, we have shown that

$$|\operatorname{sol}(F_n)| \geq |\operatorname{sol}(r)| + \frac{1}{2^{d+1}|R|^{d^2}}|\operatorname{sol}(r)|.$$

So we have constructed $\frac{|\operatorname{sol}(r)|}{2^{d+1}|R|^{d^2}}$ solutions not in r. Now we consider not only one edge but all possible subsets I of edges in $\mathcal{M}$: for a solution $\beta \in \operatorname{sol}(\mathfrak{r}(\vec{v},\vec{w}), \vec{w})$, the assignment $q_I(\beta)$ is constructed as the q above, but for all edges $e \in I$. Reasoning as above, we get

$$\left|q_I^{-1}(q_I(\beta))\right| \leq \left(2^{d+1}|R|^{d^2}\right)^{|I|}.$$

It is immediate to see that $q_I(\beta) \neq q_{I'}(\beta)$ for $I \neq I'$. Thus we get

$$\begin{aligned}|\operatorname{sol}(F'_n)| &\geq \sum_{m=0}^{\frac{1}{3}\varepsilon|X|} \binom{\frac{1}{3}\varepsilon|X|}{m} \left(\frac{1}{2^{d+1}|R|^{d^2}}\right)^m |\operatorname{sol}(r)| \\ &= \left(1 + \frac{1}{2^{d+1}|R|^{d^2}}\right)^{\frac{1}{3}\varepsilon|X|} |\operatorname{sol}(r)| = 2^{\varepsilon' n}|\operatorname{sol}(r)|.\end{aligned}$$

It follows that every X-balanced rectangle cover has to have a size of at least $2^{\varepsilon'|X|}$. With Lemma 18 and Lemma 1 it follows that any DNNF for F_n has to have a size of at least $2^{\varepsilon' n}$. ◀

Now we consider the case that $M^R_{x,y}$ is a proper block matrix, but R is not blockwise decomposable in some pair of variables x, y.

▶ **Lemma 22.** *Let $R(x, y, \vec{z})$ be a relation such that $M^R_{x,y}$ is a proper block matrix but $R(x, y, \vec{z})$ in not blockwise decomposable in x and y. Then there is a family of formulas F_n and $\varepsilon > 0$ such that a DNNF for F_n needs to have a size of at least $2^{\varepsilon \|F_n\|}$.*

The proof of Lemma 22 is a slight variant of that of Lemma 20, see [5].

We now have everything in place to prove Proposition 19.

Proof of Proposition 19. Since Γ is not strongly blockwise decomposable, there is a relation $R \in \langle\Gamma\rangle$ that is not blockwise decomposable in x and y. Then, by definition of co-clones and Lemma 11, there is a Γ-formula that defines R. If there are variables $x, y \in \tilde{x}$ such that $M^R_{x,y}$ is not a proper block matrix, then we can apply Lemma 20 to get R-formulas that require exponential size DNNF. Then by substituting all occurrences of R in these formula by the Γ-formula defining R, we get the required hard Γ-formulas. If all $M^R_{x,y}$ are proper block matrices, then there are variables x, y such that $M^R_{x,y}$ is not blockwise decomposable. Using Lemma 22 and reasoning as before, then completes the proof. ◀

6.2 Lower Bound for structured DNNF

In this section, we prove the lower bound of Theorem 9 which we formulate here again.

▶ **Proposition 23.** *Let Γ be a constraint language that is not strongly uniformly blockwise decomposable. Then there Γ-formulas F_n of size $Theta(n)$ and $\varepsilon > 0$ such that any structured DNNF for F_n has size at least $2^{\varepsilon \|F_n\|}$.*

Note that for all constraint languages that are not strongly blockwise decomposable, the result follows directly from Proposition 19, so we only have to consider constraint languages which are strongly blockwise decomposable but not strongly uniformly blockwise decomposable. We start with a simple observation.

▶ **Observation 24.** *Let $\mathfrak{r}(\vec{x}, \vec{y})$ be a rectangle with respect to the partition $(\tilde{x}, \tilde{y})$. Let $Z \subseteq \tilde{x} \cup \tilde{y}$, then $\pi_Z(\mathfrak{r}(\tilde{x}, \tilde{y}))$ is a rectangle with respect to the partition $(\tilde{x} \cap Z, \tilde{y} \cap Z)$.*

We start our proof of Proposition 23 by considering a special case.

▶ **Lemma 25.** *Let $R(x, y, \vec{z})$ be a constraint such that there are two assignments $a, b \in \mathrm{sol}(R)$ such that for every partition $\tilde{z}_1, \tilde{z}_2$ of $\tilde{z}$ we have that $a|_{\{x\}\cup\tilde{z}_1} \cup b|_{\{y\}\cup\tilde{z}_2} \notin \mathrm{sol}(R)$ or $a|_{\{y\}\cup\tilde{z}_2} \cup b|_{\{x\}\cup\tilde{z}_1} \notin \mathrm{sol}(R)$. Consider*

$$F_n := \bigwedge_{i\in[n]} R(x_i, y_i, \vec{z}_i)$$

where the $\vec{z}_i$ are disjoint variable vectors. Let $(\tilde{x}, \tilde{y})$ be a variable partition for F_n and $\mathcal{R}$ be a rectangle cover of F_n such that each rectangle $\mathfrak{r}_j$ in $\mathcal{R}$ respects the partition $(\tilde{x}, \tilde{y})$. If for all $i \in [n]$ we have that all $x_i \in \tilde{x}$ and $y_i \in \tilde{y}$ or $x_i \in \tilde{y}$ and $y_i \in \tilde{x}$, then $\mathcal{R}$ has size at least 2^n.

Proof. We use the so-called fooling set method from communication complexity, see e.g. [32, Section 1.3]. To this end, we will construct a set $\mathcal{S}$ of satisfying assignments of F_n such that every rectangle of $\mathcal{R}$ can contain at most one assignment in $\mathcal{S}$.

So let a_i be the assignment to $\{x_i, y_i\} \cup \tilde{z}_i$ that assigns the variables analogously to a, so $a_i(x_i) := a(x)$, $a_i(y_i) := a(y)$, and $a_i(\vec{z}_i) := a(\vec{z})$. Define analogously b_i. Then the set $\mathcal{S}$ consists of all assignments that we get by choosing for every $i \in [n]$ an assignment d_i as either a_i or b_i and then combining the d_i to one assignment to all variables of F_n. Note that $\mathcal{S}$ contains 2^n assignments and that all of them satisfy F_n, so all of them must be in a rectangle of $\mathcal{R}$.

We claim that none of the rectangles $\mathfrak{r}_j$ of $\mathcal{R}$ can contain more than one element $d \in \mathcal{S}$. By way of contradiction, assume this were not true. Then there is an $\mathfrak{r}_j$ that contains two assignments $d, d' \in \mathcal{S}$. Then there is an $i \in [n]$ such that in the construction of d we have chosen a_i while in the construction of d' we have chosen b_i. Let $\tilde{x}_j := \tilde{z}_j \cap \tilde{x}$ and $\tilde{y}_j := \tilde{z}_j \cap \tilde{y}$. Since $d, d' \in \mathrm{sol}(\mathfrak{r}_j)$, we have that $a_i, b_i \in \mathrm{sol}(\pi_{\{x_i,y_i\}\cup\tilde{x}_j\cup\tilde{y}_j}(\mathfrak{r}_j))$. Moreover, by Observation 24, $\pi_{\{x_i,y_i\}\cup\tilde{x}_j\cup\tilde{y}_j}(\mathfrak{r}_j)$ is a rectangle and so we have that $a_i|_{\{x_i\}\cup\tilde{x}_j} \cup b_i|_{\{y_i\}\cup y_j} \in \mathrm{sol}(\pi_{\{x_i,y_i\}\cup\tilde{x}_j\cup\tilde{y}_j}(\mathfrak{r}_j))$ and $a_i|_{\{y_i\}\cup y_j} \cup b_i|_{\{x_i\}\cup y_j} \in \mathrm{sol}(\pi_{\{x_i,y_i\}\cup\tilde{x}_j\cup\tilde{y}_j}(\mathfrak{r}_j))$. But $\mathfrak{r}$ consists only of solutions of F_n and thus $\mathrm{sol}(\pi_{\{x_i,y_i\}\cup\tilde{x}_j\cup\tilde{y}_j}(\mathfrak{r})) \subseteq \mathrm{sol}(R(x_i, y_i, \vec{z}_i))$, so $a_i|_{\{x_i\}\cup\tilde{x}_j} \cup b_i|_{\{y_i\}\cup y_j}, a_i|_{\{y_i\}\cup y_j} \cup b_i|_{\{x_i\}\cup y_j} \in \mathrm{sol}(R(x_i, y_i, \vec{z}_i))$. It follows by construction that there is a partition $(\tilde{x}, \tilde{y})$ of $\tilde{z}$ such that $a|_{\{x\}\cup\tilde{x}} \cup b|_{\{y\}\cup\tilde{y}}$ and $a|_{\{y\}\cup\tilde{y}} \cup b|_{\{x\}\cup\tilde{x}}$ are in $\mathrm{sol}(R)$. This contradicts the assumption on a and b and thus $\mathfrak{r}_j$ can only contain one assignment from $\mathcal{S}$.

Since $\mathcal{S}$ has size 2^n and all of assignments in $\mathcal{S}$ must be in one rectangle of $\mathcal{R}$, it follows that $\mathcal{R}$ consists of at least 2^n rectangles. ◀

We now prove the lower bound of Proposition 23.

Proof of Proposition 23. Since Γ is not strongly uniformly blockwise decomposable, let $R(x, y, \vec{z})$ be a constraint in $\langle\Gamma\rangle$ that is not uniformly blockwise decomposable in x and y. If $R(x, y, \vec{z})$ is such that $M^R_{x,y}$ is not a proper block matrix, then the lemma follows directly from Lemma 20, so we assume in the remainder that $M^R_{x,y}$ is a proper block matrix. We denote for every block $D_1 \times D_2$ of $R(x, y, \vec{z})$ by $R^{D_1\times D_2}(x, y, \vec{z})$ the sub-constraint of $R(x, y, \vec{z})$ we get by restricting x to D_1 and y to D_2. Since $R(x, y, \vec{z})$ is not uniformly blockwise decomposable, for every partition $(\tilde{z}_1, \tilde{z}_2)$ of $\tilde{z}$ there is a block $D_1 \times D_2$ such that $R^{D_1\times D_2}(x, y, \vec{z}) \neq \pi_{\{x\}\cup\tilde{z}_1} R^{D_1\times D_2}(x, y, \vec{z}) \times \pi_{\{y\}\cup\tilde{z}_2} R^{D_1\times D_2}(x, y, \vec{z})$.

Given a bipartite graph $G = (A, B, E)$, we construct the same formula $F(G)$ as in the proof of Lemma 20. Consider again the graphs G_n of the family from Lemma 21 and let $F_n := F(G_n)$. Fix n in the remainder of the proof. Let O be a structured DNNF representing F_n of size s. Then there is by Proposition 18 a balanced partition (X_1, X_2) of $X_A \cup X_B$ such that there is a rectangle cover of F_G of size at most s and such that all rectangles respect the partition (X_1, X_2). Let E' be the set of edges $uv \in E$ such that x_u and x_v are in different parts of the partition (X_1, X_2). By the properties of G_n, there is an induced matching $\mathcal{M}$ of size $\Omega(|A| + |B|)$ consisting of edges in E'.

For every edge $e = uv \in \mathcal{M}$ let $\tilde{z}_{e,1} := \tilde{z}_e \cap X_1$ and $\tilde{z}_{e,2} := \tilde{z}_2 \cap X_2$. Assume that $x_u \in X_1$ and $y_v \in X_2$ (the other case is treated analogously). Then we know that there is a block $D_1 \times D_2$ of $M^R_{x,y}$ such that

$$R^{D_1\times D_2}(x_u, y_v, \vec{z}_e) \neq \pi_{\{x_u\}\cup\vec{z}_{e,1}} R^{D_1\times D_2}(x_u, y_v, \vec{z}_e) \times \pi_{\{y_v\}\cup\vec{z}_{e,2}} R^{D_1\times D_2}(x_u, y_v, \vec{z}_e). \quad (11)$$

Since there are only at most $|D|$ blocks in $M^R_{x,y}$, there is a block $D_1 \times D_2$ such that for at least $\Omega\left(\frac{|A|+|B|}{|D|}\right) = \Omega(|A| + |B|)$ edges Equation (11) is true. Call this set of edges E^*.

Let $X^* := \{x_u \mid u \text{ is an endpoint of an edge } e \in E^*\}$. We construct a structured DNNF O' from O by existentially quantifying all variables not in a constraint $R(x_u, y_v, \vec{z}_e)$ for $e = uv \in E^*$ and for all $x_u \in X^*$ restricting the domain to D_1 $u \in A$ and to D_2 if $u \in B$. Note that every assignment to X^* that assigns every variable x_v with $v \in A$ to a value in D_1 and every x_v with $v \in B$ to a value in D_2 can be extended to a satisfying assignment of F_n, because $D_1 \times D_2$ is a block. Thus, O' is a representation of

$$F^* := \bigwedge_{e=uv\in E^*} R^{D_1\times D_2}(x_u, y_v, \vec{z}_e).$$

We now use the following simple observation.

▷ Claim 26. Let $R'(\vec{x},\vec{y})$ be a constraint such that $R'(\vec{x},\vec{y}) \neq \pi_{\vec{x}}(R'(\vec{x},\vec{y})) \times \pi_{\vec{y}}(R'(\vec{x},\vec{y}))$. Then there are assignments $a, b \in \mathrm{sol}(R'(\vec{x},\vec{y}))$ such that $a|_{\vec{x}} \cup b|_{\vec{y}} \notin \mathrm{sol}(R'(\vec{x},\vec{y}))$ or $a|_{\vec{x}} \cup b|_{\vec{y}} \notin \mathrm{sol}(R'(\vec{x},\vec{y}))$.

Proof. Since $R(\vec{x},\vec{y}) \neq \pi_{\vec{x}}(R(\vec{x},\vec{y})) \times \pi_{\vec{y}}(R(\vec{x},\vec{y}))$ and thus $R(\vec{x},\vec{y}) \subsetneq \pi_{\vec{x}}(R(\vec{x},\vec{y})) \times \pi_{\vec{y}}(R(\vec{x},\vec{y}))$, we have that there is $a_x \in \mathrm{sol}(\pi_{\vec{x}}(R(\vec{x},\vec{y})))$ and $b_y \in \mathrm{sol}(\pi_{\vec{y}}(R(\vec{x},\vec{y})))$ such that $a|_{\vec{x}} \cup b|_{\vec{y}} \notin \mathrm{sol}(R(\vec{x},\vec{y}))$. Simply extending a_x and b_y to an assignment in $R(\vec{x},\vec{y})$ yields the claim. ◁

Since Claim 26 applies to all constraints in F^*, we are now in a situation where we can use Lemma 25 which shows that any rectangle cover respecting the partition (X_1, X_2) for F^* has size $2^{|E^*|} = 2^{\Omega(|A|+|B|)}$. With Lemma 18, we know that $\|O'\| = 2^{\Omega(|A|+|B|)}$ and since the construction of O' from O does not increase the size of the DNNF, we get $s = \|O\| = 2^{\Omega(|A|+|B|)} = 2^{\Omega(\|F^*\|)} = 2^{\Omega(\|F\|)}$. ◀

7 The Boolean Case

In this section, we will specialize our dichotomy results for the Boolean domain $\{0,1\}$. A relation R over $\{0,1\}$ is called *bijunctive affine* if it can be written as a conjunction of the relations $x = y$ and $x \neq y$ and unary relations, so $R \subseteq E$ with $E = \langle\{=, \neq, U_0, U_1\}\rangle$ where $U_0 = \{0\}$ and $U_1 = \{1\}\}$. A set of relations Γ is called bijunctive affine if all $R \in \Gamma$ are bijunctive affine. We will show the following dichotomy for the Boolean case:

▶ **Theorem 27.** *Let Γ be a constraint language. If all relations in Γ are bijunctive affine, then there is an polynomial time algorithm that, given a Γ-formula F, constructs an OBDD for F. If not, then there are formulas F_n and $\varepsilon > 0$ such that an DNNF for F_n needs to have a size of at least $2^{\varepsilon\|F_n\|}$.*

Let us remark here that, in contrast to general domains D, there is no advantage of FDD over ODD in the Boolean case: either a constraint language allows for efficient representation by ODD or it is hard even for DNNF. So in a sense, the situation over the Boolean domain is easier. Also note that the tractable cases over the Boolean domain are very restricted, allowing only equalities and disequalities.

8 Conclusion

We have seen that there is a dichotomy for compiling systems of constraints into DNNF based on the constraint languages. It turns out that the constraint languages that allow efficient compilation are rather restrictive, in the Boolean setting they consist essentially only of equality and disequality. From a practical perspective, our results are thus largely negative since interesting settings will most likely lie outside the tractable cases we have identified.

One question that we leave for future work is that of decidability. Given a constraint language, can one decide if it is (uniformly) blockwise decomposable? We remark that for the similar property in counting complexity from [27], it is known that the dichotomy criterion is decidable. But we do currently not yet understand if a similar result is possible in our setting.

References

1 Noga Alon, Mrinal Kumar, and Ben Lee Volk. Unbalancing sets and an almost quadratic lower bound for syntactically multilinear arithmetic circuits. *Comb.*, 40(2):149–178, 2020. doi:10.1007/s00493-019-4009-0.

2 Antoine Amarilli, Pierre Bourhis, Louis Jachiet, and Stefan Mengel. A circuit-based approach to efficient enumeration. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 111:1–111:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.111.

3 Antoine Amarilli, Florent Capelli, Mikaël Monet, and Pierre Senellart. Connecting knowledge compilation classes and width parameters. *Theory Comput. Syst.*, 64(5):861–914, 2020. doi:10.1007/s00224-019-09930-2.

4 Jérôme Amilhastre, Hélène Fargier, Alexandre Niveau, and Cédric Pralet. Compiling CSPs: A complexity map of (non-deterministic) multivalued decision diagrams. *Int. J. Artif. Intell. Tools*, 23(4), 2014. doi:10.1142/S021821301460015X.

5 Christoph Berkholz, Stefan Mengel, and Hermann Wilhelm. A characterization of efficiently compilable constraint languages. *CoRR*, abs/2311.10040, 2023. doi:10.48550/ARXIV.2311.10040.

6 Christoph Berkholz and Harry Vinall-Smeeth. A dichotomy for succinct representations of homomorphisms. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 113:1–113:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.113.

7 Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Expander CNFs have exponential DNNF size. *CoRR*, abs/1411.1995, 2014. arXiv:1411.1995.

8 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.

9 Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986. doi:10.1109/TC.1986.1676819.

10 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013. doi:10.1145/2528400.

11 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.

12 Andrei A. Bulatov, Víctor Dalmau, Martin Grohe, and Dániel Marx. Enumerating homomorphisms. *J. Comput. Syst. Sci.*, 78(2):638–650, 2012. doi:10.1016/J.JCSS.2011.09.006.

13 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. *J. ACM*, 64(3):19:1–19:39, 2017. doi:10.1145/2822891.

14 Florent Capelli. Understanding the complexity of #SAT using knowledge compilation. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–10. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005121.

15 Florent Capelli and Stefan Mengel. Tractable QBF by Knowledge Compilation. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2019.18.

16 Clément Carbonnel, Miguel Romero, and Stanislav Zivný. The complexity of general-valued constraint satisfaction problems seen from the other side. *SIAM J. Comput.*, 51(1):19–69, 2022. doi:10.1137/19M1250121.

17 Arthur Choi and Adnan Darwiche. Dynamic minimization of sentential decision diagrams. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, pages 187–194. AAAI Press, 2013. doi:10.1609/aaai.v27i1.8690.

18 Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995. doi:10.1006/jcss.1995.1087.

19 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996. doi:10.1006/inco.1996.0016.

20 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM, 2001.

21 Nadia Creignou, Frédéric Olive, and Johannes Schmidt. Enumerating all solutions of a boolean CSP by non-decreasing weight. In Karem A. Sakallah and Laurent Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, volume 6695 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2011. doi:10.1007/978-3-642-21581-0_11.

22 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004. doi:10.1016/j.tcs.2004.08.008.

23 Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001. doi:10.1145/502090.502091.

24 Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In Ramón López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 328–332. IOS Press, 2004.

25 Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002. doi:10.1613/jair.989.

26 Guy Van den Broeck and Dan Suciu. Query processing on probabilistic data: A survey. *Found. Trends Databases*, 7(3-4):197–341, 2017. doi:10.1561/1900000052.

27 Martin E. Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013. doi:10.1137/100811258.

28 Hélène Fargier and Pierre Marquis. On the use of partially ordered decision graphs in knowledge compilation and quantified boolean formulae. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 42–47. AAAI Press, 2006. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-007.php.

29 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), March 2007. doi:10.1145/1206035.1206036.

30 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

31 Frédéric Koriche, Jean-Marie Lagniez, Pierre Marquis, and Samuel Thomas. Compiling constraint networks into multivalued decomposable decision graphs. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 332–338. AAAI Press, 2015. URL: http://ijcai.org/papers15/Abstracts/IJCAI15-053.html.

32 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

33 Jean-Marie Lagniez and Pierre Marquis. An improved decision-dnnf compiler. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017. doi:10.24963/ijcai.2017/93.

34 Robert Mateescu and Rina Dechter. Compiling constraint networks into AND/OR multi-valued decision diagrams (aomdds). In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 329–343. Springer, 2006. doi:10.1007/11889205_25.

35 Robert Mateescu, Rina Dechter, and Radu Marinescu. AND/OR multi-valued decision diagrams (aomdds) for graphical models. *J. Artif. Intell. Res.*, 33:465–519, 2008. doi:10.1613/jair.2605.

36 Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-dnnf compilation with sharpSAT. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence - 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings*, volume 7310 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012. doi:10.1007/978-3-642-30353-1_36.

37 Dan Olteanu. Factorized databases: A knowledge compilation perspective. In Adnan Darwiche, editor, *Beyond NP, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016*, volume WS-16-05 of *AAAI Technical Report*. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12638.

38 Dan Olteanu and Jakub Závodný. Size bounds for factorised representations of query results. *ACM Trans. Database Syst.*, 40(1):2:1–2:44, 2015. doi:10.1145/2656335.

39 Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3141–3148. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/443.

40 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.

41 Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. Comput.*, 38(4):1624–1647, 2008. doi:10.1137/070707932.

42 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

43 Johan Thapper and Stanislav Zivný. The complexity of finite-valued csps. *J. ACM*, 63(4):37:1–37:33, 2016. doi:10.1145/2974019.

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

45 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.