Characterizing Arithmetic Circuit Classes by Constraint Satisfaction Problems

Stefan Mengel*

Institute of Mathematics, University of Paderborn, D-33098 Paderborn, Germany
stefan.mengel@mail.uni-paderborn.de

Abstract. We explore the expressivity of constraint satisfaction problems (CSPs) in the arithmetic circuit model. While CSPs are known to yield VNP-complete polynomials in the general case, we show that for different restrictions of the structure of the CSPs we get characterizations of different arithmetic circuit classes. In particular we give the first natural non-circuit characterization of VP, the class of polynomial families efficiently computable by arithmetic circuits.

Acknowledgements. I am very grateful to my supervisor Peter Bürgisser for many helpful discussions and his support in making the presentation of this paper much clearer. I would also like to thank the organizers of the Dagstuhl Seminar 10481 "Computational Counting" where some of the results in this paper were conceived.

* supported by the Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo) and DFG grant BU 1371/3-1

1 Introduction and related work

The complexity class VP has a very natural definition: It is the class of families of polynomials computable by arithmetic circuits efficiently, i.e., by families of arithmetic circuits of polynomial size. Despite this apparent naturality there is one irritating aspect in which VP differs from other arithmetic circuit classes: There are no known natural complete problems for VP – artificial ones can be constructed – and no known natural characterizations of VP that do not in one form or another depend on circuits. This puzzling feature of VP raises the question whether VP is indeed the right class for measuring natural efficient computability. This scepticism is further strengthened by the fact that Malod and Portier [MP08] have shown that many natural problems from linear algebra are complete for VP_{ws}, a subclass of VP. Thus the search for complete problems or natural characterizations of VP is an interesting and meaningful problem in algebraic complexity. In this paper we give such a natural characterization of VP and other classes by constraint satisfaction problems.

Constraint satisfaction problems (CSPs) are a classical problem in complexity theory and among the first shown to be NP-complete. In a seminal paper Schaefer [Sch78] characterized the complexity of boolean CSPs by showing a famous dichotomy theorem: if all constraints are chosen from a small class which he completely describes, then the corresponding CSP is in P, otherwise it is NP-complete. This result has spawned several follow up results, in one of which Briquel and Koiran [BK09] gave a similar dichotomy result in the arithmetic circuit model. To a family (Φ_n) of CSPs they assign a polynomial family $(P(\Phi_n))$. They show that there is a small set S of constraints with the following property: If a family (Φ_n) of CSPs is built of constraints in S only, then $(P(\Phi_n)) \in \mathsf{VP}$. On the other hand if CSPs may be constructed with the help of any constraint not in S, one can construct such a CSP-family (Φ_n) such that $(P(\Phi_n))$ is VNP-complete.

Because CSPs are immensely important for practical purposes, researchers especially in database theory and AI tried to circumvent Schaefer's result by finding feasible subclasses of CSPs. The key idea here is not to restrict the individual constraints but instead to restrict the structure of the CSPs built with these constraints. It was shown [BFMY83] that if one restricts the problem to so called acyclic CSPs, then the resulting CSPs are solvable in P. It was even shown that acyclic CSPs are parallelizable, but the exact complexity of the problem was open for some time. Gottlob et. al [GLS01] solved this question by proving that acyclic CSPs are complete for the class LOGCFL. This result easily extends to CSPs of bounded treewidth.

Treewidth is a crucial graph parameter for many algorithmic problems on graphs. Often hard problems become feasible if one bounds the treewidth of the inputs by a constant. During the last years treewidth has found its way into arithmetic circuit complexity. This was started by Courcelle et al. [CMR01] who showed that generating functions of graph problems expressible in monadic second order logic have small arithmetic circuits for graphs of bounded treewidth. This line of research was continued by Flarup et al. [LKF07] who improved these

upper bounds and showed matching lower bounds for some familes of polynomials: On the one hand the permanent and the hamilton polynomial on graphs of bounded treewidth can be computed by arithmetic formulas of polynomial size. On the other hand all arithmetic formulas can be expressed this way. Briquel, Koiran and Meer [BKM11,KM08] – building on a paper by Fischer et al. [FMR08] which deals with counting problems – considered polynomials defined by CNF formulas of bounded treewidth (see also Section 3).

In this paper we unify these different lines of work: We complement the general infeasibility results of Briquel and Koiran [BK09] by showing feasible subclasses of polynomials assigned to CSPs. In this respect the results in this paper correspond to the results of Gottlob et al. [GLS01] in the boolean model. Also, our paper can be seen as an extension of the work of Briquel, Koiran and Meer [KM08,BKM11] by generalization from CNF-formulas to general CSPs. We introduce two kinds of polynomials for CSPs and show that they characterize the hierarchy $\mathsf{VP}_e \subseteq \mathsf{VP}_{ws} \subseteq \mathsf{VP} \subseteq \mathsf{VNP}$ of arithmetic circuit classes commonly considered (cf Section 2.1), respectively, for different classes of CSPs. Boolean bounded treewidth or pathwidth CSPs capture VP_e, while in the non-boolean case we get VP_{ws} for bounded pathwidth and VP for bounded treewidth. We also explain where exactly the difference in expressivity between boolean and non-boolean CSPs comes from. We prove that if each variable can take only a constant number of values in satisfying assignments of each constraint in non-boolean CSPs, then these CSPs capture VP_e again. In boolean CSPs each variable trivially takes only at most 2 values in the satisfying assignments of each constraint. This explains that non-boolean CSPs are more powerful, simply because the variables can take more values in satisfying assignments of the constraints.

2 Preliminaries

2.1 Arithmetic circuit complexity

We briefly recall the relevant definitions from arithmetic circuit complexity. A more thorough introduction into arithmetic circuit classes can be found in the book by Bürgisser [Bür00]. Newer insights into the nature of VP and especially VP_{ws} are presented in the excellent paper of Malod and Portier [MP08].

An *arithmetic circuit* over a field $\mathbb{F}$ is a labeled directed acyclic graph (DAG) consisting of vertices or gates with indegree or fanin 0 or 2. The gates with fanin 0 are called input gates and are labeled with constants from $\mathbb{F}$ or variables $X_1, X_2, \ldots, X_n$. The gates with fanin 2 are called computation gates and are labeled with $\times$ or $+$.

The polynomial computed by an arithmetic circuit is defined in the obvious way: An input gate computes the value of its label, a computation gate computes the product or the sum of its childrens' values, respectively. We assume that a circuit has only one sink which we call output gate. We say that the polynomial computed by the circuit is the polynomial computed by the output gate. The

size of an arithmetic circuit is the number of gates. The *depth* of a circuit is the length of the longest path from an input gate to the output gate in the circuit.

Sometimes we also consider circuits in which the +-gates may have unbounded fanin. We call these circuits *semi-unbounded circuits*. Observe that in semi-unbounded circuits ×-gates still have fanin 2. A circuit is called *multiplicatively disjoint* if for each ×-gate v the subcircuits that have the children of v as output-gates are disjoint. A circuit is called *skew*, if for all of its ×-gates one of the children is an input gate.

We call a sequence (f_n) of multivariate polynomials a family of polynomials or *polynomial family*. We say that a polynomial family is of polynomial degree, if there is a univariate polynomial p such that $\deg(f_n) \leq p(n)$ for each n. VP is the class of polynomial families of polynomial degree computed by families of polynomial size arithmetic circuits. VP_e is defined analogously with the circuits restricted to trees. By a classical result of Brent [Bre76], VP_e equals the class of polynomial families computed by arithmetic circuits of depth $O(\log(n))$. VP_{ws} is the class of families of polynomials computed by families of skew circuits of polynomial size. Finally, a family (f_n) of polynomials is in VNP, if there is a family $(g_n) \in \mathsf{VP}$ and a polynomial p such that $f_n(X) = \sum_{e \in \{0,1\}^{p(n)}} g_n(e, X)$ for all n where X denotes the vector $(X_1, \ldots, X_{q(n)})$ for some polynomial q.

A polynomial f is called a *projection* of g (symbol: $f \leq g$), if there are values $a_i \in \mathbb{F} \cup \{X_1, X_2, \ldots\}$ such that $f(X) = g(a_1, \ldots, a_q)$. A family (f_n) of polynomials is a p-projection of (g_n) (symbol: $(f_n) \leq_p (g_n)$), if there is a polynomial r such that $f_n \leq g_{r(n)}$ for all n. As usual we say that (g_n) is hard for an arithmetic circuit class $\mathcal{C}$ if for every $(f_n) \in \mathcal{C}$ we have $(f_n) \leq_p (g_n)$. If further $(g_n) \in \mathcal{C}$ we say that (g_n) is $\mathcal{C}$-complete.

2.2 CSPs...

Let D and X be two sets. We denote with $D^X := \{a \colon X \to D\}$ the set of functions from X to D. A *constraint* is a function $\phi \colon D^X \to \{0,1\}$ where X and D are finite sets. We call D the domain and $var(\phi) = X$ the set of variables of ϕ. We call $k = |var(\phi)|$ the arity of the constraint ϕ. If $k = 2$ we also say that ϕ is binary. An *assignment* $a \colon var(\phi) \to D$ is said to satisfy ϕ, if and only if $\phi(a) = 1$. We say that ϕ is boolean, if $D = \{0,1\}$.

A *constraint satisfaction problem (CSP)* Φ of size m in the variables $var(\Phi)$ and domain D is a set of m constraints $\{\phi_1, \ldots, \phi_m\}$ such that the domain of all ϕ_i is D and $\bigcup_{i \in [m]} var(\phi_i) = var(\Phi)$. A CSP Φ is called binary if and only if all constraints ϕ_i of Φ are binary. If $D = \{0,1\}$ we call the CSP boolean.

A CSP Φ is *satisfied* by an assignment $a \colon var(\Phi) \to D$ if for all $i = 1, \ldots m$ we have $\phi_i(a|_{var(\phi_i)}) = 1$, where $a|_{var(\phi_i)} \in D^{var(\phi)}$ is the restriction of a onto $var(\phi)$. We also say that a satisfies the constraints of Φ.

When we have an order on the variables of the CSP we sometimes identify assignments $a \colon var(\Phi) \to D$ and vectors of length $var(\Phi)$ in the obvious way by giving a value table of a. We sometimes also describe constraints by describing its satisfying assignments as a set of vectors.

A CSP defines a function $\Phi^*\colon D^{var(\Phi)} \to \{0,1\}$ by setting $\Phi^*(a) = 1$ if and only if a satisfies Φ. In a slight abuse of notation we will not distinguish between the CSP Φ and the function Φ^* in this paper, but use the same symbol Φ for both of them. It will always be clear from the context which one of the two we mean.

Many well known decision problems can be formulated as CSPs. For example 2-CNF-formulas are constraint satisfaction problems in which all constraint are of the form $a \vee b$ where a and b are literals of the form x_i or $\neg x_i$ for a variable x_i. This illustrates the fact that in general a CSP has many more variables than each of its individual constraints.

We will in the following consider families (Φ_n) of CSPs. Every Φ_n may have its own universe D_n and its own set of variables $var(\Phi_n)$. But we will always assume that the arity of all constraints in all of the CSPs Φ_n is bounded by a constant k independent of n. We say that (Φ_n) has *bounded arity* in this case. We assume bounded arity for two reasons: First, the complexity of CSPs with unbounded arity is not even well understood in the traditional decision version on Turing machines (see [Mar09]), and thus it appears difficult to adapt it to our model. Secondly if we allow our constraints to have polynomial size arity it is easy to come up with CSPs that define hard polynomials (see Section 2.3), say the permanent, and have only one single constraint. Thus restricting the structure of the CSPs – as we will do in this paper – does not make much sense for unbounded arity.

We call a family (Φ_n) of CSPs *p-bounded*, if and only if (Φ_n) has bounded arity and there is a polynomial p such that $|D_n| \leq p(n)$ and $|var(\Phi_n)| \leq p(n)$ for every n. We say that a constraint ϕ is *c-assignment bounded* if for all $x \in var(\phi)$ we have $|\{a(x) \mid a\colon var(\phi) \to D \text{ with } \phi(a) = 1\}| \leq c$, i.e., in the satisfying assignments of ϕ each variable x takes at most c values. We call a CSP c-assignment bounded if all of its constraints are c-assignment bounded. Observe that all boolean CSPs are trivially 2-assignment bounded.

We will (sometimes implicitely) use the following simple observation:

Lemma 1. *Let (Φ_n) be p-bounded family of CSPs. Then there is a p-bounded family of CSPs (Φ'_n) that defines the same family of functions such that Φ'_n is of polynomial size in n.*

Proof. Let the arity of all constraints in (Φ_n) be bounded by k and $|var(\Phi_n)| \leq p(n)$ for a polynomial p. For each n do the following: Let the constraints of Φ_n be $\phi_1, \ldots, \phi_m$. For each set $I \subseteq var(\Phi_n)$ with $|I| \leq k$ we set $\phi_I = \bigwedge_{\phi_i : var(\phi_i) = I} \phi_i$ and denote by Φ'_n the resulting CSP $\{\Phi_I \mid I \subseteq var(\Phi_n), |I| \leq k\}$. It is obvious that Φ_n and Φ'_n define the same function and the size of Φ'_n is bounded by $(p(n)+1)^k$. But k is a constant, so the size of (Φ'_n) grows polynomially. □

2.3 ... and their polynomials

To a CSP Φ we will assign two polynomials $P(\Phi)$ and $Q(\Phi)$. However, $P(\Phi)$ is only defined for boolean CSPs. So let Φ first be a boolean CSP with the set

of variables $X = \{x_1, \ldots, x_n\}$. We assign a polynomial $P(\Phi)$ in the (position) variables $Y_1, \ldots, Y_n$ to Φ in the following way:

$$P(\Phi) := \sum_{e:\{x_1,\ldots x_n\}\to\{0,1\}^n} \Phi(e) Y^e.$$

Here Y^e stands for $Y_1^{e(x_1)} Y_2^{e(x_2)} \ldots Y_n^{e(x_n)}$.

Example 1. Let the constraints in Φ be $\{x_1 \vee x_2, x_3 \neq x_2, \neg x_4 \vee x_2\}$. The satisfying assignments are then 0100, 0101, 1010, 1100 and 1101. This results in $P(\Phi) = X_2 + X_2X_4 + X_1X_3 + X_1X_2 + X_1X_2X_4$.

In contrast to $P(\Phi)$ the second polynomial $Q(\Phi)$ is also defined for non-boolean CSPs. So let Φ be a CSP with domain D. We assign to Φ the following polynomial $Q(\Phi)$ in the variables $\{X_d \mid d \in D\}$.

$$Q(\Phi) := \sum_{a:var(\Phi)\to D} \Phi(a) \prod_{x\in var(\Phi)} X_{a(x)} = \sum_{a:var(\Phi)\to D} \Phi(a) \prod_{d\in D} X_d^{\mu_d(a)},$$

where $\mu_d(a) = |\{x \in var(\Phi) \mid a(x) = d\}|$ computes number of variables mapped to d by a. Note that the number of variables in $Q(\Phi)$ is $|D|$, the size of the domain. and that $Q(\Phi)$ is homogeneous of degree $|var(\Phi)|$.

Example 2. Let $D = \{1, 2, 3, 4\}$ and let the constraints in Φ be $\{x_1 + x_2 \geq 4, x_3 = 5 - x_2, x_1 < x_2\}$. The satisfying assignments are then $(1, 3, 2)$, $(2, 3, 2)$, $(1, 4, 1)$, $(2, 4, 1)$ and $(3, 4, 1)$. This results in $Q(\Phi) = X_1X_2X_3 + X_2^2X_3 + X_1^2X_4 + X_1X_2X_4 + X_1X_3X_4$.

Remark 1. The polynomial Q has a very natural algebraic interpretation: Consider the free monoid D^* consisting of finite words of the symbols in D. Furthermore consider the free commutative monoid X_D^c on the symbols $X_D := \{X_d \mid d \in D\}$ which is essentially the set of monomials in the variables in X_D. There is a natural monoid morphism $q : D^* \to X_D^c$ with $q(a_1 \ldots a_s) = \prod_{i=1}^s X_{a_i}$. The morphism q drops the order of the symbols in a word and computes a commutative version of it.

Now we consider two rings: The first one is $\mathbb{Z}[D^*]$ consisting of formal integer linear combinations of words in D^*. Observe that we can think of any finite set $S \subseteq D^*$ as an element of $\mathbb{Z}[D^*]$ by encoding it as $\sum_{a\in S} a$. The second ring we consider is $\mathbb{Z}[X_D]$ which is simply the polynomial ring over $\mathbb{Z}$ in the variables X_D. The monoid morphism q induces the ring morphism $Q : \mathbb{Z}[D^*] \to \mathbb{Z}[X_D]$ by $Q(\sum_a c_a a) = \sum_a c_a q(a)$. Given the encoding $\sum_{a\in S} a$ of a set S, Q computes a commutative version of it. This is exactly what the polynomial $Q(\Phi)$ defined above does: To a CSP Φ it computes a commutative version of the set of satisfying assignments.

Remark 2. If Φ is a boolean CSP, i.e. $D = \{0, 1\}$, we can get $Q(\Phi)$ from $P(\Phi)$ easily. $Q(\Phi)$ has only two variables X_0 and X_1 and is homogeneous of degree $|var(\Phi)|$. Substituting Y_i of $P(\Phi)$ by $\frac{X_1}{X_0}$ we get

$$X_0^{|var(\Phi)|} P(\Phi)\left(\frac{X_1}{X_0}, \ldots, \frac{X_0}{X_1}\right) = Q(\Phi)(X_1, X_0).$$

In Section 2.4 we will see that in general $P(\Phi)$ can be expressed as $Q(\Phi)$.

Both $P(\Phi)$ and $Q(\Phi)$ are expressive enough to characterize VNP (see Section 4). Thus in order to characterize subclasses of VNP we introduce structural restrictions to CSPs in the next section.

2.4 Treewidth

An excellent introduction to treewidth, its properties and algorithmic consequences can be found in [FG06, Chapter 11]. For the convenience of the reader we recall the definitions and facts needed in the remainder of this paper.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, (B_t)_{t\in T})$, where $\mathcal{T} = (T, F)$ is a tree and $(B_t)_{t\in T}$ is a family of subsets of V such that:

- $\bigcup_{t\in T} B_t = V$.
- For every $v \in V$, the set $B^{-1}(v) := \{t \in T \mid v \in B_t\}$ is nonempty and connected in $\mathcal{T}$.
- For every edge $uv \in E$ there is a $t \in T$ such that $u, v \in B_t$.

The sets B_t are called the *bags* of the decomposition. The *width* of the decomposition is $\max\{|B_t| \mid t \in T\} - 1$. The *treewidth* $\mathrm{tw}(G)$ of a graph G is defined as the minimum of the widths of all tree-decompositions of G. With this definition trees have a treewidth of 1.

A *path decomposition* is a tree decomposition in which $\mathcal{T}$ is a path. The *pathwidth* $\mathrm{pw}(G)$ is defined in a completely analogous fashion to $\mathrm{tw}(G)$. Clearly for all graphs G we have $\mathrm{tw}(G) \leq \mathrm{pw}(G)$.

We state a well known property of tree-decompositions (see [FG06, Chapter 11]).

Proposition 1. *Let $G = (V, E)$ be a graph and $(\mathcal{T}, (B_t)_{t\in T})$ be a tree-decomposition of G. Then for each clique $C \subseteq V$ in G there is a bag B_t such that $C \subseteq B_t$.*

To a CSP Φ we assign two graphs: The *primal graph* G^P_Φ has the vertex set $var(\Phi)$ and there is an edge between two vertices x and y if and only if there is a constraint ϕ in Φ such that $\{x, y\} \subseteq var(\phi)$. Note that the constraints in Φ yield cliques in G^P_Φ. The *incidence graph* G^I_Φ has the vertex set $var(\Phi) \cup \{\phi \mid \phi \text{ constraint in } \Phi\}$. There is an edge between $x \in var(\Phi)$ and ϕ if and only if $x \in var(\phi)$. There are no other edges in G^I_Φ, thus G^I_Φ is bipartite.

We define the *treewidth* of a CSP G as $\mathrm{tw}(\Phi) := \mathrm{tw}(G^P_\Phi)$. We say that a family of CSPs (Φ_n) has bounded treewidth, if and only if $\mathrm{tw}(G_{\Phi_n}) \leq d$ for a constant d independent of n. We could also have defined the treewidth of Φ as the treewidth of the incidence graph G^I_Φ, but the following lemma tells us that there is not much difference if we consider CSPs with bounded arity.

Lemma 2. *For every CSP Φ we have:*

a) $tw(G^I_\Phi) \leq tw(G^P_\Phi) + 1$.
b) If all constraints in Φ have arity at most k, then $tw(G^P_\Phi) \leq k(tw(G^I_\Phi)+1)-1$.

Lemma 2 appears to be folklore, for example part a) can be found in [FMR08] without proof. For completeness we sketch a proof.

Proof. Let Φ be a CSP with the primal graph $G^P_\Phi = (var(\Phi), E_P)$ and the incidence graph $G^I_\Phi = (var(\Phi) \cup \{\phi \mid \phi \text{ constraint in } \Phi\}, E_I)$.

For a) consider a tree-decomposition $(\mathcal{T}, (B_t)_{t\in T})$ of G^P_Φ with $\mathcal{T} = (T, F)$. We construct a tree decomposition of G_I from it in the following way: For each constraint ϕ of Φ we pick a vertex $t_\phi \in T$ such that $var(\phi) \subseteq B_{t_\phi}$, i.e. B_{t_ϕ} contains all of ϕ's variables. Such a bag always exists, because constraints of Φ yield cliques in G^P_Φ and by Proposition 1 we know that in a tree decomposition there is for each clique a bag containing its vertices. We add a vertex t'_ϕ to T and connect it to t_ϕ with an edge $t_\phi t'_\phi$. Furthermore we set $B_{t'_\phi} := B_{t_\phi} \cup \{\phi\}$. It is easy to check that the result is indeed a tree decomposition of G^I_Φ and that the width increases only by 1. This proves a).

For b) we start from a tree-decomposition $(\mathcal{T}, (B_t)_{t\in T})$ of G^I_Φ with width $\mathrm{tw}(G^I_\Phi)$. In each bag B_t we substitute all vertices ϕ corresponding to constraints of Φ by $var(\phi)$ calling the resulting sets $\bar{B}_t$. It is easy to check that $(\mathcal{T}, (\bar{B}_t)_{t\in T})$ is a tree decomposition of G^P_Φ with at most $k(\mathrm{tw}(G^I_\Phi) + 1)$ vertices in each bag. In particular, $\bar{B}^{-1}(x)$ is connected for each $x \in var(\Phi)$, because $B^{-1}(x)$ was connected, for each constraint ϕ $B^{-1}(\phi)$ was connected and for each ϕ with $x \in var(\phi)$ there is a t such that $x, \phi \in B_t$. It follows that $\bar{B}^{-1}(x) = B^{-1}(x) \cup \bigcup_{\phi:\, x\in var(\phi)} B^{-1}(\phi)$ is connected. This yields b). □

The following lemma relates the expressivity of P and Q.

Lemma 3. *For every boolean CSP Φ in s variables there is a 2-assignment bounded CSP Ψ with domain size $|D| = 2s$ such that $P(\Phi) \leq Q(\Psi)$ and $G^P_\Phi \simeq G^P_\Psi$.*

Proof. Let $var(\Phi) = \{x_1 \ldots, x_s\}$. We construct Ψ with the variable set $var(\Psi) = \{\bar{x}_1, \ldots, \bar{x}_s\}$. For each boolean variable $x_i \in var(\Phi)$ the domain D contains two elements $(i, 0)$ and $(i, 1)$.

For each constraint ϕ_m in Φ with the variables $x_{j_1}, \ldots, x_{j_k}$ we construct a constraint ψ_m in Ψ with the variables $\bar{x}_{j_1}, \ldots, \bar{x}_{j_k}$. The new constraint ψ_m is satisfied by an assignment a with $\bar{x}_{j_t} \mapsto (i_t, b_t)$, if and only if $x_{j_t} \mapsto b_t$ is a satisfying assignment of ϕ_m and $i_t = j_t$ for $t = 1, \ldots, k$.

This way in each satisfying assignment of Ψ the only variable that can take the values $(i, 0)$ and $(i, 1)$ is the variable $\bar{x}_i$. Also satisfying assignments of Φ and Ψ directly correspond: If x_i has the value b, then $\bar{x}_i$ has the value (i, b). Identifying Y_i in $P(\Phi)$ with $X_{(i,1)}$ in $Q(\Psi)$ and substituting all variables $X_{(i,0)}$ in $Q(\Psi)$ by 1, we get $P(\Phi) = Q(\Psi)|_{(Y_{(i,0)}=1)}$. Thus $P(\Phi) \leq Q(\Psi)$. Observing that $G_\Phi \simeq G_\Psi$ via $x_i \mapsto \bar{x}_i$ completes the proof. □

3 Statement of the results

Having introduced all necessary definitions we will now formulate our results in this section. Our first theorem characterizes the expressive power of boolean and non-boolean c-assignment bounded CSPs of bounded path- and treewidth.

Theorem 1 (Characterization of VP_e).

a) Let (Φ_n) be a p-bounded family of boolean CSPs with bounded treewidth. Then $(P(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such a $(P(\Phi_n))$. The same statement also holds with pathwidth instead of treewidth.

b) Let (Φ_n) be a p-bounded family of c-assignment bounded CSPs with bounded treewidth. Then $(Q(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such $(Q(\Phi_n))$. The same statement also holds with pathwidth instead of treewidth.

Observe that Theorem 1 implies that bounded pathwidth and bounded treewidth have the same computational power in this setting, although pathwidth is a far more restrictive measure. This is a striking parallel to the findings of Flarup and Lyaudet in [FL08] where in a different context of arithmetic circuit complexity the expressive power of path-width and treewidth coincides, too.

Our next Theorem shows that general non-boolean CSPs with bounded *treewidth* characterize VP.

Theorem 2 (Characterization of VP). *Let (Φ_n) be a p-bounded family of CSPs with bounded treewidth. Then $(Q(\Phi_n)) \in \mathsf{VP}$. Moreover, any family in VP is a p-projection of such a $(Q(\Phi_n))$.*

Finally we show that for general non-boolean CSPs pathwidth and treewidth differ in expressivity. With bounded *pathwidth* we get a characterization of VP_{ws}.

Theorem 3 (Characterization of VP_{ws}). *Let (Φ_n) be a p-bounded family of CSPs with bounded pathwidth. Then $(Q(\Phi_n)) \in \mathsf{VP}_{ws}$. Moreover, any family in VP_{ws} is a p-projection of such a $(Q(\Phi_n))$.*

Observe that the only difference between Theorem 1 b) and Theorem 2/3 is the c-assigment boundedness. This means that the difference between VP_e and $\mathsf{VP}/\mathsf{VP}_{ws}$ in this setting is simply that for VP and VP_{ws} the variables in the constraints may take more different values in satisfying assignments.

We will prove Theorem 1, Theorem 2 and Theorem 3 in several individual lemmas.

We now relate our results to known results. Fischer, Makowsky and Ravve [FMR08] consider the problem of counting solutions to boolean CSPs and achieve the following results:

Theorem 4 ([FMR08]).

a) There is an algorithm that given a CNF-Formula Φ of size n and a tree decomposition of G^I_Φ of width k counts the number of satisfying assignments of Φ using at most $4^k n$ operations.

b) Given a boolean CSP Φ of size n and a tree decomposition of G^P_Φ of width k, the number of satisfying assignments of Φ can be computed with $4^k n^2$ arithmetic operations.

Observe that CNF formulas are special forms of CSPs in which the constraints are disjunctive clauses. For CNF-formulas the size of the clauses need not have bounded arity to guarantee feasibility in part a) of Theorem 4. In b) there is an implicit bound on the arity of the constraints, because the treewidth of the primal graph is bounded, so the setting is more comparable to ours. Thus Theorem 2 can be seen as an extension of b) to non-boolean CSPs also adding a matching lower bound.

Briquel, Koiran and Meer [KM08,BKM11] give the following result.

Theorem 5 ([KM08,BKM11]). *For every family (Φ_n) of p-bounded CNF-formulas with bounded treewidth of $G^I_{\Phi_n}$ we have $(P(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such a $(P(\Phi_n))$.*

Again the size of the CNF-clauses is not restricted and the treewidth of the incidence graph is considered. Theorem 5 can be interpreted as translation of Theorem 4 a) into the arithmetic circuit model with a matching hardness result. Theorem 1 is an extension of Theorem 5 to general CSPs instead of CNF-formulas. Moreover, the lower bound is shown to already hold for bounded pathwidth. But in contrast to Briquel, Koiran and Meer we require a bound on the arity of the constraints to show feasibility in our setting.

4 The power of unrestricted CSP

To start off our explorations of the expressivity of CSPs we briefly examine the power of CSPs when we do not restrict their structure. We will show that in this case we get a characterization of VNP in both the boolean and non-boolean case. The hardness follows directly from the results in [BK09]. The upper bound is a standard argument.

Lemma 4. *a) If (Φ_n) is a p-bounded family of boolean CSPs, then $(P(\Phi_n)) \in \mathsf{VNP}$. Moreover, any family in VNP is a p-projection of such $(P(\Phi_n))$.*

b) If (Φ_n) is a p-bounded family of CSPs, then $(Q(\Phi_n)) \in \mathsf{VNP}$. Moreover, any family in VNP is a p-projection of such $(Q(\Phi_n))$.

Proof. Note that with Lemma 3 we only need to show the lower bound for a) and the upper bound for b). The lower bound for a) is proved in [BK09, Section 3], so it only remains to prove that $Q(\Phi_n) \in \mathsf{VNP}$ for a p-bounded family (Φ_n) of CSPs.

So let (Φ_n) be a p-bounded family of CSPs with domains (D_n). Set $d_n = |D_n|$ and $a_n = |var(\Phi_n)|$. We encode assignments $a\colon var(\Phi_n) \to D$ by $d_n \times a_n$-matrices $M = (m_{dx})_{d \in D_n, x \in var(\Phi_n)}$ with entries 0 and 1. The entry m_{dx} is 1 if and only if $a(x) = d$. Note that a matrix $M \in \{0,1\}^{d_n \times a_n}$ is an encoding of an assignment $a\colon var(\Phi_n) \to D$ if and only if there is exactly one 1 in each column of M.

For Φ_n we will construct an arithmetic formula Ψ_n of size polynomial in d_n and $|var(\Phi)|$ such that $Q(\Phi)(X) = \sum_{M \in \{0,1\}^{d_n \times a_n}} \Psi(M, X)$. We sub-divide Ψ_n into three factors $\Psi_{n,1}$, $\Psi_{n,2}$ and $\Psi_{n,3}$.

We set

$$\Psi_{n,1} = \prod_{x \in var(\Phi_n)} \sum_{d \in D_n} m_{dx} \prod_{d' \in D_n, d' \neq d} (1 - m_{d'x}).$$

It is easy to see that for $M \in \{0,1\}^{d_n \times a_n}$ we have $\Psi_{n,1}(M) \in \{0,1\}$ and $\Psi_{n,1}(M) = 1$ if and only if M is an encoding of an assignment $a : var(\Phi_n) \to D$, i.e., in every column there is exacly one 1-entry.

We set

$$\Psi_{n,2} = \prod_{\phi \text{ constraint of } \Phi_n} \psi_\phi,$$

where ψ_ϕ is the following: Assuming that the matrix M encodes an assignment $a : var(\Phi_n) \to D$ we have $\psi_\phi(M) = 1$ if $a|_{var(\phi)}$ satisfies ϕ, otherwise $\psi_\phi = 0$. The constraints all have arity bounded by a constant k, so they have at most d_n^k satisfying assignments. Each of these can be checked individually by an arithmetic formula of size $O(k) = O(1)$, so ψ_ϕ can be realized as a formula of size $O(d_n^k)$. Observing that by Lemma 1, Φ_n has w.l.o.g. at most polynomially many constraints we get that $\Psi_{n,2}$ has a formula of polynomial size.

Finally,

$$\Psi_{n,3} = \prod_{x \in var(\Phi)} (\sum_{d \in D} m_{dx} X_d).$$

It is clear that indeed $\Psi_n = \Psi_{n,1}\Psi_{n,2}\Psi_{n,3}$ can be computed by a polynomial size formula. Also we have $Q(\Phi)(X) = \sum_{M \in \{0,1\}^{d_n \times a_n}} \Psi(M, X)$ and thus $(Q(\Phi_n)) \in \mathsf{VNP}$. □

5 Lower bounds

In this section we show the lower bounds on the expressivity of polynomials defined by CSPs of bounded pathwidth and treewidth.

Lemma 5. *There is a constant $c \leq 26$ such that the following holds: For every $(f_n) \in \mathsf{VP}_e$ there is a p-bounded family (Φ_n) of boolean CSPs with pathwidth at most c such that $(f_n) \leq_p (P(\Phi_n))$.*

Proof. Let $A_1, \ldots, A_n$ be 3×3-matrices. We denote the entries in matrix A_i by $(X_{jk}^i)_{j,k \in [3]}$. Let f_n be the $(1,1)$-entry of the product $A_1 A_2 \ldots A_n$. We show that there is a family Φ_n of boolean CSPs with pathwidth 26 such that $(f_n) \leq (P(\Phi_n))$. With the well-known VP_e-completeness of 3×3-matrix product (see [BOC92]) the claim of Lemma 5 follows.

We construct a CSP Φ_n iteratively. For each i we give a CSP Φ_n^i in the variables x_{jk}^l and y_{jk}^l with $l \leq i$ and $j, k \in [3]$. The resulting polynomial $P(\Phi_n^i)$ has the variables X_{jk}^l and Y_{jk}^l. The variables Y_{jk}^i do not appear in the iterated matrix product and we get rid of them by projecting them all to 1. In order

not to clutter the construction in our proof too much with these variables, we already substitute them by 1 in the polynomials $P(\Phi_n^i)$, so they never appear in our computations.

Let f_{jk}^i be the polynomial computed in the (j,k)-entry of $A_1A_2\dots A_i$. Furthermore let $a_y(i,j,k) := y_{jk}^i \wedge \bigwedge_{(j'k')\neq(j,k)} \neg y_{j'k'}^i$. Note that $a_y(i,j,k)$ is a constraint in the variables y_{jk}^i that is satisfied only by the assignment $y_{jk}^i \mapsto 1$ and $y_{j'k'}^i \mapsto 0$ for $(j,k) \neq (j',k')$. Let $a_x(i,j,k)$ be the same for the variables x_{jk}^i.

We now construct the Φ_n^i iteratively. In a slight abuse of notation we write the Φ_n^i as a conjunction of their constraints ϕ_l. During the construction we will make sure that the following holds:

$$P\left(\Phi_n^i \wedge a_y(i,j,k)\right) = f_{jk}^i.$$

Intuitively by fixing the y_{jk}^i we can compute individual entries of the product $A_1A_2\dots A_i$.

The CSP Φ_n^1 has the single constraint

$$\phi_1 = \bigvee_{j',k'} (y_{j'k'}^1 \wedge a_x(1,j',k')).$$

We have $P(\Phi_n^1 \wedge a_y(1,j,k)) = P(a_x(1,j,k)) = X_{jk}^1$ as desired.

For the construction of Φ_n^{i+1} assume that we have already constructed Φ_n^i with the desired properties. We construct Φ_n^{i+1} from Φ_n^i by adding one constraint ϕ_{i+1}. We set

$$\Phi_n^{i+1} = \Phi_n^i \wedge \underbrace{\left(\bigvee_{(j',k')} \left(y_{j'k'}^{i+1} \wedge \bigvee_l (a_x(i+1,l,k') \wedge a_y(i,j',l))\right)\right)}_{:=\phi_{i+1}}.$$

We get

$$\begin{aligned}
P(\Phi_n^{i+1} \wedge a_y(i+1,j,k)) &= P\left(\Phi_n^i \wedge \left(\bigvee_l (a_x(i+1,l,k) \wedge a_y(i,j,l))\right)\right) \\
&= \sum_{l=1}^{3} P(\Phi_n^i \wedge a_x(i+1,l,k) \wedge a_y(i,j,l)) \\
&= \sum_{l=1}^{3} X_{l,k}^{i+1} P(\Phi_n^i \wedge a_y(i,j,l)) \\
&= \sum_{l=1}^{3} X_{l,k}^{i+1} f_{j,l}^i \\
&= f_{jk}^{i+1}
\end{aligned}$$

Having constructed Φ_n^n we easily get f_n as

$$f_n = P(\underbrace{\Phi_n^n \wedge a(n,1,1)}_{:=\Phi_n}).$$

Certainly (Φ_n) is p-bounded, so only the bound on the pathwidth remains to be shown. The primal graph $G^P_{\Phi_n}$ has the vertices x^i_{jk} and y^i_{jk} for $i \in [n]$ and $j,k \in [3]$. Each constraint ϕ_i yields a clique in $G^P_{\Phi_n}$ and there are no other edges. We give a path decomposition by a path P of the vertices $t_1, \dots t_n$ and the bags $B_{t_i} = var(\phi_i)$. It is easy to check that this is indeed a path decomposition of $G^P_{\Phi_n}$ and it has width 26. □

One could show a version of Lemma 5 for bounded tree-width with a more standard parse tree argument. We chose to instead present this version, not only because it is stronger due to its path-width formulation but mainly because we deem the proof to be more interesting. We will see parse tree arguments in the proof of Lemma 6 and Lemma 8.

Combining Lemma 5 and Lemma 3 directly yields the following corollary.

Corollary 1. *There is a constant $c \leq 26$ such that the following holds: For every $(f_n) \in \mathsf{VP}_e$ there is a p-bounded family (Ψ_n) of 2-assignment bounded CSPs with pathwidth at most c such that $(f_n) \leq_p (Q(\Psi_n))$.*

Next we will show the lower bounds for the characterizations of VP and VP_{ws}. For the proofs we use so called parse tree arguments (see e.g. [MP08, Section 4]). A parse tree T of a multiplicatively disjoint circuit C is a subgraph of C that is constructed in the following way:

- Add the output gate of C to T.
- For every gate v added to T do the following;
  - If v is a $+$-gate, add exactly one of its children to T.
  - If v is a $\times$-gate, add both of its children to T.

Observe that parse trees are binary trees. The weight $w(T)$ of a parse tree T is the product of the labels of its leaves. The polynomial computed by C is the sum of the weights of all of C's parse trees.

Lemma 6. *Let $(f_n) \in \mathsf{VP}$, then there is a p-bounded family Φ_n of binary CSPs such that $(f_n) \leq_p (Q(\Phi_n))$ and $G^P_{\Phi_n}$ is a tree for every n.*

In the proof we will use the following result.

Lemma 7. *$(f_n) \in \mathsf{VP}$ if and only if (f_n) is computed by a family of multiplicatively disjoint semi-unbounded logarithmic depth circuits.*

The proof of Lemma 7 easily follows by applying the techniques of Malod and Portier [MP08, Lemma 2] on the classical characterization of VP by logarithmic depth semi-unbounded arithmetic circuits found by Valiant et al. [VSBR83].

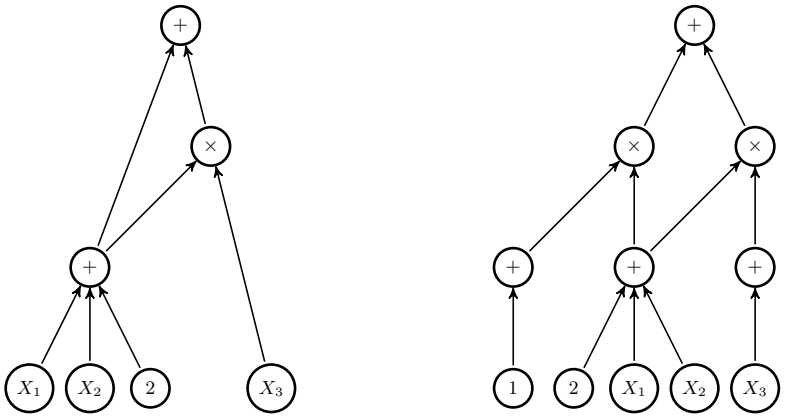

Fig. 1. The original circuit on the left is changed into one in which the gates on each level have the same operation label.

Proof (of Lemma 6). The idea of the proof is the following: We use the characterization in Lemma 7 which yields that the polynomials f_n have logarithmic depth parse trees. We encode these parse trees into polynomial size CSPs whose primal graphs are trees isomorphic to the parse trees of the f_n. Summing up over all possible encodings of parse trees we get polynomials whose projection are the f_n. We now describe the construction in more detail.

So consider a polynomial $f = f_n$ from our family. By Lemma 7 we know that f_n is computed by a logarithmic depth semi-unbounded circuit C of polynomial size. By adding dummies we can make sure that C has the following "layered" form (see Figure 1 for an illustration):

- All operation gates at the same depth have the same operation.
- All leaves are at the same depth level.

This implies that all parse trees of C are isomorphic binary trees. Let the children of the $\times$-gates in Φ be ordered, i.e., we call one of them the left child and the other one the right child. Let T be a binary tree isomorphic to the parse trees of C. The children of vertices in T that correspond to $\times$-gates in C are also ordered.

We now build a CSP Φ with $var(\Phi) = V(T)$ and $G^P_\Phi = T$. The domain is the vertex set $V(C)$ of C. To distinguish the vertices of T from the gates of C we write the vertices of T with a hat, e.g. $\hat{v} \in V(T)$. For each edge $\hat{u}\hat{v}$ in T we define a constraint $\phi_{\hat{u}\hat{v}}$ on the variables $\hat{u}$ and $\hat{v}$ in the following way: If $\hat{u}$ corresponds to a $+$-gate in C, then the satisfying assignments of $\phi_{\hat{u}\hat{v}}$ (where u and v denote the images of $\hat{u}$ and $\hat{v}$, respectively) are described by

$$\{(u, v) \mid u, v \in V(C), u \text{ is a } +\text{-gate, } v \text{ is a child of } u\}.$$

If $\hat{u}$ corresponds to a $\times$-gate and $\hat{v}$ is the left child of $\hat{u}$, then $\phi_{\hat{u}\hat{v}}$ is described by

$$\{(u, v) \mid u, v \in V(C), u \text{ is a } \times\text{-gate, } v \text{ is the left child of } u\}.$$

For right children we add constraints in an analog fashion.

It is easy to see, that Φ is satisfied by an assignment $a : V(T) \to V(C)$ if and only if a maps T onto a parse tree T_a of C. Also for each satisfying assignment a the resulting monomial $\prod_{\hat{u} \in V(T)} X_{a(\hat{u})}$ can be projected to $w(T_a)$ by doing the following: If v is an operation gate of C, then substitute X_v by 1. If v is an input gate of C with label l, then substitute X_v by l. Because each v is either an operation gate or an input gate but never both, these settings do not contradict for different satisfying assignments of Φ. It follows that $f \leq Q(\Phi)$. The primal graph G^P_Φ of Φ is by construction the tree T. The observation that the size of Φ and its domain $V(C)$ are polynomial completes the proof. □

We use a similar parse tree argument for VP_{ws}.

Lemma 8. *Let $(f_n) \in \mathsf{VP}_{ws}$, then there is a p-bounded family Φ_n of binary CSPs such that $(f_n) \leq_p (Q(\Phi_n))$. Furthermore $pw(\Phi_n) = 1$ for every n.*

Proof. The main difference to the proof of Lemma 6 is that for skew circuits not all parse trees are isomorphic and that we know no way to make them isomorphic without losing skewness. This problem is remedied by the insight that parse trees of skew circuits have a very restricted form that allows encoding them into CSPs of bounded pathwidth.

So let $(f_n) \in \mathsf{VP}_{ws}$. Then there is a family (C_n) of polynomial size skew circuits such that C_n computes f_n for every n. Let $f = f_n$ and $C = C_n$ and $s = |C|$. Each multiplication gate v of C has at least one child that is an input gate. We call this child the *leaf child* of v and the other child the *inner child* of v. If both children of v are input gates we arbitrarily choose one of them to be the leaf child and the other one the inner child. Each parse tree T of C has a very special form: T consists of a path P with some dangling leaf children. An illustration is shown in Figure 2. Note that in general not all parse trees have the same depth and that the order of the $+$- and $\times$-gates may differ in them.

We construct a CSP Φ that has the primal graph $G = G^P_\Phi = (V, E)$ with $V = \{\hat{v}_i \mid i \in \{0, \dots s\}\} \cup \{\hat{u}_i \mid i \in \{0, \dots s\}\}$ and $E = \{\hat{v}_i\hat{u}_i \mid i \in \{0, \dots s\}\} \cup \{\hat{v}_i\hat{v}_{i+1} \mid i \in \{0, \dots s-1\}\}$. We have $var(\Phi) = V$ and the domain of $D = V(C) \cup \{d\}$ for a dummy value d. Note that Φ has more variables than any parse tree of C, so that we cannot simply map $var(\Phi)$ onto the parse trees like in the proof of Lemma 6. Instead we map redundant vertices in $var(\Phi)$ onto the dummy d. This way we deal with the fact that the parse trees are not isomorphic.

For each edge e in G we have a constraint ϕ_e. For $e = \hat{v}_i\hat{u}_i$ the constraint $\phi_{\hat{v}_i\hat{u}_i}$ has the set of satisfying assignments

$$\{(v, u) \mid v \text{ }\times\text{-gate, } u \text{ its leaf child }\} \cup \{(v, d) \mid v \text{ }+\text{-gate }\} \cup \{(d, d)\}.$$

For each edge $e = \hat{v}_i\hat{v}_{i+1}$ the constraint $\phi_{\hat{v}_i\hat{v}_{i+1}}$ has the satisfying assignments

$$\begin{aligned} &\{(v, v') \mid v \text{ }\times\text{-gate, } v' \text{ its inner child }\} \\ \cup\ &\{(v, v') \mid v \text{ }+\text{-gate, } v' \text{ child of } v\ \} \\ \cup\ &\{(v, d) \mid v \text{ input gate }\} \cup \{(d, d)\}. \end{aligned}$$

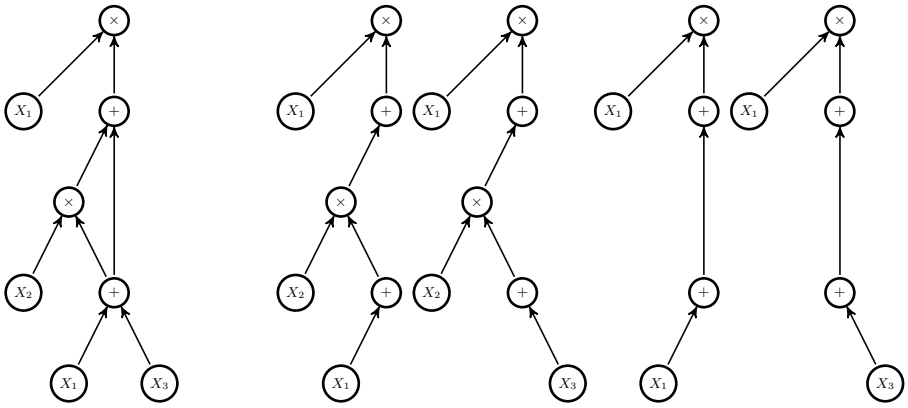

Fig. 2. A skew cicuit and all of its parse trees. The polynomial computed equals $X_1^2X_2 + X_1X_2X_3 + X_1^2 + X_1X_3$.

Furthermore Φ has one unary constraint $\phi_{\hat{v}_0}$ that is only satisfied by the assignment $a : \{\hat{v}_0\} \to V(C)$ with $a(\hat{v}_0) = v^*$ where v^* is the output gate of C.

It is easy to see that the only satisfying assignments $a \colon V \to V(C)$ of Φ are encodings of parse trees of C. The constraint $\phi_{\hat{v}_0}$ forces the satisfying assignments to map $\hat{v}_0$ to the output gate v^*. As discussed before the the satisfying assignments are "filled up" with dummies to deal with the different size and structure of the parse trees. With projections similar to those in the proof of Lemma 6 and substituting X_d by 1 we get $f \leq Q(\Phi)$. □

6 Upper bounds on the complexity

Having shown lower bounds on the expressivity of polynomials defined by CSPs, we will now prove the matching upper bounds. We will do this in detail for the case of non-boolean c-assignment bounded CSPs of bounded treewidth and then show how this proof can be adapted for the other two cases.

Lemma 9. *For every family (Φ_n) of p-bounded and c-assignment bounded CSPs of bounded treewidth we have $(Q(\Phi_n)) \in \mathsf{VP}_e$.*

The proof of Lemma 9 is conceptually similar to that in [KM08,BKM11] and [FMR08]: We make an inductive construction along the tree-decomposition. The difference here is that we do not consider CNF-formulas but arbitrary constraint satisfaction problems. In the proofs in [KM08,BKM11] and [FMR08] the authors use the fact that they can split the clauses of CNF formulas into different parts. This is not possible here, because we do not restrict the internal structure of the constraints. That is why we have to handle the individual constraints as a unit. For this reason we consider tree-decompositions of G_P and not of G_I. In the former it is guaranteed that the variables of the individual constraints are kept together by Proposition 1. Remember that for the case of bounded arity

constraints which we consider here both treewidth measures are equivalent by Lemma 2.

The authors of [FMR08] also consider a counting problem *#GENSAT* that has boolean constraints with arbitrary structure instead of CNF. They achieve results similar to ours in their setting, i.e. their results also hold for bounded treewidth of G^P_Φ or bounded arity of the constraints and bounded treewidth of G^I_Φ. Their technique for the proof is very different though: They reduce it to the CNF-case. It seems unclear how such an approach would work in our setting, because our CSPs are not boolean and thus we know of no standard form like CNF for them. Therefore we present a different approach that unfortunately is combinatorically more intricate.

Proof (of Lemma 9). Consider a family (Φ_n) of CSPs with the desired properties. To ease notation we fix n and set $\Phi = \Phi_n$ and $D = D_n$.

Fix a tree-decomposition $(\mathcal{T}, (B_t)_{t\in T})$ of the primal graph G^P_Φ with minimal width. W.l.o.g. the tree $\mathcal{T} = (T, F)$ is a rooted, binary tree and has depth $O(\log(n))$ (see [Bod89]). We give $\mathcal{T}$ a direction from the leaves to the root and will later make an induction along this direction.

Claim 1. *We may assume that there is a bijection from the vertices in T to the constraints in Φ such that $t \in T$ is mapped to a constraint ϕ_t with $var(\phi_t) = B_t$.*

Proof. In a first step we define a mapping λ_1 from the constraints to the vertices in T: For each ϕ in Φ we choose $t \in T$ such that $var(\phi) \subseteq B_t$ and set $\lambda_1(\phi) := t$. From Proposition 1 we know that this is always possible. For each $t \in T$ that is not an image of a constraint ϕ we add a constant constraint $\phi_t : U^\emptyset \to \{1\}$ and set $\lambda_1(\phi_t) = t$. It follows that λ_1 is a surjection from the constraints to the vertices in T such that $var(\phi) \subseteq B_{\lambda_1(\phi)}$ for all constraints ϕ.

From λ_1 we construct a bijection λ_2 by combining all constraints that are mapped onto t to one constraint ϕ_t by $\phi_t := \bigwedge_{\phi\in\lambda_1^{-1}(t)} \phi$. Note that $var(\phi_t) \subseteq B_t$.

In a last step we make sure that for all t we have $B_t \subseteq var(\phi_t)$. If there exists $x \in B_t \setminus var(\phi_t)$, we take an arbitrary constraint ϕ_x such that $x \in var(\phi_x)$. Because Φ is c-assignment bounded, the set $A_x := \{a(x) \mid a : var(\phi_x) \to D \text{ with } \phi_x(a) = 1\}$ has at most c elements. We set $\phi'_t := \phi_t \wedge (\bigvee_{u\in A_x} x = u)$. By doing this iteratively for all t and all $x \in B_t \setminus var(\phi_t)$, we get a new constraints that are still c-assignment bounded. For the resulting CSP Φ' with the constraint set $\{\phi'_t \mid t \in T\}$ we have that $t \mapsto \phi'_t$ is bijective and $var(\phi'_t) = B_t$.

Observe that Φ and Φ' compute the same function. Moreover, we did not change the B_t during the construction. Furthermore for each $t \in T$ the edges induced by ϕ'_t in the primal graph G^P_Φ have both their end vertices in B_t. Thus $(\mathcal{T}, (B_t)_{t\in T})$ is a tree decomposition of $G^P_{\Phi'}$, too. Also Φ' is still c-assignment bounded. So we may assume that Φ' was the CSP we started with. □

From Claim 1 it follows that we can assume $|B_t| \leq k$ for every $t \in T$, where k is the upper bound on arity of the constraints in Φ.

For each vertex t of $\mathcal{T}$ let $\mathcal{T}_t$ be the subtree of $\mathcal{T}$ that has t as its root. Let $T_t = V(\mathcal{T}_t)$ be the vertex set of $\mathcal{T}_t$. Furthermore let Φ_t be the CSP with the set of constraints $\{\phi_{t'} \mid t' \in T_t\}$ and the set of variables $var(\Phi_t) = \bigcup_{t'\in T_t} B_{t'}$.

We say that an assignment $a\colon B_t \to D$ is *good* for t or ϕ_t if it satisfies ϕ_t. Similarly we call a partial assignment to B_t good for t or ϕ_t if it can be extended to a good assignment. We are only interested in good assignments for individual constraints ϕ_t, because bad assignments do not contribute to $Q(\Phi)$ anyway.

Let $a\colon V \to D$ and $b\colon W \to D$ be assignments. We say that a and b are consistent (symbol: $a \sim b$), if $a|_{V\cap W} = b|_{V\cap W}$, i.e. they assign the same values to variables they share.

For each vertex $t \in T$ we will compute polynomials

$$f_{t,a,e} := \sum_{\substack{\alpha\colon var(\Phi_t)\to D,\\ a\sim\alpha}} \Phi_t(\alpha) \prod_{x\in var(\Phi_t)\setminus e} X_{\alpha(x)},$$

where a is a good assignment for ϕ_t and $e \subseteq B_t$. The sets $e \subseteq B_t$ will later in the construction prevent that variables $X_{a(x)}$ appear more than once in a monomial for $x \in var(\Phi)$.

Observe that for each t there are only $O(1)$ polynomials $f_{t,a,e}$: Because Φ is c-assignment bounded and its constraints ϕ have at most arity k, each ϕ has at most c^k satisfying assignments. Also there are at most $2^{|B_t|} \leq 2^k$ sets e. It follows that there are at most $c^k 2^k = O(1)$ polynomials $f_{t,a,e}$ for each t.

The *depth* of a vertex $t \in T$, denoted by $depth(t)$, is the length of the longest path from a leaf to t in $\mathcal{T}_t$.

Claim 2. *For each $t \in T$ we can compute all $f_{t,a,e}$ with a circuit of depth $O(depth(t))$.*

Lemma 9 follows easily with Claim 2: Let t^* be the root of $\mathcal{T}$. By definition

$$\begin{aligned} Q(\Phi) &= \sum_{\alpha\colon var(\Phi)\to D} \Phi(\alpha) \prod_{x\in var(\Phi)} X_{\alpha(x)} \\ &= \sum_{a\colon B_{t^*}\to D} \sum_{\alpha\colon var(\Phi)\to D, a\sim\alpha} \Phi(\alpha) \prod_{x\in var(\Phi)} X_{\alpha(x)} \\ &= \sum_{a\colon B_{t^*}\to D, \Phi_{t^*}(a)=1} f_{t^*,a,\emptyset} \end{aligned}$$

The tree $\mathcal{T}$ has depth $O(\log(n))$, so with Claim 2 we can compute $Q(\Phi)$ with a circuit of depth $O(\log(n))$. It follows that $(\Phi_n) \in \mathsf{VP}_e$. Thus all that is left to do is to prove Claim 2. □

Proof (of Claim 2). We make an induction along $\mathcal{T}_t$ from the leaves to t. So let t be a leaf. We have $f_{t,a,e} = \prod_{x\in B_t : x\notin e} X_{a(x)}$. Because of $|B_t| \leq k$ all the $f_{t,a,e}$ can be computed with constantly many arithmetic operations and thus in constant depth.

For the induction step we consider a vertex $t \in T$ with children t_1 and t_2. With the induction hypothesis it suffices to show how to compute the $f_{t,a,e}$ from the polynomials f_{t_j,a_j,e_j} in a constant number of steps. We will show this in two

substeps: First we compute reduced polynomials f'_{t_j,a'_j,e'_j} from the f_{t_j,a_j,e_j}. Then we compute $f_{t,a,e}$ from the f'_{t_j,a'_j,e'_j}.

Let $a'_j\colon B_{t_j} \cap B_t \to D$ be an assignment of $B_{t_j} \cap B_t$ that is good for t_j and let $e'_j \subseteq B_{t_j} \cap B_t$. We define

$$\begin{aligned} f'_{t_j,a'_j,e'_j} &:= \sum_{\alpha : var(\Phi_{t_j}) \to D, a'_j \sim \alpha} \Phi_{t_j}(\alpha) \prod_{x \in var(\Phi_{t_j}) \setminus e'_j} X_{\alpha(x)} \\ &= \sum_{a' : B_{t_j} \to D, a'_j \sim a'} \sum_{\alpha : var(\Phi_{t_j}) \to D, a' \sim \alpha} \Phi_{t_j}(\alpha) \prod_{x \in var(\Phi_{t_j}) \setminus e'_j} X_{\alpha(x)} \\ &= \sum_{a' : B_{t_j} \to D, a'_j \sim a'} f_{t_j,a',e'_j}. \end{aligned} \tag{1}$$

There are at most c^k good assignments for t_j. Thus the computation of the f'_{t_j,a'_j,e'_j} from the f_{t_j,a_j,e_j} can be done with $O(1)$ additions.

We now compute the $f_{t,a,e}$ from the f'_{t_j,a'_j,e'_j}. To do so we split B_t into disjoint sets $B_t = B_{(1)} \dot\cup B_{(2)} \dot\cup B_b \dot\cup B_*$ as follows: $B_* := B_t \cap B_{t_1} \cap B_{t_2}$, $B_{(1)} := (B_t \cap B_{t_1}) \setminus B_*$, $B_{(2)} := (B_t \cap B_{t_2}) \setminus B_*$, $B_b := B_t \setminus (B_{t_1} \cup B_{t_2})$.

Note that

$$var(\Phi_t) \setminus (var(\Phi_{t_1}) \cup var(\Phi_{t_2})) = B_b. \tag{2}$$

We claim that

$$var(\Phi_{t_1}) \cap var(\Phi_{t_2}) = B_*. \tag{3}$$

Indeed, $var(\Phi_{t_1}) \cap var(\Phi_{t_2}) \supseteq B_*$ is clear by the definition of B_*. For the other direction consider $x \in var(\Phi_{t_1}) \cap var(\Phi_{t_2})$. There must be $t_1 \in T_1$ and $t_2 \in T_2$ such that $x \in B_{t_1}$ and $x \in B_{t_2}$. This means that $t_1, t_2 \in B^{-1}(x)$. The vertex set $B^{-1}(x)$ is connected in $\mathcal{T}$, because $(\mathcal{T}, (B_t)_{t \in T})$ is a tree decomposition. This yields that $t, t_1, t_2 \in B^{-1}(x)$ which means that indeed $x \in B_t \cap B_{t_1} \cap B_{t_2}$. Thus $var(\Phi_{t_1}) \cap var(\Phi_{t_2}) \subseteq B_*$ and (3) follows.

For $e \subseteq B_t$ we define $e_* := e \cap B_*$ and $e_{(j)} := e \cap B_{(j)}$. Moreover, we chose a decomposition $B_* = e'_1 \cup e'_2$ with $e^* = e'_1 \cap e'_2$. This will allow us to prevent below that $X_{a(x)}$ is multiplied twice for $x \in B_*$. We claim that we have the following disjoint decomposition:

$$var(\Phi_t) \setminus e = \big(var(\Phi_{t_1}) \setminus (e'_1 \cup e_{(1)})\big) \dot\cup \big(var(\Phi_{t_2}) \setminus (e'_2 \cup e_{(2)})\big) \dot\cup (B_b \setminus e). \tag{4}$$

To see this note that $B_b \setminus e$ is disjoint from the other sets by (2). Moreover, from (3) and $B_* = e'_1 \cup e'_2$ we get $\big(var(\Phi_{t_1}) \setminus (e'_1 \cup e_{(1)})\big) \cap \big(var(\Phi_{t_2}) \setminus (e'_2 \cup e_{(2)})\big) = \emptyset$. Thus the union is indeed disjoint. Equality in (4) can be seen like this: From (2) we get $var(\Phi_t) = var(\Phi_{t_1}) \cup var(\Phi_{t_2}) \cup B_b$. Also, by the respective definitions, $e = (e \cap B_b) \dot\cup e_{(1)} \dot\cup e_{(2)} \dot\cup (e'_1 \cap e'_2)$. With these equalities, $e^* = e'_1 \cap e'_2$ and (2) one can show (4).

For $a\colon B_t \to D$ we set $a_{(j)} := a|_{B_{(j)}}$. With (4) we can compute $f_{t,a,e}$ in the following way:

$$\begin{aligned} f_{t,a,e} &= \sum_{a:var(\Phi_t)\to D} \Phi_t(a) \prod_{x\in var(\Phi_t)} X_{a(x)} \\ &= \sum_{\substack{\alpha\colon var(\Phi_{t_1})\to D,\\ \beta\colon var(\Phi_{t_2})\to D,\\ a\sim\alpha, a\sim\beta}} \Phi_{t_1}(\alpha)\Phi_{t_2}(\beta) \prod_{x\in var(\Phi_{t_1})\setminus(e'_1\cup e_{(1)})} X_{\alpha(x)} \\ &\qquad \prod_{x\in var(\Phi_{t_2})\setminus(e'_2\cup e_{(2)})} X_{\beta(x)} \prod_{x\in B_b\setminus e} X_{a(x)} \\ &= \left(\prod_{x\in B_b\setminus e} X_{a(x)}\right) f'_{t_1,a_{(1)},e'_1\cup e_{(1)}} f'_{t_2,a_{(2)},e'_2\cup e_{(2)}}. \end{aligned}$$

□

As a corollary of Lemma 9 we get with Lemma 3:

Corollary 2. *For every family (Φ_n) of boolean p-bounded CSPs of bounded treewidth we have $(P(\Phi_n)) \in \mathsf{VP}_e$.*

The proof of Lemma 9 can be adapted to show the following missing upper bounds for VP and VP_{ws}. We only sketch the proofs here, because they are very similar.

Lemma 10. *For every family (Φ_n) of p-bounded CSPs of bounded treewidth we have $(Q(\Phi_n)) \in \mathsf{VP}$.*

Proof (Sketch). If we do not require c-assignment boundedness, the construction of the proof of Lemma 9 still works. It is easy to check that Claim 1 still holds. The difference is that instead of Claim 2 we formulate another claim:

Claim 3. *For each $t \in T$ we can compute all $f_{t,a,e}$ with an arithmetic circuit of size $|T_t||D_n|^{O(1)}$.*

Claim 3 follows easily by the same techniques as in the proof of Claim 2. The difference is that we have (for fixed t) $|D_n|^{O(1)}$ polynomials $f_{t,a,e}$ and thus in the computation of the f'_{t_j,a'_j,e'_j} (see Equation (1) on page 19) there is a sum of $|D_n|^{O(1)}$ terms. Lemma 10 follows directly from Claim 3 . □

Lemma 11. *For every family (Φ_n) of p-bounded CSPs of bounded pathwidth we have $(Q(\Phi_n)) \in \mathsf{VP}_{ws}$.*

Proof (Sketch). The proof is very similar to that of Lemma 10. Instead of a tree decomposition we have a path decomposition $(\mathcal{P}, (B_t)_{t\in T})$ with $\mathcal{P} = (P, F)$ being a path. The key step is proving the following new claim.

Claim 4. *For each $t \in P$ we can compute all $f_{t,a,e}$ with a skew circuit of size $|P_t||D_n|^{O(1)}$.*

Claim 4 can easily be proved analogously to Claim 3. There are two new important insights;

1. Each vertex t has only one child t'. Thus the $f_{t,a,e}$ can be computed from $f'_{t',a',e'}$ defined analogously to the f'_{t_j,a'_j,e'_j}. In this computation every multiplication has a variable $X_{a(x)}$ as one factor. We never have to multiply any of the $f'_{t',a',e'}$.
2. The computation of the $f'_{t',a',e'}$ from the $f_{t',a,e}$ requires only additions, so it is uncritical concerning skewness of the resulting circuit.

Lemma 11 follows directly with Claim 4 □

References

[BFMY83] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.

[BK09] Irénée Briquel and Pascal Koiran. A dichotomy theorem for polynomial evaluation. In *MFCS09*, pages 187–198, 2009.

[BKM11] Irénée Briquel, Pascal Koiran, and Klaus Meer. On the expressive power of cnf formulas of bounded tree- and clique-width. *Discrete Applied Mathematics*, 159(1):1 – 14, 2011.

[BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.

[Bod89] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In *Graph-theoretic concepts in computer science (Amsterdam, 1988)*, volume 344 of *Lecture Notes in Comput. Sci.*, pages 1–10. Springer, Berlin, 1989.

[Bre76] R.P. Brent. The complexity of multiple-precision arithmetic. In R P Brent R S Andersson, editor, *The Complexity of Computational Problem Solving*, pages 126–165. Univ. of Queensland Press, 1976.

[Bür00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory.* Springer Verlag, 2000.

[CMR01] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[FG06] J. Flum and M. Grohe. *Parameterized complexity theory.* Springer-Verlag New York Inc, 2006.

[FL08] Uffe Flarup and Laurent Lyaudet. On the expressive power of permanents and perfect matchings of matrices of bounded pathwidth/cliquewidth. *CoRR*, abs/0801.3408:x, 2008.

[FMR08] Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008.

[GLS01] Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001.

[KM08] Pascal Koiran and Klaus Meer. On the expressive power of CNF formulas of bounded tree- and clique-width. In *WG08*, pages 252–263, 2008.

[LKF07] Laurent Lyaudet, Pascal Koiran, and Uffe Flarup. On the expressive power of planar perfect matching and permanents of bounded treewidth matrices. *CoRR*, abs/0705.3751:xx, 2007.

[Mar09] Dániel Marx. Tractable structures for constraint satisfaction with truth tables. In *STACS*, pages 649–660, 2009.

[MP08] Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *J. Complexity*, 24(1):16–38, 2008.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC78*, pages 216–226, 1978.

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.