Characterizing Arithmetic Circuit Classes by Constraint Satisfaction Problems (Extended Abstract)

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.

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 [14] 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 [15] 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 [5] gave a similar dichotomy result in the arithmetic circuit

* supported by the Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo)

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 [1] 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 [11] 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. [8] 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. [13] 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 [6, 12] – building on a paper by Fischer et al. [9] 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 [5] by showing feasible subclasses of polynomials assigned to CSPs. Also, our paper can be seen as an extension of the work of Briquel, Koiran and Meer [12, 6] 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.

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.

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 [7]. Newer insights into the nature of VP and especially VP_{ws} are presented in the excellent paper of Malod and Portier [14].

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 $\times$-gates still have fanin 2. A circuit is called *multiplicatively disjoint* if for each $\times$-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 $\times$-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 [4], 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 iff 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 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 can normalize CSPs with the following straightforward lemma:

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.*

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 $Q(\Phi)$ 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

$S \in D^*$ 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.

In general the $P(\Phi)$ and $Q(\Phi)$ are expressive enough to characterize VNP. 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 [10, 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 [10, 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 folklore 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 $tw(G^P_\Phi) \leq k(tw(G^I_\Phi)+1)-1$.

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 $G^P_\Phi \simeq G^P_\Psi$.*

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 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.

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. Because of space restrictions most of the proofs are omitted. They can be found in the upcoming full version of this paper.

We now relate our results to known results. Fischer, Makowsky and Ravve [9] consider the problem of counting solutions to boolean CSPs and achieve the following results:

Theorem 4 ([9]).

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 [12, 6] give the following result.

Theorem 5 ([12, 6]). *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 Lower bounds

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

Lemma 4. *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 $(f_n) \leq_p (P(\Phi_n))$.*

A proof of Lemma follows by using an encoding of iterated 3×3-matrix multiplication (see [2]) into CSPs. Combining Lemma 4 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. [14, 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 5. *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 6. *$(f_n) \in \mathsf{VP}$ if and only if (f_n) is computed by a family of multiplicatively disjoint semi-unbounded logarithmic depth circuits.*

The proof of Lemma 6 easily follows by applying the techniques of Malod and Portier [14, Lemma 2] on the classical characterization of VP by logarithmic depth semi-unbounded arithmetic circuits found by Valiant et al. [16].

Proof (of Lemma 5). The idea of the proof is the following: We use the characterization in Lemma 6 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 6 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:

- 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 $G_\Phi^P = T$. 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

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

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_Φ^P 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 7. *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.*

The key insight for the proof for Lemma 7 is that parse trees of skew circuits have a very restricted form that allows encoding them into CSPs of bounded pathwidth.

5 Upper bounds on the complexity

Lemma 8. *For every family (Φ_n) of p-bounded and c-assignment bounded CSPs of bounded treewidth we have $(Q(\Phi_n)) \in \mathsf{VP}_e$.*

Proof. 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_Φ^P 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 [3]). We give $\mathcal{T}$ a direction from the leaves to the root and will later make an induction along this direction. We use teh following helpful claim:

Claim 6. *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 $var(\phi_t) = B_t$.*

It follows that $|B_t| \leq k$ 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 7. *For each $t \in T$ we can compute all $f_{t,a,e}$ with a circuit of depth $O(depth(t))$.*

Lemma 8 follows easily with Claim 7: 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 7 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 7. $\square$

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 8 can be adapted to show the following lemmas:

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

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

References

1. Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.
2. Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.
3. 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.
4. 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.
5. Irénée Briquel and Pascal Koiran. A dichotomy theorem for polynomial evaluation. In *MFCS09*, pages 187–198, 2009.
6. 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.
7. Peter Bürgisser. *Completeness and reduction in algebraic complexity theory.* Springer Verlag, 2000.
8. 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.
9. 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.
10. J. Flum and M. Grohe. *Parameterized complexity theory.* Springer-Verlag New York Inc, 2006.
11. Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48(3):431–498, 2001.
12. Pascal Koiran and Klaus Meer. On the expressive power of CNF formulas of bounded tree- and clique-width. In *WG08*, pages 252–263, 2008.
13. 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.
14. Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *J. Complexity*, 24(1):16–38, 2008.
15. Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC78*, pages 216–226, 1978.
16. 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.