Discrete Applied Mathematics 159 (2011) 1–14

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

On the expressive power of CNF formulas of bounded tree- and clique-width

Irenée Briquel [a], Pascal Koiran [a], Klaus Meer [b,*]

[a] Laboratoire de l'Informatique du Parallélisme, ENS Lyon, France

[b] Lehrstuhl Theoretische Informatik, BTU Cottbus, Germany

ARTICLE INFO

Article history:
Received 2 November 2009
Received in revised form 27 August 2010
Accepted 6 September 2010
Available online 16 October 2010

Keywords:
Expressive power of polynomials
Permanent function
Conjunctive normal form formulas
Tree- and clique-width
Valiant's complexity theory for polynomial families

ABSTRACT

We study representations of polynomials over a field $\mathbb{K}$ from the point of view of their expressive power. Three important examples for the paper are polynomials arising as permanents of bounded tree-width matrices, polynomials given via arithmetic formulas, and families of so called CNF polynomials. The latter arise in a canonical way from families of Boolean formulas in conjunctive normal form. To each such CNF formula there is a canonically attached incidence graph. Of particular interest to us are CNF polynomials arising from formulas with an incidence graph of bounded tree- or clique-width.

We show that the class of polynomials arising from families of polynomial size CNF formulas of bounded tree-width is the same as those represented by polynomial size arithmetic formulas, or permanents of bounded tree-width matrices of polynomial size. Then, applying arguments from communication complexity we show that general permanent polynomials cannot be expressed by CNF polynomials of bounded tree-width. We give a similar result in the case where the clique-width of the incidence graph is bounded, but for this we need to rely on the widely believed complexity theoretic assumption $\#P \not\subseteq FP/poly$.

1. Introduction

An active field of research in complexity is devoted to the design of efficient algorithms for subclasses of problems which in full generality are likely hard to solve. It is common in this area to define such subclasses via bounding some significant problem parameters. Typical such parameters are the tree- and clique-width if a graph structure is involved in the problem's description.

At the center of the present paper stand problems related to families of polynomials. These families are given in a particular manner through certain Boolean formulas in conjunctive normal form, CNF formulas for short.

More precisely, we consider a Boolean CNF formula φ representing a function from $\{0,1\}^n \to \{0,1\}$. If no confusion can arise we denote this function again by φ. For n variables $x_1, \ldots, x_n$ ranging over a field $\mathbb{K}$ and an $e \in \{0,1\}^n$ define the monomial $x^e := x_1^{e_1} \cdot \cdots \cdot x_n^{e_n}$, where $x_i^0 := 1$ and $x_i^1 := x_i$. Now define a function $f : \mathbb{K}^n \to \mathbb{K}$ by

$$f(x) = \sum_{e \in \{0,1\}^n} \varphi(e) \cdot x^e \quad \text{for } x \in \mathbb{K}^n. \tag{$*$}$$

The function f is a kind of enumerating polynomial for φ. We are interested in the question of how expressive such a representation of polynomials by CNF formulas is, and under which additional conditions the polynomial $f(x)$ in $(*)$ can

* Corresponding author. Tel.: +49 355 693883; fax: +49 355 693810.
E-mail addresses: irenee.briquel@ens-lyon.fr (I. Briquel), pascal.koiran@ens-lyon.fr (P. Koiran), meer@informatik.tu-cottbus.de (K. Meer).

0166-218X/$ – see front matter
doi:10.1016/j.dam.2010.09.007

be evaluated efficiently. Fischer et al. [3], extending earlier results from [2], have shown that the counting SAT problem, i.e. computing $\sum_{e\in\{0,1\}^n}\varphi(e)$ for a CNF formula φ can be solved in time $O(n\cdot 4^k)$ if a certain bipartite graph G_φ canonically attached to φ is of bounded tree-width k. Similar results concerning clique-width as well are given in [3].

Our first main result (Theorem 5) precisely characterizes the expressive power of polynomials of form $(*)$ when G_φ is of polynomial size and bounded tree-width. It is shown that the class of these polynomials describes the same functions representable by arithmetic formulas of polynomial size and the functions represented as permanents of matrices of bounded tree-width and polynomially bounded dimension. Here, equality of the latter two concepts was known before due to a result of Flarup et al. [4].

Recall that in Valiant's algebraic model of computation for families of polynomials the permanent is VNP complete and thus unlikely to be efficiently computable. Though an unconditional proof of this conjecture seems extremely difficult, we can at least show that trying to obtain an efficient algorithm for computing permanents through formulas of the type $(*)$ with G_φ of bounded tree-width must fail. Such an algorithm would exist if the boolean function PERMUT_n recognizing $n\times n$ permutation matrices could be written as a (polynomial size) CNF formula of bounded tree-width. We show that such a CNF formula does not exist. This result is unconditional in that it does not rely on any open conjecture in complexity theory. In an earlier version of this paper [8] this impossibility result was obtained by reduction to an OBDD lower bound. Here we appeal instead to arguments from communication complexity (incidentally, this seems to be the first time that the PERMUT_n function is studied from the point of view of communication complexity). The present approach provides a new point of view on this problem; it also has the advantage of providing at little additional cost some new lower bounds for other functions, derived from communication complexity lower bounds in the so-called "best case" model.

Finally, we pose the corresponding question for CNF formulas of bounded clique-width. Using another result from [3] we show that expressing the permanent of an arbitrary matrix by formulas of type $(*)$, this time with G_φ of bounded clique-width would imply $\#P\subseteq FP/poly$ and thus is unlikely.

The paper is organized as follows. In Section 2 we recall basic definitions as well as the needed results from [3] and [4]. Section 3 first shows how functions represented via permanents of matrices of bounded tree-width can be expressed via polynomials of form $\sum_{e\in\{0,1\}^n}\varphi(e)x^e$ with G_φ of bounded tree-width. Then, we extend a result from [3] to link such polynomials to arithmetic formulas. The results in [4] now imply equivalence of all three notions. In Section 4 the above mentioned negative results concerning expressiveness of (general) permanents by CNF formulas of bounded tree- or clique-width are proven.

Our results contribute to the comparison of Boolean and algebraic complexity. In particular, we consider it to be interesting to find more results like Theorem 8 which states that certain properties *cannot* be expressed via (certain) graphs of bounded tree-width.

2. Basic definitions

In this section we collect the basic definitions and results that are needed below. We try to keep the section as short as possible since most of the notions are well known. Nevertheless, for the readers' convenience we collect all notions needed at one place.

2.1. Arithmetic circuits

Definition 1. (a) An arithmetic circuit is a finite, acyclic, directed graph. Vertices have indegree 0 or 2, where those with indegree 0 are referred to as *inputs*. A single vertex must have outdegree 0, and is referred to as *output*. Each vertex of indegree 2 must be labeled by either $+$ or $\times$, thus representing computation. Vertices are commonly referred to as *gates*. By choosing as input nodes either some variables x or constants from a field $\mathbb{K}$ a circuit in a natural way represents a multivariate polynomial over $\mathbb{K}$.

(b) An *arithmetic formula* is a circuit for which all gates except the output have outdegree 1.

(c) The *size* of a circuit is the total number of *gates* in the circuit.

(d) A family $\{\phi_n\}_{n\in\mathbb{N}}$ of arithmetic formulas is said to be of polynomial size if there is a polynomial function p such that for all $n\in\mathbb{N}$ the size of ϕ_n is at most $p(n)$.

Note that for formulas the reuse of partial results is not allowed. For more on different subclasses of arithmetic circuits see [11].

2.2. Tree- and clique-width

Tree-width for undirected graphs is defined as follows:

Definition 2. Let $G=\langle V,E\rangle$ be a graph. A k-tree-decomposition of G is a tree $T=\langle V_T,E_T\rangle$ such that:

(i) For each $t\in V_T$ there is an associated subset $X_t\subseteq V$ of size at most $k+1$.
(ii) For each edge $(u,v)\in E$ there is a $t\in V_T$ such that $\{u,v\}\subseteq X_t$.
(iii) For each vertex $v\in V$ the set $\{t\in V_T\mid v\in X_t\}$ forms a (connected) subtree of T.

The tree-width $twd(G)$ of G is then the smallest k such that there exists a k-tree-decomposition for G.
If we require the decomposition trees to be paths, then we obtain the *path-width* of the given graph.

The path-width of a graph G with n nodes can be bounded from above by $O(twd(G) \cdot \log n)$, see [9].
In this paper we frequently deal with the tree-width of matrices with entries from a field $\mathbb{K}$. This is defined as follows:

Definition 3. (a) The tree-width of an $(n \times n)$ matrix $M = (m_{i,j})$ is defined as the tree-width of the edge-weighted graph $G_M = (V_m, E_M, w)$, where $V_M = \{1, \ldots, n\}$, $(i, j) \in G_M$ iff $m_{i,j} \neq 0$ and $w(i, j) := m_{i,j}$ denote the edge weights.
(b) A family $(M_n)_n$ of matrices is said to be of polynomial size if for all $n \in \mathbb{N}$ the dimension of M_n is at most $p(n)$ for some fixed polynomial p.

For the algorithmic treatment of CNF formulas below we recall the definition of H-sums of graphs, see [3].

Definition 4. Let H, G_1, G_2 be graphs such that G_1 and G_2 have induced subgraphs H_1, H_2 which are isomorphic to H by isomorphisms h_1, h_2, respectively. Let G' be the disjoint union of G_1 and G_2. The H-sum $G := G_1 \oplus_{H,h_1,h_2} G_2$ of G_1 and G_2 is the graph obtained from G' by identifying the two copies (via h_1, h_2) of H in the disjoint union.

Given a k-tree decomposition of a graph G with sets of vertices X_t, we can consider the subgraph H_t of G induced by X_t and reconstruct G using a sequence of H_t-sums.
Next we recall the clique-width notion.

Definition 5. A graph G has *clique-width* at most k iff there exists a set of k labels $\mathcal{S}$ such that G can be constructed using a finite number of the following operations:

(i) $vert_a, a \in \mathcal{S}$ (create a single vertex with label a);
(ii) $\phi_{a\to b}(H), a, b \in \mathcal{S}$ (relabel all vertices having label a by label b);
(iii) $\eta_{a,b}(H), a, b \in \mathcal{S}, a \neq b$ (add edges between all vertices having label a and all vertices having label b);
(iv) $H_1 \oplus H_2$ (disjoint union of graphs).

To each graph of clique-width k we can attach a (rooted) parse-tree whose leaves correspond to singleton graphs and whose vertices represent one of the operations above. The graph G then is represented at the root.

2.3. Permanent polynomials

Definition 6. The permanent of an (n, n)-matrix $M = (m_{i,j})$ is defined as

$$perm(M) := \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)},$$

where S_n denotes the set of all permutations of $\{1, \ldots, n\}$.

We are interested in representing functions via particular polynomials, the permanents. If M above has as entries either variables or constants from some field $\mathbb{K}$, then $f = perm(M)$ is a polynomial with coefficients in $\mathbb{K}$ (in Valiant's terms f is a projection of the permanent polynomial). One main result in [4] characterizes arithmetic formulas of polynomial size by certain such polynomials. The tree-width of a matrix $M = [m_{ij}]$ is defined to be the tree-width of the graph including an edge (i, j) iff $m_{ij} \neq 0$.

Theorem 1 ([4]). *Let $(f_n)_{n\in\mathbb{N}}$ be a family of polynomials with coefficients in a field $\mathbb{K}$. The following properties are equivalent:*

(i) *$(f_n)_{n\in\mathbb{N}}$ can be represented by a family of polynomial size arithmetic formulas.*
(ii) *There exists a family $(M_n)_{n\in\mathbb{N}}$ of polynomial size, bounded tree-width matrices such that the entries of M_n are constants from $\mathbb{K}$ or variables of f_n, and $f_n = perm(M_n)$.*

Note that the complexity of the permanent of matrices of bounded tree-width was discussed already in [2].

2.4. Clause graphs

One of our goals is to relate Theorem 1 to yet another concept, namely CNF formulas of bounded tree-width. The latter will be defined in this subsection. Our presentation follows closely [3].

Definition 7. Let φ be a Boolean formula in conjunctive normal form with clauses $C_1, \ldots, C_m$ and Boolean variables $x_1, \ldots, x_n$.

(a) The *signed clause graph* $SI(\varphi)$ is a bipartite graph with the x_i and the C_j as nodes. We call the former v-vertices and the latter c-vertices. Edges connect a variable x_i and a clause C_j iff x_i occurs in C_j. An edge is signed $+$ or $-$ if x_i occurs positively or negated in C_j.
(b) The *incidence graph* $I(\varphi)$ of φ is the same as $SI(\varphi)$ except that we omit the signs $+, -$.

(c) The *primal graph* $P(\varphi)$ of φ has only the x_i's as its nodes. An edge connects x_i and x_j iff both occur in one of the clauses.
(d) The tree- or clique-width of a CNF formula φ is defined to be the tree- or clique-width of $I(\varphi)$, respectively.
If below we want to speak about the tree-width of $P(\varphi)$ we mention this explicitly.

There is no difference in defining the tree-width for the signed graph. Concerning the tree-width of the primal and the incidence graph it is remarked in [5] that $twd(I(\varphi)) \le twd(P(\varphi))$ if the former is at least 2 (and $\le twd(P(\varphi)) + 1$ if the former equals 1). With respect to clique-width things are a bit more complicated. There is an own notion of clique-width for signed graphs which we do not employ here. As to the results we are looking for note that the counting problem for CNF formulas remains hard even if the clique-width of the primal graph is bounded. Thus here only the incidence graph is considered. For a more detailed discussion on this and a proof of the above statements see [3].

In [3] the authors prove as well:

Theorem 2. (a) *Given φ and a tree-decomposition of $I(\varphi)$ of width k one can compute the number of satisfying assignments $\sum_{x\in\{0,1\}^n} \varphi(x)$ of φ in $4^k n$ arithmetic operations.*
(b) *Given a CNF formula φ and a parse-tree for the signed clause graph $SI(\varphi)$ of clique-width $\le k$ the number $\sum_{x\in\{0,1\}^n} \varphi(x)$ of satisfying assignments of φ can be computed in $O(2^{ck} n)$ many arithmetic operations.*

Below, we extend the algorithm proving Theorem 2(a) in order to relate CNF formulas to arithmetic formulas and to Theorem 1. Note that similar results to those of part (a) of Theorem 2 are given in [12].

3. Expressiveness of CNF polynomials of bounded tree-width

In this section we prove our first main result. We study how expressive polynomials p_n are which are given via CNF formulas φ_n of bounded tree-width. It turns out that functions represented by permanents of bounded tree-width matrices can as well be represented by such CNF polynomials, whereas functions represented by the latter in turn are representable by short arithmetic formulas. Given the equivalence stated in Theorem 1 all three concepts have the same expressive power.

3.1. From permanents to clause graphs

For the definition of the edge-weighted graph G_M related to a matrix M recall Definition 3.

Theorem 3. *Let $M = [m_{ij}]$ be an $n \times n$ matrix over a field $\mathbb{K}$ such that the corresponding directed weighted graph $G_M = (V_M, E_M)$ is of tree-width k.*

There is a CNF formula φ of tree-width $O(k^2)$, of size polynomially bounded in n and depending on n^2 variables $e_{i,j}$ and on $t(n)$ variables θ where $t \in O(n)$ such that

(a)
$$perm(M) = \sum_{\substack{e\in\{0,1\}^{n^2},\\ \theta\in\{0,1\}^{t(n)}}} \varphi(e,\theta)\cdot m^e.$$

Here, $e = \{e_{i,j}\}$ denotes variables representing the edges of G_M, $m = \{m_{i,j}\}$ denotes the entries of M and $m^e := \prod_{i,j} m_{i,j}^{e_{i,j}}$, where $m_{i,j}^{e_{i,j}} = \begin{cases} m_{i,j} & \text{if } e_{i,j} = 1 \\ 1 & \text{if } e_{i,j} = 0 \end{cases}$.
(b) *For every e there exists θ such that $\varphi(e,\theta) = 1$ if and only if e is a cycle cover of G_M; in this case, the corresponding θ is unique.*
(c) *A tree decomposition of $I(\varphi)$ of width $O(k^2)$ can be obtained from a decomposition of G_M in time $O(n)$.*

Remark 1. 1. The use of the auxiliary variables θ will become clearer in the proof. Basically, they are needed to keep the tree-width of formula φ small. Towards this aim they store certain truth values when a tree-decomposition of G_M is processed.
2. In the above CNF polynomial $\sum_{e,\theta} \varphi(e,\theta)\cdot m^e$ there are no monomials corresponding to θ. Formally one could introduce another block y of variables and add to each monomial m^e another factor y^θ. Then $perm(M)$ is obtained as a projection (in Valiant's sense) of a CNF-polynomial $\sum_{e,\theta} \varphi(e,\theta)\cdot m^e \cdot y^\theta$ by plugging in for each y-variable the value 1.

Proof. Let $(T, \{X_t\}_t)$ be a tree decomposition of width k for G_M. Without loss of generality T is a binary tree. In order to define φ, we construct the graph $I(\varphi)$ and precise how clauses of φ correspond to c-vertices of $I(\varphi)$. The graph $I(\varphi)$ to be constructed contains two blocks of v-vertices, one being the edge-variables $e_{i,j}$ of G_M and another block θ of auxiliary variables to be explained below. The tree decomposition $(T, \{X'_t\}_t)$ that we shall construct for $I(\varphi)$ uses the same underlying tree T as the tree decomposition of G_M, but the boxes X'_t will be different from the boxes X_t in the initial decomposition.

A straightforward set of clauses to describe cycle covers in G_M is the following collection:

(i) for each vertex $i \in V_M$ clauses Out_i and In_i containing as its literals all outgoing edges from and all incoming edges into i, respectively;
(ii) for each $i \in V_M$ and each pair of outgoing edges $e_{i,j}, e_{i,l}$ a clause $\neg e_{i,j} \vee \neg e_{i,l}$; similarly for incoming edges to i.

A tree decomposition of the resulting graph $I(\varphi)$ is then obtained from T by taking the same tree and joining in a box X'_t for every $i \in X_t$ all vertices resulting from (i) and (ii). However, due to the conditions under (ii) this might not result in a decomposition of bounded width.

To resolve this problem for each box $t \in T$ and each $i \in V_M$ we add additional v-vertices $check^t_{i+}$, $check^t_{i-}$. Fix t and the subtree T_t of T that has t as its root. For any assignment of the $e_{i,j}$ indicating which edges in G_M have been chosen for a potential cycle cover a condition $check^t_{i+} = 1$ indicates that an edge starting in i has already been chosen with respect to those vertices of G_M occurring in the subtree T_t.

Further clauses are introduced to guarantee that each i finally is covered exactly once for a satisfying assignment of $\varphi(e, \theta)$, where θ is the collection of all check variables. More precisely, we proceed bottom up. Let t be a leaf of T. For every $i \in X_t$ in addition to the v-vertices $check^t_{i+}$, $check^t_{i-}$ in $I(\phi)$, introduce c-vertices representing the following clauses:

(1) $\bigvee_{j \in X_t} e_{i,j} \vee \neg check^t_{i+}$;
   Interpretation: if none of the $e_{i,j}$'s were chosen yet, then $check^t_{i+} = 0$.
(2) $\neg e_{i,j} \vee \neg e_{i,l}$ for all $j, l \in X_t$;
   Interpretation: at most one outgoing edge covers i.
(3) $\neg e_{i,j} \vee \neg check^t_{i+}$ for all $j \in X_t$;
   Interpretation: if an $e_{i,j}$ was chosen (i.e. $e_{i,j} = 1$), then $check^t_{i+} = 1$.

Analogue c-vertices are added for $check^t_{i-}$.

For the box X'_t in the decomposition of $I(\varphi)$ that corresponds to box X_t of T all v-vertices $e_{i,j}$, $check^t_{i+}$, $check^t_{i-}$, $i, j, \in X_t$ as well as the c-vertices resulting from (1)–(3) above are included. These are $O(k^2)$ many elements in X'_t. Now T' is constructed bottom up. The check variables propagate bottom up the information whether a partial assignment for those $e_{i,j}$ that already occurred in a subtree can still be extended to a cycle cover of G_M. At the same time, the width of the new boxes of T' constructed will not increase too much. Suppose in T there are boxes t, t_1, t_2 such that t_1 is the left and t_2 the right child of t. Let $i \in X_t \cap X_{t_1} \cap X_{t_2}$. The case where i only occurs in two or one of the boxes is treated similarly. Assuming X'_{t_1}, X'_{t_2} has already been constructed c-vertices corresponding to the following clauses are included in X_t:

(1′) $\bigvee_{j \in X_t \setminus \{X_{t_1} \cup X_{t_2}\}} e_{i,j} \vee check^{t_1}_{i+} \vee check^{t_2}_{i+} \vee \neg check^t_{i+}$;
   Interpretation: if all new $e_{i,j}$'s and the previous check variables are 0, then the new check variable $check^t_{i+}$ is 0 as well;
(2′) $\neg x \vee \neg y$ for all $x, y \in \{e_{i,j} : j \in X_t \setminus \{X_{t_1} \cup X_{t_2}\}\} \cup \{check^{t_1}_{i+}, check^{t_2}_{i+}\}$; $x \neq y$
   Interpretation: at most one among the old check variables and the new edge variables gets the value 1;
(3′) $\neg x \vee \neg check^t_{i+}$ for all $x \in X_t \setminus \{X_{t_1} \cup X_{t_2}\} \cup \{check^{t_1}_{i+}, check^{t_2}_{i+}\}$;
   Interpretation: if one among the values $e_{i,j}$ or $check^{t_1}_{i+}$, $check^{t_2}_{i+}$ is 1, then $check^t_{i+} = 1$.

Again, analogue c-vertices are added for the ingoing edges to i. Box X'_t contains all related edge vertices $e_{i,j}$ for the new $j \in X_t \setminus \{X_{t_1} \cup X_{t_2}\}$, the six check vertices and the $O(k^2)$ many c-vertices resulting from (1′)–(3′).

This way $(T, \{X'_t\}_t)$ is obtained. Finally, for each $i \in T$ two new c-vertices corresponding to clauses containing the single literals $check^r_{i+}$ and $check^r_{i-}$, respectively, are included in that box X_r which represents the root r of the subtree of T generated by all boxes that contain i. This is to guarantee that i is covered in both directions.

Clearly, $(T, \{X'_t\}_t)$ is a binary tree with each X'_t containing at most $O(k^2)$ many vertices. Let θ denote the vector of all check variables and $t(n)$ their number. Clearly, $t \in O(n)$. It is also obvious from the construction that

$$\exists \theta \ \varphi(e, \theta) \Leftrightarrow e \text{ represents a cycle cover}$$

(via those $e_{i,j}$ that have value 1). Moreover, for each assignment of e^* giving a cycle cover there is precisely one assignment θ^* such that $\varphi(e^*, \theta^*) = 1$ because e^* uniquely determines which check variables have to be assigned the value 1. Therefore

$$perm(M) = \sum_{\substack{e \in \{0,1\}^{n^2}, \\ \theta \in \{0,1\}^{t(n)}}} \varphi(e, \theta) \cdot m^e.$$

Finally, it remains to show that $(T', \{X'_t\}_t)$ actually is a tree decomposition of the graph $I(\varphi)$. Vertices resulting from check variables at most occur in two consecutive boxes of T' and thus trivially satisfy the connectivity condition. C-vertices related to one of the construction rules (1), (3), (1′)–(3′) for a fixed $t \in T$ only occur in the single box X'_t. Finally, an edge variable $e_{i,j}$ occurs in a box X'_t iff both i and j occur in X_t. Thus, the fact that $(T, \{X_t\}_t)$ is a tree decomposition implies that the connectivity condition also holds for these vertices and $(T', \{X'_t\}_t)$. □

3.2. From clause graphs to arithmetic formulas

In the next step we link CNF polynomials to arithmetic formulas. More precisely, the next theorem shows the latter concept to be strong enough to capture the former.

Theorem 4. *Let $\mathbb{K}$ be a field and $k \in \mathbb{N}$ be fixed. Let $\{\varphi_n\}_n$ be a family of CNF formulas of tree-width at most k and with n variables. Let $\{f_n\}_n$ denote the family of functions $f_n : \mathbb{K}^n \to \mathbb{K}$ represented by these formulas, i.e.,*

$$f_n(x) = \sum_{z \in \{0,1\}^n} \varphi_n(z) \cdot x^z$$

for all $x \in \mathbb{K}^n$.

Then there is a family of arithmetic formulas over $\mathbb{K}$ and of polynomial size which represents $\{f_n\}_n$.

The proof is based on an extension of results in [3], namely Theorem 1.3. In the latter it is shown how to count efficiently satisfying assignments for a CNF formula φ, i.e. how to compute $\sum_{z\in\{0,1\}^n} \varphi_n(z)$, where $I(\varphi_n)$ is of bounded tree-width and a tree decomposition is given. Our extension is dealing with finding short arithmetic formulas for polynomials of the form $\sum_{z\in\{0,1\}^n} \varphi(z) \cdot x^z$. The proof of Theorem 1.3. in [3] proceeds along a tree-decomposition of $I(\varphi_n)$ analyzing how the evaluation can be done for a clause graph G obtained as an H-sum of two other clause graphs G_1 and G_2, see Definition 4.

Our proof of Theorem 4 works as follows. We extend the ideas of [3] in order to show how one can efficiently evaluate $f_n(x)$ for all $x \in \mathbb{K}^n$ when f_n is defined as in the statement. Note that counting satisfying assignments corresponds to evaluating $f_n(1, \ldots, 1)$. Taking the x_i's as variables the efficient algorithm obtained can then be converted into an arithmetic formula that has polynomial size.

Let us first adapt some notation from [3]. Let Σ be a set of clauses over a set V of variable, let $W \subseteq V$ and $z : W \to \{0, 1\}$ an assignment for the variables in W. Denote by $\varphi(\Sigma)$ the CNF formula $\bigwedge_{C\in\Sigma} C$ and by $\Sigma^{(z)}$ the set of clauses obtained from Σ when replacing each $v \in W$ by the value $z(v)$. The main part of the proof of Theorem 4 is to analyze how the decomposition along H-sums can be used to obtain short arithmetic formulas for $\sum_z \varphi(z)$.

We need an additional definition:

Definition 8. Let $\mathbb{K}$ be a field, G a clause graph with v-vertices V, $|V| = n$, φ_G the corresponding CNF formula, $W \subseteq V$ and $z : W \to \{0, 1\}$ a (partial) assignment for the variables in W.

Then the polynomial $f_{(G,W,z)}$ is defined as

$$\forall x \in \mathbb{K}^n \quad f_{(G,W,z)}(x) := \sum_{z' : V\setminus W \to \{0,1\}} \left(\varphi_G(z', z) \cdot \prod_{i \in V\setminus W} x_i^{z'_i} \right).$$

Above, the partial assignments z', z are plugged into φ_G in the obvious way.

In particular, if $W = \emptyset$ (and thus $z = \emptyset$) we define

$$f_{(G,\emptyset,\emptyset)}(x) := \sum_{z' : V \to \{0,1\}} \varphi_G(z') \cdot x^{z'}.$$

The following technical proposition shows how $f_{(G,\emptyset,\emptyset)}(x)$ can be computed along an H-sum decomposition of G. Theorem 4 follows as a consequence. For the definition of an H-sum recall Definition 4.

Proposition 1. *Let G, G_1, G_2 be clause graphs with v-vertices V, V_1, and V_2, and c-vertices C, C_1, and C_2, respectively. Suppose that $G = G_1 \oplus_{H,h_1,h_2} G_2$, where H is isomorphic via h_1, h_2 to two induced subgraphs H_1, H_2 of G_1, G_2, respectively. Denote the v-vertices of H by W and the c-vertices by D.*

(a) *Suppose $D = \emptyset$; then for all $x \in \mathbb{K}^n$ we have*

$$f_{(G,\emptyset,\emptyset)}(x) = \sum_{z : W \to \{0,1\}} \left(f_{(G_1,W,z)}(x) \cdot f_{(G_2,W,z)}(x) \right) \cdot \prod_{i\in W} x_i^{z_i}.$$

(b) *Suppose $W = \emptyset$ and $D = \{D_1, \ldots, D_m\}$, $m \in \mathbb{N}$. For an $X \subseteq [m] := \{1, \ldots, m\}$ and $i = 1, 2$ let $S_i(X)$ denote the set of clauses obtained from all $D_j, j \in X$ when only maintaining literals related to V_i. Let $G_i(X)$ be the clause graph with v-vertices V_i and c-vertices $C_i \setminus S_i(X)$. Finally, let $\mathcal{A}$ be the set of all pairs (X_1, X_2) with $X_1, X_2 \subseteq [m]$, $X_1 \cap X_2 = \emptyset$.*

Then for each $(X_1, X_2) \in \mathcal{A}$ there is an integer $s(X_1, X_2)$ such that

$$f_{(G,\emptyset,\emptyset)}(x) = \sum_{(X_1,X_2)\in\mathcal{A}} s(X_1, X_2) \cdot f_{(G_1(X_1),\emptyset,\emptyset)}(x) \cdot f_{(G_2(X_2),\emptyset,\emptyset)}(x).$$

(c) *Let both $W \neq \emptyset$, $D = \{D_1, \ldots, D_m\} \neq \emptyset$, then using the same notation as in (b) one has for all $x \in \mathbb{K}^n$:*

$$f_{(G,\emptyset,\emptyset)}(x) = \sum_{z : W \to \{0,1\}} \left(\sum_{(X_1,X_2)\in\mathcal{A}} s(X_1, X_2) \cdot f_{(G_1(X_1),W,z)}(x) \cdot f_{(G_2(X_2),W,z)}(x) \right) \cdot \prod_{i\in W} x_i^{z_i}.$$

Proof. (a) In [3] it is shown that

$$\sum_{z':V\to\{0,1\}} \varphi_G(z') = \sum_{z:W\to\{0,1\}} \left(\sum_{z^{(1)}:V_1\setminus W\to\{0,1\}} \varphi_1(z^{(1)}, z) \right) \cdot \left(\sum_{z^{(2)}:V_2\setminus W\to\{0,1\}} \varphi_2(z^{(2)}, z) \right).$$

In order to extend this to the evaluation of polynomials one has to take care not to include the factor $\prod_{i\in W} x_i^{z_i}$ twice. This is the reason for defining $f_{(G,W,z)}$ as above. One gets

$$\begin{aligned} f_{(G,\emptyset,\emptyset)}(x) &= \sum_{z':V\to\{0,1\}} \left(\varphi_G(z') \cdot \prod_{i\in V} x_i^{z'_i} \right) \\ &= \sum_{z:W\to\{0,1\}} \left(\sum_{z^{(1)}:V_1\setminus W\to\{0,1\}} \varphi_1(z^{(1)}, z) \cdot \prod_{i\in V_1\setminus W} x_i^{z_i^{(1)}} \right) \cdot \left(\sum_{z^{(2)}:V_2\setminus W\to\{0,1\}} \varphi_2(z^{(2)}, z) \cdot \prod_{i\in V_2\setminus W} x_i^{z_i^{(2)}} \right) \cdot \prod_{i\in W} x_i^{z_i} \\ &= \sum_{z:W\to\{0,1\}} \left(f_{(G_1,W,z)}(x) \cdot f_{(G_2,W,z)}(x) \cdot \prod_{i\in W} x_i^{z_i} \right) \end{aligned}$$

as was claimed. Note that once a $z : W \to \{0, 1\}$ has been fixed the expressions $f_{(G_i,W,z)}$, $i = 1, 2$ have again the form $f_{(G'_i,\emptyset,\emptyset)}$, where G'_i results from G_i by plugging the values for z into the clauses and then formally removing W. We thus can continue similarly for further decompositions of the subgraphs obtained.

(b) For each pair $(X_1, X_2) \in \mathcal{A}$ let $A(X_1, X_2)$ denote those assignments to all variables in V which satisfy $C_1 \setminus S(X_1)$ and $C_2 \setminus S(X_2)$. Note that for clauses D_j in D for which $j \notin X_1 \cup X_2$ an assignment $z \in A(X_1, X_2)$ satisfies both the part of D_j related to V_1 and the one related to V_2. The requirement $X_1 \cap X_2 = \emptyset$ in the definition of $\mathcal{A}$ is necessary to guarantee that $A(X_1, X_2)$ only contains satisfying assignments of φ.

Before we obtain the claimed formula a few more facts about the $G_i(X_i)$, $i = 1, 2$ have to be derived. Let $z : V \to \{0, 1\}$ be an assignment and denote by $z^{(1)}, z^{(2)}$ the restrictions of z to V_1, V_2, respectively. Recall that $W = V_1 \cap V_2 = \emptyset$. If $\phi_G(z) = 1$ then there exists $(X_1, X_2) \in \mathcal{A}$ such that

$$\varphi_G(z) \cdot \prod_{i\in V} x_i^{z_i} = \left(\varphi_{G_1(X_1)}(z^{(1)}) \cdot \prod_{i\in V_1} x_i^{z_i^{(1)}} \right) \cdot \left(\varphi_{G_2(X_2)}(z^{(2)}) \cdot \prod_{i\in V_2} x_i^{z_i^{(2)}} \right). \tag{1}$$

Moreover, if $\phi_G(z) = 0$ then the above equation holds for all $(X_1, X_2) \in \mathcal{A}$. Note that the monomial factors in (1) are independent of X_1 and X_2. Therefore one gets a decomposition of the expression for $f_{(G,\emptyset,\emptyset)}$ basically by computing correctly the number of satisfying assignments for the entire formula φ_G via counting it for subformulas consisting of clauses of type $C_i \setminus S(X_i)$.[1]

For doing so, first apply the principle of inclusion and exclusion for determining the number $\#\varphi_G$ of satisfying assignments of φ_G. It is given as

$$\begin{aligned} \#\varphi_G &= \left| \bigcup_{(X_1,X_2)\in\mathcal{A}} A(X_1, X_2) \right| \\ &= \sum_{(X_1,X_2)\in\mathcal{A}} |A(X_1, X_2)| - \sum_{\substack{(X_1,X_2)\in\mathcal{A} \\ (X_3,X_4)\in\mathcal{A}}} |A(X_1, X_2) \cap A(X_3, X_4)| \pm \cdots (-1)^{|\mathcal{A}|+1} \cdot \left| \bigcap_{(X_1,X_2)\in\mathcal{A}} A(X_1, X_2) \right| \end{aligned}$$

The following property of the sets $A(X_1, X_2)$ simplifies a bit the above formula: If two (or more) such $A(X_1, X_2), A(X_3, X_4)$ have a non-empty intersection one gets as result the set $A(X_1 \cap X_3, X_2 \cap X_4) =: A(Y_1, Y_2)$ with $(Y_1, Y_2) \in \mathcal{A}$. Applying this remark we can replace the cardinalities of all intersections by suitable integer multiples of $|A(Y_1, Y_2)|$. If $s(Y_1, Y_2)$ denotes the (possibly negative) integral factor with which $|A(Y_1, Y_2)|$ occurs in the inclusion/exclusion formula it follows

$$\left| \bigcup_{(Y_1,Y_2)\in\mathcal{A}} A(Y_1, Y_2) \right| = \sum_{(Y_1,Y_2)\in\mathcal{A}} s(Y_1, Y_2) \cdot |A(Y_1, Y_2)|.$$

For our purposes it is not necessary to know the precise values of $s(Y_1, Y_2)$. It is only important that they exist and that the number of elements in $\mathcal{A}$ is bounded by a function in m, and thus in the tree-width only. More precisely, $|\mathcal{A}| \le 3^m \le 3^{k+1}$.

[1] The latter in principle is done in Lemma 4.6 of [3]. However, it seems that the formula given there is not quite correct, f.e., it gives a negative counting result in case $m = 1$. Therefore, we give a more elaborated proof here.

Finally, the arguments following (1) now imply that

$$f_{(G,\emptyset,\emptyset)}(x) = \sum_{(Y_1,Y_2)\in\mathcal{A}} s(Y_1, Y_2) \cdot f_{(G_1(Y_1),\emptyset,\emptyset)}(x) \cdot f_{(G_2(Y_2),\emptyset,\emptyset)}(x).$$

(c) The mixed case now follows from (a) and (b) above in the same manner. □

Proof of Theorem 4. Without loss of generality we suppose the tree decomposition $(T, \{X_t\}_t)$ of a given G to be of depth $O(\log n)$, see [1]. This will increase the tree-width from the original k to at most $3k + 2$. [2] In order to find an arithmetic formula for $\sum_{z:V\to\{0,1\}} \varphi_n(z) \cdot x^z = f_{(G,\emptyset,\emptyset)}(x)$ we perform the dynamic programming algorithm provided by Proposition 1 bottom up along T. For each subgraph represented by a leaf node the evaluation easily results in $2^{O(k)}$ arithmetic formulas of length $2^{O(k)}$. When climbing up the tree at each node representing an H-sum operation the formulas resulting from the three cases of Proposition 1 contribute to the formula size by a factor of $2^{O(k)}$. Thus, since T has logarithmic depth the total formula size is of order at most $n^{O(k)}$. □

Theorems 1, 3 and 4 imply

Theorem 5. *Let $(f_n)_{n\in\mathbb{N}}$ be a family of polynomials with coefficients in a field $\mathbb{K}$. The following properties are equivalent:*

(i) *$(f_n)_{n\in\mathbb{N}}$ can be represented by a family of polynomial size arithmetic formulas.*
(ii) *There exists a family $(M_n)_{n\in\mathbb{N}}$ of polynomial size, bounded tree-width matrices such that the entries of M_n are constants from $\mathbb{K}$ or variables of f_n, and $f_n = perm(M_n)$.*
(iii) *There exists a family $(\varphi_n)_{n\in\mathbb{N}}$ of CNF formulas of size polynomial in n and of bounded tree-width such that $f_n(x)$ can be expressed as a projection: $f_n(x) = \sum_d \varphi_n(d) \cdot z^d$. Here, projection means that the z_i's can be taken either as constants from $\mathbb{K}$ or as variables among the x_j's. The vector d is Boolean and has polynomial length in n.*

As one of the referees kindly pointed out to us the following result by Gurski and Wanke gives a possibility for characterizing when a family of bounded clique-width graphs is of bounded tree-width.

Theorem 6 (*Gurski and Wanke [6]*)**.** *Let $k, s \in \mathbb{N}$. Every graph of clique-width k which does not contain the complete bipartite graph $K_{s,s}$ as a subgraph has tree-width at most $3k \cdot (s-1) - 1$.*

A direct consequence of this theorem is the following characterization for families of graphs of bounded clique-width to be of bounded tree-width:

Corollary 1. *Let $k \in \mathbb{N}$ and $(G_n)_{n\in\mathbb{N}}$ be a family of graphs each of clique-width at most k. Then the family is of bounded tree-width if and only if there exists an $s \in \mathbb{N}$ such that no G_n contains $K_{s,s}$ as subgraph.*

Proof. The proof follows from the above theorem by noting that the tree-width of $K_{s,s}$ is s and thus grows to infinity with increasing s. □

The corollary gives a possibility to rephrase Theorem 5 in terms of clique-width.

Theorem 5 (*Clique-width version*)**.** *Let $(f_n)_{n\in\mathbb{N}}$ be a family of polynomials with coefficients in a field $\mathbb{K}$. The following properties are equivalent:*

(i) *$(f_n)_{n\in\mathbb{N}}$ can be represented by a family of polynomial size arithmetic formulas.*
(ii) *There exist an $s \in \mathbb{N}$ and a family $(M_n)_{n\in\mathbb{N}}$ of polynomial size, bounded clique-width matrices such that the entries of M_n are constants from $\mathbb{K}$ or variables of f_n, none of the graphs G_{M_n} has the complete bipartite graph $K_{s,s}$ as a subgraph, and $f_n = perm(M_n)$.*
(iii) *There exists an $s' \in \mathbb{N}$ and a family $(\varphi_n)_{n\in\mathbb{N}}$ of CNF formulas of size polynomial in n and of bounded clique-width such that none of the graphs $I(\varphi_n)$ has the complete bipartite graph $K_{s',s'}$ as subgraph, and $f_n(x)$ can be expressed as a projection: $f_n(x) = \sum_d \varphi_n(d) \cdot z^d$. Here, projection means that the z_i's can be taken either as constants from $\mathbb{K}$ or as variables among the x_j's.*

Note however that a 'pure' clique-width version of the theorem seems unlikely since for a matrix M with all entries different from 0 the clique-width of G_M is bounded by 2 whereas the computation of the permanent is hard.

4. Lower bounds

Given Theorem 3 together with the efficient algorithm resulting from Theorem 4 the following question arises: How far does the approach of reducing permanent computations to computations of the form $\sum_{e,\theta} \varphi(e, \theta) \cdot m^e$ lead, when φ comes from a clause graph of bounded tree- or bounded clique-width?

[2] Though it is de facto unnecessary to first balance T it makes the complexity arguments a bit easier.

Define the boolean function $\mathrm{PERMUT}_n : \{0,1\}^{n\times n} \to \{0,1\}$ as the characteristic function for $n \times n$ permutation matrices, i.e., PERMUT_n takes the value 1 on boolean matrices that have exactly one 1 in each row and one 1 in each column, and 0 otherwise. Note that the permanent of an $(n \times n)$-matrix $M = (m_{i,j})$ is given by $\sum_{e\in\{0,1\}^{n^2}} \mathrm{PERMUT}_n(e) \cdot m^e$. Formulated a bit differently we ask the following: Is there a CNF formula $\varphi(e, \theta)$ of bounded tree- or clique-width, respectively, such that $\varphi(e, \theta) = 1$ iff $e \in \{0,1\}^{n\times n}$ is a permutation matrix; in addition, we would like for each permutation matrix e to have exactly one θ s.t. $\varphi(e, \theta) = 1$.

In this section we prove that such a formula does not exist in the bounded tree-width case. A second result shows that when replacing tree- by clique-width a formula with the above properties does not exist unless $\#P \subseteq FP/poly$.

4.1. Lower bound for tree-width

Towards our goal we employ results from communication complexity. We will relate it to the path-width of the primal graphs of formulas. Recall that the path-width of a graph with n nodes is bounded from above by $O(t \cdot \log n)$, where t denotes its tree-width [9].

In order to be able to argue on primal graphs we need the following result that justifies the replacement of a formula's incidence graph by its primal graph.

Proposition 2. *Let $\varphi = C_1 \wedge \cdots \wedge C_m$ be a CNF formula with n variables $x_1, \ldots, x_n$ such that its incidence clause graph $I(\varphi)$ has tree-width k. Then there is a CNF formula $\tilde{\varphi}(x, y)$ such that the following conditions are satisfied:*

- *each clause of $\tilde{\varphi}$ has at most $k + 3$ literals;*
- *the primal graph $P(\tilde{\varphi})$ has tree-width at most $4(k + 1)$. A tree-decomposition can be constructed in linear time from one of $I(\varphi)$;*
- *the number of variables and clauses in $\tilde{\varphi}$ is linear in n;*
- *for all $x^* \in \{0,1\}^n$ we have $\varphi(x^*) = 1$ iff there exists a y^* such that $\tilde{\varphi}(x^*, y^*) = 1$. Such a y^* moreover is unique.*

Proof. Let $(T, \{X_t\}_t)$ be a (binary) tree-decomposition of $I(\varphi)$. The construction below combines the use of check variables in the proof of Theorem 3 with the usual way of reducing a general CNF formula instance to one with a bounded number of literals in each clause. Let C be a clause of φ and T_C the subtree of T induced by C. We replace C bottom up in T_C by introducing $O(n)$ new variables and clauses. More precisely, start with a leaf box X_t of T_C. Suppose it contains k variables that occur in literals of C, without loss of generality say $x_1 \vee \cdots \vee x_k$. Note that since C itself is contained in X_t there are at most k variables included. Introduce a new variable y_t together with $k + 1$ clauses expressing the equivalence $y_t \Leftrightarrow x_1 \vee \cdots \vee x_k$. Each of the new clauses has at most $k + 1$ literals. Next, consider an inner node t of T_C having two sons t_1, t_2. Suppose $x'_1, \ldots, x'_k$ to be those variables in X_t that occur as literals in C, again without loss of generality in the form $x'_1 \vee \cdots \vee x'_k$. If y_{t_1}, y_{t_2} denote the new variables related to C that have been introduced for X_{t_1}, X_{t_2}, for X_t define a new variable y_t together with clauses expressing $y_t \Leftrightarrow y_{t_1} \vee y_{t_2} \vee x'_1 \vee \cdots \vee x'_k$. Again, there are at most $k + 3$ new clauses containing at most $k + 3$ literals each. Finally, if t is the root of T_C we define y_t as before and add a clause saying $y_t = 1$. Thus, we add for each node X_t at most $k + 4$ new clauses as well as one new variable.

Do the same for all clauses of φ. This results in a CNF formula $\tilde{\varphi}$ which depends on $O(m \cdot n)$ additional variables y and contains $O(m \cdot n \cdot k)$ clauses. The construction guarantees that $\varphi(x)$ iff there exists a y such that $\tilde{\varphi}(x, y)$, and in that case y is unique.

A tree-decomposition of the primal graph $P(\tilde{\varphi})$ is obtained as follows. For each occurrence of a clause C in X_t of T replace the c-vertex by the newly introduced variables of the tuple y related to the clause and the box X_t. In addition, for boxes X_t, X_{t_1}, X_{t_2} such that t_1, t_2 are sons of t include the variables y_{t_1}, y_{t_2} also in the upper box X_t. The x_i variables that previously occurred are maintained. Since for a single box X_t at most three y_j are included for each clause, and since there are at most $k + 1$ c-vertices in an original box, the tree-width of $P(\tilde{\varphi})$ is $\leq 4(k + 1)$. The decomposition satisfies the requirements of a tree-decomposition since we did not change occurrences of the x_i's and the only y_t-variables that occur in several boxes occur in two consecutive ones. □

Our proofs below rely on the notion of communication complexity. The model generally considered in communication complexity was introduced by Yao [13]. In this model, an input is divided between two parties, that we call processors. Those processors must compute a given function of this input. To do so, since each processor has only a partial input, they need to share information: they will send bits to each other until one processor, say the second one, returns the value of the function on the given input. We then say, that the processors have computed the function in common. We briefly recall some definitions. For more on this see [10].

Definition 9. Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a Boolean function.

(a) Consider a partition of the n variables of f into two disjoint sets $x = \{x_1, \ldots, x_{n_1}\}$, $y = \{y_1, \ldots, y_{n_2}\}$, $n_1 + n_2 = n$. The communication complexity of f with respect to (x, y) is the lowest amount of bits that two processors, the first working on the variables x and the second on the variables y, need to exchange in order to compute f in common.

(b) The one-way communication complexity of f with respect to (x, y) is the lowest amount of exchanged bits needed to compute f if only one processor is allowed to send bits to the other.

(c) If above we only allow partitions of the variables of same cardinality, i.e., n is even and $|x| = |y|$, and minimize over all of them we obtain the best-case and best-case one-way communication complexity, respectively.
(d) The non-deterministic communication complexity of f with respect to (x, y) is the lowest amount of bits that two processors, the first working on the variables x, the second on the variables y, and each having access to a source of non deterministic bits, need to exchange in order to compute in common the function f in the following sense:
- If $f(x) = 1$, at least one of the possible non-deterministic computations must be accepting
- If $f(x) = 0$, all the non-deterministic computations must be non-accepting.

A useful approach in communication complexity consists in considering for a given function $f(u, v)$ the matrix associated with it:

Definition 10. Let $f : U \times V \to \{0, 1\}$ be a boolean function.

(a) We call the matrix of f the matrix $(f(u, v))$, where the different assignments of u denote the rows and those to v denote the columns. Note that the matrix is a $|U| \times |V|$ matrix.
(b) A rectangle of the matrix $(f(u, v))$ is a set of entries composed of the intersection of a certain set of rows and a certain set of columns. That is, a set of entries R is a rectangle if and only if the following is true: $\exists \tilde{U} \subseteq U, \tilde{V} \subseteq V$ such that $R = \tilde{U} \times \tilde{V}$. Equivalently, a set of entries R is a rectangle if and only if the following is true:

$$\forall (u_1, u_2, v_1, v_2) \in U^2 \times V^2, (u_1, v_1) \in R \wedge (u_2, v_2) \in R \Rightarrow (u_1, v_2) \in R.$$

(c) A rectangle of the matrix $(f(u, v))$ is called monochromatic if f has the same value on each entry of the rectangle.

The following two results are classical in communication complexity [10,13]:

Theorem 7. *Let $f(x, y)$ be a function over two boolean vectors x and y.*

(i) *The one-way communication complexity of f equals the logarithm of the number of different rows in the matrix $(f(u, v))$.*
(ii) *The non-deterministic communication complexity of f equals the logarithm of the minimal number of monochromatic rectangles of the matrix $(f(u, v))$ needed to cover all values 1 in the matrix.*

For the lower bound proof, the non-deterministic communication complexity with respect to certain partitions is the crucial notion. The following lemma relates it to the path-width of primal graphs.

Lemma 1. *Let $\phi(e, \theta)$ be a CNF formula depending on $n + s$ variables and $f : \{0, 1\}^n \mapsto \{0, 1\}$ a Boolean function such that:*

- *if $\phi(e, \theta) = 1$, then $f(e) = 1$*
- *if $f(e) = 1$, then there exists a θ such that $\phi(e, \theta) = 1$.*

Consider an arbitrary path-decomposition $(X_1, \dots, X_p)$ of $P(\phi)$ of width k. Choose a node X_i of the decomposition and a partition x, y of the variables e such that all variables of type e that have already occurred among those in $X_1, \dots, X_{i-1}$ are distributed to x and all the ones that never occur in $X_1, \dots, X_i$ to y. Then the non-deterministic communication complexity of f with respect to (x, y) is at most $k + 2$.

Proof. We split ϕ as follows into two CNF formulas ϕ_1 and ϕ_2 such that $\phi = \phi_1 \wedge \phi_2$ and ϕ_1 and ϕ_2 have at most $k + 1$ variables in common. Formula ϕ_1 is made of all clauses in ϕ that only contain variables that appear in $X_1, \dots, X_{i-1}$. The remaining clauses are collected in ϕ_2. Due to the path-width conditions only variables in X_i can be common variables of ϕ_1 and ϕ_2.

Note that all variables in x that appear in ϕ_2 must belong to X_i, and that no variables in y appear in ϕ_1.

Now given an assignment of the variables (x, y), let the first processor complete its assignment x by guessing non-deterministically the values of the remaining variables needed to compute ϕ_1 — that is, variables of θ since no variables in y appear in ϕ_1. Similarly, the second processor completes its assignment of y by guessing the values of the remaining variables appearing in ϕ_2 — variables of θ, and variables of x appearing in X_i as remarked previously.

Let the first processor send to the second processor the result of its computation of ϕ_1 along with the values of the variables in its assignment that ϕ_2 also uses. Those are variables in x appearing in ϕ_2, and variables from θ that are common to ϕ_1 and ϕ_2. Thus they all appear in X_i. As a result, the first processor sends at most $|X_i| + 1 \leq k + 2$ bits.

With those values, the second processor can check if the values of its guesses are consistent with the values the first processor had, and if both the computations of ϕ_1 and ϕ_2 are accepting.

Thus, if $e = (x, y)$ does not satisfy f, no guesses of the variables θ could complete e in an assignment that satisfies both ϕ_1 and ϕ_2 and the protocol will never be accepting; and if $f(e) = 1$, then if the two processors guess the proper values to compute ϕ_1 and ϕ_2 on the existing assignment (e, θ) that satisfies ϕ, both ϕ_1 and ϕ_2 will be satisfied, and the protocol will be accepting. $\square$

Remark 2. At the end of this section we obtain a similar lemma in order to obtain some results of independent interest relating best-case deterministic communication complexity and path-width.

An outline of the lower bound proof is as follows: Given a CNF formula for the function PERMUT_n and a partition of the variables as above we next define certain permutations called balanced. The number of balanced permutations can be upper bounded in terms of the non-deterministic communication complexity, by Lemma 2. Then in Lemma 3 we show that a CNF formula for the permanent function gives rise to a partition of the variables with sufficiently many balanced permutations. Combining this with Lemma 1 and the well known relation between path- and tree-width gives the following lower bound result:

Theorem 8 (*Lower Bound for the Permanent*)**.** *Let $(\phi_n)_{n\in\mathbb{N}}$ be a family of CNF formulas $\phi_n(e,\theta)$ in n^2 variables $e=(e_{ij})$ and s_n auxiliary variables θ such that:*

- *if $\phi_n(e,\theta)=1$, then the matrix $e\in\{0,1\}^{n\times n}$ is a permutation matrix*
- *if $e\in\{0,1\}^{n\times n}$ is a permutation matrix, then there exists θ such that $\phi_n(e,\theta)=1$.*

Then the path-width $p(n)$ of the primal graphs $P(\phi_n)$ verifies $p(n)=\Omega(n)$, and the tree-width $t(n)$ verifies $t(n)=\Omega(n/\log(n+s_n))$.

As a result, the general permanent function cannot be expressed by a family of CNF formulas with a polynomial number of auxiliary variables and an incidence graph of bounded tree-width.

Remark 3. The above lower bounds are independent of the size of the CNF formulas.

Remark 4. It seems possible to improve the $t(n)=\Omega(n/\log(n+s_n))$ lower bound by working directly with tree decompositions instead of path decompositions. The proofs would get more cumbersome but do not seem to require new ideas. We therefore stick to path decompositions in the remainder of this section.

We proceed as outlined above with

Definition 11. For $n\in\mathbb{N}$ let $\phi_n(e,\theta)$ be a CNF formula in n^2 variables $(e_{ij})_{1\le i,j\le n}$ and s variables $\theta_1,\ldots,\theta_s$, s arbitrary. Consider a partition (x,y) of the variables e into two disjoint blocks x and y. A permutation $\pi:\{1,\ldots,n\}\mapsto\{1,\ldots,n\}$ is called *balanced with respect to the partition* (x,y) if among the n variables $e_{i,\pi(i)}$, $1\le i\le n$ precisely $\lceil\frac{n}{2}\rceil$ belong to x and $\lfloor\frac{n}{2}\rfloor$ belong to y.

Thus, if (e_{ij}) represents the matrix of a permutation π and if π is balanced with respect to (x,y), then (almost) half of those e_{ij} with value 1 belong to x and the other half to y.

Lemma 2. *Let $\phi_n(e,\theta)$ be a CNF formula which evaluates to 1 only if e is a permutation matrix as in the statement of Theorem 8. Suppose ϕ_n has n^2 variables $e=(e_{ij})$ and s_n variables θ, and let x,y be a partition of e. If there are m balanced permutations with respect to (x,y), then the non-deterministic communication complexity c of the function $f_n:=\mathrm{PERMUT}_n$ with respect to (x,y) satisfies*

$$m\le 2^c\cdot(\lceil n/2\rceil!)^2\,.$$

Proof. Consider the matrix $(f_n(x,y))$ as defined in Theorem 7, where rows and columns are marked by the possible assignments for x and y, respectively. If π is a permutation which is balanced with respect to (x,y), we denote by $(x(\pi),y(\pi))$ the corresponding assignments for the (e_{ij}) and we denote by $R(\pi)$ the row of index $x(\pi)$ in the communication matrix $(f_n(x,y))$.

We wish to compute an upper bound K such that any monochromatic rectangle covers at most K balanced permutations. The point then is that the communication matrix will have at least m/K distinct rectangles since there are m balanced permutations. We can then conclude that $m\le 2^c\cdot K$ by Theorem 7.

Towards this aim let A be a rectangle covering the value 1 corresponding to π in the matrix. This rectangle is the intersection of a certain set of rows and a certain set of columns. Since π is covered by A, $R(\pi)$ belongs to that set of rows. Let C be one of the columns.

The intersection of $R(\pi)$ and C belongs also to A, and thus contains a 1. Thus, the assignment y_c indexing C completes $x(\pi)$ in a satisfying assignment of f_n. Since π is balanced, there are $\lceil n/2\rceil$ variables set to 1 in $x(\pi)$. If $x(\pi)$, y_c are to form a permutation matrix, y_c must have exactly $\lfloor n/2\rfloor$ variables set to 1, distributed in the intersection of the $\lfloor n/2\rfloor$ rows and columns without any 1 in the assignment $x(\pi)$.

Thus, there are at most $\lfloor n/2\rfloor!$ possible values for y_c, and thus at most $\lfloor n/2\rfloor!$ possible columns in A. Symmetrically, there are at most $\lceil n/2\rceil!$ possible rows in A. Finally one can take $K=\lceil n/2\rceil!\cdot\lfloor n/2\rfloor!$, and the conclusion of the lemma follows from the inequality $m\le 2^c\cdot K$. $\square$

The final ingredient for the lower bound proof is

Lemma 3. *Let ϕ_n be as in Lemma 2. There exists a partition of e into two sets of variables x,y such that this partition is as in the statement of Lemma 1 and such that there are at least $n!\cdot n^{-2}$ many balanced permutations with respect to (x,y).*

Proof. Let $(X_1, X_2, \ldots, X_p)$ be the nodes of a path-decomposition of $P(\phi_n)$ (in that order). We define an ordering on the e_{ij}'s as follows: for an e_{ij} let $\underline{X}(e_{ij})$ be the first node in the path-decomposition containing e_{ij}. We put $e_{ij} < e_{kl}$ if $\underline{X}(e_{ij}) < \underline{X}(e_{kj})$. If both values are equal for e_{ij} and e_{kl} we order them arbitrarily but in a consistent way to achieve transitivity.

Consider a permutation π. There are precisely n variables of the form $e_{i\pi(i)}$. We pick according to the above order the $\lceil \frac{n}{2} \rceil$-th among those and denote it by e_π. Thus, among the $e_{i\pi(i)}$ exactly $\lfloor \frac{n}{2} \rfloor$ are greater than e_π and $\lceil \frac{n}{2} \rceil$ are less than or equal to e_π with respect to the defined order. By the pigeonhole principle there is at least one variable e_ℓ among the n^2 many e_{ij}'s such that for at least $\frac{n!}{n^2}$ many permutations of $\{1, \ldots, n\}$ we get that same e_ℓ by the above procedure, i.e., $e_\pi = e_\ell$ for all those π. We choose a partition (x, y) of the e_{ij} as follows. The part x consists of all the variables e_{ij} that are less than or equal to e_ℓ, and the part y of the variables that are greater than v_ℓ. The partition (x, y) is as stated in Lemma 1, where the node $\underline{X}(e_\ell)$ plays the role of the X_i in the Lemma. The above arguments imply that at least $\frac{n!}{n^2}$ many permutations are balanced with respect to this partition. □

Proof of Theorem 8. Let ϕ_n be as in the theorem's statement. According to Lemma 3 there is a partition of the variables with at least $\frac{n!}{n^2}$ many balanced permutations. According to Lemmas 1 and 2 the path-width k of $P(\phi_n)$ satisfies

$$\frac{n!}{n^2} \leq 2^{k+2} \times (\lceil n/2 \rceil !)^2.$$

Using Stirling's formula we deduce that $k = \Omega(n)$. Now the tree-width t of ϕ_n satisfies $t \in \Omega(k/\log(n+s_n))$ which results in $t \in \Omega(n/\log(n+s_n))$. Finally, the statement about the tree-width of ϕ_n's incidence graph follows from Proposition 2. □

Remark 5. The lower bound obtained above does not seem derivable from the known lower bounds on computing the permanent with monotone arithmetic circuits, see, e.g., [7]. The tree-width based algorithms for polynomial evaluation like the one in [3] are not monotone since they rely on the principle of inclusion and exclusion.

We close this subsection by strengthening slightly Lemma 1 in order to apply it also to the best-case communication complexity (Definition 9) and obtain some lower bound results of independent interest.

Lemma 4. *Let ϕ be a CNF formula depending on $2n$ variables. Assume that the primal graph $P(\phi)$ has path-width $k-1$. Then ϕ can be expressed as $\phi_1 \wedge \phi_2$ for CNF formulas ϕ_1, ϕ_2 such that both have at most k variables in common and both depend on at least $n - \frac{k}{2}$ variables which do not occur in the other formula.*

Proof. We briefly sketch how the splitting of ϕ done in Lemma 1 can be performed more carefully such that both formulas ϕ_1 and ϕ_2 depend at least on a certain number of variables. For notational simplicity assume k to be even. Let $(X_1, X_2, \ldots, X_p)$ be a path-decomposition of $P(\phi)$; order the variables once again as done in the proof of Lemma 3. Denote the ordered sequence by $v_1 < \cdots < v_{2n}$. Choose $i := n + \frac{k}{2}$ and let $X_\ell := \underline{X}(v_i)$. Define ϕ_1 as conjunction of those clauses in ϕ containing only variables among the $v_1, \ldots, v_i$ and ϕ_2 as a conjunction of all remaining clauses. Remark, that the $n - k/2$ variables $v_{i+1}, \ldots, v_{2n}$ do not occur in ϕ_1. Due to the path-width conditions the common variables in ϕ_1 and ϕ_2 must be variables in X_ℓ. Thus, there are at most k many. Moreover, X_ℓ contains at most k among the $n + \frac{k}{2}$ variables $x_1, \ldots, x_i$. Therefore at least $n - \frac{k}{2}$ of these occur for the last time in some $X_{\ell'}$, where $\ell' < \ell$ and ϕ_2 cannot depend on them. □

As consequence, Lemma 1 now also holds with respect to the best-case communication complexity (Definition 9) of the function represented by ϕ.

Corollary 2. *The best-case communication complexity of a function $f : \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is lower than $k + 1$, where $k - 1$ is the path-width of the primal graph of any CNF formula computing f.*

Proof. Let ϕ be a formula computing f, and $k - 1$ be its path-width. By Lemma 4, one can write ϕ as a conjunction $\phi_1 \wedge \phi_2$, where ϕ_1 and ϕ_2 have each $n - k/2$ variables not shared with the other formula. Let us consider a partition (x, y), where x contains the $n - k/2$ variables, that belong to ϕ_1 exclusively, and y the $n - k/2$ variables, that belong to ϕ_2 exclusively, the remaining variables being distributed arbitrarily so that $|x| = |y| = n$.

With this partition, the communication complexity of $f(x, y)$ is lower than $k + 1$. Indeed, two processors, one having the variables x and the other one the variables y, can exchange the at most $k/2$ variables that the first needs to compute ϕ_1, and the at most $k/2$ variables that the second needs to compute ϕ_2. Then, if the first processor sends the result of its computation on ϕ_1 — which is a single bit — the second can return the value of f.

Thus, the best case communication complexity is lower than $k + 1$. □

If for a function f the best-case communication complexity is known, then we can use the corollary to deduce lower bounds for the path- and tree-width of CNF formulas representing f.

Example 1. For $x, y \in \{0, 1\}^n$, $1 \leq i \leq n$ consider the boolean function $SEQ(x, y, i)$ which gives result 1 iff the string $x = x_0 x_1 \ldots x_n$ equals the string y shifted circularly by i bits to the right, that is to $y_i y_{i+1} \ldots y_{n-1} y_0 \ldots y_{i-1}$. It is known [10] that SEQ has a best case communication complexity which is at least linear in the size of the input. Thus, the path-width of the primal graph of any CNF formula computing SEQ is at least linear in the input.

The same argument holds as well for the function $PROD(a, b, i)$ which computes the i-th bit of the product $a \cdot b$, for the function $MATCH$ which on a $3m$-string x and an m-string y returns 1 iff y is a substring of x, and for the function $USTCON$ which on a graph with ℓ vertices and two given vertices s and t outputs 1 if there exists a path from s to t. As noted in [10] the best-case communication complexity of those functions is, respectively, linear, $\Omega(m/\log(m))$ and $\Omega(\sqrt{n})$. Consequently, they do not admit CNF formulas of path-width, respectively, linear, $\Omega(m/\log(m))$ and $\Omega(\sqrt{n})$.

Since the path-width p and the tree-width t are related via $p = O(t \cdot \log n)$, all the above mentioned examples do not admit CNF formulas with a primal graph of bounded or even logarithmic tree-width.

4.2. Hardness for clique-width

The question answered negatively by Theorem 8 for tree-width can be posed as well in relation to the clique-width parameter. That is: Can the permanent function be described via CNF formulas of bounded clique-width and polynomial size? Next we relate this question to Theorem 2(b) and show that such a representation is only possible if the conjecture $\#P \not\subseteq FP/poly$ fails to be true.

Theorem 9. *Suppose there is a family $\{\varphi_n\}_n$ of CNF formulas of polynomial size such that all $I(\varphi_n)$ are of clique-width at most k for some fixed k and for each $Y \in \{0,1\}^{n^2}$ we have that $\varphi_n(Y)$ holds iff Y is a permutation matrix. Then $\#P \subseteq FP/poly$.*

The result holds similarly if we allow additional variables in $\varphi_n(Y)$ as in the statement of Theorem 8.

Proof. Suppose $\{\varphi_n\}$ is given as in the assumption. For a matrix $X \in \{0,1\}^{n^2}$ we shall construct from φ_n and a parse-tree of it (given as non-uniform advice) another CNF formula $\psi_n^X(Y)$ of bounded clique-width together with a parse-tree for ψ_n^X such that

$$Perm(X) = \sum_{Y \in \{0,1\}^{n^2}} \psi_n^X(Y).$$

Theorem 2(b) implies that the latter can be computed in polynomial time. Given $\#P$-completeness of the permanent function on 0–1-matrices the claim follows.

The construction of ψ_n^X works as follows. It is $Perm(X) = \sum_{Y \in \{0,1\}^{n^2}} \varphi_n(Y) \cdot X^Y$, where $X^Y = \prod_{i,j} x_{i,j}^{y_{i,j}}$ and $x_{i,j}^{y_{i,j}} = \begin{cases} x_{i,j} & \text{if } y_{i,j} = 1 \\ 1 & \text{otherwise} \end{cases}$.

We replace the monomial X^Y by the conjunctions $\bigwedge_{i,j}(x_{i,j} \vee \neg y_{i,j})$. The clause graph $I(\psi_n)$ of the CNF formula

$$\psi_n(X, Y) \equiv \varphi_n(Y) \wedge \bigwedge_{i,j} (x_{i,j} \vee \neg y_{i,j})$$

can easily be seen to have clique-width $\leq k + 2$. Each time when in the clique-width construction of $I(\varphi_n)$ along the parse-tree a node $y_{i,j}$ is created, in the corresponding construction for $I(\psi_n)$ two new nodes for $x_{i,j}$ and the clause $D_{i,j} := x_{i,j} \vee \neg y_{i,j}$ are created with an own label each. Then $D_{i,j}$ is connected to both $x_{i,j}$ and $y_{i,j}$ (respecting the necessary signs of the edges). Finally the labels for $D_{i,j}$ and $x_{i,j}$ are removed again.

Now for a fixed given matrix X we plug the values of the $x_{i,j}$ into the CNF formula $\psi_n(X, Y)$. Clauses that are satisfied by the assignment are removed. In clauses that are not satisfied by the assignment all occurrences of $x_{i,j}$ literals are removed. That way a new CNF formula ψ_n^X is obtained. The clause graph $I(\psi_n^X)$ results from $I(\psi_n)$ by

(i) removing certain nodes (the $x_{i,j}$ as well as some clause nodes) and
(ii) identifying certain clause nodes.

Both operations do not increase the clique-width. Being clear for (i) it is also true for (ii) since two or several clauses that are identified after having assigned values to the $x_{i,j}$'s must contain the same $y_{i,j}$'s. Thus, this part has been dealt with in the parse-tree construction for $I(\psi_n)$ already and can be taken as well for the parse-tree construction of $I(\psi_n^X)$.

The proof when including additional variables in $\varphi_n(Y)$ works the same. □

It remains an open question whether Theorem 9 can be strengthened to hold unconditionally, like Theorem 8:

Conjecture. *The hypothesis of Theorem 9 is impossible.*

Acknowledgements

We thank the two anonymous referees for their very careful reading of the manuscript and the many useful comments. Support of the following institutions is gratefully acknowledged: ENS Lyon; the French embassy in Denmark, Service de Coopération et d'Action Culturelle, Ref.:39/2007-CSU 8.2.1; the Danish Agency for Science, Technology and Innovation FNU.

References

[1] H. Bodlaender, T. Hagerup, Parallel Algorithms with Optimal Speedup for Bounded Treewidth, SIAM Journal on Computing 27 (6) (1998) 1725–1746.
[2] B. Courcelle, J.A. Makowsky, U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic, Discrete Applied Mathematics 108 (1–2) (2001) 23–52.

[3] E. Fischer, J. Makowsky, E.V. Ravve, Counting truth assignments of formulas of bounded tree-width or clique-width, Discrete Applied Mathematics 156 (2008) 511–529.
[4] U. Flarup, P. Koiran, L. Lyaudet, On the expressive power of planar perfect matching and permanents of bounded tree-width matrices, in: Proc. 18th International Symposium ISAAC, in: Lecture Notes in Computer Science, vol. 4835, Springer, 2007, pp. 124–136.
[5] G. Gottlob, R. Pichler, Hypergraphs in model checking: Acyclicity and hypertree-width, in: Proc. 28th International Colloquium on Automata, Languages and Programming ICALP, in: Lecture Notes in Computer Science, vol. 2077, Springer, 2003, pp. 237–248.
[6] F. Gurski, E. Wanke, The tree-width of clique-width bounded graphs without $K_{n,n}$, in: Proc. Graph-Theoretical Concepts in Computer Science, in: Lecture Notes in Computer Science, vol. 1938, Springer, 2000, pp. 196–205.
[7] M. Jerrum, M. Snir, Some exact complexity results for straight-line computations over semirings, Journal of the ACM 29 (3) (1982) 874–897.
[8] P. Koiran, K. Meer, On the expressive power of CNF formulas of bounded tree- and clique-width, in: Proc. 34th International Workshop on Graph-Theoretic Concepts in Computer Science WG, in: Lecture Notes in Computer Science, vol. 5344, Springer, 2008, pp. 252–263.
[9] E. Korach, N. Solel, Tree-width, path-width, and cut-width, Discrete Applied Mathematics 43 (1) (1993) 97–101.
[10] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge University Press, 1997.
[11] G. Malod, N. Portier, Characterizing Valiant's Algebraic Complexity Classes, in: Proc. 31st International Symposium on Mathematical Foundations of Computer Science MFCS, in: Lecture Notes in Computer Science, vol. 4162, Springer, 2006, pp. 704–716.
[12] M. Samer, S. Szeider, Algorithms for Propositional Model Counting, in: Proc. LPAR 2007, in: Lecture Notes of Computer Science, vol. 4790, Springer, 2007, pp. 484–498.
[13] A. Yao, Some complexity questions related to distributive computing, in: Proceedings of the 11th Annual ACM Symposium on Theory of Computing STOC, 1979, pp. 209–213.