Electronic Colloquium on Computational Complexity, Report No. 38 (2018)

Tight Bounds using Hankel Matrix for Arithmetic Circuits with Unique Parse Trees*

Nathanaël Fijalkow[1,2,3], Guillaume Lagarde[4], and Pierre Ohlmann[4]

1 **CNRS, LaBRI, Bordeaux, France**
2 **Alan Turing Institute of Data Science, London, United Kingdom**
3 **University of Warwick, United Kingdom**
4 **IRIF, Université Paris 7, France**

Abstract

This paper studies lower bounds for arithmetic circuits computing (non-commutative) polynomials. Our conceptual contribution is an exact correspondence between circuits and weighted automata: algebraic branching programs are captured by weighted automata over words, and circuits with unique parse trees by weighted automata over trees.

The key notion for understanding the minimisation question of weighted automata is the Hankel matrix: the rank of the Hankel matrix of a word or tree series is exactly the size of the smallest weighted automaton recognising this series. For automata over words, the correspondence we establish allows us to rephrase Nisan's celebrated tight bounds for algebraic branching programs. We extend this result by considering automata over trees and obtain the first tight bounds for all circuits with unique parse trees.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Algebraic Branching Programs, Arithmetic Circuits, Weighted Automata

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Arithmetic circuits is the algebraic analogue of boolean complexity. Boolean functions are replaced by *multivariate polynomials*, and the computation models are *arithmetic circuits*, which compute polynomials through additions and multiplications. The measure of complexity of a circuit is the number of such operations. As in boolean complexity, obtaining tight bounds for general circuits is hard. For instance, we do not know of any explicit polynomial with super polynomial lower bounds. The best lower bounds were given by Baur and Strassen [2] for the polynomial $\sum_{i=1}^{n} x_i^d$, which requires $\Omega(n \log d)$ operations. We refer to [13] for a recent survey on arithmetic circuits.

Non-commutative polynomial computation was initiated by Nisan [10]. In this setting the variables do not commute, and therefore xy and yx are considered two distinct monomials. Non-commutative computations arise in different scenarios, for instance when performing matrix multiplication. Another interesting aspect is that the non-commutativity constraints make the computation of a polynomial harder than its commutative counterpart and therefore the quest for lower bounds is easier. In his seminal paper Nisan [10] showed exponential lower bounds for the non-commutative permanent computed by formulas or

* This project has received funding from the Alan Turing Institute under EPSRC grant EP/N510129/1 and the DeLTA project (ANR-16-CE40-0007).

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ISSN 1433-8092

algebraic branching programs (ABPs). Later, Limaye, Malod and Srinivasan [9] proved exponential lower bounds for a generalisation of ABPs called *skew circuits* where each multiplication gate has at most one argument which is not an input.

Circuits with unique parse trees were introduced by Malod, Perifel, and the second author [8] as an orthogonal generalisation of ABPs. A non commutative monomial can be computed in different ways: for example the monomial $x_1x_2x_3$ can be computed either as $x_1(x_2x_3)$ or $(x_1x_2)x_3$. These decompositions yield the notion of *parse trees* (we refer to Section 3 for a formal definition) that are allowed to appear in the circuits.

One of the main motivation for studying UPT circuits is to see them as building blocks. To obtain lower bounds for more general circuits one may decompose them into UPT circuits and prove lower bounds for these. This is the approach taken for instance in [7].

The paper [8] provides an algebraic characterisation of the size of circuits with unique parse trees (known as *unambiguous circuits* or *UPT circuits*) in canonical form together with exponential lower bounds for polynomials such as the permanenent. Later, Limaye, Srinivasan and the second author [7] extended these results to obtain exponential lower bounds for polynomials computed by circuits with up to $2^{d^{1/4}}$ different parse trees.

Saptharishi and Tengse [12] consider circuits with parse tree restrictions and give quasi-polynomial hitting sets for them implying algorithms for polynomial identity testing. In the same fashion, Arvind and Raja [1] show lower bounds for restricted commutative set-multilinear circuits, where a given monomial can be computed inside the circuit by a unique parse tree but two distinct monomials may have different parse trees.

Correspondence with weighted automata. In this paper we build a bridge between automata theory and arithmetic complexity. The correspondence is summarised in this table.

Arithmetic complexity	**Weighted automata**
variable	letter
monomial	word
polynomial	word or tree series
algebraic branching program	layered weighted automaton over words
circuit with unique parse trees	layered weighted automaton over trees

In Section 2, we show that Nisan's results about algebraic branching programs follow from theorems coming from automata theory. We introduce weighted automata (WA) over words, and show the following correspondence:

- any ABP can be seen as a WA,
- under some syntactic restriction called layered, a WA can be seen as an ABP,
- for a given word series f, the size of the minimal WA recognising f is the rank of the Hankel matrix of f,
- if the word series f represents a homogeneous polynomial P, then the minimal WA recognising f is an ABP computing P.

This gives an alternative approach to state and prove Nisan's result about minimal ABPs.

Our main technical contribution is to extend this result from ABPs to UPT circuits in Section 3. To this end we show a more subtle and involved correspondence between UPT circuits and weighted automata over trees. Our main result is to obtain tight bounds on the size of a UPT circuit computing a given polynomial.

A similar result was obtained in [8] for UPT circuits in *canonical form*; here we remove this assumption, leading to an exponential improvement in some examples as explained in Section 4.

Throughout the paper X is a finite set of non-commuting variables. We consider homogeneous polynomials over X: all monomials have the same degree d. For instance, $xyx + 4x^2y + 2y^3$ is a homogeneous polynomial of degree 3 over the variables $\{x, y\}$. In all examples we use real numbers, but all of our results are valid for any field.

2 Tight bounds for algebraic branching programs

▶ **Definition 1.** An *algebraic branching program* (ABP) is a directed acyclic graph with a distinguished source vertex s. The vertices are partitioned into $d + 1$ layers, starting with layer 0 which contains only the vertex s, and ending in layer d. Each edge is between two consecutive layers and is labeled by a homogeneous linear function over the variables X and real-valued constants. Each vertex in the last layer has a real output value.

A path from s to a vertex in layer d induces a homogeneous polynomial of degree d obtained by multiplying all the labels of the edges and the output value. An algebraic branching program $\mathcal{C}$ computes a homogeneous polynomial $P_{\mathcal{C}}$ defined by summing the polynomials over all paths from s to vertices of the layer d.

The size of an ABP is its number of vertices.

One of the motivations for studying algebraic branching programs is that they capture matrix multiplication. They can be proved to be equivalent to left skew circuits (*i.e.* circuits for which the left argument of any multiplication gate is an input).

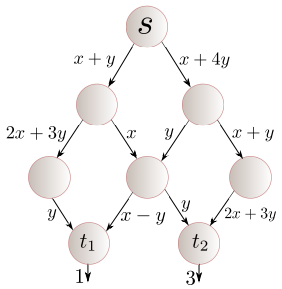

Figure 1 An example of an ABP with 4 layers and of size 8.

Nisan's theorem

Nisan's theorem gives for a homogeneous polynomial P the size of the smallest ABP computing P. Let d be the degree of P, for each $i \in \{0, \dots, d\}$, we define a matrix $M_{P,i}$ as follows. The rows are indexed by monomials of degree i, and the columns by monomials of degree $d - i$ (there are $|X|^i$ rows and $|X|^{d-i}$ columns). Then for u a monomial of degree i and v of degree $d - i$, define $M_{P,i}(u, v)$ to be the coefficient of uv in P.

▶ **Theorem 2** (Nisan's theorem). *Let P be a homogeneous polynomial of degree d, and*

$$n = \sum_{i=0}^{d} rank(M_{P,i}).$$

- *Any ABP computing P has size at least n,*
- *There exists an ABP computing P of size exactly n.*

In this section, we give an alternative proof of this theorem using a correspondence with weighted automata over finite words. The number n will appear as the rank of a single matrix called the Hankel matrix, which will in the case at hand consists of $d+1$ independent blocks, hence the summation in Nisan's theorem.

The point of this section is to serve as an introduction to our main result in Section 3. Indeed, we will prove a theorem extending Nisan's theorem to a wider class of arithmetic circuits, namely circuits with unique parse trees, following the same schema. In this section we deal with weighted automata over words, in the next section we will consider weighted automata over trees.

Weighted automata over words

An element of X^* can be seen either as a monomial over the variables X, as in ABPs, or as a (finite) word over the alphabet X. A word series is a function $f : A^* \to \mathbb{R}$.

A polynomial P with variables in X can be seen as a word series $X^* \to \mathbb{R}$ which we also write P, such that $P(w)$ is the coefficient of w in P.

▶ **Definition 3.** A *weighted automaton over words* (WA) is given by
- a finite set of states Q,
- an initial state $q_0 \in Q$,
- a transition function $\Delta : Q \times X \times Q \to \mathbb{R}$,
- an output function $F : Q \to \mathbb{R}$.

The usual definition proceeds with introducing runs, explaining that along a run the weights of the transitions are *multiplied*, and that the value of a word is the *sum* of the values of its accepting runs. We use here a more algebraic equivalent definition. Equivalently, we see the transition function as $\Delta : X \to \mathbb{R}^{Q\times Q}$, *i.e.*, $\Delta(x)$ is a matrix defined by $\Delta(x)(p, q) = \Delta(p, x, q)$. The initial state q_0 (seen as an element of $\mathbb{R}^Q$) and the transition function induce $\Delta^* : X^* \to \mathbb{R}^Q$ defined by $\Delta^*(\varepsilon) = q_0$ and $\Delta^*(wx) = \Delta^*(w) \cdot \Delta(x)$, where $\cdot$ is matrix multiplication. Similarly, we see F as a vector in $\mathbb{R}^Q$.

The weighted automaton $\mathcal{A}$ recognises the word series $f_\mathcal{A} : X^* \to \mathbb{R}$ defined by

$$f_\mathcal{A}(w) = \Delta^*(w) \cdot F,$$

where $\cdot$ is the dot product in $\mathbb{R}^Q$.

The size of a WA is its number of states.

Algebraic branching programs as weighted automata over words

ABPs form a subclass of WA over words that we define now.

▶ **Definition 4.** A weighted automaton $\mathcal{A} = (Q, q_0, \Delta, F)$ is *d-layered* if Q can be partitionned into $d + 1$ subsets $Q_0, \dots, Q_d$ such that

(1) $Q_0 = \{q_0\}$,
(2) for all $x \in X, q, q' \in Q$, if $\Delta(q, x, q') \neq 0$ then there exists $i \in \{0, \dots, d-1\}$ such that $q \in Q_i$ and $q' \in Q_{i+1}$,
(3) for all $q \in Q$, if $F(q) \neq 0$ then $q \in Q_d$.

▶ **Lemma 5.**

- *For all ABP $\mathcal{C}$, there exists a WA over words $\mathcal{A}$ of the same size such that $f_{\mathcal{A}} = P_{\mathcal{C}}$.*
- *For all WA over words $\mathcal{A}$ which is layered, there exists an ABP $\mathcal{C}$ of the same size such that $P_{\mathcal{C}} = f_{\mathcal{A}}$.*

Proof. Both claims are syntactic easy transformations. We explicit the construction to help the reader's intuitions.

Let $\mathcal{C}$ be an ABP. We define a WA over words $\mathcal{A}$ as follows. The set of states is the set of vertices of $\mathcal{C}$, the initial state of $\mathcal{A}$ is the source vertex of $\mathcal{C}$, and the output function $F : Q \to \mathbb{R}$ is defined by $F(v)$ is the output value of v if v is on the last layer, and 0 otherwise. For the transition function, let $\Delta(v, x, v')$ be the coefficient of x in the linear function labeling the edge (v, v') if $(v, v') \in E$, and 0 otherwise. We have $f_{\mathcal{A}} = P_{\mathcal{C}}$.

For the second claim, the definition of d-layered WA over words exactly says that the above construction can be reverted. ◀

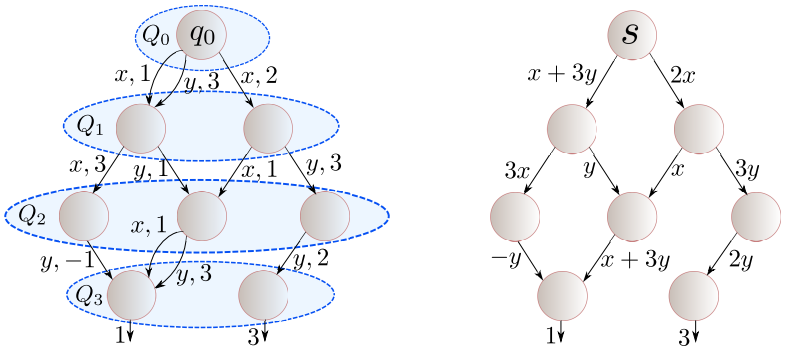

Figure 2 An example of 3-layered WA on the left, and its corresponding ABP on the right.

Fliess' theorem

The key notion in this paper is the Hankel matrix of a series.

▶ **Definition 6.** Let $f : X^* \to \mathbb{R}$, we define the (infinite) Hankel matrix $H^{(f)} \in \mathbb{R}^{X^*} \times \mathbb{R}^{X^*}$, whose rows and columns are indexed by words, by

$$H^{(f)}(u, v) = f(u \cdot v).$$

The notion of Hankel matrix and the rank of a formal non-commutative series were introduced by Carlyle and Paz [5]. One of the main results of Fliess' PhD thesis was the following theorem [6].

▶ **Theorem 7.** *Let $f : X^* \to \mathbb{R}$ be a word series such that rank$(H^{(f)})$ is finite.*

- *Any WA recognising f has size at least rank$(H^{(f)})$.*
- *There exists a WA recognising f of size exactly rank$(H^{(f)})$.*

This article is in French. However, one can find great exposition of the ideas in the handbooks of Berstel and Reutenauer [3] and Sakarovitch [11]. The proof of the second item gives a construction of the WA recognising f that we detail now as we will need it to prove Theorem 2.

Recall that the rows of $H^{(f)}$ are indexed by words in X^*. For $u \in X^*$, let $H_u^{(f)}$ be the row corresponding to u in $H^{(f)}$, which we see as $H_u^{(f)} \in \mathbb{R}^{X^*}$. Let $Q \subseteq X^*$ such that

$\left\{H_u^{(f)} \mid u \in Q\right\}$ is a basis of Span $\left\{H_u^{(f)} \mid u \in X^*\right\}$. We furthermore assume that $\varepsilon \in Q$, which is possible since $H_\varepsilon^{(f)} \neq 0$ unless f is the constant zero function.

We now construct the WA recognising f. The set of states is Q, the initial state is ε, and the output function is defined by $F(u) = f(u)$ for $u \in Q$. We now define the transition function. For $u \in Q$, there is a unique decomposition of $H_{ux}^{(f)}$ on the basis $\left\{H_u^{(f)} \mid u \in Q\right\}$:

$$H_{ux}^{(f)} = \sum_{v \in Q} \lambda(u, x, v) H_v^{(f)},$$

we define $\Delta(u, x, v) = \lambda(u, x, v)$.

Proof of Nisan's theorem

▶ **Lemma 8.** *Let P be a homogeneous polynomial of degree d. The automaton constructed above for recognising P is d-layered.*

Proof. Let $\mathcal{A} = (Q, \varepsilon, \Delta, F)$ be the automaton described in the previous subsection.

Since for u of length larger than d we have $H_u^{(f)} = 0$, it implies that $Q \subseteq X^{\leq d}$. For $i \in \{0, \dots, d\}$, we let $Q_i = Q \cap X^i$. The conditions (1) and (3) are clearly satisfied, so we focus on (2).

For $i \in \{0, \dots, d\}$, let V_i denote the vector space spanned by $\left\{H_u^{(P)} \mid u \in Q_i\right\}$. Note that for $u \in Q_i$ and v of length j, if $i + j \neq d$ then $H_u^{(P)}(v) = P(u \cdot v) = 0$, hence the same is true for any $L \in V_i$: if v has length j such that $i + j \neq d$, then $L(v) = 0$.

We claim that the subspaces $V_0, V_1, \dots, V_d$ are in direct sum. Indeed, assume that $\sum_{i=0}^d L_i = 0$ with $L_i \in V_i$. Let $j \in \{0, \dots, d\}$ such that $L_j \neq 0$, and consider a word $v \in X^{d-j}$. For $i \neq j$ we have $L_i(v) = 0$ thanks to the remark above. It follows that $L_j(v) = 0$ for all $v \in X^{d-j}$, implying again with the remark above that $L_j = 0$, a contradiction. Thus the subspaces $V_0, V_1, \dots, V_d$ are in direct sum.

Let $u \in Q$ and $x \in X$. By definition

$$H_{ux}^{(P)} = \sum_{v \in Q} \Delta(u, x, v) H_v^{(P)} = \sum_{i=0}^{d} \sum_{v \in Q_i} \Delta(u, x, v) H_v^{(P)}.$$

But since $H_{ux}^{(P)} \in V_{|u|+1}$ and the vector spaces $V_0, \dots, V_d$ are in direct sum, it follows that for $v \in Q$ of length $i \neq |u| + 1$ we have $\sum_{v \in Q_i} \Delta(u, x, v) H_v^{(P)} = 0$. Since the vectors $\left\{H_v^{(P)} \mid v \in Q_i\right\}$ are linearly independent, this implies that $\Delta(u, x, v) = 0$. Thus the property (2) is satisfied. ◀

We now explain how to obtain the proof of Nisan's theorem (Theorem 2) from the correspondence. Let P be a homogeneous polynomial of degree d. Thanks to the first item of Theorem 7 any WA recognising P has size at least $\operatorname{rank}(H^{(P)})$, and thanks to the first item of Lemma 5 this implies that any ABP computing P has size at least $\operatorname{rank}(H^{(P)})$. Thanks to the second item of Theorem 7, there exists a WA recognising P of size $\operatorname{rank}(H^{(P)})$, and thanks to Lemma 8, it is d-layered. Thanks to the second item of Lemma 5, it induces an ABP computing P of size $\operatorname{rank}(H^{(P)})$.

Let us have a closer look at the Hankel matrix of P for a homogeneous polynomial P of degree d. Most of the matrix is filled with zeros, except for $d+1$ independent blocks, which

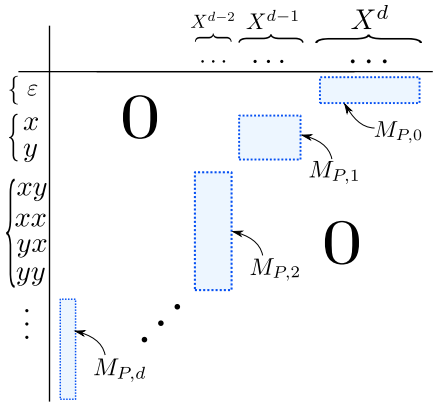

Figure 3 The Hankel matrix of P consists of $d+1$ independent blocks.

are precisely the matrices $M_{P,i}$ for $i \in \{0, \dots, d\}$. This explains the summation in Nisan's statement of the theorem, since $\operatorname{rank}(H^{(P)}) = \sum_{i=0}^{d} \operatorname{rank}(M_{P,i})$.

As we shall see in the next section, when considering UPT circuits the blocks will no longer be independent, allowing to share the result of partial computations.

3 Tight bounds for circuits with unique parse trees

▶ **Definition 9.** A circuit is a directed acyclic graph whose vertices, called gates, are of four different types.

- The *input gates* have indegree zero and are labeled with variables $x \in X$.
- The *addition gates* have unbounded fan-in and perform a linear combination of their inputs, with the associated coefficients α in $\mathbb{R}$ given on the edges.
- The *multiplication gates* have fan-in two, their arguments are ordered and the multiplication is interpreted according to this order (the left argument is multiplied before the right argument).
- The *output gates* have outdegree zero and are labeled with a real output value.

The *size* of an algebraic circuit is its number of addition gates.

While this definition allows multiple output gates, the circuits we construct only have one single output.

The reason for not taking the multiplication gates into account is an easy lemma from [8] proving that for UPT circuits (which are the only circuits we consider), if s is the number of addition gates, then the number of multiplication gates can always be bounded by s^2. Therefore, the number of addition gates gives a pretty good idea on the total size of UPT circuits. Observe also that the situation is similar in Nisan's work for ABPs as the vertices of an ABP correspond exactly to addition gates when this ABP is converted into circuits.

We also normalise the circuits by requiring that all paths alternate between addition gates and multiplication gates and start and finish with an addition gate, which increases the size by at most a linear factor.

▶ **Definition 10.** Given a circuit $\mathcal{C}$, we define its set of *parse trees* by induction on $\mathcal{C}$:

- if $\mathcal{C}$ has size 1 it has only one parse tree: itself.
- if the output gate of $\mathcal{C}$ is an addition gate g, then for each gate g' which is an argument of g and each parse tree of the subcircuit rooted at g', we obtain a parse tree of $\mathcal{C}$ by adding an edge from g' to g.

- if the output gate of $\mathcal{C}$ is a multiplication gate g whose arguments are the gates g_1 and g_2, the parse trees of $\mathcal{C}$ are obtained by taking a parse tree of the subcircuit rooted at g_1, a parse tree of a disjoint copy of the subcircuit rooted at g_2, and the edges from g_1 and g_2 to g.

The *shape* of a parse tree is the binary tree obtained by contracting the addition gates and removing all labels. Two parse trees C_1, C_2 are said to be *equivalent*, written $C_1 \sim C_2$, if their shapes are identical.

▶ **Definition 11.** A *circuit with unique parse tree* (UPT circuit) is a circuit where all parse trees are equivalent. The *shape* of a UPT circuit is the shape shared by all parse trees.

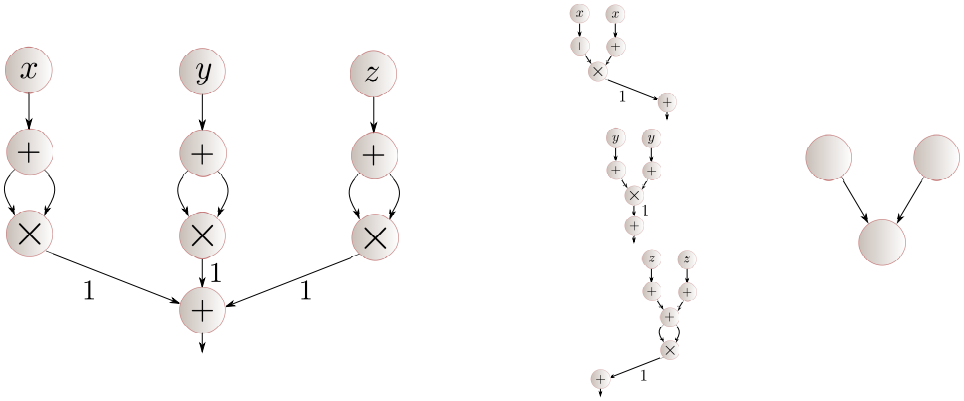

Figure 4 An example of a UPT circuit computing $x^2 + y^2 + z^2$. The circuit is on the left hand side, the set of parse trees in the middle, and the unique shape on the right hand side.

UPT circuits can be seen as circuits for which each monomial is computed in the same way given by the underlying shape. One can observe that ABPs also have this property. ABPs are exactly UPT circuits with the shape of a comb. Thus, UPT circuits subsume ABPs, and are more expressive. For instance, the *palindrome* polynomial over a set of n variables (defined in [9] or [8] for example) is computed by a UPT circuit of polynomial size, while the smallest ABP is of size $\Omega(n^{n/2})$, as witnessed for instance by the rank of the Hankel matrix. The importance of UPT circuits comes from the fact we can decompose less restricted circuits as sum of UPT circuits (see [7]). Having a good understanding of them can therefore help to get lower bounds for more expressive circuits.

Statement of the result

Shapes are binary trees without any labels, we use $T, T', \dots$ for shapes. They can be built inductively: a leaf is a shape, and given two shapes T_1, T_2, we construct the shape $T_1 \cdot T_2$.

We also consider labeled trees (later refered to as trees), which are binary trees whose leaves are labeled by variables $x \in X$. We let $\mathrm{Tree}(X)$ denote the set of trees, and use $t, t', \dots$ for them. Trees can be built inductively similarly as shapes, except that the basic trees are variables $x \in X$. A tree series is a function $f : \mathrm{Tree}(X) \to \mathbb{R}$.

For $t \in \mathrm{Tree}(X)$, we let $\widetilde{t}$ be the underlying shape of t, and for a shape T, we let $\mathrm{Tree}(X, T)$ denote the set of trees t such that $\widetilde{t} = T$. For instance for the shape T represented in Figure 4, $\mathrm{Tree}(X, T)$ has 9 elements obtained by labeling each of the two leaves by one of the three variables x, y, z.

We define the Hankel matrix for tree series. We first need the notion of contexts: a context C is an element of $\mathrm{Tree}(X \cup \{\Box\})$ with a unique leaf labeled $\Box$. We let $\mathrm{Context}(X)$ denote the set of contexts, and use $C, C', \dots$ for them. A context C and a tree t can be composed into a tree $C \circ t$ by replacing the placeholder $\Box$ by the tree t.

▶ **Definition 12.** Let $f : \mathrm{Tree}(X) \to \mathbb{R}$, we define the (infinite) Hankel matrix $H^{(f)} \in \mathbb{R}^{\mathrm{Tree}(X)} \times \mathbb{R}^{\mathrm{Context}(X)}$ by $H^{(f)}(t, C) = f(C \circ t)$.

Our main theorem gives tight bounds on the size of UPT circuits. More precisely, we consider a homogeneous polynomial P of degree d and a shape T, and find the size of the smallest UPT circuit with shape T computing P.

A necessary condition for the existence of such a circuit is that the number of leaves of T is d. Under this condition a homogeneous polynomial of degree d and a shape T induce a function $f_{P,T} : \mathrm{Tree}(X) \to \mathbb{R}$: for $t \in \mathrm{Tree}(X, T)$, we see t as a monomial and define $f_{P,T}(t)$ as the coefficient of this monomial in P, the function $f_{P,T}$ is zero outside of $\mathrm{Tree}(X, T)$. We refer to Figure 5 for an illustration of this definition.

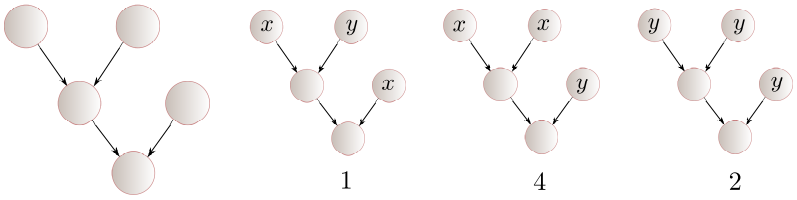

Figure 5 Given the shape T displayed on the left hand side, the polynomial $P = xyx + 4x^2y + 2y^3$ can be seen as the tree series $P : \mathrm{Tree}(X) \to \mathbb{R}$ associating with these three trees the values indicated below and zero to all other trees.

For the sake of simplicity we use $H^{(P,T)}$ instead of $H^{(f_{P,T})}$.

▶ **Theorem 13.** *Let P be a homogeneous polynomial of degree d, and T be a shape with d leaves.*

- *Any UPT circuit with shape T computing P has size at least $\mathrm{rank}(H^{(P,T)})$,*
- *There exists a UPT circuit with shape T computing P of size exactly $\mathrm{rank}(H^{(P,T)})$.*

Weighted automata over trees

▶ **Definition 14.** A *weighted automaton over trees* (WA) is given by

- a finite set of states Q,
- an initial function $\iota : X \times Q \to \mathbb{R}$,
- a transition function $\Delta : Q \times Q \times Q \to \mathbb{R}$,
- an output function $F : Q \to \mathbb{R}$.

Equivalently, we write the transition function as a bilinear function $\Delta : \mathbb{R}^Q \times \mathbb{R}^Q \to \mathbb{R}^Q$ defined by $\Delta(p, q)(r) = \Delta(p, q, r)$.

The initial and transition functions induce $\Delta^* : \mathrm{Tree}(X) \to \mathbb{R}^Q$ defined by $\Delta^*(x) = \iota(x) \in \mathbb{R}^Q$ and $\Delta^*(t_1 \cdot t_2) = \Delta(\Delta^*(t_1), \Delta^*(t_2))$.

The weighted automaton $\mathcal{A}$ recognises the tree series $f_{\mathcal{A}} : \mathrm{Tree}(X) \to \mathbb{R}$ defined by

$$f_{\mathcal{A}}(t) = \Delta^*(t) \cdot F,$$

where $\cdot$ is the dot product in $\mathbb{R}^Q$.

UPT circuits as weighted automata over trees

UPT circuits form a subclass of WA over trees that we define now.

Let T be a shape and v a node of T, we let T_v be the subshape of T rooted in v. We let $[T]$ denote $\{T_v \mid v \text{ node of } T\}$.

▶ **Definition 15.** A weighted automaton $\mathcal{A} = (Q, \iota, \Delta, F)$ is *T-layered* if there exists a map $m : Q \to [T]$ such that

(1) for all $x \in X$, if $\iota(x, q) \neq 0$ then $m(q)$ is the shape reduced to a single leaf,
(2) for all $q, q_1, q_2 \in Q$, if $\Delta(q, q_1, q_2) \neq 0$, then $m(q) = m(q_1) \cdot m(q_2)$,
(3) for all $q \in Q$, if $F(q) \neq 0$ then $m(q) = T$.

▶ **Lemma 16.**

- *For all UPT $\mathcal{C}$ with shape T, there exists a T-layered WA over trees $\mathcal{A}$ of the same size such that $f_{\mathcal{A}} = P_{\mathcal{C}}$.*
- *For all WA over trees $\mathcal{A}$ which is T-layered, there exists a UPT $\mathcal{C}$ with shape T of the same size such that $P_{\mathcal{C}} = f_{\mathcal{A}}$.*

Proof. Let $\mathcal{C}$ be a UPT circuit. We define a WA over trees as follows. The set of states is the set of addition gates of $\mathcal{C}$. The initial function is defined by $\iota(g, x)$ is the label of the edge coming from an input gate with label x to g, and 0 if there is no such edge. The output function is defined by $F(g)$ is the label of g if g is an output gate, and 0 otherwise. The transition function is defined as follows: $\Delta(g_1, g_2, g)$ is the label of (the unique) multiplication gate g' using g_1 and g_2 as arguments and g' argument of g. Then $f_{\mathcal{A}} = P_{\mathcal{C}}$.

For the second claim, the definition of layered WA over trees exactly says that the above construction can be reverted. ◀

Minimisation of weighted automata over trees

The following theorem extends Fliess' theorem.

▶ **Theorem 17** ([4]). *Let $f : Tree(X) \to \mathbb{R}$ be a tree series such that rank$(H^{(f)})$ is finite.*

- *Any WA recognising f has size at least rank$(H^{(f)})$.*
- *There exists a WA recognising f of size exactly rank$(H^{(f)})$.*

We detail the construction for the second item as we will need it to prove Theorem 13.

Recall that the rows of $H^{(f)}$ are indexed by trees in $\mathrm{Tree}(X)$. For $t \in \mathrm{Tree}(X)$, let $H^{(f)}_t$ be the row corresponding to t in $H^{(f)}$, which we see as $H^{(f)}_t \in \mathbb{R}^{\mathrm{Context}(X)}$. Let $Q \subseteq \mathrm{Tree}(X)$ such that $\left\{H^{(f)}_t \mid t \in Q\right\}$ is a basis of $\mathrm{Span}\left\{H^{(f)}_t \mid t \in \mathrm{Tree}(X)\right\}$. We furthermore assume that Q contains at least one tree reduced to a single variable $x \in X$, which is possible since $H^{(f)}_x \neq 0$ for some $x \in X$ unless f is the constant zero series.

We now construct the WA recognising f. The set of states is Q. The initial function is defined by $\iota(x, t) = 1$ if t is the variable $x \in X$, and 0 otherwise. We now define the transition function. For $t_1, t_2 \in Q$, there is a unique decomposition of $H^{(f)}_{t_1 \cdot t_2}$ on the basis $\left\{H^{(f)}_t \mid t \in Q\right\}$:

$$H^{(f)}_{t_1 \cdot t_2} = \sum_{t \in Q} \lambda(t_1, t_2, t) H^{(f)}_t,$$

we define $\Delta(t_1, t_2, t) = \lambda(t_1, t_2, t)$. The output function is defined by $F(t) = f(t)$ for $t \in Q$.

Proof of Theorem 13

▶ **Lemma 18.** *Let P be a homogeneous polynomial of degree d and T a shape with d leaves. The automaton constructed above for recognising P is T-layered.*

Proof. Let $\mathcal{A} = (Q, \iota, \Delta, F)$ be the automaton described in the previous subsection.

We define $m : Q \to [T]$ by $m(t) = \widetilde{t}$. To see that indeed $\widetilde{t} \in [T]$, we remark that if $\widetilde{t} \notin [T]$ then $H_t^{(P,T)} = 0$, hence t cannot be in Q. The conditions (1) and (3) are clearly satisfied, so we focus on (2).

For $T' \in [T]$, let $V_{T'}$ denote the vector space spanned by $\left\{ H_t^{(P,T)} \mid \widetilde{t} = T' \right\}$.

We claim that the subspaces $V_{T'}$ for $T' \in [T]$ are in direct sum. It follows from the fact that if $L \in V_{T'}$, then for a context C' such that $C' \circ T' \neq T$, we have $L(C') = 0$.

Let $t_1, t_2 \in Q$. By definition

$$H_{t_1 \cdot t_2}^{(P,T)} = \sum_{t \in Q} \Delta(t_1, t_2, t) H_t^{(P,T)} = \sum_{T' \in [T]} \underbrace{\sum_{t \in Q \mid \widetilde{t} = T'} \Delta(t_1, t_2, t) H_t^{(P,T)}}_{\in V_{T'}}.$$

Since $H_{t_1 \cdot t_2}^{(P,T)} \in V_{\widetilde{t_1} \cdot \widetilde{t_2}}$ and the vector spaces $V_{T'}$ for $T' \in [T]$ are in direct sum, it follows that for $T' \in [T]$ such that $\widetilde{t_1} \cdot \widetilde{t_2} \neq T'$ we have

$$\sum_{t \in Q \mid \widetilde{t} = T'} \Delta(t_1, t_2, t) H_t^{(P,T)} = 0.$$

Since the vectors $\left\{ H_t^{(P,T)} \mid t \in Q \right\}$ are linearly independent, this implies that $\Delta(t_1, t_2, t) = 0$. Thus the property (2) is satisfied. ◀

We now prove our main result, Theorem 13. Let P be a homogeneous polynomial of degree d and T a shape with d leaves. Thanks to the first item of Theorem 17 any WA recognising P has size at least $\mathrm{rank}(H^{(P,T)})$, and thanks to the first item of Lemma 16 this implies that any UPT circuit with shape T computing P has size at least $\mathrm{rank}(H^{(P,T)})$. Thanks to the second item of Theorem 17, there exists a WA recognising P of size $\mathrm{rank}(H^{(P,T)})$, and thanks to Lemma 18, it is T-layered. Thanks to the second item of Lemma 5, it induces a UPT circuit with shape T computing P of size $\mathrm{rank}(H^{(P,T)})$.

4 Applications

In this section, we apply our main theorem to concrete polynomials. The first example illustrates the difference between general UPT circuits and UPT circuits in canonical form as studied in [8], witnessing an exponential gap between the two models. Our second example is the permanent.

An exponential gap between UPT circuits and their canonical restrictions

Consider the polynomial $P(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 + x_3x_4x_1x_2$ and let T be a complete binary tree with 4 leaves. Figure 6 shows the smallest UPT circuit with shape T computing P given by the construction of Theorem 13. It witnesses an interesting phenomenon: both computations x_1x_2 and x_3x_4 are shared and used twice each. This is captured in the Hankel matrix by observing that two blocks contribute only by one to the rank since the two rows

are identical. Informally they correspond to isomorphic subshapes. This circuit is not in the canonical form studied in [8], which does not allow such shared computations.

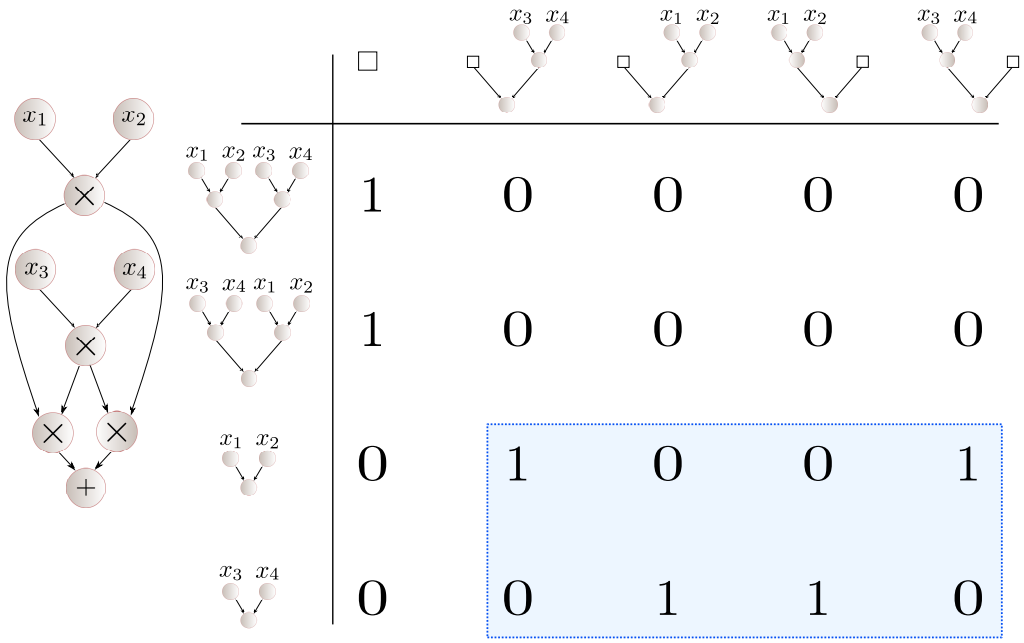

Figure 6 On the left hand side, the smallest UPT circuit computing $x_1x_2x_3x_4 + x_3x_4x_1x_2$. We have not depicted some addition gates to keep the figure simple. On the right hand side a sample of the corresponding Hankel matrix. The blue rectangle corresponds to an interaction between two different type of contexts.

We push this further to obtain an exponential gap between UPT circuits and UPT circuits in canonical form. Let $n \in \mathbb{N}$ and T the complete binary tree with 2^n leaves. Consider the polynomial $P(x) = x^{2^n}$. Inspecting the Hankel matrix (see Figure 7) yields $\text{rank}(H^{(P,T)}) = n$. Thus thanks to Theorem 13 the smallest UPT circuit with shape T computing P has exactly n addition gates, illustrated in Figure 8. The characterisation obtained in [8] shows that the smallest UPT circuit with shape T in canonical form has $2^{n+1} - 1$ addition gates, yielding an exponential gap. Note however that such a large gap can only be obtained for circuits with large degrees.

The permanent

We look at the permanent polynomial

$$P = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

over the n^2 variables $X = \{x_{i,j} \mid u, j \in [n]\}$. We examine the Hankel matrix $H^{(P,T)}$ for any shape T with n leaves and obtain the size of the smallest UPT circuit of shape T which computes the permanent.

For v a node of T, let d_v be the number of leaves in T_v.

▶ **Lemma 19.**

$$\mathit{rank}(H^{(P,T)}) = \sum_{v \in T} \binom{n}{d_v}.$$

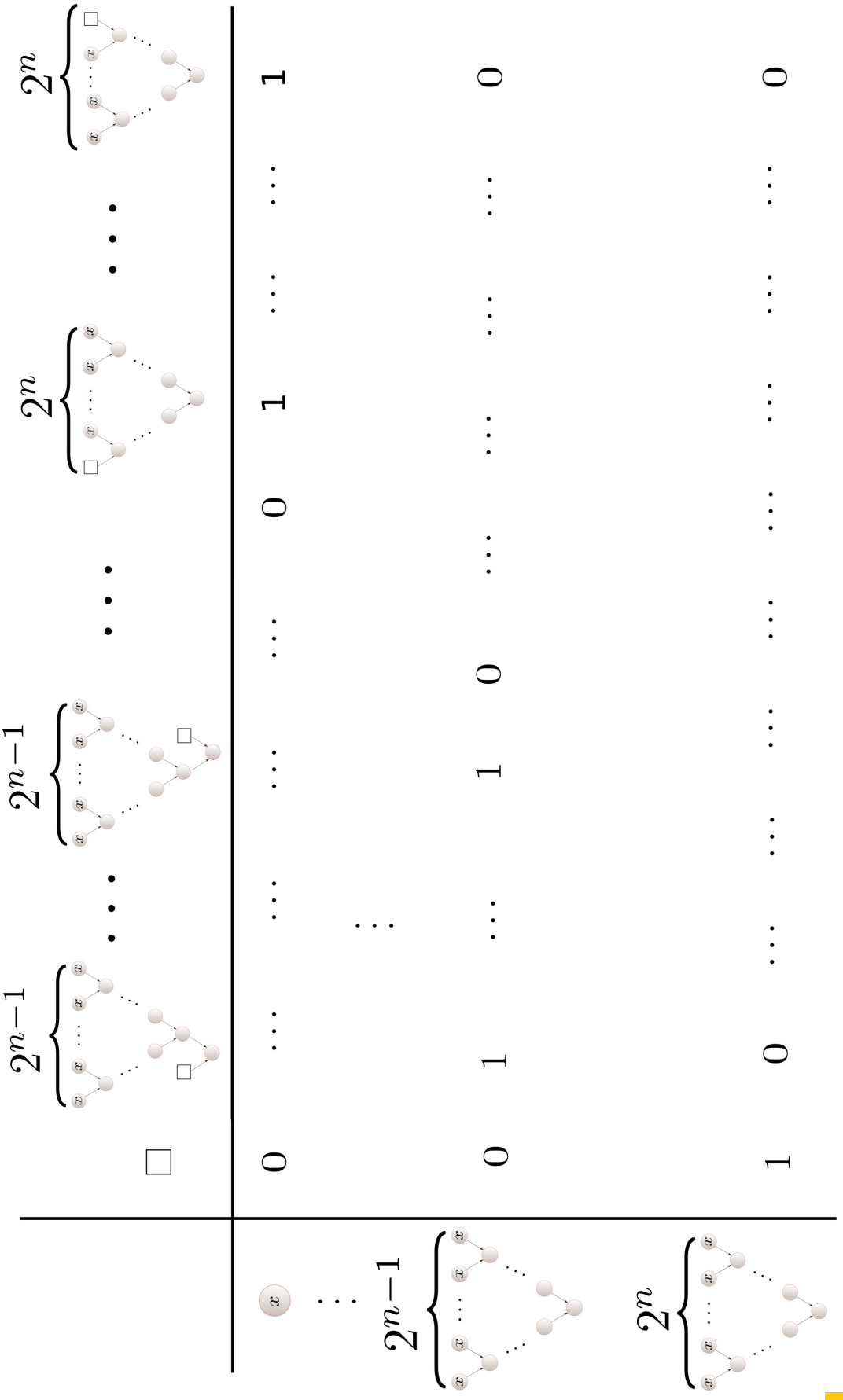

Figure 7 The Hankel matrix for the polynomial x^{2^n} and the shape T being the full binary tree.

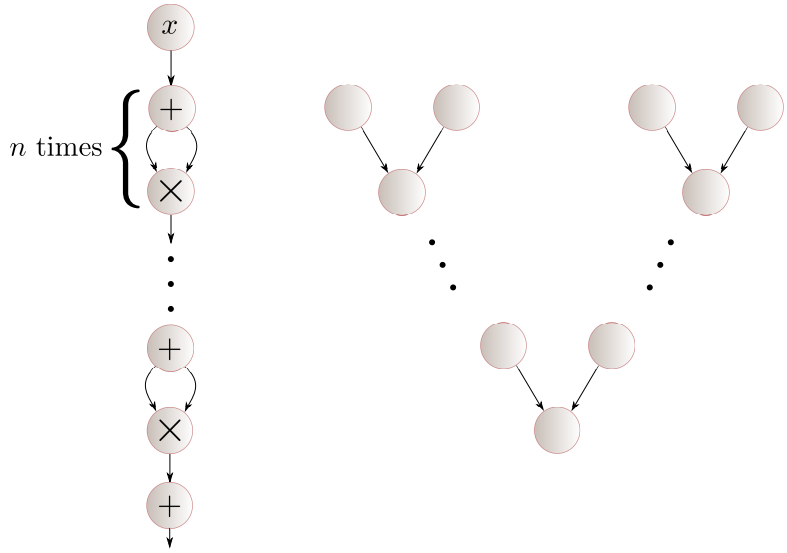

Figure 8 An example of a UPT circuit computing x^{2^n}. The circuit is on the left hand side and its shape on the right hand side, it is the complete binary tree of height n.

Proof. Let T be a shape with n leaves $\{\ell_1, \dots, \ell_n\}$ and $v \in T$ be a node in T. We let i_v denote the leftmost index of a leaf in T_v, *i.e.* T_v has leaves $\ell_{i_v}, \ell_{i_v+1}, \dots, \ell_{i_v+d_v-1}$. Moreover, let S be a subset of $\{1, \dots, n\}$ of size d_v.

We argue that in the Hankel matrix there are $\sum_{v \in T} \binom{n}{d_v}$ independent blocks, and that the set of these blocks is in bijection with pairs (v, S) where v is a node of T and S a subset of size d_v.

Let $S = \{s_1, \dots, s_{d_v}\}$ and its complement $\{1, \dots, n\} \setminus S = \{q_1, \dots, q_{n-d_v}\}$. Let U_v^S be the set of labelings of the leaves $\ell_{i_v}, \dots, \ell_{i_v+d_v-1}$ of T_v by variables $x_{i_v,\sigma(s_1)}, \dots, x_{i_v+d_v-1,\sigma(s_{d_v})}$ in this order, ranging over permutations σ of S.

Likewise, we let C denote the unlabeled context obtained by removing T_v from node v in T and replacing it by a placeholder $\Box$, and put B_v^S to be the set of labelings of all leaves but those in T_v of C by variables

$$x_{1,\tau(q_1)}, \dots, x_{i_v-1,\tau(q_{i_v-1})}, x_{i_v+d_v,\tau(q_{i_v})}, \dots, x_{n,\tau(q_{n-d_v})},$$

ranging over permutations τ of the complement of S.

For any σ, τ, the corresponding labeled tree $t_\sigma \in U_v^S$ and labeled context $c_\tau \in B_v^S$ are such that $f_P(c_\tau[t_\sigma]) = H^{(P,T)}(t_\sigma, c_\tau) = 1$, inducing a block of 1's indexed by $U_v^S \times B_v^S$ in $H^{(P,T)}$.

Conversely, we see that any $(t, c) \in \mathrm{Tree}(X) \times \mathrm{Context}(X)$ such that $H^{(P,T)}(t, c) = 1$ is in some $U_v^S \times B_v^S$, and that for any two distinct $(S, v), (S', v')$, both U_v^S and $U_{v'}^{S'}$ and B_v^S and $B_{v'}^{S'}$ are disjoint, hence the blocks cover all 1's in $H^{(P,T)}$ and are independent. This concludes. ◀

We instantiate this result for two shapes:

- If T is a comb, this yields that the smallest ABP computing the permanent has size $\sum_{i=1}^{n} \binom{n}{i} + \sum_{i=1}^{n} \binom{n}{0} = 2^n + n$,

- If T is a full binary tree of depth $k = \log(n)$, this yields that the smallest UPT circuit with this shape computing the permanent has size $\sum_{i=0}^{k} 2^i \binom{2^k}{2^{k-i}} = \Theta(\frac{2^n}{\sqrt{n}})$.

Hence the latter UPT circuit is more efficient.

However, recall that in our model circuits have unbounded fan-in on addition gates. In this setting a natural and a more accurate estimation of the size if the circuit (or number of operations that are performed) is to count directly the total number of arguments of addition gates. Examining more closely the automata and the circuits we construct, we obtain the following formula that gives the number of such edges for a UPT circuit with shape T computing the permanent

$$\sum_{v \in T} \binom{d_v}{f_v} \binom{n}{d_v},$$

where f_v is the number of leaves in $T_{v'}$, with v' an argument of v (indeed, it does not depend upon which argument is chosen). Note that our optimality result does not apply to this new measure, but we can still consider the size of the circuits constructed by Theorem 13. It yields an ABP of size $n2^{n-1} + n$, which asymptotically matches the well known optimal ABP for the permanent constructed using the Ryser formula. For the case of the full binary tree, we obtain a UPT circuit of size $\Theta(2^{\frac{3}{2}n}/\log(n))$, a bit worse than the ABP.

References

1. Vikraman Arvind and S. Raja. Some lower bound results for set-multilinear arithmetic computations. *Chicago Journal of Theoretical Computer Science*, 2016, 2016.
2. Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.
3. Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series with Applications*. Cambridge University Press, 2011. Second Edition of the English translation. Original title: Les séries rationnelles et leurs langages, Masson 1984.
4. Symeon Bozapalidis and Olympia Louscou-Bozapalidou. The rank of a formal tree power series. *Theoretical Computer Science*, 27:211–215, 1983.
5. Jack W. Carlyle and Azaria Paz. Realizations by stochastic finite automata. *Journal of Computer System Science*, 5(1):26–40, 1971.
6. Michel Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 53:197–222, 1974.
7. Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower bounds and PIT for non-commutative arithmetic circuits with restricted parse trees. In *MFCS*, 2017.
8. Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. Non-commutative computations: lower bounds and polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:94, 2016.
9. Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Theory of Computing*, 12(1):1–38, 2016.
10. Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *STOC*, pages 410–418, 1991.
11. Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
12. Ramprasad Saptharishi and Anamay Tengse. Quasi-polynomial hitting sets for circuits with restricted parse trees. *CoRR*, abs/1709.03068, 2017.
13. Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il