The Complexity of Tensor Circuit Evaluation

Article *in* computational complexity · January 2007

DOI: 10.1007/s00037-007-0222-0 · Source: DBLP

2 authors, including:

Markus Holzer
Justus-Liebig-Universität Gießen
256 PUBLICATIONS **2,041** CITATIONS

SEE PROFILE

The complexity of tensor circuit evaluation [1]

EXTENDED ABSTRACT

Martin Beaudry[2] Markus Holzer[3]

Abstract

The study of tensor calculus over semirings in terms of complexity theory was initiated by Damm *et al.* in [9]. Here we first look at tensor circuits, a natural generalization of tensor formulas; we show that the problem of asking whether the output of such circuits is non-zero is complete for the class NE $=$ NTIME$(2^{O(n)})$ for circuits over the boolean semiring, $\oplus$E for the field $\mathbb{F}_2$, and analogous results for other semirings. Common-sense restrictions such as imposing a logarithmic upper bound on circuit depth are also discussed. Second, we analyze other natural problems concerning tensor formulas and circuits over various semirings, such as asking whether the output matrix is diagonal, or a null matrix.

1 Introduction

In a recent paper [9], Damm *et al.* initiated the study of the computational complexity of *multilinear algebra* expressed by tensors. More precisely, they studied the evaluation problem for formulas over algebras of matrices defined over a finitely generated semiring $\mathcal{S}$, where the operations are matrix addition, matrix multiplication, and the tensor product—also known as the Kronecker, or outer, or direct product. Central to their work is the analysis of the *non-zero tensor problem*, denoted by $0 \neq \text{val}_{\mathcal{S}}$, which consists in asking whether a given formula yields a 1×1 matrix whose unique entry differs from the zero of $\mathcal{S}$. The complexity of this problem is indicative of where tensor calculus over the specified semiring sits in the hierarchy of complexity classes. Among their results we list in Table 1 those which are most significant to our work.

This is one more way of characterizing complexity classes in algebraic terms, which comes after the problem of evaluating formulas and circuits over the Boolean semiring (see [11, 7, 8]), and the computational models of programs over monoids (see [3, 4]) and leaf languages (see [6]), among others. Using tensor calculus in this context is especially appealing, if only because of the many applications matrix algebra finds in various areas, such as the specification of parallel algorithms. Also, since the design of algorithms for quantum computers makes extensive use of unitary matrices (see [10]), tensor formulas and circuits may become useful tools for better understanding the power of this computational model.

In this article we extend the research of Damm *et al.* from formulas to circuits. We show that over unrestricted tensor circuits, the completeness statements for their two basic problems transfer smoothly to the exponential-time counterparts to the complexity classes encountered in the formula case; to achieve this, however, we have to develop a more powerful proof technique which, as a byproduct, enables us to improve some of their completeness results, in that we can write them using polytime

[1] Work supported by Québec FCAR, NSERC of Canada, and Deutsche Forschungsgemeinschaft.

[2] Département de mathématiques et d'informatique, Université de Sherbrooke, Sherbrooke, Québec J1K 2R1 Canada. email: beaudry@dmi.usherb.ca

[3] Institut für Informatik, Technische Universität München, Arcisstraße 21, D-80290 München, Germany. email: holzer@informatik.tu-muenchen.de. Part of the work was done while the author was at Département d'I.R.O., Université de Montréal.

many-one instead of the polytime Turing reductions they had to use. More precisely, with notation E for the class DTIME($2^{O(n)}$), we obtain that:

1. evaluating a tensor circuit over the natural numbers is #E-complete under polylin many-one reductions, i.e., reductions computable in deterministic polytime producing linear output, and
2. problems $0 \neq \text{val}_{\mathbb{B}}$ and $0 \neq \text{val}_{\mathbb{F}_2}$ are respectively NE-complete and $\oplus$E-complete under polylin many-one reductions.

We also consider other natural questions pertaining to tensor circuits; in this extended abstract, we discuss the problems of deciding whether a circuit's output is a diagonal matrix, or an identity matrix. Since the problems were not addressed by Damm *et al.*, we also solve them for the case of tensor formulas.

The next section introduces the definitions and notations needed in the article. Section 3 discusses the complexity of the evaluation of tensor circuits in general. Other problems on tensor circuits and formulas are discussed in Section 4.

Problem	Complete for...	...under reducibility...
$0 \neq \text{val}_{\mathbb{B}}$	NP	polytime many-one
$0 \neq \text{val}_{\mathbb{Z}_q}$	MOD$_q$-P	polytime many-one
$0 \neq \text{val}_{\mathbb{F}_2}$	$\oplus$P	polytime many-one
$\text{val}_{\mathbb{N}}$	#P	polytime Turing
$\text{val}_{\mathbb{Z}}$	*Gap*P	polytime Turing

TABLE 1: Complexity results obtained by Damm *et al.*

2 Definitions, Notations, and Basic Techniques

We assume that the reader is familiar with the standard notation from computational complexity (see e.g. [2, 17]), in particular with the first three items in the inclusion chain

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}.$$

Here, we use EXP to denote DTIME($2^{n^{O(1)}}$) (NEXP, NTIME($2^{n^{O(1)}}$), respectively), and we denote by E the class DTIME($2^{O(n)}$). We refer to the latter as (deterministic) exponential time; its nondeterministic counterpart is referred to as NE. Note that E and NE are not closed under polytime or logspace reductions due to the possibly nonlinear output length of these reductions.

We also assume that the reader is familiar with counterparts to nondeterministic decision classes like NP defined by "counting" or defined as function classes. We will use the following chains of versions of NP, PSPACE, NE, and NEXP.

- #P, #PSPACE, #E, and #EXP, classes of functions f defined with machines M with the same resources as the underlying base class, such that $f(x)$ equals the number of accepting computations of M on x (see [1, 15, 16]),
- the parity versions $\oplus$P, $\oplus$PSPACE, $\oplus$E, and $\oplus$EXP, where $\oplus$P is the class of sets of type $\{\, x \mid f(x) \neq 0 \pmod 2 \,\}$ for some $f \in \#\text{P}$, with analogous definitions for the other classes,
- the classes MOD$_q$-P, MOD$_q$-PSPACE, MOD$_q$-E, and MOD$_q$-EXP, defined similarly with respect to counting modulo q,

- the classes *Gap*P $= \{ f - g \mid f, g \in \#\mathrm{P} \}$, *Gap*PSPACE, *Gap*E, and *Gap*EXP.

We also use the class $C_{=}$P of sets of the type $\{ x \mid f(x) - g(x) = 0 \}$ for some $f, g \in \#$P, the class USP of sets of type $\{ x \mid f(x) = 1 \}$ for some $f \in \#$P (see [5]), and DP, the class of those languages which can be written as the difference of two NP languages, or equivalently as the intersection of an NP and a co-NP language (see [13]). The exponential time classes $C_{=}$E, USE and DE are similarly defined. Finally, recall that a function h is polynomial time many-one reducible to a function g if there is a polytime computable mapping f such that for all x, $h(x) = g(f(x))$.

We borrow from Damm *et al.* [9, Definition 11] an algebraic computation model which enables us to obtain complexity results for the evaluation problem of tensor circuits, with a single proof valid for all finitely generated semirings.

An *algebraic Turing machine over a semiring* $\mathcal{S}$ is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where Q, $\Sigma \subseteq \Gamma$, Γ, $q_0 \in Q$, $B \in \Gamma$, and $F \subseteq Q$ are defined as for ordinary Turing machines and δ is the *transition relation* taking the form $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, S, R\} \times \mathcal{S}$. In this machine, a move from using the transition $(p, a, q, b, m, s) \in \delta$, is assigned a *weight* s. A weight may also be associated with any particular computation, as the *product* in $\mathcal{S}$ of the weights of its successive moves. For completeness, we define the weight of a length-zero computation to be 1.

On input w machine M computes the value $f_M(w)$, a function $f_M : \Sigma^* \to \mathcal{S}$, which is defined as the *sum* of the weights of all accepting computations from the initial configuration on input w to an accepting configuration. A language $L_M = \{ w \in \Sigma^* \mid f_M(w) \neq 0 \}$ can also be defined.

The definition is adapted in a straightforward way to define algebraic algebraic space or time bounded Turing machines, etc. We define the *generalized counting class* $\mathcal{S}$-$\#$P as the set of all functions $f : \Sigma^* \to \mathcal{S}$ such that there is a polytime algebraic Turing machine M over $\mathcal{S}$ which computes f. The *generalized language class* $\mathcal{S}$-P is the set of all languages $L \subseteq \Sigma^*$ such that there is a polytime algebraic Turing machine M over $\mathcal{S}$ for which $w \in L$ if and only if $f_M(w) \neq 0$.

Let $\mathcal{S}$-$\#\mathcal{C}$ be a generalized counting class over semiring $\mathcal{S}$ which is closed under the pairing operation $\langle \cdot, \cdot \rangle$. Then the "polynomial" counting operator $\#^p$ defines the function class $\#^p \cdot \mathcal{S}$-$\#\mathcal{C}$ as the set of all functions f for which a polynomial p and a function $f_M \in \mathcal{S}$-$\#\mathcal{C}$ can be found, such that

$$f(x) = \sum_{|y| \leq p(|x|)} f_M(x, y),$$

where summation is taken in $\mathcal{S}$. Based on the counting operator $\#^p$ the "polynomial" operators $\exists^p$, $\forall^p$, and $C_{=}^p$ define the following language classes: $\exists^p \cdot \mathcal{S}$-$\#\mathcal{C}$ (resp. $\forall^p \cdot \mathcal{S}$-$\#\mathcal{C}$, $\exists!^p \cdot \mathcal{S}$-$\#\mathcal{C}$, $C_{=}^p \cdot \mathcal{S}$-$\#\mathcal{C}$) is the set of all languages L for which there are some functions f, g in $\#^p \cdot \mathcal{S}$-$\#\mathcal{C}$ such that $x \in L$ if and only if $f(x) \neq 0$, (resp. iff $f(x) = 0$, $f(x) = 1$, $f(x) = g(x)$). Obviously, $\#^p \cdot \mathcal{S}$-$\#$P $= \mathcal{S}$-$\#$P. Table 2 states without proof other significant special cases. The "exponential" operators $\#^e$, $\exists^e$, $C_{=}^e$, and $\forall^e$ are similarly defined as their polynomial counterparts, by replacing the polynomial p with the function $2^{c \cdot |x|}$ for some constant c.

A *semiring* is a tuple $(\mathcal{S}, +, \cdot)$ with $\{0, 1\} \subseteq \mathcal{S}$ and binary operations $+, \cdot : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ (sum and product), such that $(\mathcal{S}, +, 0)$ is a commutative monoid, $(\mathcal{S}, \cdot, 1)$ is a monoid, multiplication distributes over sum, and $0 \cdot a = a \cdot 0 = 0$ for every a in $\mathcal{S}$ (see, e.g., [12]). A semiring is *commutative* if and only if $a \cdot b = b \cdot a$ for every a and b, it is *finitely generated* if there is a finite set $\mathcal{G} \subseteq \mathcal{S}$ generating

class	semiring $\mathcal{S}$			
	$\mathbb{B}$	$\mathbb{Z}_q$	$\mathbb{N}$	$\mathbb{Z}$
S-#P	$\{\chi_L \mid L \in \text{NP}\}$	$\{f \pmod q \mid f \in \#\text{P}\}$	#P	*Gap*P
S-P	NP	MOD$_q$-P	NP	co-$C_=$P
$\exists^p \cdot \mathcal{S}$-#P	NP	MOD$_q$-P	NP	co-$C_=$P
$\forall^p \cdot \mathcal{S}$-#P	co-NP	co-MOD$_q$-P	co-NP	$C_=$P
$\exists!^p \cdot \mathcal{S}$-#P	NP	co-MOD$_q$-P	USP	$C_=$P
$C_=^p \cdot \mathcal{S}$-#P	co-DP	co-MOD$_q$-P	$C_=$P	$C_=$P

TABLE 2: Polytime classes defined with algebraic TMs.
Here χ_L is the characteristic function of the language L.

all of $\mathcal{S}$ by summation, and is a *ring* if and only if $(S, +, 0)$ is a group. The special choice of $\mathcal{G}$ has no influence on the complexity of problems we study in this paper. Throughout the paper we consider the following semirings: the Booleans $(\mathbb{B}, \vee, \wedge)$, residue class rings $(\mathbb{Z}_q, +, \cdot)$, the naturals $(\mathbb{N}, +, \cdot)$, and the integers $\mathbb{Z} = (\mathbb{Z}, +, \cdot)$.

Let $\mathbb{M}_\mathcal{S}$ denote the set of all *matrices* over $\mathcal{S}$, and define $\mathbb{M}_\mathcal{S}^{k,\ell} \subseteq \mathbb{M}_\mathcal{S}$ to be the set of all *matrices of order* $k \times \ell$. For a matrix A in $\mathbb{M}_\mathcal{S}^{k,\ell}$ let $I(A) = [k] \times [\ell]$, where $[k]$ denotes the set $\{1, 2, \ldots, k\}$. The (i, j)th entry of A is denoted by $a_{i,j}$ or $(A)_{i,j}$. Addition and multiplication of matrices in $\mathbb{M}_\mathcal{S}$ are defined in the usual way. Additionally we consider the *tensor product* $\otimes : \mathbb{M}_\mathcal{S} \times \mathbb{M}_\mathcal{S} \to \mathbb{M}_\mathcal{S}$ of matrices, also known as Kronecker product, outer product, or direct product, which is defined as follows: for $A \in \mathbb{M}_\mathcal{S}^{k,\ell}$ and $B \in \mathbb{M}_\mathcal{S}^{m,n}$ let $A \otimes B \in \mathbb{M}_\mathcal{S}^{km,\ell n}$ be

$$A \otimes B := \begin{pmatrix} a_{1,1} \cdot B & \ldots & a_{1,\ell} \cdot B \\ \vdots & \ddots & \vdots \\ a_{k,1} \cdot B & \ldots & a_{k,\ell} \cdot B \end{pmatrix}.$$

Hence $(A \otimes B)_{i,j} = (A)_{q,r} \cdot (B)_{s,t}$ where $i = k \cdot (q-1) + s$ and $j = \ell \cdot (r-1) + t$.

The following notation is heavily used: let I_n be the order n identity matrix, e_i^n the ith unit row vector of length n, $(0)_n$ the all-zero row vector of length n, $(1)_n$ the all-one row vector of length n, f_i^n the vector $\sum_{\substack{j=1 \\ j \neq i}}^n e_j^n$, and let $A^{\otimes n}$ stand for the n-fold iteration $A \otimes A \otimes \cdots \otimes A$.

We now define tensor circuits over semirings.

Definition 1. *A* tensor circuit *C over a semiring $\mathcal{S}$ is a finite directed acyclic graph and its* order *are defined as follows—we call the nodes of C* gates *and its edges* wires*:*

1. *Each gate with in-degree zero is labeled with some F from $\mathbb{M}_\mathcal{S}^{k \times \ell}$ with entries from $\mathcal{S}$ and its order is $k \times \ell$. These gates are called* inputs.
2. *Non-input gates, i.e., inner gates, have in-degree exactly two and are labeled by matrix operations, i.e., addition, multiplication, and tensor product. Let f be an inner gate whose left child g is of order $k \times \ell$ and its right child h is of order $m \times n$ labeled by $\circ$, for $\circ \in \{+, \cdot, \otimes\}$. Then we simply say that f is a $\circ$-gate, write $f = (g \circ h)$, and its order is defined as:*
   $k \times \ell$ if gate f is a $+$-gate, $k = m$, and $\ell = n$,
   $k \times n$ if gate f is a $\cdot$-gate and $\ell = m$, and
   $km \times \ell n$ if gate f is a $\otimes$-gate.

3. *There is a unique gate with out-degree zero, which is called* output *gate. The order of the circuit C is defined to be the order of the output gate.*

For a tensor circuit C of order $k \times \ell$ let $I(C) = [k] \times [\ell]$ be its "set of indices." Let $\mathbb{T}_{\mathcal{S}}$ denote the set of all tensor circuits over $\mathcal{S}$, and define $\mathbb{T}_{\mathcal{S}}^{k,\ell} \subseteq \mathbb{T}_{\mathcal{S}}$ to be the set of all tensor circuits of order $k \times \ell$. A tensor circuit C is called a tensor formula *if its finite directed acyclic graph is a binary tree. We call a circuit of order $k \times k$ a* square tensor circuit.

In this paper we only consider finitely generated $\mathcal{S}$, and we assume that entries of the inputs to a tensor circuit are from $\mathcal{G} \cup \{0\}$, where $\mathcal{G}$ is a generating set of $\mathcal{S}$. Hence, label of input gates, i.e., matrices, can be string-encoded using list notation such as "$[[001][101]]$." Tensor circuits can be encoded over the alphabet $\Sigma = \{0\} \cup \mathcal{G} \cup \{[,], (,), \cdot, +, \otimes\}$. Strings over Σ which do not encode valid circuits are deemed to represent the trivial tensor circuit 0 of order 1×1.
Let C be a tensor circuit of order $m \times n$, Its *diameter* is $\max\{m, n\}$, its *size* is the number of gates and wires, and its *depth* is the maximum number of gates along a path connecting some input gate with the output gate (leaf-root path). Besides these usual notions we introduce the *tensor depth* of a circuit as the maximum number of tensor gates found along a leaf-root path. Note that in those restrictions concerning depth $d(n)$ and/or tensor depth $t(n)$, an instance, i.e., the coding of the circuit, must also contain parameters c_1 and/or c_2, respectively, and must satisfy the constraints that the actual depth is less or equal than $c_1 \cdot d(n)$ and its tensor depth is less or equal than $c_2 \cdot t(n)$, respectively. This is needed in order to be able to check the imposed restrictions with a Turing machine.

Proposition 1. *Verifying that a given tensor circuit meets the requirements to be of depth $d(n)$ (and tensor depth $t(n)$) can be done on a deterministic Turing machine in time $O(n \cdot \log d(n))$. If only tensor depth $t(n)$ has to be checked, the running time is $O(n \cdot \log t(n))$.*
Testing whether a string encodes a valid tensor circuit and if so, computing its order, is feasible in deterministic time $(n \cdot 2^{t(n)} \log n)^{O(1)}$, where $t(n)$ denotes the tensor depth of the given tensor circuit.
(Proof omitted.)

Concerning tensor circuits we observe that they are much more "powerful" than the tensor formulas [9], since they allow to blow up matrix diameter at a rate which is double-exponential.

Proposition 2. *If C is a tensor circuit of tensor depth t which has input matrices of diameter at most p, then $|C| \leq p^{2^t}$, and there exists a circuit which outputs a matrix of exactly this diameter.*
(Proof omitted.)

Definition 2. *For each semiring $\mathcal{S}$ and each k and ℓ we define $\mathrm{val}_{\mathcal{S}}^{k,\ell} : \mathbb{T}_{\mathcal{S}}^{k,\ell} \to \mathbb{M}_{\mathcal{S}}^{k,\ell}$, that is, we associate with gate f of order $k \times \ell$ of a tensor circuit C its $k \times \ell$ matrix "value," defined as follows:*

1. $\mathrm{val}_{\mathcal{S}}^{k,\ell}(f) = F$ *if f is an input gate labeled with F,*
2. $\mathrm{val}_{\mathcal{S}}^{k,\ell}(f) = \mathrm{val}_{\mathcal{S}}^{k,\ell}(g) + \mathrm{val}_{\mathcal{S}}^{k,\ell}(h)$ *if $f = (g + h)$,*
3. $\mathrm{val}_{\mathcal{S}}^{k,\ell}(f) = \mathrm{val}_{\mathcal{S}}^{k,m}(g) \cdot \mathrm{val}_{\mathcal{S}}^{m,\ell}(h)$ *if $f = (g \cdot h)$ and g is of order $k \times m$, and*
4. $\mathrm{val}_{\mathcal{S}}^{k,\ell}(f) = \mathrm{val}_{\mathcal{S}}^{k/m,\ell/n}(g) \otimes \mathrm{val}_{\mathcal{S}}^{m,n}(h)$ *if $f = (g \otimes h)$ and h is of order $m \times n$.*

The value $\mathrm{val}_{\mathcal{S}}^{k,\ell}(C)$ of a tensor circuit C of order $k \times \ell$ is defined to the value of the unique output gate. Tensor circuits of order 1×1 are called scalar tensor circuits*, and we simply write $\mathrm{val}_{\mathcal{S}}$ for the $\mathrm{val}_{\mathcal{S}}^{1,1}$ function.*

We require the notion of a *certificate* for a tensor circuit index, which was introduced by Damm *et al.* [9]. Intuitively, a certificate is to a tensor circuit entry what a proof tree is to a Boolean circuit. Certificates and their weights are defined in such a way that the entry at index $(i, j) \in [k] \times [\ell]$ in $\mathrm{val}_{\mathcal{S}}^{k,\ell}(C)$ is equal to the sum of the weights of all the certificates of C at (i, j). We refer the reader to [9] for the appropriate definitions and a detailed discussion.

Lemma 1. *Let C be a scalar tensor circuit over a semiring $\mathcal{S}$. Then $\mathrm{val}_{\mathcal{S}}(C)$ is the sum, over all certificates for the output value, of the weights of these certificates.* (Proof omitted.)

3 Computational Complexity of the Non-Zero Problem

This section contains completeness results on the computational complexity of the non-zero tensor circuit problem and of its close relative, the tensor circuit evaluation problem. Our theorems follow immediately from Lemmas 2 and 3.

Definition 3. *Let $\mathcal{S}$ be a semiring. The* non-zero tensor circuit problem *over semiring $\mathcal{S}$ is the set $0 \neq \mathrm{val}_{\mathcal{S}}$ of all scalar tensor circuits C for which $\mathrm{val}_{\mathcal{S}}(C) \neq 0$.*

Lemma 2. *For any finitely generated semiring $\mathcal{S}$, the evaluation problem $\mathrm{val}_{\mathcal{S}}$ for scalar tensor circuits of depth $d(n)$ and tensor depth $t(n)$ belongs to the class $\mathcal{S}$-#TISP$(2^{d(n)}(n \cdot 2^{t(n)} \log n)^{O(1)}, (n \cdot 2^{t(n)} \log n)^{O(1)})$.* (Proof in the Appendix.)

Lemma 3. *Let M be an algebraic Turing machine over a finitely generated semiring $\mathcal{S}$ running in time $t(n)$ and space $s(n)$, with $s(n) \geq n$, and set $d(n) = \max\{\log t(n), \log s(n)\}$ and $g(n) = \max\{n, d(n)\}$. There is a $(g(n))^{O(1)}$ time and $O(g(n))$ space output computable function f, which on input w, computes a scalar tensor circuit $f(M, w)$ of size $O(g(n))$, depth $O(d(n))$, and tensor depth $O(\log s(n))$ such that $\mathrm{val}_{\mathcal{S}}(f(M, w)) = f_M(w)$.*

Proof sketch for Lemma 3 (full proof in the Appendix.) It is straightforward to verify that the iterated matrix multiplication at the core of Warshall's algorithm is meaningful over arbitrary semirings; then we are justified to implicitly use this algorithm in our reduction. Since $M = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$ is fixed we can impose three assumptions without loss of generality.

1. The tape alphabet is $\Gamma = \{0, 1, B\}$; hence input $w \in \Sigma^*$ is binary.
2. Machine M is single-tape and the tape is cyclic with $s(n)$ tape cells, numbered 0 to $s(n) - 1$, so that tape-head motions are from cell i to cell $(i \pm 1) \pmod{s(n)}$.

Our second assumption helps us overcome the major hurdle one meets upon building a generic proof: finding an efficient way of encoding the tape content and of describing the motions of the read-write head. Here, we keep the head static and move the tape instead; using a cyclic tape enables us to encode this as a rotation matrix, which can be constructed recursively.

In the case of a machine for a complexity class defined solely in terms of a time bound, the tape has just enough cells to make sure that the machine cannot see the difference between a cyclic tape and one that is infinite in both directions. Moreover, let all computations of M have length exactly $t(n)$ and the accepting ones be determined solely by the state, regardless of the tape content.

3. We split a single transition of M into two phases: first, the machine reads and overwrites the content of the tape cell its read-write head points to, and changes accordingly the state of its finite control. Second, the new state determines whether there will be a head movement, and if so, this is implemented without changing state or tape content by taking a transition assigned the weight 1.

Because the tape is cyclic, it is not the absolute position of the head on the tape that is relevant, but the position relative to or within the tape region which contains nontrivial cell contents. Hence, the set of all configurations equals $Q \times \Gamma^{s(n)}$, where $\Gamma^{s(n)}$ is a shorthand notation for the set of all strings of length exactly $s(n)$.
Let $Q = \{q_1, \ldots, q_m\}$ and $\Gamma = \{a_1, a_2, a_3\}$. Define mappings $c_{_Q} : Q \to \mathbb{M}_{\mathcal{S}}^{1,m}$ and $c_{_\Gamma} : \Gamma \to \mathbb{M}_{\mathcal{S}}^{1,3}$ as: $c_{_Q}(q_i) = e_i^m$ and $c_{_\Gamma}(a_i) = e_i^3$. With a configuration $C = (q, a_0 \ldots a_{s(n)-1})$ in $Q \times \Gamma^{s(n)}$ we associate the unit vector

$$c(C) = c_{_Q}(q) \otimes \bigotimes_{i=0}^{s(n)-1} c_{_\Gamma}(a_i).$$

The value of the function f_M on input w equals the sum of all labels of accepting paths of length $t(n)$ in the configuration graph G_M with node set $Q \times \Gamma^{s(n)}$. We can associate a matrix A_{G_M} of order $m \cdot 3^{s(n)} \times m \cdot 3^{s(n)}$ with G_M, such that the (i,j)th entry of $A_{G_M}^{t(n)}$ equals the sum of all labels of paths of length $t(n)$ linking configuration C_i with C_j. Let $A = (A_{G_M})^{\mathsf{T}}$. For an input $w = a_0 \ldots a_n$, one has $f_M(w) = V_{\text{accept}} \cdot A^{t(n)} \cdot (V_{\text{init}})^{\mathsf{T}}$, where $V_{\text{accept}} = \sum_{C \in F \times \Gamma^{s(n)}} c(C)$ is a row column vectors representing all accepting configurations of M on input w, while $V_{\text{init}} = c(C_0)$ and $C_0 = (q_1, a_0 \ldots a_{n-1} B^{s(n)-n})$ is the initial configuration of M. The rest of the proof shows how to define an appropriate tensor circuit which meets our requirements on size, depth, and tensor depth.

Theorem 1. *For any finitely generated semiring $\mathcal{S}$,*

1. *problem* $\text{val}_{\mathcal{S}}$ *is* $\mathcal{S}$-#E*-complete under polylin many-one reduction;*
2. *problem* $0 \neq \text{val}_{\mathcal{S}}$ *is* $\exists^e \cdot \mathcal{S}$-#P*-complete under polylin many-one reduction.*

In Tables 3 and 4 we list consequences of these theorems on concrete instances of the semiring $\mathcal{S}$ and commonsense restrictions on the parameters $t(n)$, $d(n)$ and $s(n)$. Our generic reduction uses only linear workspace; this observation yields completeness statements with respect to polylin many-one reduction, for the class NE and its counting counterparts, in the case case of unrestricted depth and tensor depth (results not listed in the table).
We also observe that the proof of Lemma 3 can be rewritten it in terms of tensor formulas instead of circuits. This enables us to significantly improve the results of Damm *et al.* [9] on the complexity of the evaluation and non-zero problem on scalar tensor formulas by (*i*) extending to the case of arbitrary finitely generated semirings, and (*ii*) showing completeness under polytime many-one reduction. The third item, in particular, is not provable with the techniques used by Damm *et al.*

Theorem 2. *Let $\mathcal{S}$ denote any finitely generated semiring. For tensor formulas,*

1. *problem* $\text{val}_{\mathcal{S}}$ *is* $\mathcal{S}$-#P*-complete with respect to polytime many-one reduction;*
2. *problem* $0 \neq \text{val}_{\mathcal{S}}$ *is* $\exists^p \cdot \mathcal{S}$-#P*-complete with respect to polytime many-one reduction;*
3. *problem* $\text{val}_{\mathbb{N}}$ *is* #P*-complete with respect to polytime many-one reduction.*

depth $d(n)$	tensor depth $t(n)$		Semiring $\mathcal{S}$
	$O(\log n)$	unrestricted	
$O(\log n)$	$\#$P		$\mathbb{N}$
	*Gap*P		$\mathbb{Z}$
$O(\log^k n)$	$\#\text{TISP}(2^{O(\log^k n)}, n^{O(1)})$	$\#\text{TIME}(2^{O(\log^k n)})$	$\mathbb{N}$
	$\mathit{Gap}\text{TISP}(2^{O(\log^k n)}, n^{O(1)})$	$\mathit{Gap}\text{TIME}(2^{O(\log^k n)})$	$\mathbb{Z}$
unrestricted	$\#$PSPACE	$\#$EXP	$\mathbb{N}$
	*Gap*PSPACE	*Gap*EXP	$\mathbb{Z}$

TABLE 3: Complexity of the scalar tensor circuit evaluation problem $\text{val}_{\mathcal{S}}$
Completeness results are meant with respect to polytime many-one reductions.

depth $d(n)$	tensor depth $t(n)$		Semiring $\mathcal{S}$
	$O(\log n)$	unrestricted	
$O(\log n)$	NP		$\mathbb{B}$
	MOD_q-P		$\mathbb{Z}_q$
	$\oplus$P		$\mathbb{F}_2$
$O(\log^k n)$	$\text{NTISP}(2^{O(\log^k n)}, n^{O(1)})$	$\text{NTIME}(2^{O(\log^k n)})$	$\mathbb{B}$
	$\text{MOD}_q\text{-TISP}(2^{O(\log^k n)}, n^{O(1)})$	$\text{MOD}_q\text{-TIME}(2^{O(\log^k n)})$	$\mathbb{Z}_q$
	$\oplus\text{TISP}(2^{O(\log^k n)}, n^{O(1)})$	$\oplus\text{TIME}(2^{O(\log^k n)})$	$\mathbb{F}_2$
unrestricted	PSPACE	NEXP	$\mathbb{B}$
	MOD_q-PSPACE	MOD_q-EXP	$\mathbb{Z}_q$
	$\oplus$PSPACE	$\oplus$EXP	$\mathbb{F}_2$

TABLE 4: Complexity of the non-zero scalar tensor circuit problem $0 \neq \text{val}_{\mathcal{S}}$
Completeness results are meant with respect to polytime many-one reductions.

4 Computational Complexity of Some Related Problems

A large number of decision problems on tensor circuits can be defined, based on natural linear-algebraic questions: asking whether such a circuit outputs a null matrix, or a diagonal square matrix, etc. These questions were not addressed by Damm *et al.* [9], so they can be looked at also in the case of tensor formulas; these restrictions are denoted with an initial "F-", e.g. F-ZERO$_{\mathcal{S}}$ for ZERO$_{\mathcal{S}}$, etc. Our analysis of these problems uses two ancillary problems defined below, and the generalization of a construction of Damm *et al.* The proof of Lemma 4 is in the Appendix; it is based on the generic reduction of Lemma 3, since the technique of Damm et al. does not seem to work for arbitrary circuits. Recall that a function f is called *exponential* if $f = O(2^n)$.

Lemma 4. *Let M be an algebraic polytime Turing machine with binary input alphabet $\Sigma = \{0, 1\}$ over a finitely generated semiring $\mathcal{S}$ and $p(n)$ a polynomial or an exponential function. There is a polylin computable function f, which on input 1^n computes a tensor circuit $C_{M,n} = f(M, 1^n)$ of linear size and of depth $d(n) = O(\log n)$ if $p(n)$ is a polynomial and $d(n) = O(n)$ if $p(n)$ is an exponential function, such that for all $w = a_0 \dots a_{p(n)-1}$ with $a_i \in \Sigma$ and $0 \leq i \leq p(n) - 1$,*

$$f_M(w) = \left(\bigotimes_{i=0}^{p(n)-1} e^2_{a_i+1} \right) \cdot C_{M,n} \cdot \left(\bigotimes_{i=0}^{p(n)-1} e^2_{a_i+1} \right)^{\mathsf{T}} . \tag{1}$$

In other words the matrix $\mathrm{val}_{\mathcal{S}}^{2^{p(n)},2^{p(n)}}(C_{M,n})$ *contains the values of the function* f_M *on all possible inputs of length* $p(n)$ *on the main diagonal, and zero elsewhere.*

Definition 4. *Let* $\mathcal{S}$ *be a finitely generated semiring. Problem* $\mathrm{ZERO}_{\mathcal{S}}$ *is the set of all triples* (C, i, j)*, where* C *is a tensor circuit of order* $I(C) = k \times \ell$*, integers* i *and* j *are expressed in binary such that* $(i, j) \in [k] \times [\ell]$*, such that* $(\mathrm{val}_{\mathcal{S}}^{k,\ell}(C))_{i,j} = 0$*.*
Problem $\mathrm{ONE}_{\mathcal{S}}$ *is defined similarly with the condition* $(\mathrm{val}_{\mathcal{S}}^{k,\ell}(C))_{i,j} = 1$*.*

Lemma 5. *Let* $\mathcal{S}$ *be a finitely generated semiring.*
Problem F-$\mathrm{ZERO}_{\mathcal{S}}$ *is* $\forall^p \cdot \mathcal{S}$*-*#P*-complete (*F-$\mathrm{ONE}_{\mathcal{S}}$*,* $\exists!^p \cdot \mathcal{S}$*-*#P*-complete, respectively) under polytime many-one reduction.*
Problem $\mathrm{ZERO}_{\mathcal{S}}$ *is* $\forall^e \cdot \mathcal{S}$*-*#P*-complete (*$\mathrm{ONE}_{\mathcal{S}}$*,* $\exists!^e \cdot \mathcal{S}$*-*#P*-complete, respectively) under polylin many-one reduction.*

Once Lemma 5 is available, proving this Lemma is a standard exercise (see the proof for F-$\mathrm{ZERO}_{\mathcal{S}}$ in the Appendix). As a sample application we discuss the complexity of the following decision problems for tensor circuits; proving the results is readily done using Lemma 5. Note that because tensor circuits can output matrices of diameter double exponential in terms of the circuit size, validity of inputs for these problems demands deterministic exponential time.

Definition 5. *Let* $\mathcal{S}$ *be a finitely generated semiring.*
• *The* null tensor circuit problem *over* $\mathcal{S}$ *is the set* NULL_S *of all tensor circuits* C *over* $\mathcal{S}$ *of order* $k \times \ell$*, such that* $\mathrm{val}_{\mathcal{S}}^{k,\ell}(C)$ *equals the* $k \times \ell$ *all-zero matrix.*
• *The* diagonal tensor circuit problem *over* $\mathcal{S}$ *is the set* DIAG_S *of all square tensor circuits* C *over* $\mathcal{S}$ *of order* $k \times \ell$*, whose output satisfies* $(\mathrm{val}_{\mathcal{S}}^{k,\ell}(C))_{i,j} = 0$ *for all* $i \neq j$ *with* $(i, j) \in [k] \times [\ell]$*.*

Theorem 3. *If* $\mathcal{S}$ *is a finitely generated semiring, then problems* $\mathrm{NULL}_{\mathcal{S}}$ *and* $\mathrm{DIAG}_{\mathcal{S}}$ *are* $\forall^e \cdot (\forall^p \cdot S$*-*#P*)-complete under polylin many-one reduction.*

For specific examples of semirings, the statement translates into
for the booleans $\mathbb{B}$ and the naturals $\mathbb{N}$: co-NE-completeness;
for the modular integers $\mathbb{Z}_q$: $\forall^e \cdot$ co-MOD_q-P-completeness;
for the field $\mathbb{F}_2$: $\forall^e \cdot \oplus$P-completeness;
for the integers $\mathbb{Z}$: $C_=$E-completeness.

Meanwhile, for the restrictions of these problems to tensor formulas, completeness results with respect to polytime many-one reductions hold for the polynomial-time counterparts of these complexity classes. In the full paper, we apply these proof techniques to a range of other problems, such as: testing whether the output is an identity, a symmetric or an orthogonal matrix, and verifying whether two tensor circuits received as input are equivalent, i.e. whether they output the same matrix.

The authors thank Carsten Damm for helpful discussions and pointers to useful references. Also thanks to Pierre McKenzie and Jose Manuel Fernandez for fruitful discussions.

References

1. C. Àlvarez and B. Jenner. A very hard log space counting class. *Theoretical Computer Science*, 107:3–30, 1993.

2. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 2nd edition, 1995.

3. D. A. Mix Barrington. Bounded-width polynomial size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

4. D. A. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the Association of Computing Machinery*, 35:941–952, 1988.

5. A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 82:80–88, 1982.

6. G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace MOD-classes. *Mathematical Systems Theory*, 25:223–237, 1992.

7. S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proceedings 19th Symposium on Theory of Computing*, pages 123–131. ACM Press, 1987.

8. S.R. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 21(4):755–780, 1992.

9. C. Damm, M. Holzer, and P. McKenzie. The complexity of tensor calculus. In *Proceedings of the* 15*th Conference on Computational Complexity*, pages 70–86, Florence, Italy, 2000. IEEE Computer Society Press.

10. L. Fortnow. One complexity theorist's view of quantum computing. Technical Report quant-ph/0003035, Los Alamos arXiv, 2000.

11. N. D. Jones and W. T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3:105–117, 1976.

12. W. Kuich and A. Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1986.

13. C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

14. R. Tolimieri, M. An, and Ch. Lu. *Algorithms for Discrete Fourier Transform and Convolution*. Springer Verlag, 1997.

15. L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

16. V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings 6th Structure in Complexity Theory*, pages 270–284. IEEE Computer Society Press, 1991.

17. I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner series in computer science. B. G. Teubner & John Wiley, Stuttgart, 1987.

APPENDIX

Helpful properties of matrix algebra

The following hold when the expressions are defined:

1. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.
2. $(A + B) \otimes (C + D) = A \otimes C + A \otimes D + B \otimes C + B \otimes D$.
3. $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$.
4. $(A \cdot B)^{\otimes n} = A^{\otimes n} \cdot B^{\otimes n}$ for any $n \geq 1$.
5. $(A + B)^{\mathsf{T}} = A^{\mathsf{T}} + B^{\mathsf{T}}$, $(A \cdot B)^{\mathsf{T}} = B^{\mathsf{T}} \cdot A^{\mathsf{T}}$, and $(A \otimes B)^{\mathsf{T}} = A^{\mathsf{T}} \otimes B^{\mathsf{T}}$.

Stride permutations, which play a crucial role in the implementation of efficient parallel programs for block recursive algorithms such as the fast Fourier transform (FFT) and Batcher's bitonic sort (see [14]) will be useful in our proofs. The *mn-point stride n permutation* $P_n^{mn} \in \mathbb{M}_{\mathcal{S}}^{mn,mn}$ is defined as

$$P_n^{mn} \left(e_i^m \otimes e_j^n\right)^{\mathsf{T}} = \left(e_j^n \otimes e_i^m\right)^{\mathsf{T}},$$

where $e_i^m \in \mathbb{M}_{\mathcal{S}}^{1,m}$ and $e_j^n \in \mathbb{M}_{\mathcal{S}}^{1,n}$. In other words, the matrix P_n^{mn} permutes the elements of a vector of length mn with stride distance n.

Properties of stride permutations; the first three identities are taken from [14].

1. $(P_n^{mn})^{-1} = P_m^{mn}$;
2. $P_{mn}^{\ell mn} = P_m^{\ell mn} \cdot P_n^{\ell mn}$;
3. $P_n^{\ell mn} = \left(P_n^{\ell n} \otimes I_m\right) \cdot (I_\ell \otimes P_n^{mn})$;
4. $P_n^{mn} \cdot (A \otimes B) = (B \otimes A) \cdot P_n^{mn}$, whenever $\mathcal{S}$ is commutative, or A and B contain only 0's and 1's;
5. $P_n^{mn} = \sum_{i=1}^{n} (e_i^n)^{\mathsf{T}} \otimes I_m \otimes (e_i^n)$;
6. $(P_n^{mn})^{-1} = \sum_{i=1}^{n} (e_i^n) \otimes I_m \otimes (e_i^n)^{\mathsf{T}}$.

Proof of Lemma 2

By Proposition 1, checking whether the input string encodes a valid tensor circuit, and whether the given requirements on depth and tensor depth are satisfied can be done on a deterministic Turing machine in time and space $(n \cdot 2^{t(n)} \log n)^{O(1)}$ and $O(n \cdot \log d(n))$, respectively. Then we use an adaptation of the algorithm for the evaluation problem $\mathrm{val}_{\mathcal{S}}$ described in [9, Lemma 16]; it works by guessing step-by-step a certificate and recursively computing its weight on an algebraic Turing machine over semiring $\mathcal{S}$. Thus, the tree traversal plus the manipulation of entry indices and values takes $O(2^{d(n)})$ times $(2^{t(n)} \log n)^{O(1)}$ individual operations. The workspace required is a description of the current root-node path in the circuit, plus entry positions for all the matrices along this path, and the space necessary to perform operations; this means at most $O(d(n) \cdot 2^{t(n)} \log n)$ memory space. Thus the running time of the whole algorithm is bounded by $O(2^{d(n)}(n \cdot 2^{t(n)} \log n)^{O(1)})$ and its workspace by $(n \cdot 2^{t(n)} \log n)^{O(1)}$.

Full proof of Lemma 3

As it was specified in the proofsketch, we will implicitly use Warshall's algorithm in the forthcoming reduction.

Recall that since $M = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$ is fixed we impose the following assumptions without loss of generality:

1. The tape alphabet is $\Gamma = \{0, 1, B\}$; hence input $w \in \Sigma^*$ is binary. This increases running time and space by a constant factor.
2. Machine M is single-tape and the tape is cyclic with $s(n)$ tape cells, numbered 0 to $s(n) - 1$, so that tape-head motions are from cell i to cell $(i \pm 1) \pmod{s(n)}$. Moreover, all computations of M have length exactly $t(n)$ and accepting ones are determined solely by the state, regardless of the tape content.
3. A single transition of M is split in two phases: first, the reading and overwriting are done without head motion; second, the new state determines whether there will be a head movement, and if so, this is implemented without changing state or tape content by taking a transition assigned the weight 1. The transition functions for the two phases are denoted by δ_1 and δ_2, respectively.

We remind the reader that we regard M as a machine where the head is static, always pointing to the leftmost position on the tape, and where all motions are implemented with cyclic shiftings of the tape content. The set of all configurations equals $Q \times \Gamma^{s(n)}$, where $\Gamma^{s(n)}$ is a shorthand notation for the set of all strings of length exactly $s(n)$.

Let $Q = \{q_1, \ldots, q_m\}$ and $\Gamma = \{a_1, a_2, a_3\}$. Define mappings $c_{_Q} : Q \to \mathbb{M}_{\mathcal{S}}^{1,m}$ and $c_{_\Gamma} : \Gamma \to \mathbb{M}_{\mathcal{S}}^{1,3}$ as: $c_{_Q}(q_i) = e_i^m$ and $c_{_\Gamma}(a_i) = e_i^3$. If there is no danger of confusion we simply write c for $c_{_Q}$ and $c_{_\Gamma}$, respectively. With a configuration $C = (q, a_0 \ldots a_{s(n)-1})$ in $Q \times \Gamma^{s(n)}$ we associate the unit vector

$$c(C) = c(q) \otimes \bigotimes_{i=0}^{s(n)-1} c(a_i).$$

The value of the function f_M on input w equals the sum of all labels of accepting paths of length $t(n)$ in the configuration graph G_M with node set $Q \times \Gamma^{s(n)}$. We can associate a matrix A_{G_M} of order $m \cdot 3^{s(n)} \times m \cdot 3^{s(n)}$ with G_M, such that the (i, j)th entry of $A_{G_M}^{t(n)}$ equals the sum of all labels of paths of length $t(n)$ linking configuration C_i with C_j. Let $A = A_{G_M}^{\mathsf{T}}$. In order to obtain $f_M(w)$ one has to pre- and post-multiply $A^{t(n)}$ by appropriate row and column vectors representing all accepting configurations and the initial configuration of M on input w, respectively. This reads as $f_M(w) = V_{\text{accept}} \cdot A^{t(n)} \cdot (V_{\text{init}})^{\mathsf{T}}$, where $V_{\text{accept}} = \sum_{C \in F \times \Gamma^{s(n)}} c(C)$ encodes the accepting configurations, $V_{\text{init}} = c(C_0)$, and $C_0 = (q_1, a_0 \ldots a_{n-1} B^{s(n)-n})$ is the initial configuration of M on $w = a_0 \ldots a_n$. We now show how to define a tensor circuit $f_M(w)$ which meets our requirements on size, depth, and tensor depth. In order to construct V_{accept} we use the identity

$$V_{\text{accept}} = \sum_{\substack{q \in F \\ \boldsymbol{a} \in \Gamma^{s(n)}}} c(q) \otimes \bigotimes_{i=0}^{s(n)-1} c(a_i) \tag{2}$$

$$= \left(\sum_{q \in F} c(q) \right) \otimes \left(\sum_{\boldsymbol{a} \in \Gamma^{s(n)}} \bigotimes_{i=0}^{s(n)-1} c(a_i) \right) \left(\sum_{q \in F} c(q) \right) \otimes (1)_3^{\otimes s(n)}, \tag{3}$$

where $\boldsymbol{a}$ is a shorthand for $a_0 \dots a_{s(n)-1}$ and $(1)_n$ denotes the all-one row vector of length n.
Thus, V_{accept} can be computed by subcircuits of size, depth, and tensor depth $O(\log\log 3^{s(n)}) = O(\log s(n))$. The same upper bound applies (loosely) to the subcircuit for V_{init}.
Thus, it remains to construct a tensor circuit which evaluates to A. Because of assumption (3) we have $A = D \cdot T$, where matrices T and D simulate the first and second phases, respectively.
Consider the matrix T_{loc} of order $3m \times 3m$, which is defined as

$$\Big(c(q') \otimes c(b)\Big) \cdot T_{loc} \cdot \Big(c(q) \otimes c(a)\Big)^{\mathsf{T}} = \begin{cases} s & \text{if } (q, a, q', b, S, s) \in \delta_1, \\ 0 & \text{otherwise.} \end{cases}$$

The action of T on configurations can be implemented in parallel on all pairs of configurations by the matrix

$$T = T_{loc} \otimes I_3^{\otimes s(n)-1},$$

working on the vectors associated with configurations, so that with $C = (q, aa_1 \dots a_{s(n)-1})$, the product $T \cdot c(C)^{\mathsf{T}}$ simulates one computation step $C = (q, aa_1 \dots a_{s(n)-1}) \vdash_M (q', ba_1 \dots a_{s(n)-1})$ with weight s in the former case and no legal move in the latter. Note that all possible moves are traced simultaneously.

Matrix T_{loc} has fixed size, is obtained from the specification of M and can be given as an input to the circuit we are building, while a subcircuit of size, depth, and tensor depth $O(\log\log 3^{s(n)-1}) = O(\log s(n))$ can compute $I_3^{\otimes s(n)-1}$ starting from I_3, and from this the whole matrix T. This takes care of the first phase of a transition of M.

The states of M are partitioned into left-, right-shifting, and static. Define three "filter" matrices F_L, F_R, and F_S of order $3m \times 3m$ as

$$c(q) \cdot F_X \cdot c(q)^{\mathsf{T}} = \begin{cases} 1 & \text{if } (q, a, q, a, X, 1) \in \delta_2, \\ 0 & \text{otherwise.} \end{cases}$$

Then matrix D which simulates the second phase of a time step of M can be written as

$$D = (F_L \otimes R_{s(n)}) + (F_S \otimes I_3^{\otimes s(n)}) + (F_R \otimes L_{s(n)}),$$

where $R_{s(n)}$ and $L_{s(n)}$ are $3^{s(n)} \times 3^{s(n)}$ permutation matrices which perform a right and left shift of the tape, respectively. If $\alpha \in \Gamma^{s(n)}$ is the tape content and $\alpha = a_0 \beta a_{s(n)-1}$ where a_0 and $a_{s(n)-1}$ are the extremal bits of α and β represents the middle $s(n) - 2$ bits, then after a shift of the tape one position to the right (left, respectively), the new encoding for the tape becomes $a_{s(n)-1} a_0 \beta$ ($\beta a_{s(n)-1} a_0$, respectively). Observe, that a head movement to the left corresponds to a right tape shift and *vice versa*.
Define $R_{s(n)} = P_3^{3^{s(n)}}$ and $L_{s(n)} = P_{3^{s(n)-1}}^{3^{s(n)}}$, where P_n^{mn} is the stride n permutation of order $mn \times mn$. Let $\alpha = a_0 \dots a_{s(n)-1}$ be the content of the tape. To see that $R_{s(n)}$ implements a cyclic tape shift to the right, verify that

$$R_{s(n)} \cdot \left(\bigotimes_{i=0}^{s(n)-1} c(a_i) \right)^{\mathsf{T}} = \left(c(a_{s(n)-1}) \otimes \bigotimes_{i=0}^{s(n)-2} c(a_i) \right)^{\mathsf{T}}.$$

Analogously one shows that $L_{s(n)}$ implements a cyclic tape shift to the left. In order to construct $R_{s(n)}$ and $L_{s(n)}$, we observe that $P_n^{mn} = \sum_{i=1}^{n} (e_i^n)^{\mathsf{T}} \otimes I_m \otimes (e_i^n)$, which leads to the identities

$R_{s(n)} = \sum_{i=1}^{3} \left(e_i^3\right)^{\mathsf{T}} \otimes I_3^{\otimes s(n)-1} \otimes \left(e_i^3\right)$ and $L_{s(n)} = \sum_{i=1}^{3} \left(e_i^3\right) \otimes I_3^{\otimes s(n)-1} \otimes \left(e_i^3\right)^{\mathsf{T}}$

This enables us to construct subcircuits with size, depth, and tensor depth bounded by $O(\log s(n))$. From our analysis of T and D, we conclude that matrix $A = D \cdot T$ can be realized within the same size, depth, and tensor depth bounds. Hence matrix $A^{t(n)}$ requires size and depth $O(d(n))$, where $d(n) = \max\{\log t(n), \log s(n)\}$, and tensor depth $O(\log s(n))$.

Proof of Lemma 4

Lemma 4 generalizes a construction of Damm *et al.* [9, Lemma 7], which was the main tool in the construction of a tensor formula for the permanent and for some tensor formula completeness results. For this we use two intermediate results, Lemmas 6 and 7.

Lemma 6. *Let m, n be natural numbers. Let two sequences $\mathbf{A} = (A_i)$, with $1 \le i \le m$, of $k \times k$ matrices and $\mathbf{B} = (B_j)$, with $1 \le j \le n$, of $\ell \times \ell$ matrices both over a semiring be given. Then*

$$\begin{pmatrix} A_1 & & 0 \\ & A_2 & \\ & \ddots & \\ 0 & & A_m \end{pmatrix} \otimes \begin{pmatrix} B_1 & & 0 \\ & B_2 & \\ & \ddots & \\ 0 & & B_n \end{pmatrix}$$

equals the matrix

$$\left(I_m \otimes P_n^{kn} \otimes I_\ell\right)^{-1} \cdot \begin{pmatrix} A_1 \otimes B_1 & & & 0 \\ & A_1 \otimes B_2 & & \\ & & \ddots & \\ 0 & & & A_m \otimes B_n \end{pmatrix} \cdot \left(I_m \otimes P_n^{kn} \otimes I_\ell\right).$$

Proof. Rewrite the matrix

$$\begin{pmatrix} A_1 & & 0 \\ & A_2 & \\ & \ddots & \\ 0 & & A_m \end{pmatrix} = \sum_{i=1}^{m} D_i^m \otimes A_i$$

and do similar for the other matrix containing the B_j's. Then

$$\left(\sum_{i=1}^{m} D_i^m \otimes A_i\right) \otimes \left(\sum_{j=1}^{n} D_j^n \otimes B_j\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} D_i^m \otimes A_i \otimes D_j^n \otimes B_j.$$

Now consider, e.g., a term $D_i^m \otimes A_i \otimes D_j^n \otimes B_j$ pre- and post-multiplied by $I_m \otimes P_n^{kn} \otimes I_\ell$ and its inverse, respectively. Then this reads as

$$\begin{aligned}\left(I_m \otimes P_n^{kn} \otimes I_\ell\right) \cdot \left(D_i^m \otimes A_i \otimes D_j^n \otimes B_j\right) \cdot \left(I_m \otimes P_n^{kn} \otimes I_\ell\right)^{-1} = \\ = (I_m \cdot D_i^m \cdot I_m) \otimes \left(P_n^{kn} \cdot \left(A_i \otimes D_j^n\right) \cdot \left(P_n^{kn}\right)^{-1}\right) \otimes (I_\ell \cdot B_j \cdot I_\ell) = \\ = D_i^m \otimes D_j^n \otimes A_i \otimes B_j.\end{aligned}$$

Hence we have

$$\begin{aligned}\left(I_m \otimes P_n^{kn} \otimes I_\ell\right) \cdot \left(\sum_{i=1}^{m}\sum_{j=1}^{n} D_i^m \otimes A_i \otimes D_j^n \otimes B_j\right) \cdot \left(I_m \otimes P_n^{kn} \otimes I_\ell\right)^{-1} = \\ = \sum_{i=1}^{m}\sum_{j=1}^{n} D_i^m \otimes D_j^n \otimes A_i \otimes B_j = \sum_{i=1}^{mn} D_i^{mn} \otimes A_r \otimes B_s,\end{aligned}$$

where $i = m \cdot (r-1) + s$ such that $1 \leq r \leq m$ and $1 \leq s \leq n$. □

To understand the next Lemma, keep in mind the situation in which it is required to compute, say $(A+B) \otimes (C+D+E) \otimes (F)$, where A, B, C, D, E, and F are $k \times k$ matrices. Lemma 7 describes a preliminary step which uses tensors to produce a large block matrix having the expansion Kronecker products $A \otimes C \otimes F$, $A \otimes D \otimes F$, $A \otimes E \otimes F$, $B \otimes C \otimes F$, $B \otimes D \otimes F$, and $B \otimes E \otimes F$ as diagonal blocks. This particular application of Lemma 7 would require the parameters $n = 3$, $m_1 = 2$, $m_2 = 3$, and $m_3 = 1$.

Lemma 7. *Let $m_1, \ldots, m_n$ be natural numbers, and write $N = \prod_{i=1}^{n} m_i$. Let a sequence $\mathbf{A} = (A_{i,j_i})$, with $1 \leq i \leq n$ and $1 \leq j_i \leq m_i$, of $k \times k$ matrices over a semiring be given. Consider the $k^n \times k^n$ matrix*

$$\bigotimes_{i=1}^{n}(A_{i,1} + \cdots + A_{i,m_i}) = \sum_{\substack{1 \leq j_1 \leq m_1 \\ \vdots \\ 1 \leq j_n \leq m_n}} A_{1,j_1} \otimes A_{2,j_2} \otimes \cdots \otimes A_{n,j_n}. \tag{4}$$

There is a polytime Turing machine producing linear output which computes, on input $\mathbf{A}$, a tensor circuit $C_n(\mathbf{A})$, of linear size and of depth and tensor depth $O(\log n)$ evaluating to a $k^n N \times k^n N$ matrix having each $k^n \times k^n$ summand $A_{1,j_1} \otimes A_{2,j_2} \otimes \cdots \otimes A_{n,j_n}$ occurring in Equation (4) as a $k^n \times k^n$ block along its diagonal, i.e,

$$A_{1,j_1} \otimes \cdots \otimes A_{n,j_n} = \left(e_{j_1}^{m_1} \otimes \cdots \otimes e_{j_n}^{m_n} \otimes I_{k^n}\right) \cdot C_n(\mathbf{A}) \cdot \left(e_{j_1}^{m_1} \otimes \cdots \otimes e_{j_n}^{m_n} \otimes I_{k^n}\right)^{\mathsf{T}},$$

and zero elsewhere.

Proof. The statement is proved by induction on n. Obviously,

$$C_1(\mathbf{A}) = \sum_{j=1}^{m_1} \left(D_j^{m_1} \otimes A_{1,j} \right)$$

has the required form for $n = 1$, where D_i^m is the $m \times m$ "dot matrix" having one in position (i, i) and zeros elsewhere. Then, for $n > 1$ let $n_1 + n_2 = n$. By induction hypothesis we may assume that for the sequences $\mathbf{A}_1 = (A_{i,j})$, with $1 \leq i \leq n_1$, and $\mathbf{A}_2 = (A_{n_1+i,j})$, with $1 \leq i \leq n_2$, we already have matrices $C_{n_1}(\mathbf{A}_1)$ and $C_{n_2}(\mathbf{A}_2)$, respectively, of appropriate form representing the summands of $\bigotimes_{i=1}^{n_1}(A_{i,1} \otimes \cdots \otimes A_{i,m_i})$ and $\bigotimes_{i=1}^{n_2}(A_{n_1+i,1} \otimes \cdots \otimes A_{n_1+i,m_i})$, respectively. To simplify notations write $N_1 = \prod_{i=1}^{n_1} m_i$ and $N_2 = \prod_{i=1}^{n_2} m_{n_1+i}$. In order to obtain $C_n(\mathbf{A})$ we have to apply Lemma 6. This is done as follows:

$$C_n(\mathbf{A}) = \left(I_{N_1} \otimes P_{N_2}^{k^{n_1}N_2} \otimes I_{k^{n_2}} \right) \left(C_{n_1}(\mathbf{A}_1) \otimes C_{n_2}(\mathbf{A}_2) \right) \left(I_{N_1} \otimes P_{N_2}^{k^{n_1}N_2} \otimes I_{k^{n_2}} \right)^{-1},$$

where $I_{N_1} = \bigotimes_{i=1}^{n_1} I_{m_i}$ and $I_{k^{n_2}} = \bigotimes_{i=1}^{n_2} I_k$. Moreover, the stride permutation used here obeys

$$P_{N_2}^{k^{n_1}N_2} = \prod_{i=1}^{n_2} P_{m_{n_1+i}}^{k^{n_1}N_2}$$

and a similar identity holds for the inverse of $P_{N_2}^{k^{n_1}N_2}$, too. Then by easy means one observes that $C_n(\mathbf{A})$ is polytime constructible. If we choose in the above construction of circuit $C_n(\mathbf{A})$ the values of n_1 and n_2 to be close to $n/2$, this results in a circuit of linear size and $O(\log n)$ depth and tensor depth, respectively. □

Proof of Lemma 4.

Let $M = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$ be an algebraic Turing machine running in time $t(n)$ and space $s(n) = t(n)$, having m states, and fulfilling the properties (1), (2), and (3) as in the proof of Lemma 3. Define $n' = p(n)$ to be the length of the input given to the machine M. Then we proceed as in the proof Lemma 3.

Recall, that a configuration $C = (q, a_0 \ldots a_{s(n')-1}) \in Q \times \Gamma^{s(n')}$ is coded by the unit vector

$$c(C) = c(q) \otimes \bigotimes_{i=0}^{s(n')-1} c(a_i),$$

and that we can build tensor circuits of linear size, and depth and tensor depth $O(\log t(n'))$ for

$$\sum_{C \in F \times \Gamma^{s(n')}} c(C), \quad A^{t(n')}, \quad \text{and} \quad c(C_0),$$

where A is the transition matrix of M and C_0 the initial configuration if M starts on input $0^{n'}$. These circuits can be constructed on a Turing machine in polytime producing linear output.

Instead of computing f_M on input $0^{n'}$ only we make use of the copying feature of the Kronecker product and of Lemma 7 to "parallelize" computations in a controlled way on *all* possible inputs of

length n'. Observe, that every coding of a possible initial configuration on inputs of length n' can be generated starting from $c(C_0)$ by appropriately applying a $|\Gamma| \times |\Gamma|$ permutation matrix P realizing

$$c(a) \cdot P = \begin{cases} c(1) & \text{if } a = 0, \\ c(0) & \text{if } a = 1, \text{ and }, \\ c(B) & \text{if } a = B. \end{cases}$$

or the identity on the input bits $0 \leq i \leq n' - 1$ in sequence. Now we apply Lemma 7 to the sequence $\mathbf{A} = (A_{i,j_i})$, with $0 \leq i \leq n' + 1$ and $j_0 = 1$, $1 \leq j_i \leq 2$ for $1 \leq i \leq n'$, and $j_{n'+1} = 1$, where $A_{1,1} = I_m$, $A_{i,1} = I_3$ and $A_{i,2} = P$ for $1 \leq i \leq n'$, and $A_{n'+1,1} = I_3^{\otimes s(n')-n'}$. This results in a tensor circuit $C_{n'} = C_{n'}(\mathbf{A})$ evaluating to a square matrix of order $m3^{s(n')} \cdot 2^{n'}$ with $2^{n'}$ blocks of size $m3^{s(n')}$ along the main diagonal and zeros elsewhere, such that the non-zero blocks are the $m3^{s(n')} \times m3^{s(n')}$ permutations matrices $A_{1,j_1} \otimes \cdots \otimes A_{n'+1,j_{n'+1}}$ for some j_i's in the appropriate range. Let

$$\left(I_2^{\otimes n'} \otimes c(C_0)\right) \cdot C_{n'} = \begin{pmatrix} c(C_0) & & & 0 \\ & c(C_1) & & \\ & & \ddots & \\ 0 & & & c(C_{2^{n'}}) \end{pmatrix}, \tag{5}$$

where C_i is a configuration of the form $(q_1, a_0 \ldots a_{n'-1} B^{s(n')-n'})$ and $C_i \neq C_j$ if $i \neq j$. This matrix is of order $2^{n'} \times 2^{n'} \cdot m3^{s(n')}$, it contains the coding of every possible configuration on inputs of length n' being in start state q_1, in a brick like fashion, from its upper left corner to its lower right corner, and it vanishes everywhere else. Thus, by Equations (2) and (5) this results in

$$C_{M,n} = \left(I_2^{\otimes n'} \otimes \sum_{C \in F \times \Gamma^{s(n')}} c(C)\right) \cdot \left(I_2^{\otimes n'} \otimes A^{t(n')}\right) \cdot \left(\left(I_2^{\otimes n'} \otimes c(C_0)\right) \cdot C_{n'}\right)^{\mathsf{T}},$$

which evaluates to the $2^{n'} \times 2^{n'}$ matrix containing the values f_M on all possible inputs of length n' on the main diagonal, and zero elsewhere. Due to the recursive nature of the construction given in Lemma 7 one observes that Equation (1) holds and that the resulting tensor circuit $C_{M,n}$ is deterministic polylin constructible. According to Lemma 7 the depth of $C_{M,n}$ is bounded by $O(\log s(n'))$, which is logarithmic in n if $n' = p(n)$ is a polynomial and linear in n in case $n' = p(n)$ is exponential. □

Proof of Lemma 5

We only prove the statement concerning problem F-ZERO$_{\mathcal{S}}$. Let M be the following algebraic Turing machine over $\mathcal{S}$: Given an input (F, i, j), the machine M checks whether $(i, j) \in I(F)$, and if so guesses a string and verifies that it encodes a certificate for (F, i, j). If it is a certificate, the weight of

this particular computation equals the weight of the certificate. In the other case M rejects. Obviously, M computes the function

$$f_M(F, i, j) = \sum_{(S,\epsilon)} w(S, \epsilon)$$

where $w(S, \epsilon)$ is the weight of a certificate and where the sum runs over all certificates for the entry i, j of F's output. Machine M runs in polytime by [9, Proposition 5]. Therefore, f_M is in $\mathcal{S}$-#P, which equals $\#^p \cdot \mathcal{S}$-#P (see Page 4). Then $(F, i, j) \in$ F-ZERO$_\mathcal{S}$ if and only if $f_M(F, i, j) = 0$. Thus, F-ZERO$_\mathcal{S}$ belongs to $\forall^p \cdot \mathcal{S}$-#P. Thus, it remains to prove hardness. By Theorem 2 the function class $\mathcal{S}$-#P $= \#^p \cdot \mathcal{S}$-#P polytime many-one reduces to the evaluation problem on tensor formulas over $\mathcal{S}$. Therefore, F-ZERO$_\mathcal{S}$ is hard for $\forall^p \cdot \mathcal{S}$-#P.

Complexity of problems NULL$_\mathcal{S}$ **and** DIAG$_\mathcal{S}$

We write the proof in terms of problem DIAG$_\mathcal{S}$; the reasoning for NULL$_\mathcal{S}$ is identical.

First we show containment in $\forall^e \cdot (\forall^p \cdot S$-#P). Let C be a tensor formula instance of DIAG$_\mathcal{S}$ and assume that $I(C) = [k] \times [\ell]$. Then $C \in$ DIAG$_\mathcal{S}$ if and only if C is a square tensor circuit and

$$\forall (i, j) \in [k] \times [\ell] : \big((i \neq j) \Rightarrow (C, i, j) \in \text{ZERO}_\mathcal{S}\big).$$

By Proposition 1, whether a circuit outputs a square matrix can be tested in deterministic exponential time. Next, since ZERO$_\mathcal{S}$ is in $\forall^e \cdot \mathcal{S}$-#P by Lemma 5, membership in DIAG$_\mathcal{S}$ can be checked in $\forall^e \cdot (\forall^p \cdot \mathcal{S}$-#P); note that given (exponential-length) binary encodings for indices i and j, an algorithm for ZERO$_\mathcal{S}$ will have a computation time polynomial in the length of its input, which consists in i, j, and a description of C.
For the hardness part we argue as follows. Let L be a language in $\forall^e \cdot (\forall^p \cdot S$-#P). Then by definition there is a constant c and an algebraic polynomial time Turing machine M over $\mathcal{S}$ such that

$$x \in L \quad \text{if and only if} \quad \forall |y| = 2^{c|x|} : \ f_M(x, y) = 0.$$

Define C to be the tensor circuit constructed in polytime, according to Lemma 4, from M and the constant c such that C evaluates to a matrix of order $2^{2^{c|x|}} \times 2^{2^{c|x|}}$ containing all values of f_M on input $\langle x, y \rangle$ for all possible y's of appropriate length on the main diagonal. Then it is easily seen that $x \in L$ if and only if tensor formula

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes C$$

evaluates to the all-zero matrix (or equivalently to a diagonal matrix). This completes the construction and proves the stated claim.