Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 345 (2005) 206–234

Theoretical Computer Science

www.elsevier.com/locate/tcs

A common algebraic description for probabilistic and quantum computations☆,☆☆

Martin Beaudry[a], José M. Fernandez[b, 1], Markus Holzer[c, *, 1]

[a]*Département d'informatique, Université de Sherbrooke, 2500, boul. Université, Sherbrooke, Québec, Canada J1K 2R1*

[b]*Département de génie informatique, École Polytechnique de Montréal, C.P. 6079, succ. Centre-Ville, Montréal, Québec, Canada H3C 3A7*

[c]*Institut für Informatik, Technische Universität München Boltzmannstraße 3, D-85748 Garching bei München, Germany*

Abstract

Through the study of gate arrays we develop a unified framework to deal with probabilistic and quantum computations, where the former is shown to be a natural special case of the latter. On this basis we show how to encode a probabilistic or quantum gate array into a sum-free tensor formula which satisfies the conditions of the partial trace problem, and vice-versa; that is, given a tensor formula F of order $n \times 1$ over a semiring $\mathscr{S}$ plus a positive integer k, deciding whether the kth partial trace of the matrix $\mathrm{val}_{\mathscr{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ fulfills a certain property. We use this to show that a certain promise version of the sum-free partial trace problem is complete for the class pr- BPP (promise BPP) for formulas over the semiring $(\mathbb{Q}^{+}, +, \cdot)$ of the positive rational numbers, for pr-BQP (promise BQP) in the case of formulas defined over the field $(\mathbb{Q}^{+}, +, \cdot)$, and if the promise is given up, then completeness

☆ A preliminary version of this paper appeared in the Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS) [5].

☆☆ Supported by the National Sciences and Engineering Research Council (NSERC) of Canada, by the *Fonds pour la Formation de Chercheurs et l'Aide à la Recherche* (FCAR) of Québec, and by the Deutsche Forschungsgemeinschaft (DFG).

* Corresponding author.

E-mail addresses: martin.beaudry@usherbrooke.ca (M. Beaudry), jose.fernandez@polymtl.ca (J.M. Fernandez), holzer@informatik.tu-muenchen.de (M. Holzer).

1 Part of the work was done while these authors were at the Département d'I.R.O., Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal, Québec, H3C 3J7 Canada.

0304-3975/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.tcs.2005.07.008

for PP is shown, regardless whether tensor formulas over positive rationals or rationals in general are used. This suggests that the difference between probabilistic and quantum polytime computers may ultimately lie in the possibility, in the latter case, of having destructive interference between computations occurring in parallel. Moreover, by considering variants of this problem, classes like $\oplus$P, NP, $\mathrm{C_{=}P}$, its complement co-$\mathrm{C_{=}P}$, the promise version of Valiant's class UP, its generalization promise SPP, and unique polytime US can be characterized by carrying the problem properties and the underlying semiring.

1. Introduction

The "algebraic" approach in the theory of computational complexity consists in characterizing complexity classes within unified frameworks built around a computational model or problem involving an algebraic structure (usually finite or finitely generated) as the main parameter. In this way, various complexity classes are seen to share the same definition, up to the choice of the underlying algebra. Successful examples of this approach include the description of NC^1 and its subclasses AC^0 and ACC^0 in terms of polynomial-size programs over finite monoids [26], and analogous results for PSPACE, the polynomial hierarchy and the polytime mod-counting classes, through the use of polytime leaf languages [22]. A more recent example is the complexity of problems whose input is a tensor formula, i.e., a fully parenthesized expression where the inputs are matrices (given in full) over some finitely generated algebra and the allowed operations are matrix addition, multiplication, and tensor product, also known as outer, or direct, or Kronecker product. Evaluating tensor formulas with explicit tensor entries is shown by Damm et al. [9] to be complete for $\oplus$P, for NP, and for #P as the semiring varies. Recently also other common sense computational problems on tensor formulas and tensor circuits were analyzed by Beaudry and Holzer [6]. Tensor formulas are a compact way of specifying very large matrices. As such, they immediately find a potential application in the description of the behavior of circuits, be they classical Boolean, arithmetic (tensor formulas over the appropriate semiring) or quantum (formulas over the complex field, or an adequately chosen sub-semiring thereof).

In this paper we formalize and confirm this intuition, that basic tensor calculus not only captures natural complexity classes in simple ways, but also yields a simpler and unified view on classical probabilistic and quantum computation, which gives probabilistic and quantum computations the exact same definition, up to the underlying algebra. Apart from offering a first application of the algebraic approach to quantum computing, our paper thus reasserts the point made by Fortnow [15], that for the classes BPP and BQP, the jump from classical to quantum polynomial-time computation consists in allowing negative matrix entries for the evolution operators, which means that different computations done in parallel may interfere destructively. Based on this unified framework, we define a meaningful computational problem on tensor formula, called the *partial trace tensor formula problem*, which is fundamental to our studies, and allows us to capture important complexity classes. Our precise characterizations are as follows:

- We present probabilistic computation as a natural special case of quantum computation using the unified framework on gate arrays, instead of presenting quantum as a more or less artificial extension of probabilistic computation.

- The partial trace *sum-free* tensor formula problem enables us to capture the significant complexity classes (pr-)P (promise P), NP, pr-BPP (promise BPP), and PP and some of their quantum counterparts pr-EQP (promise EQP), NQP, and pr-BQP (promise BQP), by showing completeness results of the problem under consideration.
- By bringing back sums into tensor formulas, we obtain completeness statements for further complexity classes like $\oplus$P, NP, C$_=$P, its complement co-C$_=$P, Valiant's class pr-UP (promise UP), its generalization pr-SPP (promise SPP), and unique polytime US.

Some of these classes are "semantic" classes, i.e., the underlying machine must obey a property for all inputs, which is not obvious to check, or even undecidable. An example would be UP, since for a non-deterministic machine to define a language in UP, it must have the property that for all inputs either exactly one accepting path exists or none. Therefore, the obtained completeness results are subject to a certain promise.

The paper is organized as follows: In the next section we introduce the complexity classes needed in later sections and the basics on semirings. In Section 3 we provide the reader with the necessary background on deterministic, probabilistic, and quantum computation and develop our unified view of all these computations based on gate arrays. Then in Section 4 we introduce the terminology and basic parsing techniques for tensor formulas. Section 5 shows how to transform a gate array into a sum-free tensor formulas of special type and vice versa, which then is applied in Section 6 to prove the main results of the papers. Then in Section 7 we consider the unrestricted partial trace tensor formula problem, and finally, in the last section, we conclude and discuss, related results.

2. Definitions

We use standard notation from computational complexity [2,19,29]. In particular we recall the inclusion chains:

$$\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{PP} \supseteq \mathrm{NP} \quad \text{and} \quad \mathrm{EQP} \subseteq \mathrm{BQP} \subseteq \mathrm{PP} \supseteq \mathrm{NQP}.$$

Here P (NP, respectively) denotes the set of problems solvable by deterministic (non-deterministic) Turing machines in polytime, and the probabilistic class PP (BPP, respectively) is the set of all languages accepted by non-deterministic Turing machine in polytime with majority (strict majority, respectively). Moreover, EQP, NQP, and BQP denote the quantum analog of P, NP, and BPP, respectively. In the sequel, whenever we simultaneously deal with probabilism and quantum, we use the notations and vocabulary from the quantum case, in order to make the text easier to read.

A *semiring* [16,23] is a tuple $(\mathcal{S}, +, \cdot)$ with $\{0, 1\} \subseteq \mathcal{S}$ and two 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, i.e.,

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{and} \quad (a + b) \cdot c = a \cdot c + b \cdot c,$$

for every a, b, and c in $\mathcal{S}$, and $0 \cdot a = a \cdot 0 = 0$ for every a in $\mathcal{S}$. A semiring $\mathcal{S}$ 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 all of $\mathcal{S}$ by summation, and is a *ring* if and only if $(\mathcal{S}, +, 0)$ is a group. The special choice of $\mathcal{G}$ has no influence on the complexity of problems we study in this paper. In this paper we consider the following semirings, which are finitely representable, i.e., every element from $\mathcal{S}$ can be encoded and easily manipulated over a finite alphabet: the field of rationals $(\mathbb{Q}, +, \cdot)$ and the commutative semiring of positive rationals $(\mathbb{Q}^+, +, \cdot)$. Moreover, we refer also to the field of complex numbers $(\mathbb{C}, +, \cdot)$.

Let $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ denote the set of all *matrices* over $\mathcal{S}$ 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 $(A)_{i,j}$, the transpose of A by $A^\top$, and its inverse, if A is an invertible square matrix, by A^{-1}. A square matrix A over the complex numbers is *unitary*, if and only if $A^\dagger = A^{-1}$, where $A^\dagger$ denotes the conjugate transpose of A, and for a matrix A with rational entries this translates into $A^\top = A^{-1}$, which means that A is *orthogonal*. Observe, that an orthogonal matrix which contains only non-negative rational entries is in fact a permutation matrix. The *trace* of an order $n \times n$ square matrix A, denoted by $\operatorname{trace}(A)$, equals the sum of its diagonal elements, i.e.,

$$\operatorname{trace}(A) = \sum_{i=1}^{n} (A)_{i,i}.$$

For $k \geqslant 0$, the *kth partial trace* of A, for short $\operatorname{trace}_k(A)$, is the sum of its first k diagonal elements, counting downwards from the upper left corner. For completeness, if k exceeds the order of the matrix A, then the kth partial trace coincides with the trace of A.

Scalar multiplication, addition and multiplication of matrices form the basics of matrix calculus and are defined in the usual way. Scalar multiplication, addition, and multiplication of matrices over a semiring are compatible with transposition, i.e., $(a \cdot A)^\top = a \cdot A^\top$, $(A \cdot a)^\top = A^\top \cdot a$, $(A + B)^\top = A^\top + B^\top$, and $(A \cdot B)^\top = B^\top \cdot A^\top$. Furthermore, if A and B are invertible square matrices having the inverses A^{-1} and B^{-1}, then $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$. Additionally we consider the *tensor product* $\otimes : \mathbb{M}_{\mathcal{S}}^{k,\ell} \times \mathbb{M}_{\mathcal{S}}^{m,n} \to \mathbb{M}_{\mathcal{S}}^{km,\ell n}$ of matrices, also known as Kronecker product [21], 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 defined as

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

Hence

$$(A \otimes B)_{i,j} = (A)_{q,r} \cdot (B)_{s,t},$$

where $i = k \cdot (q-1) + s$ and $j = \ell \cdot (r-1) + t$. For the n-folded iteration $A \otimes A \otimes \cdots \otimes A$ we use $A^{\otimes n}$ as a shorthand notation; define $A^{\otimes 0}$ to be the scalar 1.

The main properties of the Kronecker product of matrices are gathered in the following identities. These properties are classical [18] and will be restated for reference only.

They hold true over arbitrary semirings, unless otherwise stated, whenever the corresponding operations are defined:

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C,$$
$$(A + B) \otimes (C + D) = A \otimes C + A \otimes D + B \otimes C + B \otimes D,$$

and

$$(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D),$$

if the underlying semiring is commutative. Moreover, for arbitrary semirings the last equation also holds, if B or C are zero-one matrices. The ultimate equation is probably the most important one, since it relates ordinary and Kronecker product of matrices. Moreover, $a \otimes A = a \cdot A$ and $A \otimes a = A \cdot a$ if a is a scalar, $(A \otimes B)^{\mathsf{T}} = A^{\mathsf{T}} \otimes B^{\mathsf{T}}$, and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ if A and B are invertible square matrices having the inverses A^{-1} and B^{-1}, respectively.

Next we define stride permutation matrices, which play a central role in tensor calculus over commutative semirings. The *mn-point stride n permutation* matrix P_n^{mn} in $\mathbb{M}_{\mathcal{S}}^{mn,mn}$ is defined by Ledermann [24] 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 $e_j^n \in \mathbb{M}_{\mathcal{S}}^{1,n}$ are row unit vectors of appropriate length. In particular, $P_1^n = P_n^n = I_n$, where I_n is the order n identity matrix. In other words, matrix P_n^{mn} permutes the elements of a zero-one vector of length mn with stride distance n, i.e., the matrix vector product $P_n^{mn} \cdot x$ takes a “card deck” x, splits the card deck into m piles of length n each, and then takes one card from each pile in turn until the deck is reassembled. If the underlying semiring $\mathcal{S}$ is commutative, then stride permutations obey the following commutation theorem [24]

$$P_m^{km} \cdot (A \otimes B) = (B \otimes A) \cdot P_n^{\ell n},$$

where $A \in \mathbb{M}_{\mathcal{S}}^{k,\ell}$ and $B \in \mathbb{M}_{\mathcal{S}}^{m,n}$. Thus, over commutative semirings one can reverse the order of a Kronecker product $A \otimes B$ into $B \otimes A$ by post-multiplying the equation given above on both sides with the appropriate inverse of a stride permutation. Very importantly, looking more closely, it reveals that the commutation theorem holds also true over arbitrary semirings if A or B are zero-one matrices.

The main identities of stride permutations are listed below [33]:

$$P_{mn}^{\ell mn} = P_m^{\ell mn} \cdot P_n^{\ell mn}$$

and

$$P_n^{\ell mn} = (P_n^{\ell n} \otimes I_m) \cdot (I_\ell \otimes P_n^{mn}),$$

where I_n denotes the identity matrix of order $n \times n$. Recall also, that the inverse of P_n^{mn} and in general for every permutation matrix P exists and equals its transposed, i.e., $P^{-1} = P^{\mathsf{T}}$.

3. Background on gate arrays and complexity

In this section we introduce gate arrays in order to handle the two types of computations, i.e., probabilistic and quantum. The word *circuit* is reserved for the traditional idea of an acyclic network with a unique output bit, and we use the word *gate array* to describe those computational networks, which satisfy the below given requirements.

It is useful to think of gate arrays as natural extensions of classical leveled Boolean circuits. The usual notion of depth and size on Boolean circuits naturally carries over to gate arrays. These consist of gates interconnected without fan-out [2] or feedback, by wires. Each wire represents a path of a single bit in time or space, forward from left to right, and it can be described by a state in a two-dimensional space with orthonormal basis $|0\rangle$ and $|1\rangle$. The gates have the same number of inputs and outputs, and a gate of k inputs operates on the set of k-bit vectors mapping each of the 2^k possibilities of input values to a combination of output values, i.e., it can be specified by a square matrix over a certain semiring $\mathcal{S}$, which describes its action on the specified entries and may obey certain properties. Without loss of generality we may assume, that each gate acts on neighboring wires. This requirement can easily be achieved at the cost of inserting a quadratic number of extra levels of "swap" gates, which interchange the values carried by two adjacent wires. Entries to the gate array are either input bits or non-input bits also called *ancilla* bits. Thus, an n-bit input to a gate array over semiring $\mathcal{S}$ can be seen as a formal sum of the form

$$|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w |w\rangle, \tag{1}$$

where α_w is in $\mathcal{S}$ and $|\varphi\rangle$ may obey some additional properties, and gates act on certain bits $|0\rangle$ and $|1\rangle$ of $|\varphi\rangle$ in the natural way. The vector of bits received as input by a gate array can be regarded as a linear combination of (*pure*) *states*. Finally, at the end of the gate array the decision whether the input is accepted or rejected is done by a particular observation on the output vector, which is also of the form (1). Next we compare quantum and probabilistic computation. We continue with the former one—a more detailed discussion can be found in [4].

Quantum computation was originally defined by Deutsch [10] in terms of quantum Turing machines: Here the data (qubits) handled by this machine are formally represented as a vector whose complex components give the distribution of amplitudes for the probability that the qubits be in a certain combination of values and each transition of the machine acts as a unitary transformation on this vector. Later it was shown by Yao [36] that polytime quantum Turing machines (and their inputs) can be encoded in deterministic polytime into an equivalent quantum gate array, if one allows a small probability of error. In our general view on gate arrays the properties of wires, gates, input and output vectors, and measurement

[2] Fan-in and fan-out are electrical engineering terms which refer to the *joining* and *branching* of wires; sometimes logical devices output k wires with the same signal, hence providing broadcasting, and this is known as fan-out k.

at the very end of the gate array read as follows: (1) Wires in a quantum gate array carry qubits and they can be described by a state in a two-dimensional Hilbert space with basis $|0\rangle$ and $|1\rangle$. Just as classical bit strings can represent the discrete states of arbitrary finite dimensionality, so a length n string of qubits can be used to represent quantum states in any Hilbert space of dimensionality up to 2^n. (2) The action of a k-input gate is a unitary operation of the group $U(2^k)$ of $2^k \times 2^k$ unitary matrices, i.e., a generalized rotation in a Hilbert space of dimension 2^k. The unitarity (orthogonality) property of the square matrices, which describe the performance of the gates, implies reversibility, i.e., computations where the input and output is uniquely retrievable from each other. In this way, it is always possible to un-compute or reverse the computation. It has been shown [3,11,25,32] that a small set of one- and two-qubit gates suffices to build quantum arrays, in that any k-qubit gate can be simulated by a gate array consisting of two-qubit gates, and the number thereof is at most an exponential in k. As two-qubit gates it suffices to take the controlled NOT-gate, which is defined as

$$\begin{aligned} x &\mapsto x, \\ y &\mapsto x \oplus y, \end{aligned}$$

where $\oplus$ is the two-input one-output XOR function. Moreover, the power of quantum gate arrays remain unchanged if gates are restricted to implement unitary operations with entries taken form a small set of rationals [1]. (3) The coefficients α_w in vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w |w\rangle$ are called *amplitudes* and they satisfy

$$\sum_{w\in\{0,1\}^n} |\alpha_w|^2 = 1. \tag{2}$$

Without loss of generality the input ancilla qubits are prepared to be in state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Assuming an even number of ancillae, we are back with the rationals since

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle). \tag{3}$$

Later, we will use a similar trick for probabilistic computations. (4) Finally, there is a measurement done on the array's output, which consists in projecting the output vector onto a subspace, usually defined by setting a chosen subset of the qubits to $|1\rangle$, the accepting subspace. If the qubits are numbered 1 to n, then a k-qubit accepting subset can be chosen to be qubits 1 to k, at the cost of inserting a quadratic number of extra swap gates. Thus, the probability of acceptance on input w equals the 2^kth partial trace of the matrix $|\psi\rangle\langle\psi|$, where the input is mapped to $|\psi\rangle$ by the gate array under consideration.

As quantum classes, also probabilistic complexity classes are usually defined in terms of Turing machines. Here the Turing machine picks one random bit at a time and acts deterministically otherwise.[3] In fact, deterministic computations, or to be more precise Boolean circuits, can be made reversible with little cost in efficiency [7], since there exists a three-bit universal gate for reversible computations, that is, a gate which when applied in succession to different triplets of bits in a gate array, could be used to simulate arbitrary

[3] When considering probabilistic Turing machines as Turing machines in which some transitions are random choices among finitely many alternatives, the below given argument results in gate arrays, where the gates can be described by stochastic matrices, the only ℓ_1-norm preserving linear mapping over the positive rationals.

Table 1
Probabilistic and quantum computations on gate arrays compared

	Probabilistic	Quantum
Semiring	$\mathbb{Q}^+$	$\mathbb{Q}$ ($\mathbb{C}$, resp.)
Wires	Bits $\|0\rangle$ and $\|1\rangle$	Qubits $\|0\rangle$ and $\|1\rangle$
Gates	Permutation (Stochastic, resp.)	Orthogonal (Unitary, resp.)
Vector entries α_w	Probability	Amplitude
Vectors of unit length in …	ℓ_1-norm	ℓ_2-norm
Ancilla $\alpha_0\|0\rangle + \alpha_1\|1\rangle$	$\frac{1}{2}(\|0\rangle + \|1\rangle)$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
Measurement on $\|\psi\rangle = \sum_w \alpha_w\|w\rangle$	$\sum_v \alpha_v$	$\sum_v \|\alpha_v\|^2$

reversible computations. This universal gate is called the Toffoli-gate, and is also known as the double-controlled NOT- or controlled–controlled NOT-gate and its behaviour is

$$\begin{aligned} x &\mapsto x, \\ y &\mapsto y, \\ z &\mapsto (x \wedge y) \oplus z, \end{aligned}$$

where $\oplus$ is the two-input one-output XOR function. One can easily prove that the Toffoli gate is universal; by setting z to 1 at the input, the Toffoli gate produces the NAND function, which is a two-input one-output universal gate for classical irreversible computation. Thus, from a probabilistic Turing machine an equivalent circuit and in turn a gate array over the positive rationals can be built, in which an appropriate number of random bits are fed alongside the input bits. Whether the input belongs to the language specified by the Turing machine is verified by counting those combinations of random bits, for which the output bit takes value 1, assuming that all random bit combinations have equal length and are equally likely. In this way, the constraints of a gate array read as follows: (1) Wires carry bits $|0\rangle$ and $|1\rangle$. (2) Gates implement deterministic reversible computations, i.e., they carry out permutation operations and thus can be described by matrices with 0–1 entries. (3) The coefficients α_w in vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w|w\rangle$ are called *probabilities* and they satisfy

$$\sum_{w\in\{0,1\}^n} \alpha_w = 1. \tag{4}$$

Moreover, input ancilla (probabilistic) bits are prepared to be equally likely, i.e., set to $\frac{1}{2}(|0\rangle + |1\rangle)$. (4) The measurement at the end of the gate array consists in determining the probability that the decision bits take some predefined values, usually set to $|1\rangle$, at the output level. Thus, the probability of acceptance equals the sum of some the coefficients α_w corresponding to the accepting subspace of the output vector $|\psi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w|w\rangle$. A comparison of probabilistic and quantum computation is shown in Table 1.

When restricting to rational numbers, the essential difference between probabilistic and quantum computation lies in the way, the probability of acceptance is determined. In most papers, quantum computation is presented as a natural extension of probabilistic computation. This is not convenient for us, therefore we go the other way around and want to explain how to see probabilistic computation as a natural special case of quantum. In this respect, we

Table 2
Probabilistic computations as a natural special case of quantum computation

	Probabilistic	Quantum
Semiring	$\mathbb{Q}^+$	$\mathbb{Q}$
Wires	Bits $\|0\rangle$ and $\|1\rangle$	Qubits $\|0\rangle$ and $\|1\rangle$
Gates	Orthogonal Permutation	Orthogonal
Vector entries α_w	Amplitude	Amplitude
Vectors of unit length in …	ℓ_2-norm	ℓ_2-norm
Ancilla $\sum_{w\in\{0,1\}^2} \alpha_w \|w\rangle$	$\frac{1}{2}(\|0\rangle + \|1\rangle)^{\otimes 2}$	$\frac{1}{2}(\|0\rangle + \|1\rangle)^{\otimes 2}$
Measurement on $\|\psi\rangle = \sum_w \alpha_w \|w\rangle$	$\sum_v \|\alpha_v\|^2$	$\sum_v \|\alpha_v\|^2$

are already half the way towards this goal. Consider probabilistic gate arrays in more detail. Since all the gates in the array do classical reversible computations they only permute the different vector components without ever combining them, i.e., no interference ever takes place along the array's computation, so that it does not matter in terms of overall outcome, whether the vector entries are probabilities represented as such or as amplitudes. However, using amplitudes enables us to describe the measurement at the end of the computation as in the quantum case, by determining the partial trace of a matrix. Nevertheless, we face the problem, that the amplitudes compared to the probabilities in the original vector may not be rational anymore. For instance, the amplitudes to $\frac{1}{2}(|0\rangle + |1\rangle)$ are $\frac{1}{\sqrt{2}}$ for both $|0\rangle$ and $|1\rangle$. As argued in the quantum case, when considering an even number of ancilla bits, we can overcome this problem still staying rational—see Eq. (3). This observation allows us to see probabilistic computation as a natural special case of quantum computation as follows: (1) Wires carry bits $|0\rangle$ and $|1\rangle$ and (2) gates are describable by orthogonal matrices over the positive rationals. It is elementary to verify that these are exactly the permutation matrices. In other words, these gates are still classical reversible gates. (3) Instead of dealing with probabilities we compute with amplitudes, which implies that the vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w |w\rangle$ preserves ℓ_2-norm, and the even number of ancilla bits are prepared to be equally likely, which means, that they are set to $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. (4) The measurement at the end of the gate array is done as in the case of quantum gate arrays. Our view on probabilistic computation as a natural restriction of quantum computation is depicted in Table 2.

The above given discussion motivates and satisfies the following definition and theorem.

Definition 1. Let $\mathcal{S}$ be the set of positive rationals $\mathbb{Q}^+$ or the set of rationals $\mathbb{Q}$. Define $R, A \subseteq \mathcal{S}$ with $R \cap A = \emptyset$ and $R \cup A \subseteq [0, 1]$. A logspace (polytime) uniform family of polynomial size gate arrays over $\mathcal{S}$ determines a language L as follows: Assume $1^{|w|} \mapsto C$ with an even number of ancilla bits, vector $|\varphi\rangle$ is built by input w and appropriately set ancilla bits, and $|\varphi\rangle \mapsto |\psi\rangle$ running C on input w. Then

$w \in L$ implies $f_{C(w)} \in A$

and

$$w \notin L \text{ implies } f_{C(w)} \in R,$$

where $f_{C(w)}$ denotes the probability that $|\psi\rangle$ is projected onto the accepting subspace, i.e, equals the partial trace of $|\psi\rangle\langle\psi|$ restricted to the accepting subspace.

The class $\mathcal{C}_{\mathcal{S}}(R, A)$ consists of all languages $L \subseteq \Sigma^*$ that can be accepted by gate arrays over $\mathcal{S}$ satisfying the above property.

The following theorem is immediate by the previous discussion on quantum and probabilistic computations. We state it without proof.

Theorem 2. (1) *For the positive rationals we find*:
(a) $\mathcal{C}_{\mathbb{Q}^+}(R, A) = \mathrm{P}$ *if* $R = [0]$ *and* $A = [1]$.
(b) $\mathcal{C}_{\mathbb{Q}^+}(R, A) = \mathrm{NP}$ *if* $R = [0]$ *and* $A = (0, 1]$.
(c) $\mathcal{C}_{\mathbb{Q}^+}(R, A) = \mathrm{PP}$ *if* $R = [0, \frac{1}{2}]$ *and* $A = (\frac{1}{2}, 1]$.
(d) $\mathcal{C}_{\mathbb{Q}^+}(R, A) = \mathrm{BPP}$ *if* $R = [0, \frac{1}{3}]$ *and* $A = [\frac{2}{3}, 1]$.
(2) *For the rationals we find*:
(a) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{EQP}$ *if* $R = [0]$ *and* $A = [1]$.
(b) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{NQP}$ *if* $R = [0]$ *and* $A = (0, 1]$.
(c) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{PP}$ *if* $R = [0, \frac{1}{2}]$ *and* $A = \frac{1}{2}, 1]$.
(d) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{BQP}$ *if* $R = [0, \frac{1}{3}]$ *and* $A = [\frac{2}{3}, 1]$.

Concerning the above theorem, there are three points to mention: (1) Observe, that the result on BPP (BQP, respectively) includes the constraint, that the cutpoint $\frac{1}{2}$ is isolated, i.e., the probability never falls inside the open interval $(\frac{1}{3}, \frac{2}{3})$. Nevertheless classes PP, BPP, and BQP can be redefined with a cutpoint other than $\frac{1}{2}$. (2) The quantum analog to PP is in fact no different than PP itself. We recall the simple argument, which leads to this observation. An alternative characterization of PP reads as follows [13]: A language $L \subseteq \Sigma^*$ belongs to PP if and only if there is a *Gap* P function f whose value on input w is positive, i.e., $f(w) > 0$ if and only if w is in L. Now given a quantum gate array, which checks membership of w in L with unique accepting and rejecting configurations, summing all the positive and negative contributions to the total amplitude for these configurations defines four #P functions. The difference between the probabilities of acceptance and rejection by this gate array is a quadratic polynomial in these four functions, which belongs to *Gap* P by the closure of #P under finite sum and product. Thus, language L is a member of PP. (3) Finally, it was shown by Fenner et al. [14] that NQP = co-C$_=$P, where co- denotes the complementation operation and C$_=$P is the class of sets of type $\{ x \mid f(x) = g(x) \}$, for some $f, g \in$ #P, which was introduced by Wagner [35].

4. Tensor formulas and problems

In this section we introduce tensor formulas over semirings and some basic techniques to deal with them. Moreover, we define the partial trace problem, which is fundamental to our studies.

Definition 3. The tensor *formulas* over a semiring $\mathcal{S}$ and their *order* are recursively defined as follows:

(1) Every matrix F from $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ with entries from $\mathcal{S}$ is a (*atomic*) tensor formula of order $k \times \ell$.
(2) If F and G are tensor formulas of order $k \times \ell$ and $m \times n$, respectively, then
  (a) $(F + G)$ is a tensor formula of order $k \times \ell$ if $k = m$ and $\ell = n$.
  (b) $(F \cdot G)$ is a tensor formula of order $k \times n$ if $\ell = m$.
  (c) $(F \otimes G)$ is a tensor formula of order $km \times \ell n$.
(3) Nothing else is a tensor formula.

Let $\mathbb{T}_{\mathcal{S}}$ denote the set of all tensor formulas over $\mathcal{S}$, and define $\mathbb{T}_{\mathcal{S}}^{k,\ell} \subseteq \mathbb{T}_{\mathcal{S}}$ to be the set of all tensor formulas of order $k \times \ell$.

In this paper we only consider semiring elements whose value can be given with a standard encoding over some finite $\mathcal{G}$. Hence, atomic tensor formulas, i.e., matrices, can be string-encoded using list notation such as "[[0 0 1][1 0 1]]." Non-atomic tensor formulas can be encoded over the alphabet $\Sigma = \{0\} \cup \mathcal{G} \cup \{[,], (,), \cdot, +, \otimes\}$. Strings over Σ which do not encode valid formulas are deemed to represent the trivial tensor formula 0 of order 1×1.

Let F be a tensor formula of order $m \times n$. Its *size*, denoted $|F|$, is $\max\{m, n\}$ and its *length* $L(F)$ is the number of symbols in its string representation. It is easy to show that $|F| \leqslant 2^{O(L(F))}$. The upper bound is attained when F is an iterated tensor product.

Lemma 4. *Testing whether a string encodes a valid tensor formula and if so, computing its order, can be done in deterministic polytime.*

Proof. Let M be the Turing machine which, on an input string w, rejects and halts if the bracketing or operator structure of w are illegal. This can be tested in logspace. If w is legal, then M continues by running the function *order* described by the following pseudo-code:

```
function order (tensor $F$) : (nat, nat);
var $k, \ell, m, n$ : nat;
begin case $F$ in:
    atomic:      determine order of $F$ and store it in $(k, \ell)$;
                 return $(k, \ell)$;
    $(G + H)$:   $(k, \ell) :=$ order$(G)$; $(m, n) :=$ order$(H)$;
                 if $k \neq m$ or $\ell \neq n$ then halt and reject fi;
                 return $(k, \ell)$;
    $(G \cdot H)$:   $(k, \ell) :=$ order$(G)$; $(m, n) :=$ order$(H)$;
                 if $\ell \neq m$ then halt and reject fi;
                 return $(k, n)$;
    $(G \otimes H)$: $(k, \ell) :=$ order$(G)$; $(m, n) :=$ order$(H)$;
                 return $(k\ell, mn)$;
    esac;
end.
```

The *order* function may be implemented on M, using a tape in a pushdown like fashion to handle the recursive calls. Hence M operates in polytime, since M performs a depth-first search of the formula, and since polynomial space is sufficient to keep track of the orders in binary notation. The initial call *order*(F) thus returns the order of F. $\square$

Definition 5. For each semiring $\mathcal{S}$ and each k and ℓ we define $\mathrm{val}_{\mathcal{S}}^{k,\ell} : \mathbb{T}_{\mathcal{S}}^{k,\ell} \to \mathbb{M}_{\mathcal{S}}^{k,\ell}$, as follows:

$$\mathrm{val}_{\mathcal{S}}^{k,\ell}(F) = \begin{cases} F & \text{if } F \text{ is atomic} \\ \mathrm{val}_{\mathcal{S}}^{k,\ell}(G) + \mathrm{val}_{\mathcal{S}}^{k,\ell}(H) & \text{if } F = (G + H) \\ \mathrm{val}_{\mathcal{S}}^{k,m}(G) \cdot \mathrm{val}_{\mathcal{S}}^{m,\ell}(H) & \text{if } F = (G \cdot H) \text{ and } G \in \mathbb{T}_{\mathcal{S}}^{k,m} \\ \mathrm{val}_{\mathcal{S}}^{k/m,\ell/n}(G) \otimes \mathrm{val}_{\mathcal{S}}^{m,n}(H) & \text{if } F = (G \otimes H) \text{ and } H \in \mathbb{T}_{\mathcal{S}}^{m,n}. \end{cases}$$

That is, we associate with each tensor formula F of order $k \times \ell$ its $k \times \ell$ matrix “value” in the natural way.

The partial trace evaluation problem is defined as follows:

Definition 6. Let $\mathcal{S}$ be a semiring. The partial trace evaluation problem means to determine the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ for a given tensor formula F over $\mathcal{S}$ of order $n \times 1$ and a natural number k, which is a power of two and is written in binary.

5. From gate arrays to sum-free tensor formulas and back

In this section we show how to encode gate arrays into specific tensor formulas over an appropriate semiring, and conversely, how to compute from a particular type of tensor formula F a gate array which will later be used as a mean to solve a partial trace instance built from F. In particular, we are interested in sum-free tensor formulas obeying some further easy properties.

Definition 7. A tensor formula F is *sum-free* if and only if none of F and its sub-formulas has the form $G + H$, for tensor formulas G and H. A tensor formula is *array-like* if and only if all sub-formulas of F evaluate to square matrices or column vectors. Moreover, a array-like tensor formula F is *orthogonal array-like* if and only if all sub-formulas of F evaluate to orthogonal square matrices or column vectors whose ℓ_2-norm equals 1.

We choose the term “orthogonal array-like” because as we will show, such a formula can be reorganized as a product of an orthogonal matrix with a column vector, i.e., as the specification of an orthogonal system of linear equations. Observe, that “sum-free array-like” implies that each sub-formula F of a tensor formula fulfills the following properties: If $F = (G \cdot H)$, then G is a matrix and either H is a matrix or a column vector, and if $F = (G \otimes H)$, either both G and H are matrices or both are column vectors.

In the forthcoming we use the terminology that a gate array is said to be *reversible* if and only if all gates in the gate array can be described by orthogonal matrices. Thus, both quantum and probabilistic gate arrays are reversible gate arrays.

5.1. From gate arrays to sum-free tensor formulas

The construction of a sum-free tensor formula from a given gate array is rather straightforward and is done as follows:

Lemma 8. *Let C be a (reversible) gate array operating on n wires, whose gates can be described by (orthogonal) square matrices over a semiring $\mathcal{S}$. Then there is a polytime computable function, which given a suitable encoding of C, computes a (orthogonal) array-like sum-free tensor formula F_C of order $2^n \times 2^n$ such that for each $x = (x_1, \dots, x_n) \in \{0,1\}^n$, if gate array C maps $|\varphi\rangle = |x_1 \dots x_n\rangle$ to $|\psi\rangle$, then*

$$|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,2^n}(F_C) \cdot |\varphi\rangle,$$

and $|\varphi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_x^{\mathsf{T}})$ for some polytime computable sum-free tensor formula d_x.

Proof. Let C be a m-leveled gate array, where C_i denotes the ith level of C, with C_1 is the left-most and C_m the right-most level. Without loss of generality we assume that each level contains only one gate and moreover each gate acts on neighboring wires. This can be achieved by inserting extra swap gates. In the following we describe how to construct an equivalent tensor formula F_C from C.

If level C_i contains a k-bit gate H with $1 \leqslant k \leqslant n$ acting on the wires j up to $j+k-1$, for $j+k-1 \leqslant n$, then

$$F_{C_i} = \left(I_2^{\otimes j-1} \otimes H \otimes I_2^{\otimes n-j-k+1}\right),$$

is the tensor formula of order $2^n \times 2^n$ which describes the system evolution in the ith time step. Recall, that $A^{\otimes n}$ is a shorthand notation for the n-fold iteration $A \otimes A \otimes \cdots \otimes A$.

To complete the description of the sum-free tensor formula F_C over semiring $\mathcal{S}$ let

$$F_C = F_{C_m} \cdots F_{C_2} \cdot F_{C_1},$$

since according to the usual convention, the input-to-output direction in a gate array is left-to-right, while in its matrix representation, the array's action on its input is given as a product of matrices with a column vector, and is read right-to-left. It is readily verified that for each $x_i \in \{0,1\}$ with $1 \leqslant i \leqslant n$, if C maps $|\varphi\rangle = |x_1 \dots x_n\rangle$ to $|\psi\rangle$, then

$$|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,2^n}(F_C) \cdot |\varphi\rangle,$$

and $|\varphi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_x^{\mathsf{T}})$ for the sum-free tensor formula

$$d_x = e^2_{x_1+1} \otimes e^2_{x_2+1} \otimes \cdots \otimes e^2_{x_n+1}.$$

Since F_C and d_x are polytime constructible from a suitable description of the gate array C and its input, the stated claim follows. □

Although Lemma 8 only applies to input vectors of the form $|x_1 \dots x_n\rangle$, arbitrary input vectors of the form $|\varphi\rangle = \sum_{w \in \{0,1\}^n} \alpha_w |w\rangle$ are appropriately mapped to output vectors due to the linearity of gate array "semantics." Observe, that it is not obvious that all possible vectors $|\varphi\rangle$ obey sum-free tensor formula representations. Nevertheless, input vectors for probabilistic and quantum computations do obey sum-free tensor formula representations, since for a gate array on n wires with m_1 input bits and $2m_2$ ancilla bits, i.e, $n = m_1 + 2m_2$, we find that for a particular input $x = (x_1, \ldots, x_{m_1}) \in \{0,1\}^{m_1}$ the input vector can be described by

$$|\varphi\rangle = \left(\bigotimes_{i=1}^{m_1} |x_i\rangle\right) \otimes \left(\bigotimes_{i=1}^{m_2} \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\right),$$

where $(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ can be explicitly given without summation. Thus, in both cases sum-free tensor formulas exist.

Moreover, the previous lemma is not restricted to gate arrays operating on n wires carrying (qu)bits only. In fact, one can easily generalize the result of the lemma such that it work on gate arrays with multi-valued logic, in the sense that there is a mapping from $\{1, \ldots, n\}$ to the natural numbers, defining the arity of the wires. This approach is even more general than the multi-valued bit approach presented studied in the literature [27], where each wire carries (qu)dits of same dimensionality. This more general model allows us to build gates dealing with, e.g., (qu)bits and (qu)trits simultaneously in a single gate.

5.2. From sum-free tensor formulas to gate arrays

In the formula to gate array part, we must deal with the fact that a sum-free tensor formula may contain matrices of various sizes and vectors at atypical locations. In principle, the latter can be regarded as a non-standard manner of specifying the gate array's input. The matrices of various orders, however, cannot be readily interpreted in terms of gate array computations. For instance, consider the sum-free tensor formula

$$(A \otimes B)(B \otimes A),$$

where A is of order 2×2 and B an order 3×3 matrix. Both Kronecker products independently considered may be realized on a two wire gate array, where the wires carry bits and trits, but $(A \otimes B)(B \otimes A)$ lacks a direct realization on gate arrays. This comes from the fact that the wires of the independently constructed gate arrays do not fit together, i.e., a bit in the first product must become a trit in the second one and vice versa. To overcome situations like the above described one the following solutions may be considered:

(1) We stay with "bit-logic" and thus restrict tensor formulas to suit our needs, i.e., all atomic sub-formulas are matrices whose order is a power of two or column vectors of length 2^k for some $k \geqslant 0$. This explicitly forbids tensor products as the above given ones and thus is the simplest solution to our problem.
(2) Matrices of various orders are allowed, and therefore gate arrays as introduced must be generalized to cope with this new situation. In this way we focus on a multi-valued bit approach [27], where wires carry (qu)dits, i.e., states $|0\rangle,\ |1\rangle,\ \ldots,\ |d-1\rangle$, from a d-dimensional space. In fact, this approach is even more general, since gates may act on wires with various dimensionality. For instance one can design gates acting on (qu)bits and (qu)trits simultaneously. Considering our small example, we find, that the

Kronecker product $(B \otimes A)$ can be turned into $(A \otimes B)$ by pre- and post-multiplying it with appropriate stride permutation matrices, when working in a commutative semiring. These stride permutations act on both wires—the (qu)bits and (qu)trits—simultaneously. Hence, we can come up with a gate array realizing the behaviour of $(A \otimes B)(B \otimes A)$. Nevertheless, for more complex examples like, e.g., $(A \otimes B \otimes C)(D \otimes C)$, where A and B are as above and C is a 5×5, and finally D a 6×6 matrix, further problems face up, since the order of the involved matrices in the Kronecker products may not be equal, but their products are. Here the sub-formula $(A \otimes B \otimes C)$ can be implemented on a gate array with three wires carrying (qu)bits, (qu)trits, and (qu)quints, while $(D \otimes C)$ induces a two-wire gate array, where wires carry (qu)sets and (qu)quints. Again, further restrictions have to be imposed in order to overcome these problems.

(3) Finally, gates may act on various (qu)dits as above, but instead of redefining the gate array's action, the action of the gate is embedded in a higher dimensional space of suitable size, i.e., (qu)dits are embedded in dimensionality 2^k, for suitable $k \geqslant 0$. Technically, this means that we pad our matrices and vectors in order to turn their orders into powers of two, and thus working on gate array wires carrying (qu)bits only. Observe, that the underlying computational model is quite general, since it allows, e.g., (qu)bits and (qu)trits to be processed by a single gate. This is in fact the most general scenario for the multi-valued bit approach, since the wires are "non-homogeneous" w.r.t. their (qu)dits.

Although, we favour the third approach, since the problem solution is the most general one, we first have to deal with the first approach, working with (qu)bits only.

5.2.1. Tensor formulas where all atomic sub-formulas are powers of two

Before we show how to transform a suitable tensor formula into a gate array acting on (qu)bits, we show that the postulated requirements on a given tensor formula as specified in the discussion above, can be verified in deterministic polytime. We omit the proof of following lemma, because it is quite similar to the proof of Lemma 4.

Lemma 9. *Testing whether a string encodes a valid tensor formula and if so* (a) *to check sum-freeness*, (b) *orthogonality on sum-free formulas*, (c) *the array-like property, and* (d) *whether all atomic sub-formula have orders, which are powers of two, can be done in deterministic polytime*. □

Now we are ready to prove the converse relation, i.e., transforming a array-like sum-free tensor into an equivalent gate array, if the formula obeys some additional easily checkable properties.

Theorem 10. *Let F be a* (*orthogonal*) *array-like sum-free tensor formula of order* $2^n \times 1$ *over semiring $\mathcal{S}$, where the orders of all atomic sub-formulas are powers of two. Then there is a polytime computable function, which given the tensor formula F, computes a* (*reversible*) *gate array C_F over $\mathcal{S}$ operating on n wires and an input $|\varphi_F\rangle$* (*with* $\langle \varphi_F | \varphi_F \rangle = 1$), *such that*

$$|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(F),$$

if gate array C_F maps $|\varphi_F\rangle$ to vector $|\psi_F\rangle$, and $|\varphi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_F^{\top})$ for some sum-free tensor formula d_F.

Proof. We prove the following more general statement, where we call a tensor formula F *closed* if F has order $2^n \times 1$, for some $n \geqslant 0$, and *open* if the order equals $2^n \times 2^n$, for some $n \geqslant 1$. Let F be a closed (open, respectively) array-like sum-free tensor formula F over semiring $\mathcal{S}$ having only atomic sub-formulas whose orders are powers of two. Then there is a polytime computable function, which given the tensor formula F, computes a gate array C_F over $\mathcal{S}$ operating on n wires and an (arbitrary, respectively) input $|\varphi_F\rangle$, such that $|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(F)$ ($|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(F) \cdot |\varphi_F\rangle$, respectively) if gate array C_F maps $|\varphi_F\rangle$ to vector $|\psi_F\rangle$, and $|\varphi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_F^{\top})$ for some sum-free tensor formula d_F.

The statement is shown by induction on the (orthogonal) sum-free tensor formula F. If F is an atomic sub-formula, then we distinguish the cases whether F is open or closed:

(1) If F is closed, i.e., is of order $2^k \times 1$, then it specifies the amplitudes for all possible combinations of values of k input bits. Thus, the trivial gate array C_F only consisting of k wires with no gates at all and the sum-free tensor formula $d_F = F$ satisfies [4]

$$|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^k,1}(F),$$

since C_F realizes the identity transformation on $|\varphi\rangle = \mathrm{val}_{\mathcal{S}}^{2^k,1}(d_F^{\top})$; this means $|\psi\rangle$ equals $|\varphi\rangle$.

(2) If F is a matrix of order $2^k \times 2^k$, i.e., formula F is open, then F is interpreted as the specification of a k-bit gate F. Thus, the gate array C_F consists of the single k-bit gate F acting on k wires and the input to the gate is some vector as, e.g., the unit column vector $e_1^{2^k}$ of length 2^k. Trivially, the gate array and the input vector fulfill the requirements above. Observe, that $e_1^{2^k} = (e_1^2)^{\otimes k}$.

Now assume that the statement holds for sub-formulas G and H of the tensor formula F. Thus, by induction hypothesis there are gate arrays C_G and C_H and inputs $|\varphi_G\rangle$ and $|\varphi_H\rangle$, that can be specified by sum-free tensor formulas d_G and d_H, respectively. Then we distinguish two cases:

(1) If $F = (G \cdot H)$, then we combine the sub-arrays C_H and C_G in sequential manner, where C_H is to the left of C_G, and define the input to be $|\varphi_H\rangle$. It is easy to see that C_G and $|\varphi_H\rangle$ fulfill the required properties.

[4] If working on a field $\mathcal{S}$ instead of a semiring, one can show the following result: Let $|\psi\rangle$ be a vector over the field $\mathcal{S}$ of length 2^k obeying $\langle\psi|\psi\rangle = 1$. Then there is a matrix A over an extension field of $\mathcal{S}$, whose first column equals $|\psi\rangle$, which can be decomposed into an orthogonal matrix Q and an upper diagonal matrix R whose upper left element equals 1, and both matrices are over $\mathcal{S}$, satisfying $A = Q \cdot R$. The proof relies on a careful analysis of the Gram-Schmitt [17] algorithm, which inductively computes an orthogonal (orthonormal) basis from any set of linearly independent vectors. Therefore, the orthogonal matrix Q may be interpreted as the specification of a k-bit gate Q. Thus, a gate array consisting of a single Q-gate acting on k wires maps input $|\varphi\rangle = e_1^{2^k} = (e_1^2)^{\otimes k}$ to vector $|\psi\rangle$. Observe, that if $\mathcal{S}$ is a (semi)ring, then a similar statement as that for fields is not true in general.

(2) If $F = (G \otimes H)$, then the sub-arrays are combined in parallel, where C_G is on top of C_H. Thus, the input equals $|\varphi_G\rangle \otimes |\varphi_H\rangle$, which can be described by the sum-free tensor formula $d_G \otimes d_H$. Again, the induction assertion is fulfilled.

This proves the statement. Observe, that one can easily show, that whenever F is an orthogonal array-like sum-free tensor formula, then all gates in the gate array C_F can be specified by orthogonal matrices, and moreover, the input $|\varphi_F\rangle$ obeys $\langle\varphi_F|\varphi_F\rangle = 1$ and has a sum-free tensor description. □

The proof above reveals a significant difference of probabilistic and quantum computation—see the footnote again. In the former case, the ancilla bits must be given well prepared to the gate array, since the gate array can only perform deterministic computations and thus, is not able to prepare them itself. In the latter case, this preparation is not necessary, since the gate array itself is able to prepare them properly. This means, that in the quantum case one can set all ancilla bits to, e.g., $|0\rangle$, without changing the computational power of the underlying device.

5.2.2. Tensor formulas in general

Finally, let us consider arbitrary (orthogonal) array-like tensor formulas, not necessarily obeying the property that all atomic sub-formulas are powers of two. The following lemma, which is very technical and the main ingredient of the transformation to an equivalent gate array, shows how to pad matrices and vectors in order to turn their orders into powers of two.

Lemma 11. *Let F be a (orthogonal) array-like tensor formula F of order $n \times 1$ over semiring $\mathcal{S}$. Then there is a polytime computable function, which given the tensor formula F, computes a (orthogonal) array-like tensor formula G over the same semiring of order $m \times 1$ having only atomic sub-formulas whose orders are power of two, such that $\mathrm{val}_{\mathcal{S}}^{n,1}(F)$ appears in the upper left corner of $\mathrm{val}_{\mathcal{S}}^{m,1}(G)$, i.e.,*

$$\mathrm{val}_{\mathcal{S}}^{m,1}(G) = \begin{pmatrix} \mathrm{val}_{\mathcal{S}}^{n,1}(F) \\ (0)_{n-m}^{\mathsf{T}} \end{pmatrix},$$

where $(0)_k$ denotes the all zero row vector of length k. If F is sum-free, then so is tensor formula G.

Proof. We prove the more general statement, that any (orthogonal) array-like tensor formula F of order $n \times 1$ ($n \times n$, respectively) over semiring $\mathcal{S}$ can be turned into a (orthogonal) array-like tensor formula over the same semiring of order $m \times 1$ ($m \times m$, respectively) having only atomic sub-formulas whose orders are power of two, such that $\mathrm{val}_{\mathcal{S}}^{n,1}(F)$ ($\mathrm{val}_{\mathcal{S}}^{n,n}(F)$, respectively) appears in the upper left corner of the computed tensor formula (and thus is a block diagonal matrix, respectively), when evaluated. In order to simplify representation, we speak of the computed tensor formula as the padded version of F and denote it by $\varepsilon(F)$. Observe, since $\varepsilon(F)$ has only atomic sub-formulas whose orders are power of two, the order of $\varepsilon(F)$ is a power of two, too.

We proceed by induction of F and distinguish three cases:

(1) If F is an atomic formula, then consider two subcases: (1) F is a column vector of order $n \times 1$, then define $\varepsilon(F)$ to be the $\pi(n) \times 1$ column vector with entries from F at the first n positions, and zero otherwise. Here $\pi(n)$ denotes the smallest power of two greater than or equal to n. (2) F is an order $n \times n$ matrix, then set $\varepsilon(F)$ to be the $\pi(n) \times \pi(n)$ block-diagonal matrix consisting in a copy of F at the upper left corner and the identity matrix $I_{\pi(n)-n}$ at the lower right. Obviously, in both cases $\varepsilon(F)$ satisfies the above given requirements. Moreover, formula $\varepsilon(F)$ is sum-free.

(2) If $F = (G \cdot H)$, then we argue as follows: Since F is array-like, tensor formula G does not evaluate to a column vector. Hence, by induction we may assume that $\varepsilon(G)$ is of order $2^k \times 2^k$, for some k. Then we consider two subcases: (1) If H is a column vector, then by induction hypothesis $\varepsilon(H)$ is of order $2^\ell \times 1$, for some ℓ, and we define the sum-free tensor formula

$$\varepsilon(F) = \begin{cases} (I_2^{\otimes \ell-k} \otimes \varepsilon(G)) \cdot \varepsilon(H) & \text{if } k < \ell, \\ \varepsilon(G) \cdot \varepsilon(H) & \text{if } k = \ell, \text{ and} \\ \varepsilon(G) \cdot \Big(((e_1^2)^{\otimes k-\ell})^\mathsf{T} \otimes \varepsilon(H) \Big) & \text{if } k > \ell. \end{cases}$$

(2) If H is a matrix, then the smaller matrix must undergo some padding as in the previous case. Since the construction is very similar as above, the details are left to the reader. Easy calculations show that $\varepsilon(F)$ is indeed a (orthogonal) array-like tensor formula, having only atomic sub-formulas whose orders are power of two. As in the previous case, $\varepsilon(F)$ is sum-free, whenever F is sum-free.

(3) If $F = (G \otimes H)$, then we distinguish two cases and argue as follows:

(a) First, assume that G and H are of order $m \times 1$ and $n \times 1$, respectively. Then by induction hypothesis, assume that $\varepsilon(G)$ ($\varepsilon(H)$, respectively) is a array-like sum-free tensor formula of order $2^k \times 1$ ($2^\ell \times 1$, respectively) for some k (ℓ, respectively). To improve readability of the proof, let $\mu = 2^k$ and $v = 2^\ell$. Moreover, let

$$\begin{aligned} \varepsilon(G) &= (\, g_1 \quad \dots \quad g_m \quad g_{m+1} \quad \dots \quad g_\mu \,)^\mathsf{T}, \\ \varepsilon(H) &= (\, h_1 \quad \dots \quad h_n \quad a_{n+1} \quad \dots \quad h_v \,)^\mathsf{T} \end{aligned}$$

and

$$\varepsilon(G) \otimes \varepsilon(H) = (\, g_1h_1 \quad g_1h_2 \quad \dots \quad g_\mu h_v \,)^\mathsf{T}.$$

In order to build $\varepsilon(G \otimes H)$ from $\varepsilon(G) \otimes \varepsilon(H)$ we proceed in two steps: (i) First we pre-multiply $\varepsilon(G) \otimes \varepsilon(H)$ by the stride permutation matrix $P_v^{\mu v}$, which results in

$$P_v^{\mu v} \cdot (\varepsilon(G) \otimes \varepsilon(H)) = (\, g_1h_1 \quad g_2h_1 \quad \dots \quad g_\mu h_1 \quad g_1h_2 \quad \dots \quad g_\mu h_v \,)^\mathsf{T},$$

and then (ii) we pre-multiply with a block-diagonal matrix $H_{n,\mu,v}$, whose two blocks are the stride permutation $P_\mu^{\mu n}$ (top left) and some permutation of $\mu(v-n)$ objects (bottom right). Thus, $H_{n,\mu,v}$ read as

$$H_{n,\mu,v} = \begin{pmatrix} P_\mu^{\mu n} & 0 \\ 0 & R \end{pmatrix},$$

where R is some permutation matrix of order $\mu(\nu - n) \times \mu(\nu - n)$, which will be specified later. Thus, we obtain

$$\begin{aligned} &H_{n,\mu,\nu} \cdot (g_1h_1 \quad g_2h_1 \quad \ldots \quad g_\mu h_1 \quad g_1h_2 \quad \ldots \quad g_\mu h_\nu)^\mathsf{T} \\ &\quad = \begin{pmatrix} P_\mu^{\mu n} & 0 \\ 0 & R \end{pmatrix} \cdot (g_1h_1 \quad g_2h_1 \quad \ldots \quad g_\mu h_1 \quad g_1h_2 \quad \ldots \quad g_\mu h_\nu)^\mathsf{T} \\ &\quad = (g_1h_1 \quad g_1h_2 \quad \ldots \quad g_\mu h_n \quad \ldots)^\mathsf{T}, \end{aligned}$$

where the former part of μn elements equals $G \otimes H$ and the remaining $\mu(\nu - n)$ elements are some permutation of the g_ih_j's, for $1 \leqslant i \leqslant \mu$ and $n + 1 \leqslant j \leqslant \nu$.
This completes our construction and defines

$$\varepsilon(G \otimes H) = \left(H_{n,\mu,\nu} \cdot P_\nu^{\mu\nu}\right) \cdot (\varepsilon(G) \otimes \varepsilon(H)).$$

(b) If G and H evaluate to square matrices of order $m \times m$ and $n \times n$, respectively, then it is easy to verify that pre-multiplying $\varepsilon(G) \otimes \varepsilon(H)$ by $H_{n,\mu,\nu} \cdot P_\nu^{\mu\nu}$ reorders the rows and that we can do the same job with the columns by post-multiplying with $\left(H_{n,\mu,\nu} \cdot P_\nu^{\mu\nu}\right)^\mathsf{T}$, where μ and ν are defined as in the previous case. Therefore, $\varepsilon(G \otimes H)$ is defined as

$$\varepsilon(G \otimes H) = \left(H_{n,\mu,\nu} \cdot P_\nu^{\mu\nu}\right) \cdot (\varepsilon(G) \otimes \varepsilon(H)) \cdot \left(H_{n,\mu,\nu} \cdot P_\nu^{\mu\nu}\right)^\mathsf{T}.$$

Observe, that the key to these definitions is that we have some freedom in defining $\varepsilon(G \otimes H)$, i.e., in fact, choosing it among many candidates, and that we will build $H_{n,\mu,\nu}$ as if it were the padded version of a smaller matrix. Before we concentrate on $H_{n,\mu,\nu}$, we prove that matrix $P_\nu^{\mu\nu}$ obeys a sum-free tensor formula description. By the stride permutation identities

$$P_{mn}^{\ell mn} = P_m^{\ell mn} \cdot P_n^{\ell mn} \quad \text{and} \quad P_n^{\ell mn} = (P_n^{\ell n} \otimes I_m) \cdot (I_\ell \otimes P_n^{mn}),$$

which can be found in [33], one immediately observes, that

$$P_\nu^{\mu\nu} = \left(P_2^{\mu\nu}\right)^\ell$$

and since $\mu\nu$ is a power of two, we find

$$P_2^{\mu\nu} = \left(P_2^{\mu\nu/2} \otimes I_2\right) \cdot \left(I_{\mu\nu/4} \otimes P_2^4\right),$$

if $\mu\nu \geqslant 4$. This yields a array-like sum-free tensor formula for the matrix $P_\nu^{\mu\nu}$ with atomic sub-formulas I_2 and P_2^4. Now we are ready to concentrate on $H_{n,\mu,\nu}$. We distinguish two cases, depending on G and H:

(a) If both G and H are atomic formulas, then we work ad hoc setting $\mu = \pi(m)$, $\nu = \pi(n)$, and define

$$H_{n,\mu,\nu} = \begin{pmatrix} P_\mu^{\mu n} & 0 \\ 0 & I_{\mu(\nu-n)} \end{pmatrix},$$

where $\pi(n)$ denotes the smallest power of two greater than or equal to n.

(b) Otherwise, we use our freedom to choose the lower right block in $H_{n,\mu,v}$ to write it as the kth power of some padded matrix $\varepsilon(P_2^{\mu n})$ of order $\mu v \times \mu v$, i.e.,

$$H_{n,\mu,v} = \left(\varepsilon(P_2^{\mu n})\right)^k ,$$

since $P_\mu^{\mu n} = (P_2^{\mu n})^k$ by the stride permutation identities.

In the forthcoming, we use the following facts for defining padded matrices: If A and B are square matrices having the same size, then we can set $\varepsilon(A \cdot B) = \varepsilon(A) \cdot \varepsilon(B)$, and if B is its own padded version, i.e., $B = \varepsilon(B)$, then we can set $\varepsilon(A \otimes B) = \varepsilon(A) \otimes B$.

Next by the stride permutation identities again, and our freedom to choose the padding, we write

$$\varepsilon(P_2^{\mu n}) = \varepsilon(P_2^{2n} \otimes I_{\mu/2}) \cdot \varepsilon(I_n \otimes P_2^{\mu}),$$

which simplifies to

$$I = \left(\varepsilon(P_2^{2n}) \otimes I_{\mu/2}\right) \cdot \left(\varepsilon(I_n) \otimes P_2^{\mu}\right).$$

For $\varepsilon(I_n)$ the obvious choice is $\varepsilon(I_n) = I_v$. Note, that $I_v = I_2^{\otimes \ell}$.

The sum-free tensor formula of order $2v \times 2v$ for $\varepsilon(P_2^{2n})$ is obtained as follows: Observe, that n can be expressed in polytime as $n = n_1 \cdot n_2 \ldots n_s$, where each n_i is the row or column dimension of some atomic sub-formula of F. Expressibility of n in this way can be readily verified by induction on the tensor formula F. Thus, we can write $n = s \cdot t$, where t is a row or column dimension of a atomic sub-formula. Now by the stride permutation identities we can set

$$\varepsilon(P_2^{2n}) = \varepsilon(P_2^{2s} \otimes I_t) \cdot \varepsilon(I_s \otimes P_2^{2t}),$$

since the unpadded matrices involved in the multiplication must be of same size in order to be compatible. Let $v = \sigma\tau$ with $\tau = \varepsilon(t)$. Dealing with the above factors separately yields

$$\varepsilon(P_2^{2s} \otimes I_t) = \left(H_{t,2\sigma,\tau} \cdot P_\tau^{2\sigma\tau}\right) \cdot \left(\varepsilon(P_2^{2s}) \otimes \varepsilon(I_t)\right) \cdot \left(H_{t,2\sigma,\tau} \cdot P_\tau^{2\sigma\tau}\right)^{\mathsf{T}}$$

by the general padding process; choosing $\varepsilon(I_t) = I_\tau$, which implies that $\varepsilon(P_2^{2s})$ must be of order $2\sigma \times 2\sigma$, results in a sum-free tensor formula for the first factor. Meanwhile, the second factor reads as

$$\varepsilon(I_s \otimes P_2^{2t}) = \left(H_{2t,\sigma,2\tau} \cdot P_{\sigma\tau}^{2\sigma\tau}\right) \cdot \left(\varepsilon(I_s) \otimes \varepsilon(P_2^{2t})\right) \cdot \left(H_{2t,\sigma,2\tau} \cdot P_{\sigma\tau}^{2\sigma\tau}\right)^{\mathsf{T}},$$

where we set $\varepsilon(I_s) = I_\sigma$, which implies that $\varepsilon(P_2^{2t})$ is of order $2\tau \times 2\tau$.

This completes the description of the recursive construction for $H_{n,\mu,v}$, which can be done in polynomial time, and shows that the matrix under consideration has a sum-free tensor formula implementation.

Thus, we have shown the stated claim on the transformation of an arbitrary (orthogonal) array-like formula F to a (orthogonal) array-like formula having only atomic sub-formulas whose orders are power of two such that the value of F appears in the upper left corner of the constructed tensor formula. □

As an immediate consequence of Theorem 10 and Lemma 11 we obtain the following theorem, which is the main result of this subsection.

Theorem 12. *Let F be a* (*orthogonal*) *array-like sum-free tensor formula F of order* $n \times 1$ *over semiring* $\mathcal{S}$. *Then there is a polytime computable function*, *which given the tensor formula F*, *computes a* (*reversible*) *gate array* C_F *over* $\mathcal{S}$ *operating on m wires*, *with* $n \leqslant 2^m$, *and an input* $|\varphi_F\rangle$ (*with* $\langle\varphi_F|\varphi_F\rangle = 1$), *such that*

$$|\psi_F\rangle = \begin{pmatrix} \mathrm{val}_{\mathcal{S}}^{n,1}(F) \\ (0)_{2^m-n}{}^{\mathsf{T}} \end{pmatrix},$$

if gate array C_F *maps* $|\varphi_F\rangle$ *to vector* $|\psi_F\rangle$, *and* $|\varphi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^m,1}(d_F^{\mathsf{T}})$ *for some sum-free tensor formula* d_F.

6. Complexity results for sum-free tensor formulas

In this section we prove completeness results on variants of the partial trace problem for orthogonal array-like sum-free tensor formulas. These variants are defined as follows:

Definition 13. Let $\mathcal{S}$ be a semiring.

(1) The *one partial trace problem* over semiring $\mathcal{S}$ is the set of all tensor formulas F of order $n \times 1$ together with a natural number k, which is a power of two, given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ equals 1.
(2) The *non-zero partial trace problem* over semiring $\mathcal{S}$ is the set of all tensor formulas F of order $n \times 1$ together with a natural number k, which is a power of two, given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ is non-zero.

In order to obtain our completeness results we have to deal with promise versions of the above defined problems. Moreover, we also have to introduce promise complexity classes.

Observe, that PP is a "syntactic" class, since acceptance is defined by simply counting the number of accepting paths, while BPP is a "semantic" class, i.e., for a non-deterministic machine to define a language in BPP, it must have the property that for all inputs one of the two outcomes has a clear majority. This property is not obvious how to check. Thus, it is convenient to introduce the notion of promise problems and promise complexity classes [12,31]. A *promise problem* is a formulation of a partial decision problem and can be specified in the form "$R(x)$ given the promise $Q(x)$?," where Q and R are predicates. That is, on input x, an algorithm solving a promise problem (Q, R) has to correctly decide property $R(x)$, if the promise $Q(x)$ is met; otherwise, it can give an arbitrary answer. More formally, a language L is said to be a solution to (Q, R), whenever $x \in Q$ implies that $x \in R$ if and only if $x \in L$. In particular, set R is the unique solution to (Σ^*, R). Thus, the promise problem

(Σ^*, R) is identified with the set R. Now we are ready to extend the class BPP as follows: a promise problem (Q, R) belongs to pr-BPP if and only if there is a non-deterministic Turing machine, such that if $x \in Q$, then either the number of accepting or rejecting paths has clear majority, and for $x \in Q$, the word x is in R if and only if the Turing machine accepts with clear majority. Observe, that (Σ^*, L) is in pr-BPP if and only if L is in BPP. We can similarly define the generalized class pr-BQP in terms of promise problems. Finally, our reductions are polytime many-one reductions, and we say that a promise problem (Q, R) is uniformly many-one reducible in polytime to a promise problem (S, T), if there exists a partial polytime computable function $f : \{x \in \Sigma^* \mid Q(x)\} \to \Sigma^*$, such that for every solution A of (S, T), the set B defined by $B(x) = A(f(x))$ is a solution of (Q, R).

The promise version of the one partial trace and non-zero partial trace problem restricted to tensor formulas of order $n \times 1$ such that the kth partial trace of the matrix $\mathrm{val}_S^{n,n}(F \cdot F^\top)$ evaluates to either 0 or 1, will be called the 0–1-*promise* in the forthcoming. Moreover, we refer to the promise classes associated with P and EQP, respectively, as pr-P and pr-EQP, respectively. Then the following first main theorem of this section reads as follows.

Theorem 14. (1) *The* 0–1-*promise version of the one partial trace problem over the positive rationals* $\mathbb{Q}^+$ (*rationals* $\mathbb{Q}$, *respectively*), *restricted to the domain of orthogonal array-like sum-free tensor formulas*, *is complete for* pr-P (pr-EQP, *respectively*) *under polytime many-one reductions*.

(2) *The non-zero partial problem over the positive rationals* $\mathbb{Q}^+$ (*rationals* $\mathbb{Q}$, *respectively*), *restricted to the domain of orthogonal array-like sum-free tensor formulas, is complete for* NP (NQP, *respectively*) *under polytime many-one reductions*.

Proof. We only prove the first statement, since the second one can be shown by similar arguments. The hardness of the 0–1-promise one partial trace problem on orthogonal array-like sum-free tensor formulas is shown by a generic reduction from (pr-EQP, respectively). Using Theorem 2, we start with a *m*-level reversible gate array *C* over the positive rationals working on *n* wires number from 1 to *n*, whose accepting subspace is defined by setting the first bit to $|1\rangle$. Now using Lemma 8 we build from *C* an equivalent tensor formula F_C in polytime. Meanwhile we define for the gate array's input bits x_1 up to x_n a tensor product $d_x = \bigotimes_{i=1}^n e_{1+x_i}^2$ of order 1×2^n of *n* unit row vectors e_i^2 each of order 1×2. By Lemma 8 it is easy to see that the first 2^{n-1} entries along the diagonal of

$$\mathrm{val}_{\mathbb{Q}^+}^{2^n,2^n}((F_C \cdot d_x^\top) \cdot (F_C \cdot d_x^\top))$$

add up to the value of $f_C(x)$, which is the probability that the gate array's output is projected onto the accepting subspace. The original gate array's input is accepted if and only if this partial trace is exactly one, by which acceptance by *C* is defined. Scrutiny of the reduction shows that the constraint on $f_C(x)$ is transported intact from the description of *C* and x to the partial trace orthogonal array-like sum-free tensor formula problem over $\mathbb{Q}^+$ instance $F_C \cdot d_x^\top$.

In the other direction, we use Lemma 11 and Theorem 10 to translate an instance $\langle F, 2^k\rangle$ of the partial trace problem variant under consideration into the description of a reversible gate array C_F over *m* bits, where $m \leqslant n$, if the order of F equals $2^n \times 1$, and of its input

$|\varphi\rangle$; the 2^kth partial trace of

$$\mathrm{val}_{\mathbb{Q}^+}^{2^m,2^m}(F \cdot F^\top)$$

represents the probability that the output bits of the gate array C_F be projected onto the direct sum of the dimension-1 subspaces generated by $|0\rangle = |0\ldots00\rangle$, $|1\rangle = |0\ldots01\rangle$, $|2\rangle = |0\ldots10\rangle, \ldots,$ and $|2^k-1\rangle$. The promise on the partial trace is transported unmodified from the input tensor formula to the reversible gate array. $\square$

In the remainder of this section we define meaningful problems, which capture PP, pr-BPP, and pr-BQP. Recall, that a tensor formula F is called orthogonal if and only if' all sub-formulas of F evaluate to orthogonal square matrices or vectors whose ℓ_2-norm equals 1.

Definition 15. Let $\mathcal{S}$ be either the commutative semiring of positive rationals $\mathbb{Q}^+$ or the field of rationals $\mathbb{Q}$. The *majority partial trace problem* over semiring $\mathcal{S}$ is the set of all orthogonal tensor formulas F of order $n \times 1$ together with a natural number k given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^\top)$ is superior to $\frac{1}{2}$.

As already mentioned, the classes BPP and BQP are defined *via* a semantic condition. Thus, we need a promise version of the majority partial trace problem, which captures the semantic condition in order to obtain completeness result. Let $\mathcal{S}$ be either $\mathbb{Q}^+$ or $\mathbb{Q}$. Restricting the majority partial trace problem to orthogonal tensor formulas of order $n \times 1$ such that the (partial) trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^\top)$, belongs to the interval $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, is called the *strict majority partial trace problem*. Now we are ready to prove the following theorem.

Theorem 16. (1) *The majority partial trace problem over both, the positive rationals $\mathbb{Q}^+$ and rationals $\mathbb{Q}$ in general, restricted to the domain of orthogonal array-like sum-free tensor formulas is complete for* PP *under polytime many-one reductions*.

(2) *The strict majority partial trace problem over the positive rationals $\mathbb{Q}^+$* (*rationals $\mathbb{Q}$, respectively*), *restricted to the domain of orthogonal array-like sum-free tensor formulas is complete for* pr-BPP (pr-BQP, *respectively*) *under polytime many-one reductions*.

Proof. We only prove the second statement. For the first statement, observe, that PP equals its quantum counterpart as discussed after Theorem 2. The proof of the second assertion parallels that of Theorem 14. Hardness follows from Theorem 2 and Lemma 8, while containment is shown with Theorem 10, and the fact, that the promise on the partial trace problem is transported unmodified from the input tensor formula to the reversible gate array. $\square$

We summarize our results on variants of the partial trace problem over the positive rationals or rationals in general in Table 3. It is worth mentioning that both the one partial trace and non-zero partial trace problems over the Boolean semiring $\mathbb{B}$, restricted to the domain of orthogonal array-like sum-free tensor formulas, is complete for P under polytime

Table 3
Completeness results for the sum-free case summarized

Semiring	One PTP with 0–1-promise	Non-zero PTP	Majority PTP	Strict maj. PTP
Partial trace problem (PTP) with appropriate restricted domain				
$\mathbb{B}$	P	P	–	–
$\mathbb{Q}^+$	pr-P	NP	PP	pr-BPP
$\mathbb{Q}$	pr-EQP	NQP	PP	pr-BQP

many-one reductions. Observe, that the 0–1-promise is ridiculous in the case of Booleans $\mathbb{B}$, since by definition the promise condition is met.

7. Complexity results for tensor formulas in general

We discuss the partial trace evaluation problem and its variants for tensor formulas in general. In this way, we obtain completeness results for complexity classes like, e.g., $\oplus$P, NP, $\mathrm{C_{=}P}$, and US, in more detail—a formal definition of these classes is given below. The material presented here is not essential for the further understanding of probabilistic and quantum computation, and therefore may safely be skipped by those not interested in this aspect. A direct application of Lemma 17 will be the construction of complete problems for the above mentioned classes.

To understand the statement of Lemma 17 below, keep in mind a situation in which it is required to determine the trace of say $A \oplus B \oplus C \oplus D$, where A, B, C, and D are $\ell n \times \ell n$ matrices and $\oplus : \mathbb{M}_{\mathcal{S}}^{k,k} \times \mathbb{M}_{\mathcal{S}}^{k,k} \to \mathbb{M}_{\mathcal{S}}^{2k,2k}$ is the direct sum of matrices and is defined as

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$$

for $A, B \in \mathbb{M}_{\mathcal{S}}^{k,k}$, which generalizes to an arbitrary number of matrices in the domain. Lemma 17 describes a preliminary step which uses tensors to produce a large block matrix having $n \times n$ sub-matrices compatible to A, B, C, and D on its main diagonal, i.e., the new matrix has exactly the same diagonal elements as $A \oplus B \oplus C \oplus D$, but in permuted order. This particular application of Lemma 17 would require the parameter $m = 4$.

Lemma 17. *Let a sequence* $\mathbf{A} = (A_i)$, *with* $1 \leqslant i \leqslant m$, *of* $\ell n \times \ell n$ *matrices over a semiring* $\mathcal{S}$ *be given. Consider the* $\ell mn \times \ell mn$ *matrix* $A = \bigoplus_{i=1}^{m} A_i$. *Then there is a polytime Turing machine which computes on input* $\mathbf{A}$, *a tensor formula* $F_{m,\mathbf{A}}$ *evaluating to the* $\ell mn \times \ell mn$ *matrix* $B = \bigoplus_{i=1}^{\ell m} B_i$, *where* $\mathbf{B} = (B_i)$ *with* $1 \leqslant i \leqslant \ell m$, *of* $n \times n$ *matrices satisfying*

$$\mathrm{diag}(B) = (P_{\ell}^{\ell mn})^{-1} \cdot \mathrm{diag}(A) \cdot P_{\ell}^{\ell mn}.$$

Here $\mathrm{diag}(A)$ *denotes the matrix, which consists of the diagonal entries of A and is zero elsewhere. In other words, matrix B has the exactly the same diagonal elements as matrix A, but in permuted order.*

Proof. First, we show how the diagonal elements of an arbitrary square matrix A and $P^{-1} \cdot A \cdot P$, for a permutation matrix P such that the corresponding multiplication operation is defined, are related to each other. Observe, that $P^{-1} = P^{\mathsf{T}}$ since P is a permutation matrix. Assume that $(P^{-1})_{i,j} = (P)_{j,i} = 1$ for some $1 \leqslant i, j \leqslant n$. Then it is easily seen that

$$\left(P^{-1} \cdot A \cdot P\right)_{i,i} = \sum_{j=1}^{n} \sum_{k=1}^{n} \left(P^{-1}\right)_{i,j} \cdot (A)_{j,k} \cdot (P)_{k,i} = (A)_{j,j} \,.$$

Thus, $P^{-1} \cdot \operatorname{diag}(A) \cdot P = \operatorname{diag}(P^{-1} \cdot A \cdot P)$ follows.

Next consider $A = \bigoplus_{i=1}^{m} A_i$. To simplify presentation, let us call a matrix of the above given form a m-uniform block diagonal matrix. Now consider the tensor formula $(P_\ell^{\ell mn})^{-1} \cdot A \cdot (P_\ell^{\ell mn})$, which results in a matrix

$$\begin{pmatrix} B_{1,1} & \dots & B_{1,\ell} \\ \vdots & \ddots & \vdots \\ B_{\ell,1} & \dots & B_{\ell,\ell} \end{pmatrix},$$

where the $B_{i,j}$'s are m-uniform block diagonal matrices of order $mn \times mn$. This is because pre- and post-multiplying A by $(P_\ell^{\ell mn})^{-1}$ and $P_\ell^{\ell mn}$, respectively, rearranges the rows and column in ℓ stride fashion. Thus, from every sub-matrix A_i exactly n rows and n columns are taken to form a single stride. Therefore the $B_{i,j}$'s are m-uniform block diagonal matrices of appropriate order.

Then

$$F_{m,\mathbf{A}} = \sum_{i=1}^{\ell} \left(D_i^\ell \otimes I_{mn}\right) \cdot \begin{pmatrix} B_{1,1} & \dots & B_{1,\ell} \\ \vdots & \ddots & \vdots \\ B_{\ell,1} & \dots & B_{\ell,\ell} \end{pmatrix} \cdot \left(D_i^\ell \otimes I_{mn}\right)^{\mathsf{T}}$$

equals the ℓm-uniform block diagonal matrix $\bigoplus_{i=1}^{\ell} B_{i,i}$ of order $\ell mn \times \ell mn$, where D_i^n is the order $n \times n$ "dot matrix" having one in position (i, i) and zero elsewhere. Moreover, by our previous investigation on the effect of permutation matrices to the diagonal elements of a matrix we immediately conclude that

$$\operatorname{diag}(B) = (P_\ell^{\ell mn})^{-1} \cdot \operatorname{diag}(A) \cdot P_\ell^{\ell mn}$$

holds, where $B = \operatorname{val}_{\mathcal{S}}^{\ell mn, \ell mn}(F_{m,\mathbf{A}})$.

Finally, one observes, that the tensor formula $F_{m,\mathbf{A}}$ is polytime constructible from the given input using the identities on stride permutations and observing that $I_{mn} = I_m \otimes I_n$. This proves this statement. $\square$

As an application we construct complete problems for the above mentioned complexity classes, which are defined as follows: The corresponding counting version of NP is denoted by #P and is the class of functions f, such that there is a machine M with the same resources as the underlying base class, such that $f(x)$ equals the number of accepting computations of M on x. The decision class NP is defined on *Boolean* computation models, in that they rely on the mere *existence* of accepting computations. If existence is replaced by the predicate “there is an *odd number of* accepting computations,” we obtain the parity version [30] $\oplus$P and the class MOD$_q$-P, which is similarly defined with respect to counting modulo q. More formally, $\oplus$P is the class of sets of type $\{\, x \mid f(x) \neq 0 \pmod 2 \,\}$ for some $f \in$ #P. Moreover, consider the complexity class [13] *Gap* P $= \{\, f - g \mid f, g \in \#\mathrm{P} \,\}$ as a natural generalization of #P. Additionally, we will make use of some further classes, namely consider the chains

$$\text{co-NP} \subseteq \text{US} \subseteq \text{C}_{=}\text{P} \quad \text{and} \quad \text{UP} \subseteq \text{SPP} = \text{co-SPP} \subseteq \text{C}_{=}\text{P} \cap \text{co-C}_{=}\text{P}.$$

Here US is the class of sets of type $\{\, x \mid f(x) = 1 \,\}$ for some $f \in$ #P and is called *unique* polytime [8]. Moreover, UP denotes Valiant’s class [34], which is the set of all languages whose characteristic function belongs to #P and SPP is a generalization of UP and is defined to be the set of languages whose characteristic function belongs to *Gap* P, hence is the difference of two #P functions [13,28]. Observe, that SPP $\subseteq$ MOD$_q$-P, for any q; thus, in particular SPP $\subseteq \oplus$P. Recall that C$_=$P is the class of sets of type $\{\, x \mid f(x) = g(x) \,\}$, for some $f, g \in$ #P. Finally, the promise counterparts of the classes UP and SPP, respectively, are intuitively defined and denoted by pr-UP and pr-SPP, respectively.

Before we state the main theorem of this section, we need the following result, which can be deduced from Damm et al. [9], and Beaudry and Holzer [6], respectively.

Theorem 18. *The one problem on scalar tensor formulas over semiring $\mathcal{S}$ is the set of all tensor formulas of order* 1×1 *for which* $\mathrm{val}_{\mathcal{S}}^{1,1}(F) = 1$. *The one problem for scalar tensor formulas is complete for* NP, $\oplus$P, US, *and* C$_=$P, *respectively, with respect to polytime many-one reductions in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively*.

Now we are ready to state our first main theorem.

Theorem 19. (1) *The one partial trace problem is complete for* NP, $\oplus$P, US, and C$_=$P, *respectively, with respect to polytime many-one reductions in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively*.

(2) *The non-zero partial trace problem is complete for* NP, $\oplus$P, NP, *and the class* co-C$_=$P, *respectively, with respect to polytime many-one reductions in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively*.

Proof. We only prove the first statement, since the second one can be shown by similar arguments. The containment of the one partial trace problem immediately follows from Lemma 17 and the following reasoning. Let F be an $n \times 1$ order instance of the one partial trace problem. Then $F \cdot F^{\top}$ is a tensor formula again, and can be reduced in sequence to a diagonal tensor formula G whose matrix $\mathrm{val}_{\mathcal{S}}^{n,n}(G)$ satisfies the condition that the diagonal elements are exactly those of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\top})$—even in the same order. To this end, we use Lemma 17 several times, together with appropriately constructed stride permutation matrices in order to keep the sequence of the diagonal elements. Finally, we reduce G to a scalar tensor formula, i.e., a tensor formula of order 1×1, by pre- and post-multiplying G

by f_k^n and its transpose, respectively, giving

$$\left(f_k^n\right) \cdot G \cdot \left(f_k^n\right)^{\mathsf{T}}$$

as output, where f_k^n denotes the row vector of length n whose first k entries equals 1 and is 0 elsewhere. Since the vector f_k^n may be of exponential length we use a similar technique as in the construction of unit vectors e_k^n presented by Damm et al. [9]. The main idea is that n can be expressed in polytime as $n = m_1 \cdot m_2 \dots m_t$, where each m_{r} is the row dimension of some atomic sub-formula of G. Expressibility of n in this way is readily verified by induction on G. But then, vector f_k^n can be expressed as a finite sum in polytime. This shows that the one partial trace problem polytime many-one reduces to a scalar tensor formula. Thus, containment in NP, ⊕P, US, and C$_=$P, respectively, immediately follows in case of Booleans $\mathbb{B}$, the field $\mathbb{F}_2$, the naturals $\mathbb{N}$, and the integers $\mathbb{Z}$, respectively, by Theorem 18.

For the hardness part we argue as follows: Again by Theorem 18 the classes NP, ⊕P, US, and C$_=$P, respectively, reduce to a scalar tensor formula over the Booleans $\mathbb{B}$, the field $\mathbb{F}_2$, the naturals $\mathbb{N}$, and the integers C$_=$P, respectively, such that w is in L if and only if the scalar tensor formula F evaluates to 1 (in each semiring under consideration). Deeming F to be an instance of the one partial trace problem together with the natural number 1 shows hardness in all considered cases—in case of integers we additionally need the closure of C$_=$P under union, which was shown by Gundermann et al. [20]. This completes this proof. □

The reader can verify that the above given proof can be rewritten in terms of the 0–1-promise version. Meanwhile the complexity of the one partial trace and the non-zero partial trace problem is obtained with a straightforward application of the above given argument. Thus, we state the below given corollary without proof. Observe, that the 0–1-promise is ridiculous in the case of Booleans $\mathbb{B}$ and the field $\mathbb{F}_2$, since by definition the promise condition is met.

Corollary 20. *The* 0–1-*promise versions of the below mentioned problems are complete w.r.t. polytime many-one reductions: Both the one partial trace problem and the non-zero partial trace problem are complete for* NP, ⊕P, pr-UP, *and* pr-SPP, *respectively, in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively*.

We summarize our results on the computational complexity of the one partial trace and non-zero trace problem and their promise versions in Table 4.

8. Conclusions

Through the study of gate arrays, we have developed a common algebraic description for polytime complexity classes, where the choice of the semiring (plus a possible promise) determines the complexity class. In this way, characterizations of (pr-)P, NP, ⊕P, pr-BPP and their quantum counterparts pr-EQP, NQP, and pr-BQP are obtained. In particular, for the inclusion BPP $\subseteq$ BQP, the classical model of polytime probabilistic computation turns

Table 4
Completeness results for the general case summarized

Semiring	0–1-promise		Unrestricted	
	one PTP	non-zero PTP	one PTP	non-zero PTP
Partial trace problem (*PTP*)				
$\mathbb{B}$	NP	NP	NP	NP
$\mathbb{F}_2$	$\oplus$P	$\oplus$P	$\oplus$P	$\oplus$P
$\mathbb{N}$	pr-UP	pr-UP	US	NP
$\mathbb{Z}$	pr-SPP	pr-SPP	$C_=P$	co-$C_=P$ = NQP

out to be a special case of polytime quantum computation where interference between computations is ruled out.

The definitions of variants of the partial trace problems on (sum-free) tensor formulas allowed us to obtain complete problems for the above mentioned polytime complexity classes in a very natural way. Moreover, by giving up sum-freeness, classes like $\oplus$P, NP, $C_=P$, its complement co-$C_=P$ = NQP, the promise version of Valiant's class UP, its generalization promise SPP, and unique polytime US, were captured. It would be interesting, to see whether extending our work to other semirings would yield characterizations for further complexity classes.

Acknowledgements

We wish to thank Lance Fortnow and John Watrous for helpful discussion and pointers to useful references. Insightful comments by Gilles Brassard, Michele Mosca, and Pierre McKenzie were also appreciated.

References

[1] L.M. Adleman, H. DeMarrais, M.-D.A. Huang, Quantum computability, SIAM J. Comput. 26 (5) (1997) 1524–1540.
[2] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity I, EATCS Monographs on Theoretical Computer Science, Vol. 11, Springer, Berlin, 1988.
[3] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. A 52 (1995) 3457–3467.
[4] M. Beaudry, J.M. Fernandez, M. Holzer, Deterministic and probabilistic computations from a quantum perspective, 2003, in preparation.
[5] M. Beaudry, J.M. Fernandez, M. Holzer, A common algebraic description for probabilistic and quantum computation, in: J. Fiala, V. Koubek, J. Kratochvil (Eds.), Proc. of the 29th Conference on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Vol. 3153, Springer, Berlin, Prague, Czech Republic, August 2004, pp. 851–862.
[6] M. Beaudry, M. Holzer, The complexity of tensor circuit evaluation, in: J. Sgall, A. Pultr, P. Kolman (Eds.), Proc. of the 26th Conference on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Vol. 2136, Springer, Berlin, Mariánské Lázně, Czech Republic, August 2001, pp. 173–185.
[7] C.H. Bennett, Time/space trade-offs for reversible computation, SIAM J. Comput. 18 (4) (1989) 766–776.
[8] A. Blass, Y. Gurevich, On the unique satisfiability problem, Inform. and Control 82 (1–3) (1982) 80–88.
[9] C. Damm, M. Holzer, P. McKenzie, The complexity of tensor calculus, Comput. Complexity 11 (1/2) (2003) 54–89.

[10] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, Proc. Roy. Soc. London Ser. A 400 (1985) 97–117.
[11] D.P. DiVincenzo, Two-bit gates are universal for quantum computation, Phys. Rev. A 51 (1995) 1015–1022.
[12] S. Even, A.L. Selman, Y. Yacobi, The complexity of promise problems with applications to public-key cryptography, Inform. and Control 61 (1984) 159–173.
[13] S.A. Fenner, L. Fortnow, S.A. Kurtz, Gap-definable counting classes, J. Comput. System Sci. 48 (1) (1994) 116–148.
[14] S. Fenner, F. Green, S. Homer, R. Pruim, Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy, Proc. Roy. Soc. London Ser. A 455 (1999) 3953–3966.
[15] L. Fortnow, One complexity theorist's view of quantum computing, Theoret. Comput. Sci. 292 (3) (2003) 597–610.
[16] J.S. Golan, The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science, Pitman Monographs Surveys in Pure and Applied Mathematics, Longman Scientific & Technical, 1992.
[17] G.H. Golub, C.F. Van Loan, Matrix Computations, Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.
[18] A. Graham, Kronecker Products and Matrix Calculus With Applications, Mathematics and its Applications, Ellis Horwood, Chichester, UK, 1981.
[19] J. Gruska, Quantum Computing, Advanced Topics in Computer Science, McGraw-Hill, New York, NY, 1999.
[20] Th. Gundermann, N.A. Nasser, G. Wechsung, A survey on counting classes, in: Proc. of the fifth Annual Structure in Complexity Theory Conference, Barcelona, Spain, July 1990, IEEE, New York, pp. 140–153.
[21] H.V. Henderson, F. Pukelsheim, S.R. Searle, On the history of the Kronecker product, Linear and Multilinear Algebra 14 (1983) 113–120.
[22] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, K.W. Wagner, On the power of polynomial time bit-reductions, in: Proc. of the eighth Annual Structure in Complexity Theory Conference, San Diego, California, May 1993, IEEE Computer Society Press, Silver Spring, MD, pp. 200–207.
[23] W. Kuich, A. Salomaa, Semirings, Automata, Languages, EATCS Monographs on Theoretical Computer Science, Vol. 5, Springer, Berlin, 1986.
[24] W. Ledermann, On singular pencils of Zehfuss, compound, Schäflian matrices, Proc. Roy. Soc. Edinburgh LVI (1936) 50–89.
[25] S. Lloyd, Almost any quantum logic gate is universal, Phys. Rev. Lett. 75 (1995) 346–349.
[26] D.M. Barrington, D. Thérien, Finite monoids and the fine structure of NC^1, J. ACM 35 (4) (1998) 941–952.
[27] A. Muthukrishnan, C.R. Stroud Jr., Multi-valued logic gates for quantum computation, Phys. Rev. Ser. A 62 (2000) 052309-1–8.
[28] M. Ogiwara, L. Hemachandra, A complexity theory of closure properties, J. Comput. Sys. Sci. 46 (3) (1993) 295–325.
[29] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[30] C.H. Papadimitriou, S. Zachos, Two remarks on the power of counting, in: A.B. Cremers, H.-P. Kriegel (Eds.), Proc. of the sixth GI Conference on Theoretical Computer Science, Lecture Notes in Computer Science, Vol. 145, Springer, Berlin, Dortmund, Germany, January 1983, pp. 269–275.
[31] A.L. Selman, Promise problems complete for complexity classes, Inform. Comput. 78 (1988) 87–98.
[32] T. Sleator, H. Weinfurter, Realizable universal quantum logic gates, Phys. Rev. Lett. 74 (1995) 4087–4090.
[33] R. Tolimieri, M. An, Ch. Lu, Algorithms for Discrete Fourier Transform and Convolution, Springer, Berlin, 1997.
[34] L.G. Valiant, The complexity of computing the permanent, Theoret. Comput. Sci. 8 (1979) 189–201.
[35] K.W. Wagner, Some observations on the connection between counting and recursion, Theoret. Comput. Sci. 47 (1986) 131–147.
[36] A.C.-C. Yao, Quantum circuit complexity, in: Proc. of the 34th Symposium on Foundations of Computer Science, Palo Alto, California, November 1993, IEEE, New York, pp. 352–361.