Revista Brasileira de Ensino de Física, vol. 43, suppl. 1, e20200403 (2021)
www.scielo.br/rbef
DOI: https://doi.org/10.1590/1806-9126-RBEF-2020-0403

Bosons vs. Fermions – A computational complexity perspective

Bósons e Férmions sob a perspectiva da complexidade computacional

Daniel Jost Brod[*1]

[1]Universidade Federal Fluminense, Instituto de Física, Niterói, RJ, Brasil.

Received on September 22, 2020. Revised on November 10, 2020. Accepted on November 17, 2020.

Recent years have seen a flurry of activity in the fields of quantum computing and quantum complexity theory, which aim to understand the computational capabilities of quantum systems by applying the toolbox of computational complexity theory.

This paper explores the conceptually rich and technologically useful connection between the dynamics of free quantum particles and complexity theory. I review results on the computational power of two simple quantum systems, built out of noninteracting bosons (linear optics) or noninteracting fermions. These rudimentary quantum computers display radically different capabilities—while free fermions are easy to simulate on a classical computer, and therefore devoid of nontrivial computational power, a free-boson computer can perform tasks expected to be classically intractable.

To build the argument for these results, I introduce concepts from computational complexity theory. I describe some complexity classes, starting with **P** and **NP** and building up to the less common #**P** and polynomial hierarchy, and the relations between them. I identify how probabilities in free-bosonic and free-fermionic systems fit within this classification, which then underpins their difference in computational power.

This paper is aimed at graduate or advanced undergraduate students with a Physics background, hopefully serving as a soft introduction to this exciting and highly evolving field.

Keywords: Quantum computing, quantum complexity theory, linear optics, free fermions.

Os últimos anos presenciaram uma atividade crescente nas áreas de computação quântica e teoria de complexidade quântica. Essas áreas têm como objetivo analisar a complexidade e as capacidades computacionais de diversos sistemas quânticos através do uso das ferramentas desenvolvidas, ao longo das últimas décadas, no contexto da teoria de complexidade computacional.

Este artigo explora as conexões conceitualmente ricas e tecnologicamente úteis entre a dinâmica de partículas quânticas livres e a teoria da complexidade. Eu reviso resultados sobre a complexidade computacional da simulação de dois sistemas quânticos simples, construídos com bósons não interagentes (ou seja, óptica linear), ou com férmions não interagentes. Tais sistemas podem ser vistos como tipos rudimentares de computadores quânticos, com potencialidades radicalmente distintas: por um lado, sabe-se que férmions livres são facilmente simulados por um computador clássico e, portanto, desprovidos de poder computacional não trivial; por outro lado, existe forte evidência de que um computador construído a partir de bósons livres pode realizar uma tarefa que é classicamente intratável (ou seja, é capaz de demonstrar vantagem ou supremacia quântica).

A fim de construir os argumentos que fundamentam esses resultados, este artigo oferece uma introdução básica a alguns conceitos do campo da teoria da complexidade computacional. Para isso, eu descrevo algumas classes de complexidade e as relações entre elas, partindo das bem conhecidas **P** e **NP**, e avançando para as classes #**P** e a hierarquia polinomial. Identifico então como as probabilidades de transição nos sistemas de bósons livres e de férmions livres se encaixam nessa classificação, o que fundamenta a diferença em seu poder computacional.

Este artigo foi pensado para estudantes no fim da graduação e na pós-graduação, com formação em Física. Espero que ele sirva como uma introdução leve a este campo fascinante e em rápido desenvolvimento.

Palavras-chave: Computação quâtica, teoria de complexidade quântica, óptica linear, férmions livres.

1. Introduction

Computational complexity theory and quantum mechanics lie at the hearts of two major fields of human study. Computational complexity theory [1] is the subfield of computer science that aims to understand the fundamental nature of computational problems by classifying them according to e.g. how long they take to solve. Its central open question is whether the ability to efficiently *check* that a solution to a problem is correct implies that a solution is easy to find (the **P** vs. **NP** question). This is one of the most important questions in mathematics, and one of the seven Millennium Prize Problems [2]. The classes **P** and **NP**, though

[*] Correspondence email address: danieljostbrod@id.uff.br

particularly famous, are just two among a rich landscape of complexity classes [3] which lay the foundation for our understanding of the notions of computability and computational efficiency. In essence, computational complexity theory aims to understand the limits of what computers can do, a worthwhile goal given how ubiquitous computers are in our daily life and society.

Quantum mechanics, on the other hand, forms the basis for most of our understanding of the physical world, the standard model of fundamental particles. Despite that, some aspects of it remain not fully understood—even though its mathematical previsions are trusted and sound, there is no consensus on how to interpret them, or how classical mechanics ultimately emerges as its limit. In order to understand this transition better, quantum mechanics has been progressively tested in the limits of large masses [4], large distances [5] and so on.

From the intersection of these two fields arose the subfield of quantum computing, and its more abstract cousin, quantum complexity theory [6, 7]. Quantum mechanics predicts that quantum computers will be able to solve some computational problems (e.g. factoring large integers) very efficiently [8], whereas complexity theory gives us evidence that these problems cannot be solved efficiently by classical computers [1]. The combination of these two facts has launched the field of quantum computing into rapid growth over the last few decades, and more recently shifted it from a purely academic to an industrial endeavor [9].

However, the field of quantum computing is still subject to an amount of skepticism, mostly (but not exclusively!) from the side of some computer scientists [10, 11]. Prior to the proposal of quantum computers, a central tenet of complexity theory was the Extended Church-Turing thesis, which states (informally) that all problems efficiently solvable by any realistic model of computation are efficiently solvable by a Turing machine (i.e. by classical computers). This is a useful principle, since it allows us to build a theory of computation that is abstract, and in a sense robust against different definitions of computational models. It will also, of course, be threatened by the construction of a full-purpose large-scale quantum computer. Quantum computing skeptics often argue that, despite all the progress made so far, unavoidable levels of experimental noise will defeat any practical attempt at a large-scale quantum computer.

This skepticism has motivated the concept of quantum computational advantage, or quantum supremacy [12, 13]. This is an approach where the usefulness or applications of quantum computers are temporarily left aside, in favor of constructing the simplest quantum experiment capable of performing some computational task much faster than any classical computer. In a sense, it is the exploration of yet another frontier of quantum mechanics—rather than large masses or distances, it is the frontier of large *computational complexity*. This has seen impressive recent progress, culminating in an experiment on a 54-qubit quantum computer that allegedly performs a task tens of thousands times faster than the best classical supercomputer [9].

The main approach in the paradigm of quantum computational advantage is to consider a *restricted* quantum system, e.g. one that has some limitations in its allowed dynamics which might make it weaker than a full-fledged quantum computer, but still stronger than a classical one, in some sense, and easier to implement in near-future experiments. One of the first examples of such a restricted system was linear optics. A simple experiment, consisting of preparation of single photons, followed by a sequence of beam splitters and culminating in a round of photon number detections, was shown [14] to sample from a probability distribution that cannot be reproduced efficiently by a classical computer (modulo some complexity-theoretic conjectures which we will describe in due time). Interestingly, the fermionic analogue of this system, where "beam splitters" are replaced by Hamiltonians quadratic in the fermionic operators (such as hopping Hamiltonians on a lattice), does not have the same complexity—free fermion dynamics is easy to simulate on a classical computer [15, 16].

In this paper, I review the main results that separate free-boson and free-fermion dynamics (both of which I will refer to as linear optics for simplicity) from a computational complexity theory point of view. It is aimed at graduate or advanced undergraduate students with a Physics background, but no familiarity with computational complexity theory is assumed.

In section 2, I review the formalism of second quantization. I suggest that even readers familiar with the formalism take a quick glance at this section, as it defines the assumptions, the physical system and the dynamics we consider, as well as some notation. In section 3, I give a basic introduction to computational complexity theory and some of the main complexity classes. It is not comprehensive, as I focus only on a cross-section of the theory that I use in the rest of the paper, but it is introductory.

Section 4 contains the main results I wish to review. In section 4.1, I define formally what it means, for our purposes, to simulate a quantum system. This definition, together with the structure of complexity classes described previously, culminate in section 4.2, where I discuss the classical simulability of free fermions, and section 4.3, where I outline the evidence that (bosonic) linear optics is not classically simulable. Sections 4.2 to 4.3 are the most technical sections of the paper, and can also be of interest to more advanced readers entering the field of quantum computing. In section 4.4 we return to a more leisurely speed, and I discuss the paradigm of quantum computational advantage, or quantum supremacy, reviewing some of its most recent advances. Finally, in section 4.5, I move briefly beyond linear optics to give some other examples where computational complexity theory and quantum mechanics have interacted, in particular in the context of computing ground states of (interacting) bosonic and fermionic Hamiltonians, and the well-known sign problem from condensed matter and quantum field theory.

Notation: Throughout this paper, I represent by $\{0,1\}^n$ the set of all binary strings of length n. I assume that any mathematical object of interest (an integer, a matrix, a graph) can be encoded in some binary string (see [1], chapter 0, for a careful discussion). If real numbers are involved, we can assume they have been truncated to some desired precision. A Boolean function $f(x)$ is a function that takes as input a binary string x and outputs a single bit. I denote as poly(n) any quantity which is a polynomial in n but whose details are unimportant. We refer to the number of elements of a set S as $|S|$.

Here I also assume that $\hbar = 1$ for simplicity. Though $\hbar$ is an important fundamental constant that sets the scales for many quantum phenomena, from the point of view of computational complexity it is just a constant that does not change e.g. the asymptotic behavior of algorithmic runtimes.

2. The formalism of second quantization

Let us begin with a quick review of the formalism used to describe identical quantum particles. This review will serve also to fix notation and situate the readers of various backgrounds on details of the systems I will consider here and the underlying assumptions. For an in-depth discussion of this formalism, see e.g. [17].

In this paper, I will describe bosonic or fermionic systems in the language of second quantization. We are interested only in the particles' bosonic or fermionic nature, and none of our conclusions depend on other properties such as mass, charge, spin, and so on. Our discussion will be valid for any particle for which the corresponding dynamics, to be described shortly, can be realized.

2.1. The states

Consider a set of creation operators, $\{a_i^\dagger\}_{i=1\ldots m}$, labeled by some discrete index i. We use the letter a for particle operators that can be either bosonic or fermionic. When the situation requires distinction between them, we replace a by b, for bosons, or f, for fermions. Borrowing terminology from quantum optics, we refer to labels i as *modes*. Physically, these modes could represent any degree of freedom of the particles, such as polarization, wave vector, or time-of-arrival at a detector, for photons, spin or momentum, for an electron, and so on. They could *a priori* be discrete or continuous but, for the applications we have in mind, we assume that they are discrete.[1]

[1] If we refer to a degree of freedom that is continuous, such as direction of propagation, the reader can assume that we are using some suitable coarse-grained discretization of it.

All characteristic bosonic or fermionic behaviors, of which we will see some examples shortly, follow from the canonical commutation relations, given by

$$\begin{aligned}
[b_i, b_j^\dagger] &= \delta_{ij}, \\
[b_i, b_j] &= 0 = [b_i^\dagger, b_j^\dagger], \\
\{f_i, f_j^\dagger\} &= \delta_{ij}, \\
\{f_i, f_j\} &= 0 = \{f_i^\dagger, f_j^\dagger\}.
\end{aligned}$$

Suppose now that we have a single mode. From the creation operators we can construct the *single-mode* Fock states. For bosons, these states are constructed as

$$(b^\dagger)^n |0\rangle = \sqrt{n!}\,|n\rangle,$$

whereas for fermions we have

$$(f^\dagger)\,|0\rangle = |1\rangle.$$

In both cases $|0\rangle$ is the corresponding vacuum state, defined by the fact that $a\,|0\rangle = 0$. The bosonic Fock states satisfy

$$b^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle, \tag{1a}$$
$$b\,|n\rangle = \sqrt{n}\,|n\rangle, \tag{1b}$$

while for fermionic ones we have

$$f^\dagger |n\rangle = \delta_{n,0}\,|n+1\rangle, \tag{2a}$$
$$f\,|n\rangle = \delta_{n,1}\,|n\rangle. \tag{2b}$$

In both cases, we have that

$$a^\dagger a\,|n\rangle = n\,|n\rangle.$$

From the above equations it is clear why $a^\dagger$, a and $n := a^\dagger a$ are referred to as creation, annihilation and number operators, respectively.

When we have m modes in total, these equations are replaced by a natural generalization. If again we define the vacuum state as the unique state for which $a_k\,|0\rangle = 0$ for all k, the bosonic Fock states are constructed as

$$\prod_{i=1}^{m} (b_i^\dagger)^{n_i} |0\rangle = \left(\prod_{i=1}^{m} n_i!\right)^{\frac{1}{2}} |N\rangle,$$

where N is a shorthand for $(n_1, n_2 \ldots n_m)$. Generalizing equations (1) to more than one bosonic mode, we have

$$\begin{aligned}
b_i^\dagger\,|N\rangle &= \sqrt{n_i+1}\,|n_1, \ldots, n_i+1, \ldots, n_m\rangle, \\
b_i^\dagger\,|N\rangle &= \sqrt{n_i}\,|n_1, \ldots, n_i-1, \ldots, n_m\rangle.
\end{aligned}$$

For fermions, the Fock states are constructed as

$$\left(f_1^\dagger\right)^{n_1} \left(f_2^\dagger\right)^{n_2} \ldots \left(f_m^\dagger\right)^{n_m} |0\rangle = |N\rangle.$$

Similarly, we can generalize equations (2) for more than one fermionic mode:

$$f_i^\dagger \left|N\right\rangle = \delta_{n_i,0}(-1)^{\sum_{j<i} n_j} \left|n_1, \dots, n_i + 1, \dots, n_m\right\rangle,$$
$$f_i \left|N\right\rangle = \delta_{n_i,1}(-1)^{\sum_{j<i} n_j} \left|n_1, \dots, n_i - 1, \dots, n_m\right\rangle.$$

In both cases,

$$a_i^\dagger a_i \left|N\right\rangle = n_i \left|N\right\rangle.$$

We refer to the Fock space as the space spanned by all Fock states on these m modes.

For the benefit of the reader that is more familiar with the language of first quantization, let us give a brief example that connects these two formalisms. Suppose we have particles that can have one of two values for some quantum number. These can represent horizontal or vertical polarizations of a photon, up or down Z components of electron spins, and so on, but let us label them 0 and 1 for simplicity. In a first-quantized notation, if we have a single of these particles, we would represent the two corresponding states as $\left|0\right\rangle_1$ and $\left|1\right\rangle_1$ (where the label 1 indicates this state is written in first quantization).

Suppose now we have two of these particles, and they are in a state where each assumes a different value for this quantum number. If the particles are distinguishable, this could correspond to either state $\left|01\right\rangle_1$ or $\left|10\right\rangle_1$, or in fact any superposition of them. However, if the particles are bosons (fermions), then this state must be symmetric (antisymmetric) under particle exchange, so we would write it as:

$$\frac{1}{\sqrt{2}}\left(\left|01\right\rangle_1 + \left|10\right\rangle_1\right), \text{ for bosons, and}$$
$$\frac{1}{\sqrt{2}}\left(\left|01\right\rangle_1 - \left|10\right\rangle_1\right), \text{ for fermions.}$$

In the second quantization formalism we reviewed, states are identified instead by the occupation number of each mode. In this example, the modes are identified with the quantum numbers $\{0, 1\}$. *Both* states are written simply as $\left|11\right\rangle$, and the information about the *symmetry* of the wave function is encoded in the corresponding bosonic or fermionic commutation relations.

2.1.1. Fock space dimension

What are the dimensions of the Fock spaces described so far? In principle, for fermions the dimension is 2^m, while for bosons it is infinite, as there can be arbitrarily many bosons in even a single mode. However, the dynamics we consider here always preserves the total number of particles, and so we can restrict ourselves to the subspaces of fixed total particle number[2].

Let us denote the set of all configurations of n fermions in m modes as $F_{m,n}$, and its bosonic analogue as $B_{m,n}$. The size of $F_{m,n}$, which we denote $|F_{m,n}|$, just corresponds to the number of permutations of m objects, out of which n are identical (the occupied modes), and the remaining $m - n$ are also identical (the empty modes). Therefore

$$|F_{m,n}| = \frac{m!}{n!(m-n)!} = \binom{m}{n}.$$

For bosons, we need to count all ways to drop n identical balls in m bins, and so

$$|B_{m,n}| = \frac{(m+n-1)!}{n!(m-1)!} = \binom{m+n-1}{n}.$$

For reasons that will become clear in due time, it is sometimes convenient to consider a restriction on the bosonic Fock space to only no-collision states, i.e., those in which $n_i \leq 1$. We denote this space as $B^*_{m,n}$, and note that $|B^*_{m,n}| = |F_{m,n}|$.

Unless n is very small (or, for fermions, if either n or $m - n$ is small), both $B_{m,n}$ and $F_{m,n}$ grow exponentially in dimension as we increase the number of particles and modes. This is identified as one reason why quantum systems are hard to simulate on classical computers, and why quantum computers might be better at this task [7, 18]. However, it cannot be the *only* reason, since a system of n classical particles in m bins also has an exponentially large configuration space (even larger than $B_{m,n}$ if we attach labels to the classical particles, such that they are distinguishable!). In order to see complexity arise, let us move to the matter of *dynamics*.

2.2. Bosonic and fermionic linear optics

In principle, quantum mechanics allows for arbitrary unitary transformations in Fock space. However, we are interested in the computational complexity of identical particles in a much more restricted settings, that of *linear optics*.

Suppose we have m modes associated with either bosonic or fermionic particles. A linear-optical transformation will be defined by an $m \times m$ matrix U such that the evolution of the particle operators in the Heisenberg representation is given by

$$(a_i^\dagger)_{\text{out}} = \sum_{j=1}^{m} U_{ij} (a_j^\dagger)_{\text{in}}, \quad i = 1, 2, \dots, m. \tag{3}$$

Transformations such as equation (3) are also often referred to as Bogoliubov transformations, or free-particle or noninteracting-particle evolutions. We will

[2] Some readers might object to this, given that second quantization was developed precisely to deal with states with varying particle numbers. Though that might be historically accurate, this does not mean that this formalism cannot prove valuable in other circumstances. Hopefully this paper showcases an example where second quantization is the appropriate description even in the case of fixed particle number, as most of what we discuss would be considerably more complex to write in first quantization.

restrict ourselves to discrete-time transformations, in terms of incoming and outgoing operators, rather than a continuous-time evolution. This is standard practice in the fields of quantum information and quantum optics,[3] and can be done without loss of generality, since every unitary matrix can be written as the complex exponential of some Hamiltonian. The Hamiltonian which generates an arbitrary linear-optical evolution can be written as

$$H = \sum_{ij} h_{ij} a_i a_j^\dagger, \tag{4}$$

where h is an $m \times m$ Hermitian matrix. These Hamiltonians can represent, for example, beam splitters or phase shifters in linear optics, or hopping terms between sites for particles in a lattice.

Let us prove that this Hamiltonian gives rise to an evolution as in equation (3) in the bosonic case (the fermionic case is analogous, and we leave the proof for the interested reader). To that end, consider the Heisenberg equation for a set of (time-dependent) operators $b_i^\dagger(t)$, such that $(b_j^\dagger)_{\text{in}} := b_i^\dagger(0)$:

$$\frac{d}{dt} b_i^\dagger(t) = \mathrm{i}[H, b_i^\dagger(t)]$$

By using the bosonic commutation relations we can write this as

$$\frac{d}{dt} b_i^\dagger(t) = \mathrm{i} \sum_j h_{ij} b_j^\dagger(t)$$

Since h is Hermitian, the solution to this set of coupled first-order differential equations is simply

$$b_i^\dagger(t) = \sum_j U_{ij} b_j^\dagger(0)$$

where $U = \exp(\mathrm{i}ht) \in SU(m)$. Since t is just a parameter, we can set it to 1 such that $(b_j^\dagger)_{\text{out}} := b_i^\dagger(1)$. This shows we can generate the whole of $SU(m)$ by tuning h in equation (4).

Note that U is **not** an unitary evolution operator acting on the Fock space, it is simply a matrix that describes the linear evolution of the creation operators as per equation (3). In particular, while the Fock spaces for n particles in m modes are exponentially large, U has only roughly m^2 free parameters. This shows that considering only linear-optical dynamics is a very restrictive setting. Furthermore, these transformations cover only an exponentially small corner of the space they inhabit, which creates a tension with the argument, from the previous section, of hardness of simulation based on the *dimensionality* of the Fock space.

As we will see, the complexity of these systems is dictated neither by the Fock space being too large nor by the operator space of linear optics being too small. In order to understand the actual complexity of these systems, we need to look more closely at how their transition probabilities behave.

[3] It can also be thought of as a discretized version of the standard scattering formalism.

2.2.1. Elementary linear-optical elements

Let us now illustrate the use of the second quantization formalism by analyzing simple two-mode transformations. We consider, as elementary transformations, the *beam splitter* and the *phase shifter* (borrowing nomenclature from linear optics). The phase shifter acts on a single mode and is defined by the following Hamiltonian

$$H_{PS} = \phi a^\dagger a,$$

or correspondingly

$$(a^\dagger)_{\text{out}} = e^{\mathrm{i}\phi}(a^\dagger)_{\text{in}}.$$

Physically, the phase shifter corresponds to one mode where particles gain a phase relative to the others. This can happen, for example, due to a difference in optical length of two paths, for photons, or to a difference in the local magnetic field acting on two distant electron spins. We represent its action as in Figure 1(a).

The beam splitter is defined as

$$H_{BS} = \theta(a_1^\dagger a_2 + a_2^\dagger a_1),$$

with corresponding action on the particle operators

$$\begin{aligned}(a_1^\dagger)_{\text{out}} &= \cos\theta (a_1^\dagger)_{\text{in}} - \mathrm{i}\sin\theta (a_2^\dagger)_{\text{in}} \\ (a_2^\dagger)_{\text{out}} &= -\mathrm{i}\sin\theta (a_1^\dagger)_{\text{in}} + \cos\theta (a_2^\dagger)_{\text{in}}.\end{aligned}$$

Physically it can correspond to a semi-reflective mirror for optical propagation modes, a wave-plate for polarization modes, or a lattice hopping term for massive particles, where in each case the parameter θ regulates the transmission probability. We represent its action as in Figure 1(b).

An arbitrary 2-mode linear-optical transformation can be obtained by alternated application of phase shifters and beam splitters. To see that, recall that a general

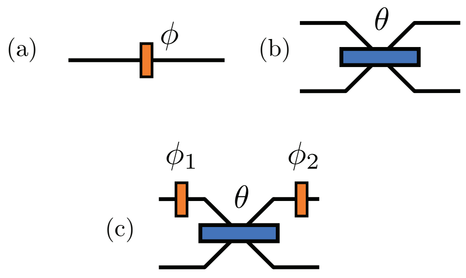

Figure 1: Elementary two-mode linear-optical transformations. (a) A phase shifter. (b) A beam splitter. (c) An arbitrary SU(2) element, with three free parameters.

SU(2) element has three free parameters, and can be always written as [7]:

$$U = \begin{pmatrix} e^{\mathrm{i}(\phi_1+\phi_2)/2}\cos\theta & -\mathrm{i}e^{\mathrm{i}(\phi_2-\phi_1)/2}\sin\theta \\ -\mathrm{i}e^{\mathrm{i}(\phi_1-\phi_2)/2}\sin\theta & e^{\mathrm{i}(-\phi_1-\phi_2)/2}\cos\theta \end{pmatrix}.$$

where $\theta \in [0,\pi]$ and $\phi_1, \phi_2 \in [0, 2\pi]$. This corresponds to the decomposition depicted in Figure 1(c), up to an irrelevant global phase.

Imagine now an experiment where we input particles in such a two-mode transformation, or *interferometer*. Let us write the interferometer matrix generically as

$$U = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

If we input a single particle in mode 1 then, according to equation (3), the output is

$$(\alpha a_1^\dagger + \beta a_2^\dagger)\,|0\rangle$$

and this is independent of whether the particle is a boson or fermion, as there is no distinction at the single-particle level.

Suppose now we input one particle in each arm of this two-mode transformation, i.e., input state $|11\rangle = a_1^\dagger a_2^\dagger\,|0\rangle$. What do we observe at the output?

Consider first the bosonic case. Using the bosonic commutation relations, it follows that the output state is given by

$$(\alpha b_1^\dagger + \beta b_2^\dagger)(\gamma b_1^\dagger + \delta b_2^\dagger)\,|0\rangle = \sqrt{2}\alpha\gamma\,|2,0\rangle + \sqrt{2}\beta\delta\,|0,2\rangle + (\alpha\delta + \beta\gamma)\,|1,1\rangle. \quad (5)$$

For the fermionic case, we have

$$(\alpha f_1^\dagger + \beta f_2^\dagger)(\gamma f_1^\dagger + \delta f_2^\dagger)\,|0\rangle = (\alpha\delta - \beta\gamma)\,|1,1\rangle, \quad (6)$$

Note that the application of the second quantization formalism is very similar in both cases. Only at the very end do we use the commutation or anti-commutation relations, as appropriate, and the expressions differ.

To see how equations (5, 6) lead to inherent bosonic and fermionic behaviors, suppose that the interferometer is a 50:50 beam splitter (i.e. $\theta = \pi/4$):

$$U = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & \mathrm{i} \\ \mathrm{i} & 1 \end{pmatrix} \quad (7)$$

By plugging equation (7) into equation (5), we obtain the output state

$$\frac{\mathrm{i}}{\sqrt{2}}\,|2,0\rangle + \frac{\mathrm{i}}{\sqrt{2}}\,|0,2\rangle,$$

and so the two photons have zero probability of exiting in separate modes. This is known as the Hong-Ou-Mandel effect [19], and is a small-scale manifestation of the natural tendency of bosons to bunch together. Equation (6), on the other hand, tells us that fermions always exit in separate modes, no matter what U is, which of course is due to the Pauli exclusion principle. Figure 2 summarizes the different behaviors on a balanced beam splitter.

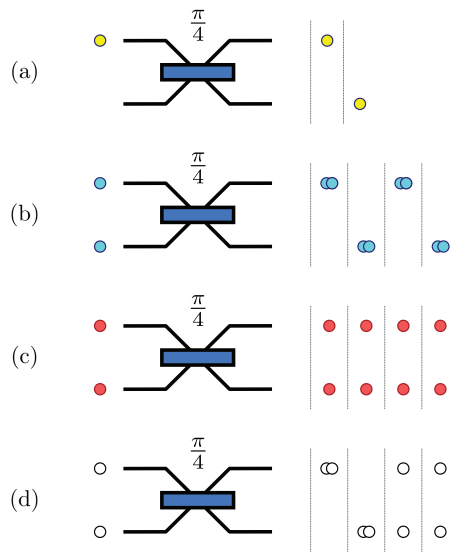

Figure 2: Effect of a 50:50 beam splitter on different particle types. In each diagram, the circles represent particles, which are input on the left of the beamsplitter. At the output, columns represent possible outcomes with their correct relative frequencies. (a) For a single particle (yellow), there is no difference in behavior between bosons and fermions. (b) The Hong-Ou-Mandel effect, where two photons (blue) impinging on a balanced beam splitter always leave it together. (c) The Pauli exclusion principle, which states that two fermions (red) cannot occupy the same mode. (d) For comparison, the behavior of distinguishable particles (white), which exit the beam splitter together or separately with equal probabilities.

Let us look more carefully now at the amplitude of the $|1,1\rangle$ outcomes in equations (5, 6). The combinations $(\alpha\delta+\beta\gamma)$ and $(\alpha\delta-\beta\gamma)$ are well-known matrix functions of U, the *permanent* and the *determinant*, respectively. Let us now see what these expressions look like for scaled-up experiments.

2.2.2. m–ports

We now define the paradigmatic large-scale linear-optical experiment. Suppose that we have a system of n particles in m modes, and they are prepared in some initial state

$$|S\rangle = |s_1, s_2, \ldots, s_m\rangle$$

such that $\sum_i s_i = n$. These particles will be input in an m-mode interferometer U, or m-port, which acts as per equation (3). It was shown by Reck et al. [20] that an arbitrary m-port can be decomposed in terms of $m(m-1)/2$ two-mode transformations, as shown in the main circuit of Figure 3. Finally, we place detectors at all output ports of the interferometer to measure the occupation number of each mode, repeating the

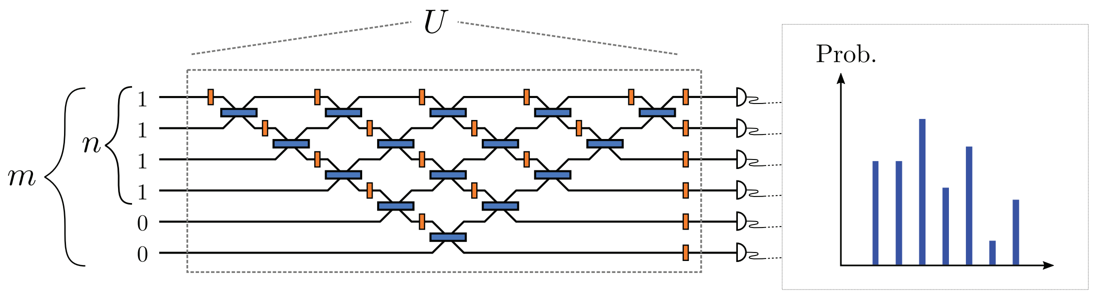

Figure 3: The standard linear-optical setup we consider, composed of (i) Fock state input with n particles (bosons or fermions) in m modes, (ii) arbitrary m-mode linear-optical transformations (which can be decomposed as a circuit of $m(m-1)$ elementary transformations [20]), and (iii) detection of occupation number at all output ports, repeated to build statistics.

experiment many times to build statistics. The full setup is represented in Figure 3.

We now consider the probability of observing an output state $|T\rangle = |t_1, t_2, \ldots, t_m\rangle$. To that end, we need to define a submatrix of U, which we call $U_{S,T}$, by the following procedure. First, we take t_i identical copies of the ith column of U and construct an intermediate $m \times n$ matrix U_T. Then, we take each s_j identical copies of each jth row of U_T to obtain the $n \times n$ matrix $U_{S,T}$. For example, suppose

$$U = \begin{pmatrix} \alpha & \beta & \gamma \\ \delta & \epsilon & \zeta \\ \eta & \theta & \iota \end{pmatrix},$$

and suppose that $|S\rangle = |011\rangle$ and $|T\rangle = |200\rangle$. Then we obtain

$$U_T = \begin{pmatrix} \alpha & \alpha \\ \delta & \delta \\ \eta & \eta \end{pmatrix}, \text{ and } U_{S,T} = \begin{pmatrix} \delta & \delta \\ \eta & \eta \end{pmatrix}$$

The transition probabilities can now be written as the generalizations of equations (5, 6) as follows:

$$\Pr(S \to T) = \frac{|\mathrm{per}(U_{S,T})|^2}{\prod_i s_i! \prod_j t_j!} \qquad \text{for bosons,} \tag{8a}$$

$$\Pr(S \to T) = |\det(U_{S,T})|^2 \qquad \text{for fermions.} \tag{8b}$$

The functions per(A) and det(A) are the permanent and determinant, respectively, given by

$$\det(A) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma_i}, \tag{9a}$$

$$\mathrm{per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma_i}, \tag{9b}$$

where S_n is the set of all permutations of the set $\{1, 2, \ldots, n\}$, and sgn(σ) is the signature of the permutation, equal to $+1$ if the permutation is even and -1 if it is odd. Notice that the only difference in these expressions is the minus signs in the determinant for odd permutations. That can be traced directly to the fermionic anti-commutation relations. From this also follows the Pauli exclusion principle, since any outcome with two fermions in the same mode would correspond to a matrix $U_{S,T}$ with repeated columns, whose determinant is always zero. I invite the interested reader to use equations (9) to verify equations (5, 6).

The connection between the transition probabilities and equations (9) is somewhat intuitive. The determinant and permanents are sums over all possible choices of one matrix element from each distinct row and column. Each term in this sum corresponds to one trajectory that the n input particles might have taken to end up as the n output particles – though, since particles are identical, these possibilities are all summed coherently. For a proof of these expressions, see [14, 21].

For completeness, we should point out that transition probabilities for classical (i.e. distinguishable) particles can also be written similarly. Of course, *a priori* classical particles can have labels, so their configuration space is larger. For particles labeled a and b, state $|ab\rangle$ is different from state $|ba\rangle$. However, imagine we perform an experiment, but forget the particle labels at the output (alternatively, imagine we perform an experiment with particles that are all in mutually orthogonal states of some internal, undetectable degree of freedom). Then the configuration space coincides with that of bosons, and the transition probabilities are given by

$$\Pr(S \to T) = \frac{\mathrm{per}(|U_{S,T}|^2)}{\prod_i s_i! \prod_j t_j!} \text{ for classical particles,} \tag{10}$$

where $|U_{S,T}|^2$ is obtained from $U_{S,T}$ by taking element-wise absolute value squared.

With such similar expressions, one might think that bosonic and fermionic probabilities can be computed with similar computational cost. That is not the case, but to give a full justification we need to take a detour into the territory of computational complexity theory.

3. Computational complexity theory

The distinction between the determinant and the permanent (which will translate into a difference between bosons and fermions) seems, at first glance, a trivial minus sign in some terms of equations (9). Both expressions seem to require computing sums with the same number of very similar terms ($n!$, in fact, for $n \times n$ matrices), which might suggest that they should be equally easy or hard to compute. This impression is misleading–computing the permanent is in fact typically much harder than the determinant. In order to give a robust justification for this claim we need employ the toolbox of *computational complexity theory*, which aims to classify computational problems based on their hardness. This theory is quite extensive and technical, so here we only look at a cross section of it that serves our purposes. We direct the interested reader to the excellent textbook by Arora and Barak [1].

3.1. P and NP

Let us begin our foray into the field of computational complexity theory with an example:

Problem 1. *Suppose you, a famous archaeologist, discovered an ancient tablet of a forgotten civilization. After some work in deciphering the tablet, you realize that it is a geographical document: it contains a list of n cities, together with a list of all pairs of cities which share a common border. Wanting to understand more about this civilization, you decide to figure out: did they inhabit a single island*[4]*?*

How should you proceed?

This problem has two notable features prominent for our discussion. The first is that it is a *decision problem*, i.e., it has a definite yes or no answer. The second is that there exists a procedure to solve this problem which takes a number of steps that grows as a *polynomial* in the size of the problem (n). One simple such procedure, based on a strategy known as *breadth-first search*, is the following:

1. Choose any city in the list, number it 1.
2. Go through the list of borders, identifying the neighbors of city 1. Label them as 2.
3. Repeat the previous step, identifying (yet unlabeled) neighbors of all cities labeled 2. Label them as 3.
4. Repeat the previous step, incrementing the labels, until the procedure stops and no new cities receive labels. If every city receives a label, then they are all in a single island. Any cities left unlabeled are not in the same island as city 1.

We represent this procedure in Figure 4. Clearly it solves the problem, but how long does it take? If there

[4] In the jargon of graph theory, this problem is called GRAPH CONNECTIVITY.

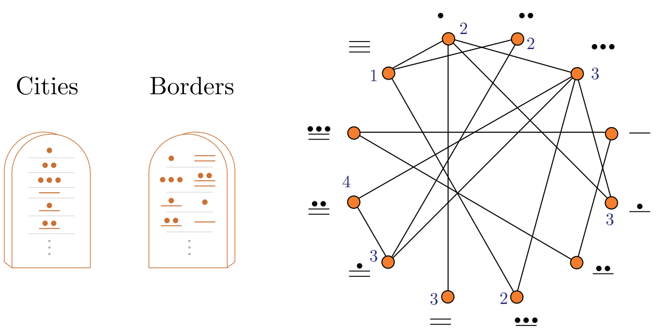

Figure 4: On the left, two ancient tablets, one with a list of cities and the other with a list of borders. To decide whether all cities are in the same island, you represent this data as a graph, on the right, where cities are vertices and edges correspond to shared borders. This does not yield a visual answer to the problem, so you resort to the procedure described in the text. You choose one city and label it as 1, its neighbors as 2, subsequent neighbors as 3, and so on. As the procedure terminates, you realize that three cities have not been labeled, and you conclude they are located on a second island.

are n cities, then the procedure will iterate for at most n steps. In each step, we may need to parse through a list containing at most $n(n-1)/2$ elements, corresponding to the maximum number of pairwise borders that can exist. Therefore, if we were to write an algorithm implementing this procedure in our preferred programming language, its runtime would grow more slowly than n^3[5]. This is definitely not optimal, but crucially it is a polynomial in the size of the problem.

Decision problems for which there exist a polynomial-time classical algorithm can be collected in what is probably the most well-known *complexity class*, named simply **P**. This is identified, informally, as the class of problems that a classical computer can solve *efficiently*. Problems outside of **P** are those for which the best classical algorithm must take time that grows faster than any polynomial (e.g. exponentially).

Definition 1. ***P*** *[informal]: the set of decision problems that can be solved by some classical algorithm that runs in time polynomial in the size of the input.*

Alternatively, the set of decision problems for which there exists some Boolean function $f(x)$ such that:

(i) *x is an encoding of the problem instance in n bits;*
(ii) *if the answer is* YES*, $f(x) = 1$, otherwise $f(x) = 0$; and*
(iii) *$f(x)$ can be computed on a classical computer in time poly(n).*

In this case, the decision problem can be formulated as "given x, is $f(x) = 1$?"

[5] In complexity theory and algorithm analysis this is denoted as $O(n^3)$. Given two functions $f(n)$ and $g(n)$, then $f = O(g)$ means that $f(n)$ is upper bounded by some positive multiple of $g(n)$ in the limit $n \to \infty$. Note that a similar notation is used in the physics literature with a similar spirit but different technical meaning. We refer the reader to chapter 0 of [1].

The alternative definition above, in terms of a Boolean function $f(x)$, may sound a bit opaque for now but will be convenient when we consider generalizations of $\mathbf{P}$. In essence, x simply encodes the question (e.g. it is some encoding of the list of cities and borders in problem 1 as a binary vector), whereas $f(x)$ encodes the answer (YES or NO).

There are objections to this definition of $\mathbf{P}$. In particular, the definition requires the existence of an algorithm that runs in polynomial time for *every* instance of the problem. What if I develop an algorithm that runs in polynomial time for *most* instances of a problem? Or for a subclass of instances relevant for a particular practical application? For problem 1, for example, one can easily devise an algorithm that works much faster than what I described above for the particular case of a map where one city borders all others. Also, what if my algorithm had runtime n^{100}, should we label that as more efficient than an algorithm that has runtime 1.00001^n, just because the first is a polynomial while the second is an exponential?

These are all valid objections. Nonetheless, the definition of $\mathbf{P}$ has historically been very useful, laying the foundations for the structure of complexity classes that has proven so far to be quite robust. From now on we take this and subsequent definitions for granted in order to make meaningful claims about the complexity of simulating quantum mechanical particles, but we direct the interested reader to chapter 1 of [1] for a more thorough discussion.

One complexity class doesn't quite make up a useful classification scheme, so let us now define another, again with an example.

Problem 2. *While studying more ancient tablets, you come across the following story.*

A king ruled the land many years ago. He had two twin children, and it was unclear which of them would inherit the throne. The king thus decided to divide his kingdom between the two. However, he wished to enforce cooperation between his bickering descendants. The king then established the following rule: each town in the kingdom was to be assigned to one of the children, but in such a way that no two neighboring towns could be under the rule of the same person. The overly optimistic monarch reasoned that, in this way, every time one his children wanted to move goods or people between their towns they would be forced to cross territory owned by their sibling, and they would be forced to cooperate.

You strongly doubt the king's reasoning but, in any case, how can you tell whether such a division is possible[6]*? You only possess the list of towns and borders, as in the problem 1.*

This problem is actually quite easy to solve by a minor modification of the algorithm I described for problem 1,

[6] This problem is known as 2-COLORING, or more generally k-COLORING if the king had k children.

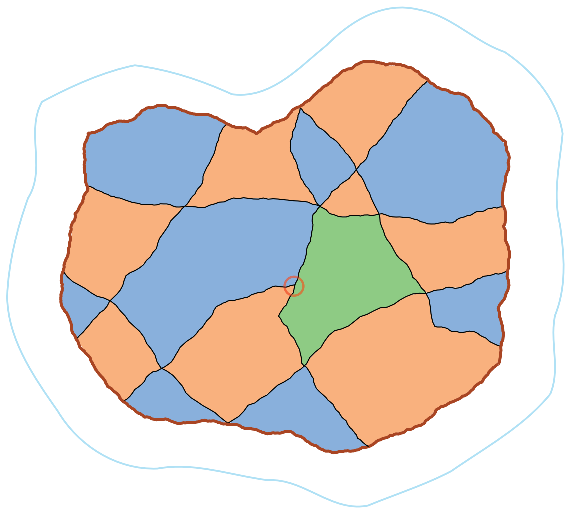

Figure 5: A map of the king's island, which he wishes to divide between his children according to the rules of problem 2. If the king only has two children, this task is impossible, as this map is not 2-colorable. A proof of this fact follows by analyzing the possibilities around the triple border in the center or the map. The map is 3-colorable, so the division *is* possible if the king has three children, as shown above. (If the king evicts everyone from the green region and removes it from the kingdom, the map does become 2-colorable.)

and it is thus also in $\mathbf{P}$. Something interesting happens, however, if we assume that the king had more than just two children. If the king had *four* children, the answer would always be yes. This follows from the four-color theorem, which states simply that any separation of a planar shape in contiguous regions (i.e. a map) can be colored with four colors in such a way that no two adjacent regions have the same color[7].

What happens if the king instead had three children? In this case, there is no known efficient algorithm to decide whether the distribution of regions desired by the king is possible. Curiously, however, *if* such a division of the kingdom is possible, and some court magician used their magic powers to find it, they could easily convince the king of this fact. To that end, they could simply provide the king with a list of the cities assigned to each heir. It would be a much easier task to *check* that the assignment follows the rules imposed by the king than it would be to *find* such an assignment. Figure 5 shows an example.

Problems with the above feature—where a solution can be verified efficiently regardless of whether it can be obtained efficiently–define the second most well-known complexity class: $\mathbf{NP}$.

Definition 2. ***NP*** *[informal]: the set of decision problems with the property that, when the answer is yes, there exists a certificate which can be used to check this fact efficiently (i.e.* verifying *the solution is in* $\boldsymbol{P}$*).*

[7] This is one the first major theorems proven using a computer!

Alternatively, the set of decision problems for which there exists some Boolean function $f(x, y)$ such that:

(i) *x is an encoding of the problem instance in n bits;*
(ii) *y is an m-bit string (known as a* certificate*), with $m = poly(n)$;*
(iii) *$f(x, y)$ can be computed on a classical computer in time $poly(n)$; and*
(iv) *if the answer to the decision problem is* YES, there exists *y such that $f(x, y) = 1$. Conversely, if the answer is* NO *there is no y such that $f(x, y) = 1$.*

In this case, the decision problem can be formulated as "given x, does there exist y such that $f(x, y) = 1$?"

There are countless known problems in **NP**. The version of problem 2 where the king has three children, known as 3-COLORING, is one example. Others including finding the prime factors of some integer (FACTORING), the traveling salesman problem, and many optimization problems. So far the definition of **NP** may seem a bit loose, simply collecting a variety of unrelated problems. However, it is made much more robust by the notion of **NP**–completeness.

An **NP**–complete problem is an **NP** problem with the property that every other **NP** problem *reduces* to it. We will see concrete examples of reductions in the next section (involving the determinant and the permanent) but, for now, it suffices to say that a reduction from problem A to problem B is a way of *efficiently* converting an instance of A to an instance of B. If an efficient algorithm for B exists, this then implies that A can be solved efficiently as well, simply by mapping it to B and using the algorithm for B as a subroutine. In other words, a reduction from A to B implies that B has to be *at least as hard as A*. Consequently, **NP**–complete problems are the hardest among all **NP** problems, since an efficient algorithm for one of them could be used as an efficient algorithm for any other.

The definition makes **NP**–complete problems sound very useful! One can wonder, then, whether they exist at all—and, if so, are they common? The first question was answered in one of the foundational results of computational complexity theory, known as the Cook-Levin theorem [22, 23], which essentially states that **NP**–complete problems do indeed exist, and provides an example (a problem known as Boolean satisfiability problem, or SAT). Soon after, Karp published 21 additional **NP**–complete problems [24], and since then thousands more have been discovered [25].

Complexity theory is the study of relations between these classes, so how are **P** and **NP** related? It is straightforward to see that **P**$\subseteq$ **NP**. If the king wants to know whether it is possible to divide the kingdom between his two children to his satisfaction, he can just ignore the solution provided by the court magician and solve the problem himself. The reverse question, whether **NP**$\subseteq$ **P** is one of the most important open problems in computer science, and one of the seven Millennium Prize Problems stated by the Clay Mathematics Institute [2]. It is impossible to overstate the importance of this question, and I cannot do justice to it here. If it turns out that **P** = **NP**, then the mere ability to check the correctness of the solution to a problem would imply the ability to solve it, and our world would likely become a very different place (for an entertaining and nontechnical discussion of the importance of the **P** vs. **NP** question, we direct the interested reader to ref. [26]).

Our interest here is not on the **P** vs. **NP** question, our goal lies in higher levels of the structure of complexity classes. However, this digression aims to illustrate the role played by some *conjectures* in computational complexity theory. Even though it is currently unknown whether **NP** is contained in **P**, many researchers consider that the answer is probably negative. There are thousands of **NP**–complete problems in several fields of knowledge, such as economics, graph theory, number theory, physics, biology and so on. Finding an efficient algorithm for *any* of them (equivalently, proving that any of them is in **P**) would cause a collapse and imply that **P** = **NP**. As a matter of fact, there are only a handful of problems which are *not* known to be either **NP**–complete or in **P** (the most famous examples being factoring and graph isomorphism [25]). And yet, decades of research have failed to produce an efficient algorithm for any one of these problems.

An analogy can be traced between the belief that **P** $\neq$ **NP** and the laws of thermodynamics prior to the invention of statistical mechanics. The laws of thermodynamics are empirical observations about the world, not proven facts. Despite no proof that it was impossible, any early-19th century scientist familiar with thermodynamic theory would be wary of investing in a project of a perpetual motion machine. Similarly, there is no proof that **P** $\neq$ **NP**, but any proposed efficient algorithm for e.g. the traveling salesman problem is regarded with skepticism unless it clearly uses novel ingredients missed by the community for the last several decades.

Of course, the **P** vs. **NP** question is different from the laws of thermodynamics in a fundamental way. Though statistical mechanics subsumes thermodynamics in some sense, it has replaced the latter's laws by microscopic principles which are still essentially empirical. The question of whether **P** = **NP**, on the other hand, is a well-posed mathematical question, which in principle has a definite truth value and presumably can be proven. Nonetheless, I can give two examples to justify why the analogy is apt. The first is the fact that, to date, one of the methods used for public-key cryptography, known as RSA [27], depends on the assumption that factoring large integers is a much harder problem than multiplication. The second is that one of the crown jewels of the field of quantum computing is Shor's algorithm [7, 8], which precisely shows how to factor large integers in polynomial time on a quantum computer. Though Shor's algorithm is no longer the main driving force behind research (and funding) on quantum computing, it certainly holds a special place kick-starting widespread interest in it. Both of these endeavors are predicated on the *conjecture* that factoring is a hard problem for classical computers—despite the fact that factoring is not even the hardest among **NP** problems!

The belief in conjectures such as that $\mathbf{P} \neq \mathbf{NP}$ has led to several results in theoretical computer science of the form "if claim X was true, it would imply $\mathbf{P} = \mathbf{NP}$, therefore X is probably not true". The entire field of quantum computing in a sense stands on similar assumptions, given that there is no *proof* that a quantum computer really is faster than its classical counterpart. In the following section we will use a similar conjecture to give evidence of the large discrepancy in complexity between the determinant and the permanent and, subsequently, between the computational power of bosons and fermions. Hopefully the preceding discussion has made it clear why these conjectures, though still unproven, are not empty or unjustified.

3.2. The polynomial hierarchy and counting problems

Let us now move a few rungs up the complexity ladder to define two new complexity classes, **PH** and **#P**, which will be central to our main argument.

The polynomial hierarchy, or **PH**, is a tower of generalizations of **P** and **NP**, which can be defined as follows.

Definition 3. ***PH***: *The set of decision problems for which there exists some Boolean function $f(x, y, z, w, \ldots)$ such that:*

(i) *x is some encoding of the problem instance in n bits;*
(ii) *y, z, w are poly(n)-sized strings;*
(iii) *$f(x, y, z, w, \ldots)$ can be computed on a classical computer in time poly(n).*

The decision problem can then be formulated as "given input x, does there exist y such that for all z, there exists w such that ... $f(x, y, z, w \ldots) = 1$?"

This formulation of **PH** justifies our inclusion of the function-based definitions of **P** and **NP**, as it is a straightforward generalization of them. Technically, **PH** is not a complexity class, but rather a union of infinitely many classes. For each number of sequential "there exists" and "for all" quantifiers, the corresponding set of problems defines one *level* of **PH**. **P** corresponds to level 0, **NP** (and the closely related co-**NP**, which we have not seen) correspond to level 1, and so on.

I will not go into further details of the definition, or even examples of problems within higher levels of **PH**. For us, the polynomial hierarchy will serve simply as a proof mechanism. It can be shown that, if any two levels of **PH** are equal, then the entire tower collapses to that level. For example, if $\mathbf{P} = \mathbf{NP}$, it follows that the tower collapses to its level 0, i.e. $\mathbf{PH} = \mathbf{P}$. Our main conclusions will then be predicated on the conjecture that this does not happen, and that the polynomial hierarchy is infinite. This is a very common generalization of the conjecture-based argument described in the previous sections, but now of the form "if X is true the polynomial hierarchy collapses to its nth level, therefore X is probably not true". This is a reasonable conjecture, as there is no reason to expect the class of problems of the form "given input x, does there exist y such that for all z, there exists w... is $f(x, y, z, w \ldots) = 1$?" to simply terminate after some number of quantifiers. Note however that it is a strictly *weaker* conjecture than $\mathbf{P} \neq \mathbf{NP}$, since it is possible for the latter to be true even if **PH** collapses (e.g. to some level above the second).

Finally, the last complexity class we will need is **#P**, which can be defined as follows:

Definition 4. ***#P*** *[informal]: The set of problems of the type "compute the number of solutions to an **NP** problem".*

Note that **#P** collects problems of a different nature than the previous classes we encountered, namely *counting* problems rather than *decision* problems. One example would be if the king from problem 2 wanted to know not only whether it is *possible* to divide the kingdom between his children as desired, but rather *how many* different ways there are to make this division. In the next section we will show how the permanent of a matrix can be cast as a **#P** problem.

How does **#P** relate to the classes we saw previously? First, **#P** problems must be harder than **NP** problems by definition. If we can count the number of solutions to an **NP** problem, we can easily find out whether a solution exists, just by checking whether the number of solutions is larger than 0! On the other hand, even if a solution to a particular problem happens to be easy to find, there might be exponentially many different solutions, and counting all of them might be much harder. A well-known result in complexity theory, Toda's theorem, establishes that **#P** is in fact harder than the entire polynomial hierarchy:

Theorem 1. *Toda's Theorem [28]:* $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$

Note that, technically, we cannot write $\mathbf{PH} \subseteq \#\mathbf{P}$. This makes no sense, since **PH** is a set of decision problems, while **#P** is a set of counting problems, so they are not directly comparable. This is resolved by defining the class $\mathbf{P}^{\#\mathbf{P}}$, which informally means the set of decision problems that a classical computer could solve efficiently if it had access to an auxiliary machine capable of solving counting problems.

We have now constructed the cross-section of the structure of complexity classes that we will need, consisting of **P**, **NP**, **PH** and **#P**. However, this is far

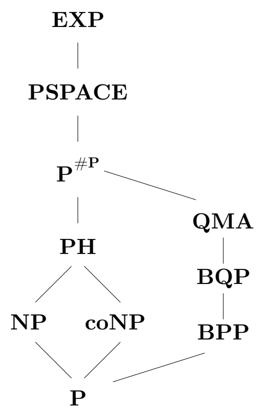

Figure 6: Abridged version of the petting zoo of complexity classes [3], focusing on the most common classes and those important for this work. The only classes not mentioned in the main text are **PSPACE**, corresponding to problems solvable by a classical computer with polynomial *memory* (regardless of how long it takes), and **EXP**, which corresponds to problems which classical computers can solve given exponential time. Lines indicates inclusions, from bottom to top (e.g. **P**$\subseteq$ **NP**), though not necessarily strict inclusions (e.g. it is unknown whether **NP** $\subseteq$ **P**).

from a complete picture of computational complexity theory. In Figure 6 we represent these classes in relation to a few well-known others, but there are in fact several hundred known complexity classes to date, and we direct the reader to the Complexity Zoo website [3] for a full listing.

3.3. The determinant and the permanent

Let us now see how the determinant and permanent functions fit within the classification scheme described in the previous section. Consider first the determinant. Recall its definition:

$$\det A = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma_i},$$

where A is an $n \times n$ matrix. This expression is a very inefficient way to perform the computation, as it requires the summation of $n!$ terms. This does not mean that there is no better alternative. Recall that the determinant has the following property:

$$\det(XY) = \det X \det Y.$$

This provides a shortcut for computing the determinant. Rather than use the definition, we can instead rewrite A as a product of matrices whose determinants are easy to compute. One example is the standard high school method for computing the determinant and solving linear systems based on Gaussian elimination. This procedure is based on applying three types of transformations on the rows of A: (i) swapping two rows, (ii) multiplying a row by a scalar c, and (iii) adding one row to another. All three steps can be written as multiplying A by particular matrices, whose respective determinants are (i) -1, (ii) the scalar c or (iii) 1. At the end of this procedure, A is taken into another matrix, A', that is in upper triangular form. In other words, we can write:

$$AB_1B_2 \dots B_m = A',$$

where the number of operations is $m = n(n+1)/2 < n^2$. Notice that $\det A'$ is easy to compute, since A' is in upper triangular form, and its determinant is just the product of the diagonal elements. Therefore, we can write

$$\det A = (\det B_1 \dots \det B_m)^{-1} \det A',$$

which requires computing less than n^2 easy determinants. If we carefully account for all the intermediate steps in the Gaussian elimination, this algorithm requires roughly n^3 steps[8]. Since this is polynomial in the size of the matrix, this places computation of the determinant in **P**[9].

Having shown that the determinant can be computed efficiently, what about the permanent? Can we use a similar trick? Alas, the answer seems to be no. In general per $XY \neq$ per X per Y, so it is not possible to perform an efficient tracking of the permanent during Gaussian elimination, as was done for the determinant. We will now give a stronger argument why it is unlikely that the permanent can be computed efficiently. Before we get to that, however, let us define another useful problem, that of perfect matchings (illustrated in Figure 7).

Problem 3. *Upon finishing your archaeological dig, you return to your regular life as a university professor. On your first class of the semester, you are assigned two groups, one composed of n junior students and the other of n senior students. To improve the performance of the junior students, you decide to pair each of them with one senior student. However, you are aware that some pairs of students simply despise each other, and will refuse to work together. Is it possible to find a pairing of the students, one from each group, such that no student is left to work alone? This is known as a perfect matching.*

How hard is this problem? It can be shown that it is in **P**, and I will now describe a simple procedure that (almost) proves this.

We begin by labeling the senior students as $(s_1, s_2, \dots s_n)$, and the junior students as $(j_1, j_2 \dots j_n)$.

[8] One caveat is that this algorithm can produce intermediate numbers with exponentially many digits [29], but there are better algorithms which are truly polynomial [30].

[9] The alert reader may protest that computing the determinant is not a *decision* problem, so it cannot be in **P**. This is technically true, though it is possible to replace the problem of computing $\det A$ by a sequence of decision problems of the type "is the nth digit of $\det A$ greater than 5?", which allows us to reconstruct $\det A$ to a desired precision. So we stick to standard imprecise terminology and state that the determinant is in **P**.

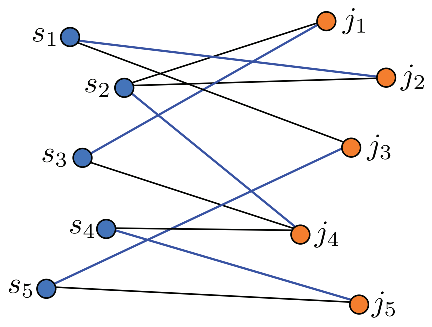

Figure 7: The problem of finding a perfect matching between a group of students. Blue and orange dots represent senior and junior students, respectively. For every pair of students that are willing to work together, we draw an edge between them. In blue, we represent one possible solution to this problem. For this example, there are three possible perfect matchings.

We then construct a matrix A (known as the Edmonds matrix in graph theory [31]) by the following prescription:

$$A_{ik} = \begin{cases} x_{ik} & \text{if } s_i \text{ and } j_k \text{ would work together,} \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

In principle the variables x_{ik} are indeterminates, though for the applications we consider we can assume they can take integer values, for simplicity.

For the example of Figure 7, this matrix is given by

$$A = \begin{pmatrix} 0 & x_{12} & x_{13} & 0 & 0 \\ x_{21} & x_{22} & 0 & x_{24} & 0 \\ x_{31} & 0 & 0 & x_{34} & 0 \\ 0 & 0 & 0 & x_{44} & x_{45} \\ 0 & 0 & x_{53} & 0 & x_{55} \end{pmatrix}. \quad (12)$$

Now consider the quantity $\det A$, as given in the definition in equation (9a). Clearly, it is a polynomial of degree n in $\{x_{ik}\}$. Each of its monomials contains the product of n elements of A that do not share rows or columns. Therefore, each monomial is directly associated with one way to pair the students, where senior student s_i is paired to junior student j_{σ_i}. If any of these pairs would refuse to work together, then the corresponding matrix element is 0 and that monomial does not contribute to the sum. Therefore, the only terms of $\det A$ that survive are those corresponding to valid perfect matchings of the students! This means that we have reduced the problem of deciding whether *some* perfect matching exists to the problem of deciding whether $\det A$ is the zero polynomial.

The next step is to decide whether $\det A$ is the zero polynomial. We can write it explicitly, e.g. for equation (12) we have

$$\det A = x_{12}x_{24}x_{31}x_{45}x_{53} - x_{12}x_{21}x_{34}x_{45}x_{53} - x_{13}x_{22}x_{31}x_{44}x_{55}.$$

However, writing $\det A$ explicitly is not efficient, as it may contain exponentially many monomials. Instead, we can use the following procedure. We first choose random values for the $\{x_{ik}\}$ variables, and then we compute $\det A$ for those values, which *can* be done efficiently, as we argued above. Suppose we repeat this procedure many times, and every time we observe that $\det A = 0$. The more samples we take, the higher our confidence that it is indeed the zero polynomial, since it is unlikely that we chose only roots of $\det A$ by chance. Conversely, if $\det A$ is not the zero polynomial, we expect to very quickly find one assignment of $\{x_{ik}\}$ for which $\det A \neq 0$. The success probability of this procedure can be formalized by the Schwartz-Zippel lemma:

Lemma 1 (Schwartz-Zippel [32–34]). *Let $P(x_1, x_2 \ldots x_m)$ be a non-zero polynomial of total degree at most d. Let S be a finite set of integers. Then, if $a_1, a_2, \ldots a_m$ are randomly chosen from S, then*

$$Pr[P(a_1, a_2 \ldots a_n) = 0] \leq \frac{d}{|S|},$$

where $|S|$ is the number of elements of S.

Clearly, the Schwartz-Zippel lemma guarantees that we can make the probability of failure of the previous procedure arbitrarily small. Therefore, we conclude that the problem of deciding whether a perfect matching of the students exists is at most as hard as computing the determinant, and thus it can be done efficiently on a classical computer.

The preceding argument does not quite place the problem of perfect matching in **P**, because the definition of **P** does not allow for randomness or a tiny (but non-zero) failure probability. It does place the problem in **BPP**, which is the generalization of **P** that does allow randomness, but which we have not defined. In any case, there are other ways to prove that perfect matching is in **P**, both in the versions of deciding whether one exists and actually *finding* one [35].

What if we now consider the problem of *counting* the number of perfect matchings? Interestingly, even though the decision version of the problem is easy, the counting version is actually #**P**–complete. Recall that an **NP**–complete problem was one of the hardest **NP** problems, and the definition of #**P**–complete is similar. A problem is #**P**–complete if every other #**P** problem reduces to it. If there exists an efficient algorithm to count the number of perfect matchings, then every #**P** problem would also be solvable efficiently. From the discussion of the previous section, this would imply that the entire polynomial hierarchy would come crashing down and collapse to **P**, due to Theorem 1. This suggests that such an efficient algorithm is very unlikely indeed!

The #**P**–completeness of counting the number of perfect matchings in a graph was shown, by Valiant, in the same paper that defined the #**P** class [36], and where he proved that the permanent is also #**P**–complete. To see why the latter is true, consider the Edmonds matrix A we constructed in equation 11, but set all (nonzero) variables x_{ik} to 1, obtaining the following matrix (known as the biadjacency matrix)

$$A' = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{13}$$

When computing $\det A$, we argued that every nonzero monomial in that sum corresponded to one valid perfect matching. By the exact same argument, each of the monomials in $\det A'$ takes value 1 for a perfect matching and 0 otherwise. If we now drop the minus signs from the definition of the determinant, the resulting expression *counts* the number of perfect matchings. But that is simply the definition of per A'

$$\text{per } A' = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A'_{i,\sigma_i}.$$

Since counting perfect matchings is #**P**–complete, this proves that computing the permanent of a matrix—furthermore, even one whose elements are 0 or 1—is #**P**–complete as well, since any #**P** problem can be reduced to counting perfect matchings and subsequently to the permanent.

Let us summarize the conclusions of this section. First, we argued that the determinant is in **P**. Then we showed that deciding *whether* a set of students can be matched perfectly can be reduced to the determinant, and hence it is also easy. We then showed that *counting* in how many ways we can match these students reduces to the permanent. But this counting is known to be #**P**–complete, and therefore the permanent is #**P**–complete. If we now look back at Figure 6, this shows us how radically different the permanent and determinant are, in terms of complexity–at least if we assume there are no major collapses in these complexity classes.

This concludes our review of computational complexity theory. We now have the tools necessary to investigate the difference in complexity of bosons and fermions. The previous paragraph suggests that bosonic linear optics should be computationally more complex, in some sense, than fermionic linear optics, due to the difference between the determinant and the permanent that appear in equations (8). However, there is more work to be done to justify that claim, because quantum-mechanical probabilities are not directly observable. Therefore, we need to define more formally what does it mean to *simulate* a quantum system.

4. Sampling of bosons and fermions

4.1. Classical simulation

There are many notions of what it means to simulate a quantum system. We might want to compute its energy levels and describe the corresponding eigenstates, or obtain the expectation value of some observable, or compute the probabilities associated with particular outcomes. Our approach will be to formulate a concrete *computational task* that we could in principle perform experimentally using fermions or bosons, and then consider how well a classical computer would perform at the *same task*.

Suppose initially we want to use a bosonic system to directly compute the permanent of a matrix. If possible, that would be quite astounding, since it would imply quantum computers can solve #**P**–complete problems—whereas currently we do not believe they can solve even **NP**–complete ones. An attempt to leverage the bosonic transition probabilities was made by Troyansky and Tishby in [37], where they showed how to encode per A for some $n \times n$ matrix A as an expectation value of a bosonic observable. Predictably, however, this did not imply an efficient quantum procedure to compute permanents, since the *variance* of this observable grows exponentially, implying we would need to repeat the measurement exponentially many times to obtain a reasonable precision in the estimate of per A[10].

A similar approach would be to directly compute the permanents via equation (8a), by repeating the experiment many times and estimating the corresponding probabilities. This suffers from the same problem, however. Recall, from section 2.1, that the configuration space for n bosons in m modes is exponentially large. As a consequence, for typical transition matrices U, the probabilities of specific events tend to be exponentially small, again requiring an exponential number of repetitions of the experiment to estimate.

Can there be a smarter way to use bosons to compute permanents? There is no proof that quantum computers cannot solve #**P**–complete problems–as I argued previously, there rarely is. But let us move towards a more feasible goal, that of providing evidence of complexity of whichever task a linear-optical experiment performs more naturally. To that end, recall the ingredients of our paradigmatic linear-optical experiment (shown in Figure 3): (i) a (bosonic or fermionic) Fock input state of n particles in m modes, $|S\rangle := |s_1 s_2 \ldots s_m\rangle$, such that $\sum s_i = n$; (ii) a linear-optical transformation U, or interferometer, acting according to equation (3); and (iii) a final measurement of the occupation number of the output modes. The output is described by the probabilities:

$$\Pr(S \to T) = \frac{|\text{per}(U_{S,T})|^2}{\prod_i t_i! \prod_j s_j!}, \text{ for bosons,}$$

$$\Pr(S \to T) = |\det(U_{S,T})|^2, \text{ for fermions.}$$

In both cases $U_{S,T}$ is a submatrix of U constructed by the prescription just above equations (8).

[10] Interestingly, they observed that the variance for the corresponding fermionic observable encoding $\det A$ was zero.

We now make two simplifying assumptions for the bosonic case. First, we assume the input state is such that $s_i \leq 1$, i.e. we only input one particle in each mode. Second, we also ignore output collision states (when two or more bosons exit in the same mode). In other words we assume $t_j \leq 1$, so the output distribution only has support on $B^*_{m,n}$, as defined in section 2.1. There are two justifications for this assumption. The first is experimental: often, photonic linear-optical experiments use avalanche photo-detectors [38], which can only distinguish whether photons arrived or not, but not how many. In that case, an experimental run is only accepted if *n different* detectors click, i.e. for no-collision outputs. Luckily, the second justification is based on the bosonic birthday paradox [14, 39], which states that no-collision states dominate the output distribution, and therefore not many experimental runs will be wasted by ignoring collisions. This only holds if $m \gg n^2$ and for typical (uniformly random) unitaries U, but this regime is required for other technical reasons anyway, which we will mention in section 4.3. Fortunately, in this aspect the theoretical limitation happens to match experimental convenience!

Having made these assumptions, bosons and fermions now inhabit configurations spaces of the same size, namely $|F_{m,n}| = |B^*_{m,n}| = \binom{m}{n}$. Therefore, we can write the output probability distributions of bosonic ($\mathcal{B}$) and fermionic ($\mathcal{F}$) linear optics, as functions of the interferometer matrix U, as follows

$$\Pr_{\mathcal{B}}(T) = |\mathrm{per}(U_{S,T})|^2 \text{, for bosons,} \tag{14a}$$

$$\Pr_{\mathcal{F}}(T) = |\det(U_{S,T})|^2 \text{, for fermions,} \tag{14b}$$

where, in both cases, we can think of T as running over all bit strings with n ones and $m - n$ zeroes.

We have made some progress in formulating a concrete computational task: since a fermionic or bosonic system behaves according to $\mathcal{B}$ or $\mathcal{F}$, our task is now to simulate either probability distribution (given as input a description of U). However, there are a still a few different things we might mean by *simulating a probability distribution.*

The first, known as *strong simulation*, can be defined as follows:

Definition 5 (Strong simulation)**.** *Given a probability distribution $\mathcal{B}$ or $\mathcal{F}$, as in equations* (14)*, an efficient strong simulation of it consists of a classical algorithm that computes any probability to poly*(m, n) *digits of precision in time poly*(m, n).

Whenever strong simulation can be done efficiently on a classical computer (as we will see happens for fermionic linear optics), there is no reason not to use this definition. However, it is not a good standard of *comparison* between the computational power of a quantum and a classical device, because a quantum device in general *cannot perform this task efficiently.* This is because probabilities are not directly observable, and the best that a quantum device can hope to do is produce a sequence of *samples* drawn from this distribution.

We could try to estimate the probability of an event by repeating the experiment many times and building a table of relative frequencies, but that does not achieve sufficiently high precision. To see why, suppose an event happens with probability one in 1000, and we wish to estimate this probability from a list of samples. We expect to need around 1000 samples for the event to happen at least once and for us to be able to ascribe it a nonzero probability. But this estimate would have precision of only 3 digits. In general, taking M samples only provides $\log M$ digits of precision. Since the definition of strong simulation requires poly(m, n) digits of precision, this in turn requires repeating the experiment exponentially many times. Therefore, a quantum device cannot efficiently strongly simulate itself.

This gives rise to the notion of *weak* classical simulation, defined as follows:

Definition 6 (Weak simulation)**.** *Given a probability distribution $\mathcal{B}$ or $\mathcal{F}$, as in equations* (14)*, an efficient weak simulation of it consists of a classical algorithm that, in time poly*(m, n)*, produces a sample drawn from this distribution.*

More simply, a weak simulation is just a procedure by which the classical computer *imitates* the behavior of the quantum device, in a sense producing the same amount of information we would get by running the experiment the same number of times. Although it is less common, it is the appropriate standard by which to judge a classical competitor that wants to computationally outperform a quantum device[11]. And, indeed, there are examples of quantum systems for which efficient weak simulation is possible but efficient strong simulation is considered unlikely [40, 41]. The two types of simulation are represented in Figure 8.

We can now finally define the precise computational task solved by a linear-optical experiment:

Problem 4. *Given as input an efficient description of the linear-optical transformation U, acting on a state of n bosons, the task of (exact) BosonSampling is to weakly simulate the distribution $\mathcal{B}$ given in equation* (14a)*, in time poly*(m, n)*. The fermionic analogue, FermionSampling, is defined equivalently in terms of the distribution $\mathcal{F}$ from equation* (14b)*.*

One can make the job of the classical competitor even easier by noticing that a real-world quantum device

[11] Often, in the definition of strong simulation one is also required to computer conditional probabilities. That is, not only the probabilities over the outcomes, as in equations (14), but also quantities like "the probability of observing a particle on mode 2 given that a particle is known to occupy output mode 1". This is necessary for the notion of strong simulation to be strictly stronger than that of weak simulation.

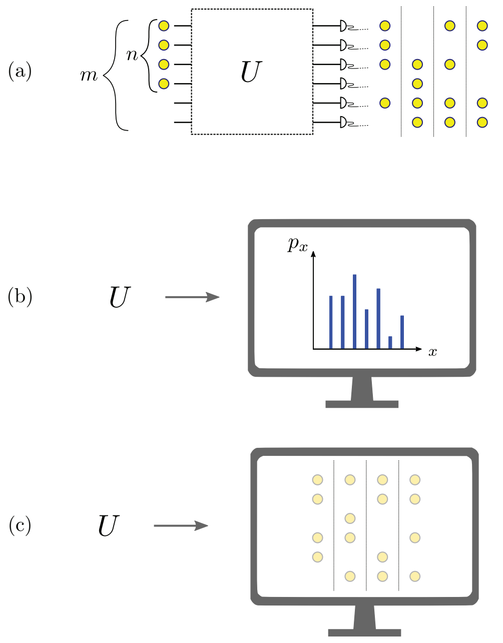

Figure 8: Two notions of classical simulation. (a) A quantum experiment within the setup of Figure 3 is completely defined by its interferometer matrix U. (b) A strong classical simulator takes as input U and outputs the value of any outcome probability. (c) A weak classical simulator takes as input U and outputs *samples* from the corresponding distribution.

cannot even provide an *exact* weak simulation of itself, due to experimental noise. Therefore, we could define notions of *approximate* weak simulation, where the algorithm is only required to produce samples from a distribution suitably close to the ideal one. I will not use this notion of simulation in detail here, since it is drastically harder to prove meaningful results in this more realistic setting. However, it is the standard used by the field of quantum computational advantage, which we return to in section 4.3.

4.2. FermionSampling

Let us begin with the easiest case, that of FermionSampling. Equation (14b) states that transition probabilities in $\mathcal{F}$ are proportional to determinants of sub-matrices of U. As we argued in section 3.3, the determinant can be computed efficiently by a classical computer (it is in **P**). Therefore, it immediately follows that strong simulation of fermionic linear optics is in **P** as well.

What about weak simulation? We can efficiently compute the probability of *any* outcome, but not of *all* outcomes, since $F_{m,n}$ has exponentially many states. In order to sample from the distribution $\mathcal{F}$, we need to be able to compute conditional or marginal probabilities. Then we can, for example, simulate the outcome of the first detector, then simulate the outcome of the second detector *conditioned* on the outcome obtained for the first, and so on. By definition this will result in a sample that was obtained according to $\mathcal{F}$. I will not go into the details of how this is done, as it is not particularly illuminating. It requires invoking Wick's theorem and the Pfaffian (a matrix function related to the determinant), as shown in [15]. This implies that weak classical simulation of FermionSampling is also easy.

This argument suggests that fermionic linear optics does not display the high level of complexity usually associated with quantum computers. That is not a bug, but a feature, since we can *leverage* the classical simulability of fermions to simulate efficiently other interesting quantum systems. One way to do this is to use the Jordan-Wigner transformation [42], which maps quadratic fermionic Hamiltonians into Hamiltonians of spin-1/2 particles.

The Jordan-Wigner transformation maps an m-mode fermionic system into a 1D chain of m spins, such that the $|0\rangle$ and $|1\rangle$ states of mode i are identified with the $|\uparrow\rangle$ and $|\downarrow\rangle$ states of the ith spin. Then, it establishes the following correspondence between fermionic and spin operators:

$$f_j^\dagger = \prod_{k<j} Z_k (X_j + \mathrm{i} Y_j)/2,$$
$$f_j = \prod_{k<j} Z_k (X_j - \mathrm{i} Y_j)/2,$$

where X_j, Y_j and Z_j are the conventional Pauli operators acting on spin j (with identity implied on the remaining spins). It is easy to check that, if the particle operators satisfy the proper anti-commutation relations, then the spin operators satisfy the correct algebra for Pauli matrices. This mapping seems very nonlocal at first sight, but it actually maps quadratic operators acting on nearest-neighbor modes into operators on nearest-neighbor spins. For example,

$$-(X_3 + \mathrm{i}Y_3)(X_4 - \mathrm{i}Y_4)/4 = f_3^\dagger f_4.$$

Note how all the extra Z_i operators for $i < 3$ cancel since $Z^2 = I$.

Now recall, from equation (4), that quadratic fermionic Hamiltonians are precisely the generators of fermionic linear optics. If we work out all possibilities for (nearest-neighbor) quadratic operators, we will see that the corresponding spin Hamiltonians are those spanned by the following set:

$$\mathcal{M} = \{X_i Y_{i+1}, Y_i Y_{i+1}, X_i X_{i+1}, Y_i X_{i+1}, Z_i, Z_{i+1}\}_{i=1\ldots m-1}.$$

The classical simulability of free fermions then directly implies the simulability of any one-dimensional condensed matter system with a Hamiltonian spanned by elements of $\mathcal{M}$, which include, e.g., the Ising and the XY (or anisotropic Heisenberg) interactions.

Beyond condensed matter applications, unitaries generated by Hamiltonians from $\mathcal{M}$ have appeared in the quantum computing literature as *matchgate circuits* [16, 43]. The fact that they are classically simulable has served as a testbed to investigate particular aspects of large restricted quantum computers that would be prohibitive in general. While an arbitrary quantum computer cannot be classically simulated beyond a few dozen particles (a point we will return to in section 4.3), a computer composed only of matchgates can be simulated up to thousands of qubits. This has been used e.g. to investigate error propagation and benchmark error correcting codes [44, 45], or as toy models for holographic duality [46].

4.3. BosonSampling

We now move to the bosonic case where, as we should already expect, the situation is drastically different from that of fermions. The task of BosonSampling was first considered formally (and named) in a seminal paper by Aaronson and Arkhipov [14] which is arguably the starting point of the field of quantum computational advantage (or supremacy). In this section, I outline the proof of their simpler result, which provides evidence that the *exact* version of BosonSampling is hard for classical computers. I also briefly touch on some aspects of their more experimentally-friendly proof that *approximate* BosonSampling is also expected to be hard[12]. I leave a more general discussion on the concept of quantum advantage for the next section.

In previous sections, we identified that transition amplitudes for bosonic linear-optical devices are given by matrix permanents. We also saw how permanents are expected to be extremely hard functions to compute (they are #**P**–complete, and so outside of the polynomial hierarchy). This already implies that strong simulation of these devices is a #**P**–complete task. However, this is not enough to attest that the quantum device is performing a computationally hard task, since it cannot strongly simulate itself either. Therefore it might be possible, in principle, to have an efficient classical algorithm that weakly simulates this device. In order to provide evidence that this is not possible, we need some more sophisticated tools.

The central ingredient we need is a classic result due to Stockmeyer [47]. Below I paraphrase it and use a version adapted for our purposes.

Theorem 2. *[Stockmeyer [47]] Let $\mathcal{D} = \{p_x\}_x$ be a probability distribution over $x \in \{0,1\}^n$. Suppose there is an efficient randomized classical algorithm that produces a sample drawn from $\mathcal{D}$. Let $g \geq 1 + \frac{1}{poly(n)}$. Then there exists an efficient randomized classical algorithm which, upon receiving $x \in \{0,1\}^n$, and* when given access to a machine capable of solving problems within the second level of ***PH**, approximates p_x to multiplicative precision. In other words, outputs q_x such that*

$$\frac{p_x}{g} \leq q_x \leq p_x g. \tag{15}$$

This is a powerful theorem because, in general, the task of computing p_x *exactly* can be #**P**–complete. Let us illustrate that with an example. Consider again problem 3, i.e. the problem of finding a perfect matching between the two sets of students. Let M be the set of all possible ways to pair the students, regardless of their preferences. Some of these will be valid perfect matchings (where no student is paired with another they despise), others will not. Consider now the probability distribution $\mathcal{PM} = \{p, 1-p\}$, where p is the probability that, by choosing a pairing of the students uniformly at random, we happen to find a valid one. To produce a sample of this distribution on a classical computer is a very simple task: we just choose a pairing at random, and check whether it is valid, which can be done efficiently. However, the value of p corresponds to the fraction of all pairings that are valid, and so *computing* p is equivalent to counting the number of perfect matchings, which is #**P**–complete.

What Stockmeyer's theorem then shows is that some #**P**–complete problems, if we allow a modest amount of error, can be solved *approximately* within very low levels of the polynomial hierarchy. An alternative formulation [14] makes it more transparent that the #**P** problems amenable to this approximation are those equivalent to estimating the sum of exponentially many *nonnegative* real numbers. One such example is counting the number of perfect matchings, as described above, or equivalently computing the permanent of a $(0,1)$ matrix.

It is important to emphasize that theorem 2 does *not* mean that these #**P** quantities can be approximated efficiently on a classical computer (though some of them can), only that this can be done within **PH**, but at a level above **NP**. Why do we care that approximating these quantities has been moved from the extremely-hard class #**P** into the still-extremely-hard **PH**? As we will see, the relations between these unrealistic classes can produce relevant claims for more realistic ones, like **P**.

The observant reader might expect that, since we just demoted approximating some permanents to a lower complexity class, this suggests BosonSampling might not be that hard either. However, that is not so. The permanents amenable to Stockmeyer's theorem are those of matrices with nonnegative entries. In fact, they are even easier to compute than theorem 2 suggests, since it was shown by Jerrum, Sinclair and Vigoda that they can be approximated efficiently on a classical computer. Luckily for bosons, the matrices that describe their transition probabilities [cf. equation (14a)], have complex-valued coefficients. And those permanents remain #**P**–complete even to approximate, in the multiplicative sense of equation (15), as shown in [14].

We now have all the ingredients to describe the seminal result for (exact) BosonSampling.

[12] The whole argument spans over 80 pages and is much too technical for the scope of this paper.

Theorem 3. *[Hardness of exact BosonSampling [14]] The exact BosonSampling problem is not efficiently solvable by a classical computer, unless $\mathbf{P}^{\#\mathbf{P}} = \mathbf{BPP}^{\mathbf{NP}}$, in which case the polynomial hierarchy collapses to its third level.*

The proof of this result can be outlined as follows[13].

(i) First, it was shown that approximating (per $X)^2$ to within multiplicative error, for an arbitrary real $n \times n$ matrix X, is $\#\mathbf{P}$–complete.

(ii) Then, it was shown that it is always possible to find a $2n \times 2n$ unitary matrix U that contains ϵX as its top left submatrix (where ϵ is a rescaling factor).

(iii) Now use the matrix U from step (ii) as the interferometer in a linear-optical experiment, and use as input the state where the n photons each occupy one of the first n modes. From equation (8a), the transition probability to the *same* output state is proportional to (per $X)^2$.

(iv) Suppose that there is a classical algorithm that weakly simulates BosonSampling, i.e. that produces a sample from the output distribution predicted for the linear-optical experiment.

(v) By Stockmeyer's theorem, the existence of the classical algorithm of point (iv) implies that it is possible to compute the transition probability, and hence (per $X)^2$, within the third level of the polynomial hierarchy[14].

(vi) Combining steps (i) and (v), we have concluded that a $\#\mathbf{P}$–complete quantity can be computed within the third level of $\mathbf{PH}$. By the discussion of section 3.2, in particular using Toda's theorem, we conclude that $\mathbf{PH}$ collapses to its third level.

This proof outline uses all the ingredients we described up until now, and can be quite dense, so feel free to read it again carefully. In essence it proves that, though linear-optical devices cannot themselves compute $\#\mathbf{P}$–complete quantities, the fact that their transition amplitudes are $\#\mathbf{P}$–complete implies that they are performing in fact a quite complex task. Since the collapse of $\mathbf{PH}$ is considered unlikely, this is regarded as evidence that it is hard for a classical computer to simulate a BosonSampling experiment. As promised, we used the relations between the up-in-the-sky complexity classes to produce a concrete claim about the power of real-world devices, namely classical computers, on one hand, and very rudimentary bosonic computers on the other.

The contribution of Aaronson and Arkhipov goes much beyond theorem 3. The main limitation of this theorem is that it only considers the task of *exact* sampling, which a realistic linear-optical device cannot perform either. It can be trivially extended to allow the bosonic and the classical computers to incur some multiplicative errors, as in equation (15). However, since the transition probabilities can be exponentially small, a multiplicative approximation implies an exponentially small error, which is still beyond the capabilities of a realistic device.

A more realistic notion of approximation is to consider an error in *total variation distance*. In this case we are satisfied if the weak classical simulator produces samples from some distribution $\mathcal{B}' = \{q_x\}_x$ such that

$$\|\mathcal{B}' - \mathcal{B}\| = \frac{1}{2}\sum_x |p_x - q_x| < \epsilon,$$

where $\mathcal{B} = \{p_x\}_x$ is the ideal BosonSampling distribution, and $\epsilon > 0$ is a small error parameter. Unfortunately, the proof of theorem 3 is not strong enough to provide hardness that is robust in this sense. The reason is that it relies on the $\#\mathbf{P}$–completeness of one *particular* probability (which we chose to be the transition between $|1,\ldots,1,0,\ldots,0\rangle$ and itself). Suppose now that we have a classical simulator that decides to cheat by declaring that this particular event has zero probability, while estimating the probabilities of all other outcomes very well. This simulator would produce a distribution that is not multiplicatively close to $\mathcal{B}$, but that might be close in total variation distance.

In order to circumvent this issue, Aaronson and Arkhipov replace the matrix U [obtained by embedding the $\#\mathbf{P}$–hard matrix X in step (ii) of the proof], with a uniform (i.e. Haar-random) matrix. They showed that, if the number of modes m is sufficiently larger than the number of photons n, all $n \times n$ sub-matrices of U look "identical", in the sense that they look like matrices of elements drawn independently from a complex Gaussian distribution. This serves as a protection against the cheating classical simulator described in the previous paragraph. Since all sub-matrices look very similar, the classical simulator cannot tell which of them to corrupt. Therefore, in order for it to produce a distribution close to $\mathcal{B}$, it has to get most of the probabilities right, and so it is likely to approximate the one we care about very well.

The downside of this more experimentally-friendly version of the argument is that it requires considering the complexity of permanents of *random* matrices, rather than just worst-case matrices, which requires a daunting amount of technical details and two additional conjectures beyond what we describe here.

To finish this Section, I need to address an issue I have been neglecting until now, namely the fact that classical (distinguishable) particles also evolve according to permanents, as in equation (10). Does this mean that weak simulation of classical particles is also hard for a classical computer? Of course not. The transition

[13] Readers familiar with quantum computing might be interested in an alternative version of this argument, based on post-selection and the Knill-Laflamme-Milburn protocol for universal quantum computing with linear optics, see [14, 48].

[14] For technical reasons, while Stockmeyer's algorithm mentions the second level of the hierarchy, the final conclusion refers to its third level. This comes from the fact that we need to compare problems in $\mathbf{PH}$ to $\mathbf{P}^{\#\mathbf{P}}$. This does not change the final conclusion too much.

probabilities for distinguishable particles are given by permanents of nonnegative elements, and these are in fact easy to approximate, as we discussed just before theorem 3. It is also trivial to perform a weak simulation in this case. To that end, we simply label the n particles, simulate them passing through the interferometer *one at a time* and reaching the output, and then erase their labels. This will produce a sample from the correct distribution with only a polynomial amount of effort.

4.4. Quantum computational advantage

We have seen that bosonic linear optics, in contrast to its fermionic counterpart, cannot be simulated efficiently on a classical computer (unless **PH** collapses). This is unfortunate if our main interest is to in fact simulate these systems, for example to guide the design of a future experiment or check whether our experimental apparatus is working properly. However, the upside is that this suggests a very simple experiment to test quantum mechanics in the limit of *high computational complexity*. This is the concept of "quantum computational advantage", or "quantum computational supremacy".

The idea for a universal quantum computer, i.e. one capable of displaying the entire power of quantum computation, has been known in the literature for a few decades [49]. However, actually building one has so far proven to be a daunting technological task. It requires very precise control of individual quantum systems, and furthermore of a very large number of them. To illustrate, consider the problem of factoring integers. So far, the record for largest integer factored by a classical computer is a 240-digit number [50], while the largest integer factored by a quantum computer is 21 [51]. An implementation of Shor's algorithm that would actually compete with current classical computers would require a quantum computer with millions of qubits [52], whereas the largest quantum computer ran to date, as far as I know, has 54 qubits [9].

By the laws of quantum mechanics, quantum computers should be "simply" a technological challenge. However, there are skeptics which claim that the noise present in real-world systems makes quantum computers fundamentally impossible [10, 11, 53]. The field of quantum computational advantage arose, in some sense, in response to these claims. It concerns the question of what is the simplest and smallest quantum device that we can build and control that solves some well-defined computational task faster than the best known classical computer. This task does not need to be particularly *useful*, except as a demonstration of computational strength. The hope then is that, by optimizing for raw computational complexity rather than usefulness, we might be able to give experimental evidence that quantum computers outperform their classical counterparts using substantially fewer than a million qubits.

The Aaronson-Arkhipov paper on BosonSampling is arguably the first full-fledged proposal of quantum computational advantage. It was not the first paper to propose that rudimentary quantum computers may have computational powers beyond classical computers [54, 55], but it was done for a very simple and appealing quantum system (namely linear optics) and with the aim of making the argument more relevant for realistic devices.

How many photons would be needed for a demonstration of quantum computational advantage? Recall that, as far as we know, an n-photon BosonSampling experiment, in order to be simulated, requires computing permanents of $n \times n$ matrices. A typical laptop computer struggles with computing permanents for n much above 30. This suggests than a linear-optical experiment with a few dozen photons in a few hundred modes would already be performing a task (namely, sampling from the BosonSampling distribution) that would pose a serious challenge for a classical competitor. Improved classical algorithms and analyses have revised this number up to something like 90 photons [56]. Such an experiment is still a challenge for present-day technology, but is definitely more feasible than using millions of qubits to factor a large number!

Since the original BosonSampling paper many different candidates of quantum advantage have been proposed. A particularly notable one is the random circuit sampling (RCS) model [9, 57]. In the RCS model the hardware is in principle capable of universal quantum computing[15], but it is only run for a small number of computation cycles. The idea is that the small number of cycles allows for sufficiently high complexity whilst maintaining the experimental errors under control. This idea has already borne fruit. In 2019 a research group used a quantum computer made of 53 qubits to run an instance of the RCS model with a runtime of 200 seconds, performing a task which a current classical supercomputer is expected to take orders of magnitude longer[16][9].

[15] I should point out that linear optics is *also* universal for quantum computing. It requires resources beyond BosonSampling, such as the ability to perform projective measurements and change the subsequent circuit depending on the outcome [58]. A natural qubit present in these systems is the polarization of a single photon, though any two modes that a photon may occupy can be used to encode a qubit. This technological platform is also currently being pursued as a viable candidate for quantum computing [59, 60].

[16] The authors claim that a classical supercomputer would take ten thousand years to simulate this experiment. This has been challenged in the literature, with claims that an optimized classical algorithm would take in fact only a few days instead [61]. Some debate remains whether the latter algorithm is a fair comparison, as it requires such massive amount of data storage that it would not work if the experiment had been a bit larger. In any case, if the best classical supercomputer takes a few days to do what the quantum device did in a few seconds, I would argue that a quantum computational advantage is hard to dismiss.

On the BosonSampling front, experiments have also been steadily increasing in complexity, ranging from 3–4 photons soon after the original proposal [62–65], to the largest experiment reported to date with 20 input photons (though only 14 made it to the output) in a 60-mode interferometer [66].

Unfortunately, it is beyond the scope of this work to describe in further detail all the developments, both theoretical and experimental, in the field of quantum advantage (and BosonSampling in particular). A non-exhaustive list includes strengthened proofs more robust against experimental imperfections [67–69], the development of better competing classical algorithms [70–77], the raising of other important issues, such as how to efficiently *verify* the output data to guarantee that the device is working correctly [78–81], or proposal of alternative models, such as BosonSampling with Gaussian input states rather than Fock states [82, 83]. We direct the interested reader to recent review articles for more thorough overviews of these developments [12, 13, 38, 84].

4.5. Fermions and bosons beyond linear optics

So far we have restricted our attention to linear optics, which is the focus of this work. However, there are some interesting connections to complexity theory to be found beyond this setup, in particular when one is interested in computing other quantities such as ground state energies or partition functions. So in this Section I will briefly discuss some of them. In particular, we will see that computing the ground state of Hamiltonians is, in general, a very hard problem, and interestingly it tends to be harder for fermions than for bosons, in contrast to our discussion so far.

Let us first, however, consider extensions of linear optics. Can we boost the computational power of either bosons or fermions by adding resources beyond those defined in our linear-optical setup shown in Figure 3? Recall that linear-optical unitaries acting on m modes inhabit a space of dimension roughly m^2, whereas the Fock space of n particles in m modes, which we denoted by $F_{m,n}$ and $B_{m,n}$ for fermions and bosons respectively, is exponentially larger. Therefore, there could conceivably be other interesting complexity regimes between linear optics and arbitrary transformations. Interestingly, that does not seem to be the case. It was shown in [85] that, except for very fringe cases, when we supplement linear optics (either bosonic or fermionic) with any number-preserving Hamiltonian that is not quadratic in the particle operators we already obtain universality, in the sense that we can generate any transformation within either $F_{m,n}$ or $B_{m,n}$. That is, even a simple non-linear Hamiltonian such as

$$H = \chi a_1 a_1^\dagger a_2 a_2^\dagger$$

is able to promote linear optics to all allowed transformations in the corresponding state space. Thus, there might not be anything new in between these two limits, as any supplementation of fermionic or bosonic linear optics makes them equally powerful (and as powerful as a general purpose quantum computer).

Another ingredient capable of boosting the power of *bosonic* linear optics are adaptive measurements, which consists of conditioning future beam splitters and phase shifters on the outcomes of intermediate measurements. It was shown in a seminal paper by Knill, Laflamme and Milburn [58] that this ingredient allows universal quantum computation with linear optics. Interestingly, the same is not true for fermions, as they remain classically simulable even in this case.

What if we are interested in other questions that do not fit the scattering-based setup we considered so far, such as finding the ground state of a Hamiltonian? To investigate the complexity of this question, let us momentarily set aside bosons and fermions and consider the following problem: given N spins and a k-local Hamiltonian acting on these spins, what is its ground state energy? By k-local Hamiltonian we just mean one that can be written as

$$H = \sum_i H_i \tag{16}$$

where each H_i acts non-trivially on at most on k spins, though there is *a priori* no restriction that these spins are *physically* close to each other.

How hard should we expect this problem to be? It is easy to show that it must be at least as hard as **NP** problems. To prove this, consider again problem 2, where the king wants to distribute N regions between his three children. We encode this problem into a collection of N 3-level systems, each encoding one region of the map. For each pair of *neighboring* regions, the Hamiltonian has a term H_i given by

$$H_i = |00\rangle\langle 00| + |11\rangle\langle 11| + |22\rangle\langle 22|,$$

with identity implied on the remaining systems. In other words, this Hamiltonian gives an energy penalty of $+1$ whenever two neighboring particles are in the same state. If we now identify the state labels $\{0, 1, 2\}$ with the three heirs to the kingdom, it is clear that this Hamiltonian will have a zero energy ground state if and only if it is possible to divide the map such that no two neighboring regions belong to the same person. Since the latter problem is **NP**–complete, it follows that, in general, deciding whether a Hamiltonian has a zero-energy ground state is at least as hard as **NP** [86].

Could this problem be even harder than **NP** in general? The answer in fact seems to be yes. Define the k-local Hamiltonian problem as follows:

Problem 5. *Given a k-local Hamiltonian*[17] *as defined in equation* (16)*, determine whether*

[17] There is a further technical constraint that the norm of each H_i be at most polynomially large in n.

(i) H has an eigenvalue less than a, or
(ii) all eigenvalues of H are larger than b,

promised that one of these is true, and where $b - a \geq 1/poly(n)$.

This problem was proven to be complete for a complexity class called **QMA**. The class **QMA** is the quantum analogue of **NP**. Recall that, in the definition of **NP**, we required that there exist a certificate y that can be checked in polynomial time on a classical computer. For **QMA**, we require instead that there exist a *quantum state* $|y\rangle$ which acts as a certificate that can be checked in polynomial time on a quantum computer (note that, analogous to the case of **NP**, we might need a court magician to prepare this state as a quantum computer might not be able to do this efficiently).

The k-local Hamiltonian is a natural candidate for problem in **QMA**. Whenever H does have a low-energy eigenvalue, we can use the corresponding eigenstate as a certificate. A quantum computer that receives this state as an input can act on it, efficiently estimate the corresponding eigenvalue (a procedure known as phase estimation [7]), and verify that condition (i) is satisfied. In fact, the general k-local Hamiltonian problem was proven by Kitaev [6] to be **QMA**–complete, and hence one of the hardest problems in the class **QMA**. This suggests that this problem cannot be *solved* efficiently by a quantum computer, even though an answer can be checked efficiently, for the same reasons that it is believed that **P**$\neq$ **NP**.

The result of Kitaev was improved in several subsequent works, most notably by Oliveira and Terhal [87], who showed that it still **QMA**–complete to solve this problem in the case where the Hamiltonians are *geometrically* 2-local, if the particles are arranged on a 2D square lattice, and if the particles are *qubits*, i.e. two-level systems. For an overview of **QMA**–complete problems, see [88].

Let us now return to our discussion on bosons and fermions. What happens to the complexity of the k-local Hamiltonian problem when we have (interacting) bosons or fermions? Interestingly, for many cases there is an inversion of complexity compared to what we saw so far. That is, fermionic systems are typically harder to solve than bosonic ones. This is known as the *sign problem* in Quantum Monte Carlo simulations, so let us give an idea of why this happens.

Suppose we wish to compute the expectation value of some operator A in a thermal state with inverse temperature β (the k-local Hamiltonian can be thought of as the low-temperature limit of this problem when $A = H$). This requires computing

$$\langle A \rangle = \frac{1}{Z}\mathrm{tr}[A\exp(-\beta H)],$$
$$Z = \mathrm{tr}[\exp(-\beta H)],$$

where Z is the partition function. One way to perform this computation is to write a series expansion

$$\begin{aligned} Z = \mathrm{tr}\left[\exp(-\beta H)\right] &= \sum_n \frac{(-\beta)^n}{n!}\mathrm{tr}H^n \\ &= \sum_n \sum_{i_1 \ldots i_n} \frac{(-\beta)^n}{n!}\langle i_1|H|i_2\rangle \ldots \langle i_n|H|i_1\rangle \\ &= \sum_x p_x, \end{aligned}$$

where we inserted a resolution of the identity between each copy of H in the second step, and $\{|i\rangle\}$ forms a basis for the entire system. Here, x runs through all sets of configurations $\{i_1, i_2, \ldots i_n\}$. Similarly, we can write

$$\langle A \rangle = \frac{1}{Z}\sum_x p_x A_x$$

for some coefficients A_x.

Suppose, for the moment, that all p_x were positive. Then we could treat $\mathcal{D} = \{p_x/Z\}_x$ as a (classical) probability distribution. There are exponentially many values of x, so we would not be able to compute $\langle A \rangle$ exactly by the sum above, but we could estimate it via a Monte Carlo [89] method, which consists of choosing a small set of configurations $\{x\}$ according to $\mathcal{D}$, and estimating $\langle A \rangle$ using the sample mean from this set.

The problem, of course, is that the p_x are not usually all positive. If many of these are negative, then the Monte Carlo procedure can incur an exponentially large error, coming from the fact that the p_x will tend to cancel each other and Z can be exponentially small. This is the so-called sign problem [90], an issue well-known in the condensed matter and quantum field theory literatures. Interestingly, this closely resembles our discussion that distinguishable particles are not hard to simulate even though their transition probabilities are given by permanents, since those permanents are of matrices with only positive entries.

The sign problem is not exclusive to fermions, bosonic Hamiltonians can display negative values of p_x as well [90]. However, it is much more prevalent in fermionic Hamiltonians, due to the anti-commutation relations. To illustrate, consider a generic quartic Hamiltonian (either bosonic or fermionic) of the type

$$H = \sum_{ijkl} H_{ijkl} a_i a_j a_k^\dagger a_l^\dagger,$$

where all H_{ijkl} are positive. The bosonic version of this Hamiltonian does not have a sign problem since, by choosing $\{|i\rangle\}$ as Fock states, when we work out each $\langle i_1|H|i_2\rangle$ term in the definition of p_x, we will use only commutation of bosonic operators, and there is no step where a minus sign could arise. On the other hand, the equivalent fermionic Hamiltonian does have a sign problem, since the anti-commutation relations can leave residual minus signs in some terms.

It is possible that, for particular Hamiltonians, the sign problem can be circumvented. For instance, it has recently been shown that for some cases the sign problem can be "cured" by performing a local basis change [91], though the same authors showed that, in general, even deciding whether this is possible is **NP**–complete. In any case, it is considered unlikely that a general solution to this problem exists. It was shown that the 2-local Hamiltonian problem is, in general, also **QMA**–complete for both the fermionic [92, 93] and the bosonic [94] cases. Therefore, even though bosonic simulations tend to be easier in practice than fermionic ones, this is not a general feature, as they can be equally hard for generic Hamiltonians.

5. Conclusion

In this review, we discussed a few ways in which computational complexity theory and quantum mechanics come together. Their intersection leads to a very active subfield, quantum complexity theory, and here we discussed some of its inhabitants.

We saw how free particle dynamics–both in the bosonic (i.e. linear optics) and in the fermionic cases–defines a natural computational problem, which consists simply of simulating the output distribution of the corresponding experiment. The two problems, known as BosonSampling and FermionSampling, respectively, are quite similar from a physical point of view. In contrast, from a complexity theory perspective, they are surprisingly different, and this difference can be traced back to their dependence on matrix functions of quite disparate complexities. Fermions evolve according to matrix determinants which, as we saw, are easy to compute (in **P**), whereas bosons evolve according to the much more complex matrix permanents (which as #**P**–complete). The degree of separation between these complexity classes can be measured by the existence of the polynomial hierarchy (**PH**) between them, and the very plausible conjecture that **PH** is infinitely high.

Leveraging the distinction in complexity between determinants and permanents to obtain meaningful claims for the complexity of free fermions and bosons requires quite a lot of work, since these functions are not computed directly by a real-world experiment either. To give evidence of the complexity of bosonic linear optics, we had to argue that these experiments solve *some* computational task that is hard for a classical computer—a hardness which follow from the hardness of the permanent even though the experiment cannot actually compute them. This led to the different notions of simulation described in section 4.1, and ultimately to the main arguments of sections 4.2 and 4.3.

Beyond linear optics, in section 4.5 we discussed differences and similarities between bosons and fermions from the point of view of a different question: instead of a scattering experiment, we considered the computational complexity of computing ground state energies (or partition functions). This leads to the well-known *sign problem* which plagues fermionic simulations (e.g. based on Monte Carlo methods) in condensed matter and quantum field theory. Though conventional wisdom tells us that, in this case, bosons are easier to simulate than fermions, that is not a general rule, as they are both complete for the **QMA** complexity class.

The discussion of the k-local Hamiltonian problem also led to an interesting instance where the influence between physics and complexity theory is reversed. Rather than using classical complexity classes to study quantum-mechanical systems, as in the case of FermionSampling and BosonSampling, we now have a complexity class that is directly defined in terms of a quantum-mechanical problem. It is one of many such quantum complexity classes that have earned their place in the Complexity Zoo [3], and have since then started to interact with their classical counterparts.

There are several interesting open problems that remain in bosonic and fermionic complexity theory. The most obvious one, to my mind, is whether we can trace the difference in complexity of free bosons and fermions to some fundamental physical property that distinguishes them. Currently, the most straightforward explanation of this is that fermions evolve according to determinants, and the latter satisfy the property that $\det(AB) = \det A \det B$. This property provides a shortcut which allows us to compute the determinant (and hence simulate free fermions) in polynomial time. Permanents do not have this property, and so one seems to be stuck taking an exponential time to compute them. Can we trace this property to some interesting physical aspect of fermions? On the physics side, fermions and bosons are distinguished mainly by their commutation relations, but there might be a more compelling connection between this and ease of simulation of fermionic systems.

As technological progress on quantum computing advances, so does the need for understanding its foundations and limitations. Linear optics provides a testbed for the power of small-scale or restricted quantum devices, a test of quantum theory itself in the limit of increasing complexity, and a conceptual toolbox for investigating foundational aspects of quantum computers. That the bosonic or fermionic nature of the identical particles, a fundamentally quantum property, influences so heavily on the resulting computational complexity is a fascinating connection between two vast and (until recently) mostly independent theories. I hope that these connections continue to be unveiled, contributing to trace the path for the future of quantum technologies, but also benefiting both the communities of Physics and Computer Science.

Acknowledgments

The author would like to acknowledge support from Brazilian funding agencies CNPq and FAPERJ.

References

[1] S. Arora and B. Barak, *Computational Complexity: A Modern Approach* (Cambridge University Press, Cambridge, 2009).

[2] https://www.claymath.org/millennium-problems.

[3] https://complexityzoo.net/Complexity_Zoo.

[4] S. Eibenberger, S. Gerlich, M. Arndt, M. Mayor and J. Tüxen, Phys. Chem. Chem. Phys. **15**, 14696 (2013).

[5] D. Salart, A. Baas, J.A.W. van Houwelingen, N. Gisin, and H. Zbinden, Phys. Rev. Lett. **100**, 220404 (2008).

[6] A.Y. Kitaev, A.H. Shen and M.N. Vyalyi, *Classical and Quantum Computation* (American Mathematical Society, Rhode Island, 2002).

[7] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[8] P.W. Shor, SIAM J. Comput. **26**, 1484 (1997).

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G.S.L. Brandao, D.A. Buell, et al., Nature **574**, 505 (2019).

[10] G. Kalai, arXiv:0904.3265 (2009).

[11] G. Kalai, arXiv:1106.0485 (2011).

[12] A.W. Harrow and A. Montanaro, Nature **549**, 203 (2017).

[13] A.P. Lund, M.J. Bremner and T.C. Ralph, npj Quantum Information **3**, 15 (2017).

[14] S. Aaronson and A. Arkhipov, Theory of Computing **4**, 143 (2013).

[15] B.M. Terhal and D.P. DiVincenzo, Phys. Rev. A **65**, 032325 (2002).

[16] L.G. Valiant, SIAM J. Comput. **31**, 1229 (2002).

[17] L.E. Ballentine, *Quantum Mechanics: A Modern Development* (World Scientific, Singapura, 1998).

[18] R.P. Feynman, Int. J. Theo. Phys. **21**, 467 (1982).

[19] C.K. Hong, Z.Y. Ou and L. Mandel, Phys. Rev. Lett. **59**, 2044 (1987).

[20] M. Reck, A. Zeilinger, H.J. Bernstein and P. Bertani, Phys. Rev. Lett. **73**, 58 (1994).

[21] S. Scheel, arXiv:quant-ph/0406127 (2004).

[22] S.A. Cook, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (Ohio, 1971).

[23] L.A. Levin, Problemy Peredači Informacii **9**, 115 (1973).

[24] R.M. Karp, in *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations*, edited by R. E. Miller, J. W. Thatcher and J. D. Bohlinger (Plenum Press, Boston, 1972), pp. 85–103.

[25] M.R. Garey and D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman and Co., New York, 1990).

[26] L. Fortnow, *The Golden Ticket: P, NP and the Search for the Impossible* (Princeton University Press, New Jersey, 2017).

[27] R.L. Rivest, A. Shamir and L. Adleman, Commun. ACM **21**, 120 (1978).

[28] S. Toda, SIAM J. Comput. **20**, 865 (1991).

[29] X.G. Fang and G. Havas, in *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation* (Hawaii, 1997).

[30] E.H. Bareiss, Mathematics of Computation **22**, 565 (1968).

[31] R. Motwani and P. Raghavan, *Randomized Algorithms* (Cambridge University Press, Cambridge, 1995).

[32] R. Zippel, in *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation* (Berlin, Heidelberg, 1979).

[33] J.T. Schwartz, J. ACM **27**, 701 (1980).

[34] R.A. Demillo and R.J. Lipton, Information Processing Letters **7**, 193 (1978).

[35] L. Lovász and M.D. Plummer, *Matching Theory* (North-Holland Mathematics Studies, Holland, 1986), v. 121.

[36] L.G. Valiant, Theoretical Computer Science **8**, 189 (1979).

[37] L. Troyansky and N. Tishby, in *Proceedings of Physics and Computation – PhysComp 96 (New England, 1996).*

[38] D.J. Brod, E.F. Galvão, A. Crespi, R. Osellame, N. Spagnolo and F. Sciarrino, Advanced Photonics **1**, 034001 (2019).

[39] A. Arkhipov and G. Kuperberg, Geometry & Topology Monographs **18**, 1 (2012).

[40] M. Van den Nest, Quant. Inf. Comp. **11**, 784 (2011).

[41] R. Jozsa and M. Van den Nest, Quant. Inf. Comp. **14**, 633 (2014).

[42] P. Jordan and E. Wigner, Z. Phys. **47**, 631 (1928).

[43] R. Jozsa and A. Miyake, Proc. R. Soc. A **464**, 3089 (2008).

[44] S. Bravyi, M. Englbrecht, R. König and N. Peard, npj Quantum Information **4**, 1 (2018).

[45] Y. Suzuki, K. Fujii and M. Koashi, Phys. Rev. Lett. **119**, 190503 (2017).

[46] A. Jahn, M. Gluza, F. Pastawski and J. Eisert, Science Advances **5**, eaaw0092 (2019).

[47] L. Stockmeyer, in *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (New York, 1983).

[48] D.J. Brod, arXiv:1412.7637 (2014).

[49] D. Deutsch, Proc. R. Soc. Lond. A. **400**, 97 (1985).

[50] F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann, in *Proceedings of the 40th Annual International Cryptology Conference* (Santa Barbara, 2020).

[51] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou and J.L. O'Brien, Nature Photonics **6**, 773 (2012).

[52] C. Gidney and M. Ekerå, arXiv:1905.09749 (2019).

[53] G. Kalai and G. Kindler, arXiv:1409.3093 (2014).

[54] B.M. Terhal and D.P. DiVincenzo, Quant. Inf. Comp. **4**, 134 (2004).

[55] M.J. Bremner, R. Jozsa and D.J. Shepherd, Proc. R. Soc. A **467**, 459 (2011).

[56] A.M. Dalzell, A.W. Harrow, D.E. Koh and R.L. La Placa, Quantum **4**, 264 (2020).

[57] S. Aaronson and L. Chen, in *Proceedings of the 32nd Computational Complexity Conference* (Schloss

Dagstuhl – Leibniz-Zentrum fuer Informatik, Dagstuhl, 2017).

[58] E. Knill, R. Laflamme and G.J. Milburn, Nature **409**, 46 (2001).

[59] T. Rudolph, arXiv:1607.08535 (2016).

[60] T.R. Bromley, J.M. Arrazola, S. Jahangiri, J. Izaac, N. Quesada, A.D. Gran, M. Schuld, J. Swinarton, Z. Zabaneh and N. Killoran, Quantum Sci. Technol. **5**, 034010 (2020).

[61] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh and R. Wisnieff, arXiv:1910.09534 (2019).

[62] M.A. Broome, A. Fedrizzi, S. Rahimi-Keshari, J. Dove, S. Aaronson, T.C. Ralph and A.G. White, Science **339**, 794 (2013).

[63] J.B. Spring, B.J. Metcalf, P.C. Humphreys, W.S. Kolthammer, X.-M. Jin, M. Barbieri, A. Datta, N. Thomas-Peter, N.K. Langford, D. Kundys, et al., Science **339**, 798 (2013).

[64] M. Tillmann, B. Dakić, R. Heilmann, S. Nolte, A. Szameit and P. Walther, Nat. Photon. **7**, 540 (2013).

[65] A. Crespi, R. Osellame, R. Ramponi, D. J. Brod, E. F. Galvão, N. Spagnolo, C. Vitelli, E. Maiorino, P. Mataloni and F. Sciarrino, Nat. Photon. **7**, 545 (2013).

[66] H. Wang, J. Qin, X. Ding, M.-C. Chen, S. Chen, X. You, Y.-M. He, X. Jiang, L. You, Z. Wang, et al., Phys. Rev. Lett. **123**, 250503 (2019).

[67] S. Aaronson and D.J. Brod, Phys. Rev. A **93**, 012335 (2016).

[68] A. Arkhipov, Phys. Rev. A **92**, 062326 (2015).

[69] A. Leverrier and R. García-Patrón, Quantum Information and Computation **15**, 489–512 (2015).

[70] P. Clifford and R. Clifford, in *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, USA, 2018).

[71] P. Clifford and R. Clifford, arXiv:2005.04214 (2020).

[72] A. Neville, C. Sparrow, R. Clifford, E. Johnston, P.M. Birchall, A. Montanaro and A. Laing, Nat. Phys. **13**, 1153–1157 (2017).

[73] M. Oszmaniec and D.J. Brod, New J. Phys. **20**, 092002 (2018).

[74] D.J. Brod and M. Oszmaniec, Quantum **4**, 267 (2020).

[75] J.J. Renema, A. Menssen, W.R. Clements, G. Triginer, W.S. Kolthammer and I.A. Walmsley, Phys. Rev. Lett. **120**, 220502 (2018).

[76] J. Renema, V. Shchesnovich and R. Garcia-Patron, arXiv:1809.01953 (2018).

[77] S. Rahimi-Keshari, T.C. Ralph and C.M. Caves, Phys. Rev. X **6**, 021039 (2016).

[78] C. Gogolin, M. Kliesch, L. Aolita and J. Eisert, arXiv:1306.3995 (2013).

[79] M. Walschaers, J. Kuipers, J.-D. Urbina, K. Mayer, M.C. Tichy, Klaus Richter and A. Buchleitner, New J. Phys. **18**, 032001 (2016).

[80] J. Carolan, J.D.A. Meinecke, P.J. Shadbolt, N.J. Russell, N. Ismail, K. Wörhoff, T. Rudolph, M.G. Thompson, J.L. O'Brien, J.C.F. Matthews, et al., Nat. Photon. **8**, 621 (2014).

[81] N. Spagnolo, C. Vitelli, M. Bentivegna, D.J. Brod, A. Crespi, F. Flamini, S. Giacomini, G. Milani, R. Ramponi, P. Mataloni, et al., Nat. Photon. **8**, 615 (2014).

[82] A.P. Lund, A. Laing, S. Rahimi-Keshari, T. Rudolph, J.L. O'Brien and T.C. Ralph, Phys. Rev. Lett. **113**, 100502 (2014).

[83] C.S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn and I. Jex, Phys. Rev. Lett. **119**, 170501 (2017).

[84] S. Mullane, arXiv:2007.07872 (2020).

[85] M. Oszmaniec and Z. Zimborás, Phys. Rev. Lett. **119**, 220502 (2017).

[86] M. Troyer and U.-J. Wiese, Phys. Rev. Lett. **94**, 170201 (2005).

[87] R. Oliveira and B.M. Terhal, Quantum Information and Computation **8**, 900 (2008).

[88] A.D. Bookatz, Quantum Information and Computation **14**, 361 (2014).

[89] K. Binder and D. Heermann, *Monte Carlo Simulation in Statistical Physics: An Introduction*, Graduate Texts in Physics (Springer, Berlin, Heidelberg, 2010).

[90] H. Fehske, R. Schneider and A. Weiße, *Computational Many-Particle Physics*, (Springer, Berlin, Heidelberg, 2008).

[91] J. Klassen, M. Marvian, S. Piddock, M. Ioannou, I. Hen and B. Terhal, arXiv:1906.08800 (2019).

[92] N. Schuch and F. Verstraete, Nature Physics **5**, 732 (2009).

[93] J.D. Whitfield, P.J. Love and A. Aspuru-Guzik, Phys. Chem. Chem. Phys. **15**, 397 (2012).

[94] T.-C. Wei, M. Mosca and A. Nayak, Phys. Rev. Lett. **104**, 040501 (2010).