Quantum Meta-Programming for Dummies

Cohort 4

Quantum Engineering CDT
University of Bristol

May 11, 2018

1 Preface

This is where the preface will be

Contents

2 Introduction

[Quantum computation] does not merely make computer science a branch of physics.
It also makes part of experimental physics into a branch of computer science

---

and the universal quantum computer
– David Deutsch

Quantum mechanics is one of the most well-tested theories in existence [REF?]. It is also one of the most unintuitive, revealing aspects of nature at nanoscopic scales which are entirely incompatible of our own experience of the world.

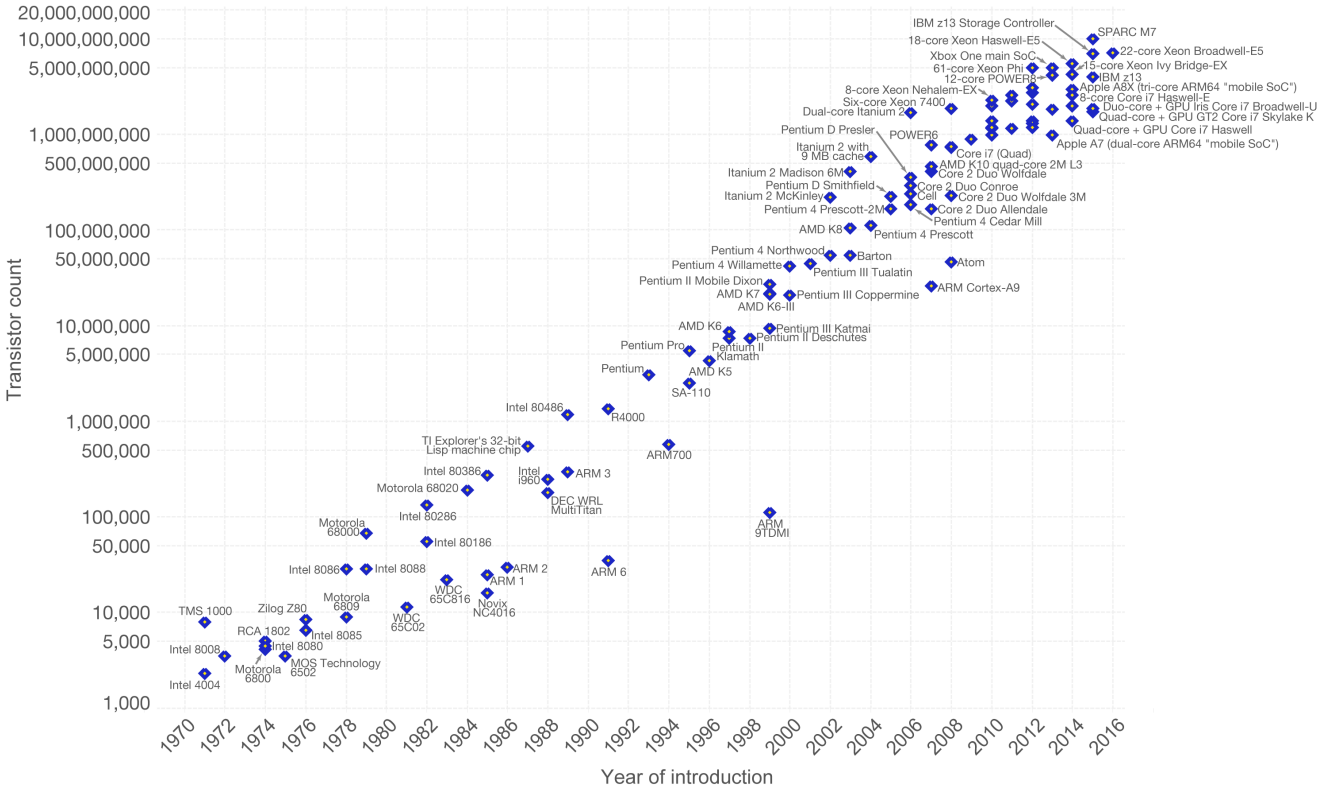
Quantum mechanics has two key facets. The first, from which the field derives its moniker, is the quantisation of properties such as energy... . This property has already changed the world substantially over the course of the last 70 years. Among technologies such as the laser and magnetic resonant imaging (MRI), perhaps the greatest impact has been made through the manipulation of semiconductor technology. Since the invention of the first transistor in 1947 [1], this semiconductor technology has laid the groundwork for scalable computers, bringing us into the information age.

The development of this technology comes at a critical time in the conventional silicon industry. The famous Moore's law, hypothesised in its current form in 1975, stated that the number of transistors per square inch would double every two years. This law models the exponential scaling of computing power since its conception incredibly well, as we can see in figure ??. This was driven partly by the cost and power consumption per transistor going down as feature sizes decreased [2]. However, now increasingly small feature sizes have resulted in energy efficiencies and profitability are starting to plateau, while technical issues continue to increase . These issues are in part due to quantum effects such as tunnelling.

This example of tunnelling is related to the so-called 'wave-particle duality' which is the second key feature of quantum mechanics. which is causing these issues may the solution in addition to the problem. The distinction between waves and particles, whose behaviour is well established in classical physics, becomes blurred. **Every object in the universe, if the correct energy and length scales are chosen, will display both of these aspects to some degree.**

It is in this property in which lies the tantalising promise of quantum computing. Particles like electrons, which have comprised the backbone of electricity and classical information for the past century, have the ability to behave in a wave-like manner: they could contain not just the binary bit choices of 0 or 1, but one of an infinite number of continuous values, called 'qubits'. Combined with entanglement to allow our qubits to influence each other while in this state, we can harness a sort of parallelism that results from the wave-like nature of controlled particles.

While in general it is doubtful that a quantum computer will be generically 'faster' than a classical computer, and it is much harder to engineer, we do have significant potential to outperform conventional computers at certain tasks. While the amount of information processing



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

in a conventional computer scales linearly with the number of bits, a quantum computer scales exponentially with the number of qubits for these tasks. Thus adding a single extra qubit could double the computing power. These are discussed, along with the quantum algorithms used to implement them, in Chapter 5. At the point when quantum computers are able to outperform classical supercomputers at a task, the so-called 'quantum supremacy' will have been achieved.

These tasks range from...

However, achieving this potential does not come without significant difficulty. Readout or detection of the information in the qubit destroys the information contained within, resulting in us reverting to the classical bit values with some probability. Furthermore, the technology is still very young and undeveloped. Algorithms exist for many of the applications above, but there may be many more as yet undiscovered. Academic institutions , large corporations (including Google [3], IBM, Intel) and small start-ups (Rigetti, [4]) alike have invested heavily in hardware. There are a wide range of platforms and architectures, including but not limited to superconducting qubits [3], ion traps [REF], quantum dots [REF], spin qubits in silicon [REF], and silicon photonics [REF].

One crucial area that remains comparatively underdeveloped is software. It will be crucial to provide this missing link between the theoretical algorithm and its implementation on a quantum computer. Ideally 'quantum' programming should adopt many of the features as its classical counterpart: it should be usable by any person without understanding the details of the hardware

being used, it

Over the next few years and decades quantum computing is likely to become a reality. It will be crucial when this becomes the case that people are able to understand how to use these machines in order to harness their applicability to the areas of mathematics, computer science, chemistry and finance. This guide is designed to be an introduction to the science of quantum computers and the current state of the field. Initially we explain in more depth how quantum computers work and their differences to classical computers in Chapter 3. Since in the short-term, quantum computers are likely to be noisy, error-prone and limited in scale, we discuss how they can be used in this regime in 4.

Once the engineering of quantum computers have been improved, then a host of more impressive applications can be demonstrated, which are considered in chapter 5. We examine the programming languages that will be able to interface between the algorithms and the quantum computer in chapter 6, and hardware-specific implementations and architectures in chapter 7.

A more complete description of quantum mechanics is given in chapter **??** for any interested party.

2.1 Why you should be interested in quantum computers

- Use laws of QM
-

2.2 What are quantum computers

2

2.3 Traditional computers shortcomings & quantum supremacy

- Information that can be stored in qubit vs bit
- Definition of quantum supremacy

3 Weird Vector things

The fundamental unit of information in classical computing is the bit. which takes one of two states usually represented with 0 or 1. Modern silicon chips realise this as the 'on' or 'off' state of a transistor. In an analogy with bits, the fundamental unit for quantum computing is the qubit. The represents the simplest, fully quantum system

```
vec = [a,b]
```

3.1 Quantum Circuits

This section briefly reviews the gate model for circuit based quantum computing and discusses the similarities between between digital and quantum computers. The gate model is one of the most popular architectures for quantum computation at the moment. *Intel [5] IBM [6], Google [7], Rigetti [8], ...* **REF!** are all using the gate model approach for quantum computing.

Both forms of computation follow the same structure, you start with bits (or qubits), operations are performed on the (qu)bits and then you measure the new values of the (qu)bits. We show an example in Fig. 1.

(a) Digital operation

(b) Quantum operation

Figure 1: resource flow

This

3.1.1 Digital logic

Every digital computing operation can be built up from NAND logic gates [9]. We call this a universal gate for computation.

(a) The NAND logic gate [10].

(b) The NOR logic gate [11].

reversible logic

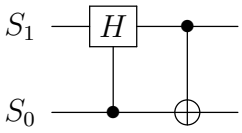


Figure 3: Schur transform for 2 qubits

4 Short term quantum computing

In this section we aim to give a comprehensive overview of quantum computing platforms which are currently available and discuss the near-term advantages that these platforms can bring.

4.1 Adiabatic quantum computing & quantum annealers

Dwave, NISC QUBO problems.

Quantum Algorithms in Dwave? optimisation? Quantum Monte Carlo speed-up? Simulating quantum systems

4.2 Rigetti- Forest

Rigetti has created an environment called Forest, amongst its contributions is a python library called PyQuil. With PyQuil one can simulate up to 26 qubits on the Quantum Virtual Machine (QVM). An example code from [8] is detailed below with comments to assist the reader:

```
from pyquil.quil import Program
from pyquil.gates import H, CNOT
from pyquil.api import SyncConnection
# construct a Bell State program
p = Program()
p.inst(H(0))
p.inst(CNOT(0, 1))
# run the program on a QVM
qvm = SyncConnection()
result = qvm.wavefunction(p)
# produces the output wavefunction of the Bell state
```

Rigetti also offers access to a 19 qubit processor they call 19Q. This API

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

```

```
                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

4.2.1 Example Codes

4.3 IBM- Project Q

IBM has launched the IBM Q experience that consists of a development environment called QISkit and a higher-level gate-building platform that allows users to compose their own algorithms. The devices used to perform the simulation and computation are based on a superconducting charge qubit implementation, which can be found described in more detail in section 7. This development environment is also based on python, as we saw with Forest by rigetti. There is ample space for development of algorithms in this environment as the visual arrays provide a clear structure to those familiar with quantum computation. The composer also displays the code on which it operates so that users interested in further development have the opportunity to learn how to code their own gates in QISkit. The associated documentation which is linked on the IBM Q website provides more detail to the structure [12]. Updates are regularly posted to the main site. Additional features such as limited access to a larger register of bits are available to academics. Specifications of the device that is being used to perform these calculations are available on the website as seen in Fig. 4.

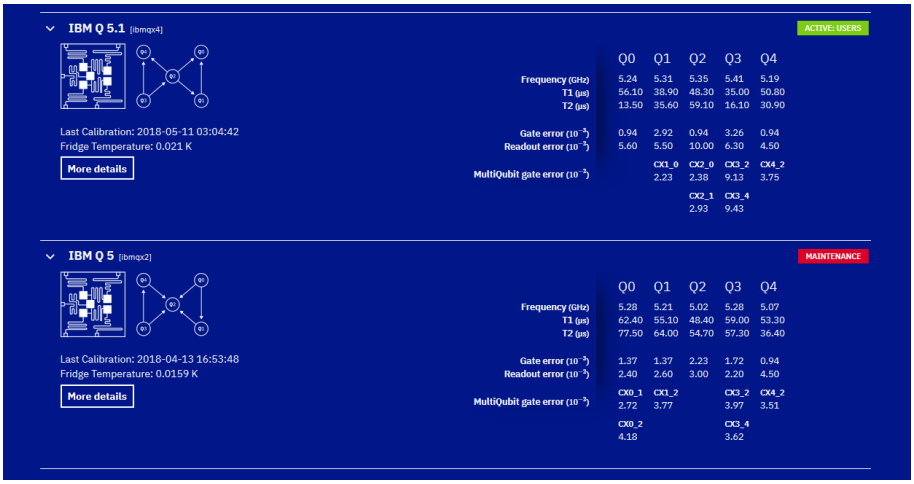


Figure 4: The description of the devices used for the IBM Q suite as of early May 2018 (must be updated closer to submission). Cooling refrigerator temperature along with the update structure is presented. A useful live display feed allows users to track both the physical and computational side of their own algorithms made on the composer.

4.3.1 Example Codes

Examples of composed codes in this environment are based on adding sequences of pre-defined gates. An example of two single-qubit operations being performed on bit 0 out of the 0-4 register.

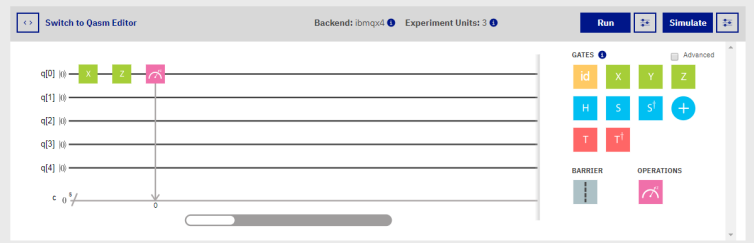


Figure 5: An example of the composition suite that allows gates to be strung together. In this instance, the first qubit in the register has the X and Z gates performed on it before measurement. The symbolism is identical to the standard gate model used within this guide, which makes the transition from other environments simpler.

5 Algorithms and applications

We now discuss the three primary types of algorithms that take advantage of a quantum computer.

5.1 Quantum transforms

1

5.1.1 Quantum Fourier transform

2

5.1.2 Schur transform

3

5.2 Number theory algorithms

4

5.2.1 Shor's algorithm

5

5.2.2 Discrete Logarithm problem

6

5.3 Oracular algorithms

7

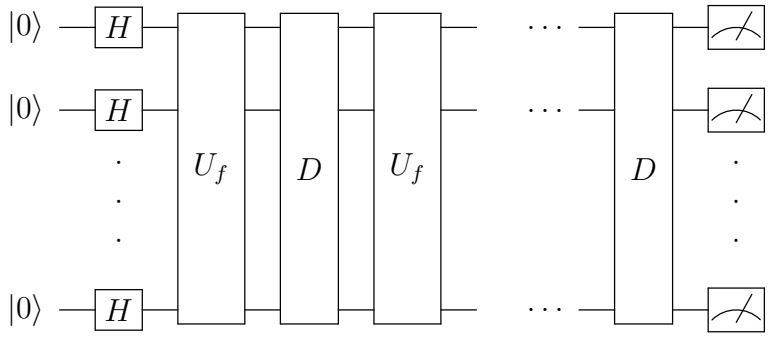
5.3.1 Grover's algorithm

Grover's algorithm solves the unstructured search problem in $O(\sqrt{N})$ queries [13, 14] compared to average number of $O(N)$ queries required for classical algorithms. Although Grover's algorithm does not allow quantum computers to solve NP-complete problems in polynomial time, it does provide a quadratic speedup over classical algorithms. Grover's algorithm is based on the principle of selective phase-shifting on one state (which satisfies a certain condition) of a quantum system at each iteration. A simple case of Grover's algorithm for $N = 2^n$ elements with exactly one marked element has been described in the following box:

We are given access to $f : \{0,1\}^n \to \{0,1\}$ with the promise that $f(x_0) = 1$ for a unique element x_0. We use a quantum circuit on n qubits with initial state $|0\rangle^{\otimes n}$. Let H denote the Hadamard gate, and let U_0 denote the n-qubit operation which inverts the phase of $|0^n\rangle$: $U_0 |0^n\rangle = -|0^n\rangle$, $U_0 |x\rangle = |x\rangle$ for all $x \neq 0^n$.

1. Apply $H^{\otimes n}$.
2. Repeat the following operations T times, for some T to be determined later:
   (a) Apply U_f, where $U_f |x\rangle = (-1)^{f(x)} |x\rangle$.
   (b) Apply D, where $D = -H^{\otimes n} U_0 H^{\otimes n}$
3. Measure all the qubits and output the result.

In circuit diagram form, Grover's algorithm appears like:

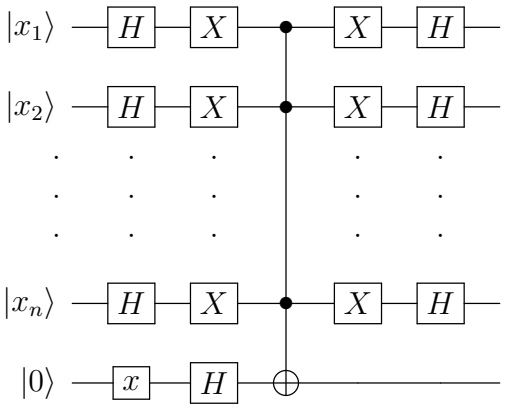


Circuit complexity

In Grover's algorithm, gates U_f and D are applied on the qubits repeatedly until the desired state is found. It is the underlying hardware of these two gates that decide the intrinsic cost of Grover's algorithm per iteration [**?**]. This means that the total time complexity of Grover's algorithm is the number of iterations ($O(\sqrt{N})$) multiplied by the "overhead" provided by the gates U_f and D. The hardware construction of gate U_f depends upon the problem being solved by the algorithm while the hardware for gate D is independent of the problem and only depends upon the number of qubits required by the algorithm.

5.3.2 Construction of gate D

The gate D for n-qubits can be implemented in the following manner. This shows that gate D requires $2n + 1$ Hadamard (H) gates, $2n + 1$ Pauli X gates and an n-control Toffoli gate.



5.3.3 Construction of gate U_f

The action of U_f on qubits is defined as,

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle \tag{1}$$

Therefore, the underlying hardware of U_f depends upon the function, $f(x)$.

5.3.4 The hidden subgroup problem

9

5.4 Approximations & Simulating quantum systems

10

5.4.1 Approximating Matrix powers

11

5.4.2 Approximating Partition functions

12

6 Programming a future universal quantum computer

Three languages will be covered here: Q#, Quil and Scaffold.

6.1 Implementing Shor's algorithm

2

7 Implementations

— PLAN —
Focus on implementation of quantum programming languages
Implementing quantum computers:

- Comparison of quantum and classical computer architectures Classical computer architectures: ALU, CPU, buses, memory, instructions, etc. equivalents for quantum computers? Quantum architectures: gates? Cluster states? Quantum vs. classical control? What are the analogues of instructions, execution, buses and memory? Are there fundamental limits to architecture design (imposed by e.g. no cloning. Does memory make sense in the same way as for classical computers? Quantum memory is more like single-use working-registers). How about classical computers with quantum instructions? what about qbranch instructions?

- Different quantum computing platforms Probably just a few – maybe linear optical quantum computers and iron traps.

- Classical language compilers

Implementing quantum programming:

- Quantum error correction Not entirely analogous to classical computers, (no instruction execution errors). Can error correction be abstracted away from the compiler level? Would error correction take place at the equivalent level of (say) instruction pipelining in a classical processor?

- Quantum language compilers What is the quantum equivalent of object code? Does it contain machine code (instructions to be executed, analogous to classical computers) or does it compile into an arrangement of gates, or some other computation mode (measurements in a cluster state?). The architecture of the quantum computer would probably heavily inform the structure of low level languages. (For example, C has basic structures essentially based on mov, branch, and arithmetic and bit manipulation instructions). Hence the low level languages of gate based quantum computers will be based on structures easily realised using gates (the gates themselves, presumably analogous to bit manipulation; coherent arithmetic, (implemented using gate arrangements lifted from half and full adders); flow control – presumably classical; and strict mov operations, i.e. excluding copy). Low level languages targeted at cluster state implementation will have cluster state measurements as primitive operations (equivalent to bit manipulation) in the language, and presumably various other generic operations (like control, branching, etc.) What kind of optimisations does the C compiler do, and what are the equivalents in the quantum cases?

— PLAN —

7.1 Computer architecture and programming languages

7.1.1 Classical computer architecture

7.1.2 Low level classical languages

7.1.3 Compilers and abstraction

7.2 Quantum computer architectures

7.2.1 Overview

7.2.2 What are the qubits?

7.2.3 What are the operations?

7.2.4 Putting it all together

7.3 Comparison of classical and quantum architectures

7.4 Low level quantum programming languages

7.5 Quantum compilers and high level languages

7.6 The future

8 Advanced topics

1

8.1 Quantum mechanics: The basics

In this section we cover some supplementary background quantum theory. Although a deeper knowledge of quantum theory would serve to further your understanding of quantum computing, it should not be necessary for sections 1 to 7. You have seen in Section (?) that the state of a system governed by quantum mechanics can be represented by a binary 1 or 0, similar to that of a classical bit. However this is not the only two options. In general the state of a bit can now be in a superposition of both 1 and 0, collapsing to one or the other when measured. In this section we will cement this description into a more formal language that will help you to probe deeper into more advanced literature.

8.1.1 Quantum States

In quantum mechanics, we describe the state of a system simply with labels. These labels we assign to the system, such as '0','1', aim to give some intuition about the state of the system. We could equally have used 'open', 'closed' if we were trying to describe the state of a door. Formally these labels are called quantum numbers and in general are not restricted to be one of two values.

For example, imagine the 4 of spades was chosen from a deck of cards, a good choice of label to describe the card would be '4' or 'spade'. Equally in a quantum system we would say that the card is in the state '4' or 'spade' which we write formally as $|4\rangle$, or $|spade\rangle$. In this scenario the quantum number '4' could have been one of 13 values, therefore the set of states needed to fully describing the system are $\{|A\rangle, |2\rangle, |3\rangle, ..., |K\rangle\}$ with quantum numbers $\{A, 2, 3, ...K\}$. If the set of quantum numbers are unique and their associated states describe all possible values a properties can take then they are said to form a Hilbert space $\mathcal{H}$ of the system. Formally we say the set of independent and orthonormal states span a Hilbert space:

$$\mathcal{H} := span\{|A\rangle, |2\rangle, |3\rangle, ..., |K\rangle\} \tag{2}$$

The Hilbert space should loosely be thought of as a vector space. It's purpose is to mathematically define all the possible states a system can be in. This is a very useful tool when we start to describe the evolution of a system because it has better be the case that our description of a system remains physically possible. For example, it would make no sense to talk about the state $|A\rangle$ evolving to the state $|spade\rangle$. In that sense, the Hilbert space helps define the boundaries of your system.

The notation $|...\rangle$ is called a 'ket' and for every 'ket' there is a 'bra' written as $\langle ...|$. The names originates from the first and second halves of the word 'braket', which when placed together resemble the most important operation in quantum computation: the inner product. The inner product is a simple function that does the following:

if state $|a\rangle$ is the same as state $|b\rangle$ return 1 else return 0

In quantum physics notation this operation is performed by turning the ket $|a\rangle$ into a bra, $\langle a|$. The 'bra' $\langle a|$ and 'ket' $|b\rangle$ are then used to form the word 'braket' and is equal to 1 or 0 depending on whether the state a is equal to the state b.

For example, returning to our deck of cards, the inner product of the states $|A\rangle$ with $|5\rangle$ is written as:

$$\langle A|5\rangle = 0 \tag{3}$$

Conversely, the inner product of the states $|J\rangle$ with $|J\rangle$ would be:

$$\langle J|J\rangle = 1 \tag{4}$$

When a set of state are unique in this way, they are said to be orthonormal. The inner product is important as it allows you to check that all the state that form your Hilbert space are orthonormal and will become very useful when we introduce superposition in the next section.

8.1.2 Superposition

In quantum mechanics, using the states that form the basis of our Hilbert space to describe the system is not enough. Observation of quantum systems tell us that when we prepare a state multiple times and measure it, the outcome will not always be the same. At this point its crucial to

point out that the formalism does not try to explain why this is the case, but only provides a way of describing the system. I highlight this fact for the following reason. Typically when confronted with a new phenomena, we turn to the mathematical description to gain some intuition of its causes. With quantum mechanics, the formal description should not be used to find a 'logical' explanation of how superposition works but only to help fully describe the observed physics of the system.

To understand how we can describe this phenomena formally, lets make our deck of cards a quantum deck of cards that now obeys the laws of quantum mechanics and see how we can reflect our observations within the maths.

The first thing to note is that the act of measurement appears to perform an operation on the system that takes it from its superposition state to a basis state of the Hilbert space.

For example, lets say you are given a face down card that when flipped multiple times is sometimes a queen and sometimes a 4. Before performing the operation of turning it over the state of the card should be thought of as being in a superposition of $|Q\rangle$ and $|4\rangle$. Only under the operation of flipping it does it become one of the two basis states. Repeating many times would allow us to build up a picture of the probabilities of getting either one of the other. Since all we have to describe the system pre-measurement are the probabilities of getting each state after measurement, this is what we will use.

Formally, given a state $|\psi\rangle$ that is said to be in superposition of the states $|a\rangle$ and $|b\rangle$ where the probability of getting the state $|a\rangle$ is $|\alpha|^2$ and the probability of getting the state $|b\rangle$ is $|\beta|^2$ then we describe the state as:

$$|\psi\rangle = \alpha\,|a\rangle + \beta\,|b\rangle \tag{5}$$

In general, both α and β can take complex values and so to ensure that the probabilities are real we take the absolute value squared. The significance of having complex values will be discussed later on but for now its sufficient to consider them as real.

Returning to our example, say we get a queen one third of the time and a four two thirds of the time. We would describe the pre-measurement state as:

$$|\psi\rangle = \frac{1}{\sqrt{3}}\,|Q\rangle + \frac{2}{\sqrt{6}}\,|4\rangle \tag{6}$$

where

$$\left|\frac{1}{\sqrt{3}}\right|^2 + \left|\frac{2}{\sqrt{6}}\right|^2 = 1 \tag{7}$$

as expected since the probabilities of getting a queen or a four must sum to one. Consider a more general case where the state is in a superposition over all possible state in the Hilbert space.

Lets call this state $|\phi\rangle$ and it is given by:

$$|\phi\rangle = \sum_{i=0}^{n} c_i |\phi_i\rangle \tag{8}$$

8.1.3 Entanglement

4

8.1.4 Errors & Decoherence

5

8.2 Error correcting codes

6

8.3

`https://github.com/ot561/qprogramming/tree/master`

References

[1] John Bardeen and Walter Hauser Brattain. The transistor, a semi-conductor triode. *Physical Review*, 74(2):230, 1948.

[2] Tim Cross. After Moore's law. *The Economist Technology Quarterly*, 2016.

[3] https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

[4] https://www.rigetti.com.

[5] https://newsroom.intel.com/press-kits/quantum-computing/.

[6] https://www.research.ibm.com/ibm-q/.

[7] https://research.google.com/teams/quantumai/.

[8] https://www.rigetti.com/forest.

[9] Henry Maurice Sheffer. A set of five independent postulates for boolean algebras, with application to logical constants. *Transactions of the American mathematical society*, 14(4):481–488, 1913.

[10] https://commons.wikimedia.org/wiki/File:NAND_ANSI_Labelled.svg.

[11] https://commons.wikimedia.org/wiki/File:NOR_ANSI_Labelled.svg.

[12] Patrick J Coles, Stephan Eidenbenz, Scott Pakin, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, et al. Quantum algorithm implementations for beginners. *arXiv preprint arXiv:1804.03719*, 2018.

[13] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[14] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79:325–328, Jul 1997.