Quantum Meta-Programming for Dummies

Cohort 4

Quantum Engineering CDT
University of Bristol

May 1, 2018

1 Preface

1

Contents

2 Introduction

1

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

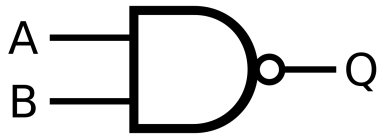
```
vec = [a,b]
```

3.1 Quantum Circuits

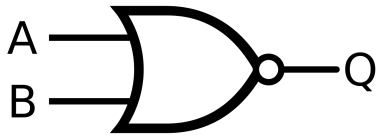
This section aims to show that it is possible to change digital computing into something that is compatible with quantum computers.

3.1.1 Digital logic

Every digital computing operation can be built up from NAND logic gates [1]



(a) The NAND logic gate [2].



(b) The NOR logic gate [3].

reversible logic

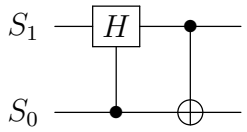


Figure 2: Schur transform for 2 qubits

.

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 [4] 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-ProjectQ

4.3.1 Example Codes

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

8

5.3.2 The hidden subgroup problem

9

5.4 Approximations & Simulating quantum systems

10

5.4.1 Approximating Matrix powers

11

6.4.1 Language 1

12

6.5 The universal quantum computer

13

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, which don't experience 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 —

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 8.

8.1.1 Quantum states & Dirac notation

2

8.1.2 Superposition

Unlike classical representations, quantum states can be formed from superpositions of the computational basis elements of the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Taking the inner product allows us to evaluate the probability of measuring a result $\langle\psi|\psi\rangle = |\alpha|^2 \langle 0|0\rangle + |\beta|^2 \langle 1|1\rangle$. α and β represent the weighting of the states they precede, hence $\mathrm{P}_0 = |\alpha|^2$ and $\mathrm{P}_1 = |\beta|^2$. In general, states may be formed of an arbitrarily long superposition of basis states $|\phi_n\rangle$, as in the example given in 1.

$$|\Phi\rangle = \sum_{0}^{n} \frac{1}{\sqrt{n!}} |\phi_n\rangle \tag{1}$$

3

8.1.3 Entanglement

4

8.1.4 Errors & Decoherence

5

8.2 Error correcting codes

6

8.3

References

[1] 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.

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

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

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