QLang: Qubit Language

Christopher Campbell Clément Canonne Sankalpa Khadka Winnie Narang
Jonathan Wong

September 24, 2014

1 Introduction

In 1965, Gordon Moore predicted that the number of transistors in integrated circuits would double every two years [Moo98]; up until now, this prediction has been seen to hold with quite an uncanny success. However, due to physical properties of the circuits, the classical architecture of computers cannot sustain such a increase rate.

Amongst the possible "fixes" that could allow the race for computational power to go on, a very promising candidate emerged from the last century's advances in Physics; namely, from the field of Quantum Mechanics. Since Richard Feynman suggested in 1982 a new paradigm of computation based on the laws of Quantum physics [Fey82], there has been great progress made in formulating the principles of *quantum computers*, and in studying both their possibilities and limitations (see e.g. [Sho98, Wil08, NC11]). The objective of this project is to design a language that exploits the mathematical foundations and syntax conventions of quantum computing to facilitate the simulation of quantum algorithms.

1.1 Quantum computing

In classical computing, data are stored in the form of binary digits or bits. A *bit* is the basic unit of information stored and manipulated in a computer, which in one of two possible distinct states (for instance: two distinct voltages, on and off state of electric switch, two directions of magnetization, etc.). The two possible values/states of a system are represented as binary digits, 0 and 1.

1.1.1 Qubit

In a quantum computer, data are stored in the form of *qubits*, or quantum bits. A quantum system of n qubits is a Hilbert space of dimension 2^n; fixing any orthonormal basis, any *quantum state* can thus be uniquely written as a linear combination of 2^n orthogonal vectors $\{|i\rangle\}_i$ where i is an n-bit binary number.

Example 1.1. A 3 qubit system has a canonical basis of 8 orthonormal states denoted $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$.

Upshot. A classical bit has only two states, either 0 or 1. However a qubit can have states $|0\rangle$ and $|1\rangle$, or any linear combination of states also known as a *superposition*:

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ are any complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

1.1.2 Quantum Gates

Logical operations, also known as *logical gates*, are the basis of computation in classical computers. Computers are built with circuit that is made up of logical gates. The examples of logical gates are AND, OR, NOT, NOR, XOR, etc. Similarly, a *quantum gate* is an operation which is a *unitary transformation* on qubits. The quantum gates are represented by matrices, and a gate acts on n qubits is represented by $2^n \times 2^n$ unitary matrix[1]. Analogous to the classical computer which is built from an electrical circuit containing wires and logic gates, quantum computers are built from quantum circuits containing "wires" and quantum gates to carry out the computation.

More on this, as well as the definition of the usual quantum gates; can be found in Appendix A.

1.2 Dirac notation of quantum computation

In quantum computing, *Dirac notation* is used to represent qubits. Dirac notation provides concise and intuitive representation of complex matrix operations.

More precisely, a column vector $\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$ is represented as $|\psi\rangle$, also read as "ket psi". In particular, the computational basis states, also know as *pure states* are represented as $|i\rangle$ where i is a n-bit binary number. For example,

$$|000\rangle = \begin{bmatrix} 1\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, |001\rangle = \begin{bmatrix} 0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix}, |010\rangle = \begin{bmatrix} 0\\0\\1\\0\\0\\0\\0\\0 \end{bmatrix}, \ldots, |101\rangle = \begin{bmatrix} 0\\0\\0\\0\\0\\1\\0\\0 \end{bmatrix}, |110\rangle = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\1\\0 \end{bmatrix}, |111\rangle = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{bmatrix}$$

Similarly, the row vector $\begin{bmatrix} c_1^* & c_2^* & \ldots & c_n^* \end{bmatrix}$, which is also complex conjugate transpose of $|\psi\rangle$, is represented as $\langle\psi|$, also read as "bra psi".

The inner product of vectors $|\varphi\rangle$ and $|\psi\rangle$ is written $\langle\varphi \mid \psi\rangle$. The tensor product of vectors $|\varphi\rangle$ and $|\psi\rangle$ is written $|\varphi\rangle \otimes |\psi\rangle$ and more commonly $|\varphi\rangle|\psi\rangle$. We list below a few other mathematical notions that are relevant in quantum computing:

- z^* (complex conjugate of elements)
  if $z = a + ib$, then $z^* = a - ib$.
- A^* (complex conjugate of matrices)
  if $A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix}$ then $A^* = \begin{bmatrix} 1 & -6i \\ -3i & 2-4i \end{bmatrix}$.

[1]That is, a matrix $U \in \mathbb{C}^{2^n \times 2^n}$ such that $U^\dagger U = I_{2^n}$, where $\cdot^\dagger$ denotes the Hermitian conjugate.

- A^{T} (transpose of matrix A)
  if $A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix}$ then $A^{\mathrm{T}} = \begin{bmatrix} 1 & 3i \\ 6i & 2+4i \end{bmatrix}$.
- $A^{\dagger}$ (Hermitian conjugate (adjoint) of matrix A)
  Defined as $A^{\dagger} = \left(A^{\mathrm{T}}\right)^{*}$; if $A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix}$ then $A^{\dagger} = \begin{bmatrix} 1 & -3i \\ -6i & 2-4i \end{bmatrix}$
- $\||\psi\rangle\|$ (ℓ_2 norm of vector $|\psi\rangle$)
  $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$. (This is often used to normalize $|\psi\rangle$ into a unit vector $\frac{|\psi\rangle}{\||\psi\rangle\|}$.)
- $\langle\varphi|A|\psi\rangle$ (inner product of $|\varphi\rangle$ and $A|\psi\rangle$).
  Equivalently[2], inner product of $A^{\dagger}|\varphi\rangle$ and $|\psi\rangle$

1.3 Quantum Algorithms

A quantum algorithm is an algorithm that, in addition to operations on bits, can apply quantum gates to qubits and measure the outcome, in order to perform a computation or solve a search problem. Inherently, the outcome of such algorithms will be probabilistic.The representation of a quantum computation process requires an input register, output register and unitary transformation that takes a computational basis states into linear combination of computational basis states. If x represents an n qubit input register and y represents an m qubit output register, then the effect of a unitary transformation U_f on the computational basis $|x\rangle_n|y\rangle_m$ is represented as follows:

$$U_f(|x\rangle_n|y\rangle_m) = |x\rangle_n|y \oplus f(x)\rangle_m, \tag{1}$$

where f is a function that takes an n qubit input register and returns an m qubit output and $\oplus$ represents mod-2 bitwise addition.

1.3.1 Deutsch-Jozsa Algorithm

Deutsch's Problem is an example of a quantum algorithm that performs computation in fewer steps than a classical computer.
Suppose $f\colon \{0,1\}^n \to \{0,1\}$ and that f is either constant or balanced. The goal is determine which one it is. Classically it is trivial to see that this would require (in worst case) querying just over half the solution space, or $2^{n-1}+1$ queries. The Deutsch-Jozsa algorithm answers this question with just *one* query!

2 Qubit Language

The objective of `QLang` language is to allow simulation of quantum algorithm.

2.1 Syntax

An operation, or language elementary unit, starts from the end of the previous one, and ends whenever a semicolon or a line return is encountered. In particular, a semicolon at the end of the line is valid, yet not necessary.

[2]Recall that we work in a complex Hilbert space: the inner product is a sesquilinear form.

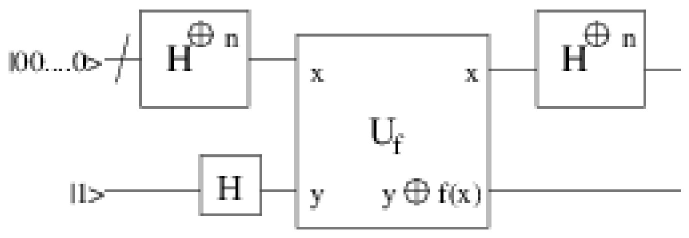

Figure 1: Quantum Circuit for Deutsch-Jozsa Algorithm

2.1.1 Comments

Comments will be preceded by symbol #; after the symbol, the remainder of the line is considered as a comment. No multiline comment is supported.

```
# This is just a comment.
```

2.1.2 Variable Declaration

Variables are declared with the name of the variable followed by equals and value assigned to the variable.

```
a = 5.9
b = 4+5i
c = |00>
```

A variable name must start with a lower case letter, and can only contain `ASCII` characters: `a-aA-Z`, `0-9`, `_`: that is, it must match the pattern `[a-z][a-zA-Z0-9_]*`.

2.1.3 Reserved keywords

The following words are reserved, and cannot be variable names: `i`, `if`, `elif`, `else`, `def`, `while`, `for`, `from`, `to`, `by`, `true`, `false`, as well as the identifiers of the builtin constructs of Section 2.1.5.

2.1.4 Data Types

- Complex numbers
  Quantum computation is done over complex vector space and hence one of the main data types is complex number.

```
a = 5.9
b = 4+5i
```

All numbers are complex; the keyword `i` is reserved and refers to the usual value (imaginary unit i). By default, all numbers are internally stored as real numbers. The implementation takes care of precision issues in a transparent fashion: the syntax does not differentiate between different floating-point precisions, and the programmer can implicitly assume all numbers have "infinite precision".

The two particular keywords `true` and `false` are constants, equal to respectively 1 and 0, and provided for notational convenience. Although we refer to them as Booleans, they are internally considered are complex numbers and can be used as such.

- Qubits
  Qubit representation follows Dirac notation.

```
a=|1000>
b =<001|
c= |100> + | 110>
```

- Vector and Matrices
  In essence, all data type are matrices.

```
a=[ 1, 1, 3]
a=[ 1, 2, 3 ; 4, 5, 6; 7, 8, 9]
```

2.1.5 Operations

Mathematical operations such as multiplication, division, addition, tensor product matrix multiplication are supported by the language.

```
a = 5*5
b = 5*(6+7i)
c = H|00>
  d = |00> @ |1>
M = [ 1, 2, 3 ; 4, 5, 6; 7, 8, 9]
N = [1, 2; 5, 6]
D = M @ N
```

Other builtin constructs are `transp`, `adj`, `isunit`, `det` (for matrices), `norm`, `conj` (for complex numbers). Note that no support by default is provided for matrix inversion; this choice is due to the fact that most matrices considered will (and have to) be unitary, in which case getting the inverse is straightforward which the above.

2.1.6 Control Flow

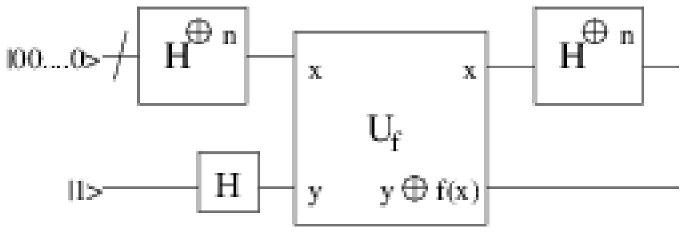

Figure 2: Quantum Circuit for Deutsch-Jozsa Algorithm

```
# if-elif-else statement
if ( variable == true){
}elif (boolean){
}else{
}
#for loop, increment by 2 (the ‘by 2’’ is optional
for k from 1 to 10 by 2 {
}
```

2.1.7 Functions

```
def output = function_name(args){
}
```

2.2 Simulation of Deutsch-Jozsa Algorithms

```
def outcome = deutschjozsa(in-qubit, U-matrix){

  input-registers = in-qubit @ |1>;
  after-Hadamard = (H @ H)*(input-registers);
  after-U = U-matrix * after-Hadamard;
  after-Had = (H @ I)*after-U;
 M0=((in-qubit*(in-qubit)^H) @ I)*after-Had;

  if (M0 == 0){
    outcome = 0;
  }
  else{
    outcome = 1;
  }

}
```

The Team

Manager: Jonathan Wong

Language Guru: Sankalpa Khadka

System Architect: Winnie Narang

System Integrator: Clément Canonne

Verification and Validation: Christopher Campbell.

These roles are not definitive: each member of the team will work on every aspect of the project, while being the main responsible for its particular area.

References

[Fey82] RichardP. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. 1

[Moo98] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, Jan 1998. 1

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011. 1

[Osk02] Mark Oskin. Quantum Computing – Lecture Notes. `http://homes.cs.washington.edu/~oskin/teaching.html` (accessed 09/2014), 2002.

[Sho98] Peter W. Shor. Quantum computing. *Documenta Mathematica*, 1:1000, 1998. 1

[Wil08] Colin P. Williams. *Explorations in Quantum Computing*. Springer Publishing Company, Incorporated, 2nd edition, 2008. 1

A More on quantum computing and the underlying mathematics

A.1 Common quantum gates

Pauli Operators

The *Pauli operators* are the special single qubit gates which are represented by the Pauli matrices $\{I, X, Y, Z\}$ as follows

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

For example, the application of X causes bit-flip in following ways:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle.$$

Hadamard Gate

The *Hadamard gate* is defined by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The Hadamard gate maps the computational basis states into superposition of states. The Hadamard gate is significant since it produces maximally entangled states from basis states in the following ways:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Controlled-U Gates

A *controlled-U gate* is the quantum gate in which the U operator acts on the n^{th} n-qubit only if the value of the preceeding qubit is 1.
For example: In a Controlled-NOT gate, the NOT operator flips the second qubit if the first qubit is 1.

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathsf{CNOT}|00\rangle = |00\rangle$$
$$\mathsf{CNOT}|01\rangle = |01\rangle$$
$$\mathsf{CNOT}|10\rangle = |11\rangle$$
$$\mathsf{CNOT}|11\rangle = |10\rangle.$$

A.2 Tensor product and its properties

Let $A = (a_{i,j})$ be a matrix with respect to the ordered basis $\mathcal{A} = (u_1, \dots, u_n)$ and $B = (b_{i,j})$ be a matrix with respect to the ordered basis $\mathcal{B} = (v_1, \dots, v_m)$. Consider the ordered basis $\mathcal{C} = (u_i \otimes v_j)$ ordered by lexicographic order, that is $u_i \otimes v_j \leq u_l \otimes v_k$ if if $i < l$ or $i = l$ and $j < k$. The matrix of $A \otimes B$ with respect to $\mathcal{C}$ is :

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,n}B \end{bmatrix}$$

This matrix is called the tensor product of the matrix A with the matrix B.

- $A \otimes B \otimes C = (A \otimes B) \otimes C = A \otimes (B \otimes C)$
- $a(|x\rangle \otimes |y\rangle) = a|x\rangle \otimes |y\rangle = |x\rangle \otimes a|y\rangle$
- $(A \otimes B) \cdot (|y\rangle|z\rangle) = A|y\rangle \otimes B|z\rangle$
- $(A \otimes B) \cdot (C \otimes D) = AC \otimes BD$
- $(A \otimes B)^H = A^H \otimes B^H$
- If A and B unitary, $A \otimes B$ is unitary.
- If $|x\rangle = |x_1\rangle|x_2\rangle$ and $|y\rangle = |y_1\rangle|y_2\rangle$ then $\langle x|y\rangle = \langle x_1|y_1\rangle\langle x_2|y_2\rangle$