Review of Quantum Computing Research

Cris Cecka

Summer 2005

This paper documents many of the quantum computing research paths taken in the summer of 2005 under the guidance of Professor Christopher Stone.

In the last decade, Quantum computing has generated a large amount of interest in Physicists, Mathematicians, and Computer Scientists alike. Physical problems such as quantum state entanglement, isolation to prevent quantum decoherence, and mechanisms for implementation of basic quantum operators have taken many paths including study in ion capture devices, NMR architectures, quantum optics, and quantum dot technologies. Mathematical problems such as error correction codes, development of new quantum algorithms, and operator decomposition. These studies have recently sparked much development into new, generalized descriptions of the structure of quantum algorithms. Computer Science issues such as language architectures, algorithms, and theoretical frameworks have had much interest and development in the last decade. Overall, this is an exciting field spanning multiple disciplines.

In this paper, I will review the topics covered in my research opportunity with Professor Stone, describe the significance of many of these topics, and remark on future research possibilities.

1 Languages and Quantum Computation

The quantum computation model, has been through, and still exists in, many forms. Classical computation models such as the Turing machine, Lambda calculus, and circuit representation have all been extended to encompass quantum information. Currently, the most efficient forms of representation of quantum algorithms include the circuit model and the associated operator calculus. In this model, quantum bits are manipulated by abstracted operators which have a mathematical construction independent of implementation.

From this model, multiple quantum computing languages have been developed which attempt to provide a complete framework for simulating and verifying algorithms within the circuit-operator model. From experience in working with quantum computing languages as well as from a theoretical standpoint, computer scientists have constructed the following list of requirements which any useful quantum computing language must satisfy.

- **Abstracted Quantum Model**

Any language must provide a high level construction to allow programmers to develop modular, intuitive, and compact code. Thus, the language must have some automated mechanism for translation to a sequence of low level instructions for a quantum machine.

- **Hardware Independence**

  Any quantum computing language must not make reference to the hardware of the quantum machine, only the operators which any quantum machine should be capable of. The automated translation to low level instructions may take responsibility for this independent of the programmer and his code.

- **Quantum Registers**

  There are three basic operations that a quantum register (qureg) must be able to perform for the user.

  - **Allocation and Deallocation -** This will likely only involve the addressing and scoping of a qureg.
  - **Initialization -** In order to initialize a qureg, it must be measured and operated upon, two attributes which clearly must be properties of any quantum language.
  - **Concatenation -** Operators must have the ability to access any subset of the quregs available to the system. This could be accomplished in a number of ways. Physical movement of quregs could allow passage through an operator together which would be implemented as addressing manipulation in the quantum language. Alternatively, operators could be constructed and understood to be used as state operators which would be inherently applied to the entire quantum machine state with identity operations for unused qubits. In this instance, the implementation would be manipulation of operator structure rather than addressing schemas.
  - **Measurement** A measurement of a quantum register will collapse the quantum state to an observable basis. Although this will be governed by the hardware, a generic `measure` command should be available.

- **Quantum Operators**

  Operators may have a number of possible implementations. Properties of operators within languages which contributed to readable, efficient code are described below.

  - **Low Level Operators** Common operators which must form a complete set (redundancy is fine) should be quickly accessible.

– **Operator Composition** From these low level operators, higher level operators can be constructed through composition. Thus, a user may conjugate many operators into a single operator which may be used more than once.

– **Operator Inversion** Because all quantum operators must be unitary, they must be reversible. Inverting an operator should be easy and accessible.

– **Classical Function and Permutation Operators** Though not available in many quantum languages which were researched, automatic construction of quantum operators from classical functions or permutation maps should be accessible.

A functional operator has the form

$$\hat{U}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

while a permutation operator has the form

$$\hat{U}_\sigma|x\rangle = |\sigma(x)\rangle$$

These attributes, and any others which make the language more clear and easier to manipulate, should make up any quantum computing language. They will be used as a basis for discussing the ease and versatility of QCL and Q-Language, the two quantum computing languages studied in depth over the summer.

1.1 QCL

Our research began with the quantum computing language QCL, written by Bernard Omer and documented in his paper "A Procedural Formalism for Quantum Computing" which can be found at http://tph.tuwien.ac.at/ oemer.

QCL is a high level programming language for quantum computers, with a syntax derived from classical procedural languages like C or Pascal. This allows for the complete implementation and simulation of quantum algorithms (including classical components) in one consistent formalism.

QCL provides a subroutine hierarchy which forces a structure on the code that provides an inherent separation of classical computing from quantum computing. There are four levels of computation.

- `procedure`s may elicit any effect on the program or machine state. This may include allocation and deallocation of classical and quantum information, interpretation of quantum measurements, take input and produce output, and are simply the most general routine type.

- `operator`s may represent any unitary operator. Since this procedure must be unitary, it must be reversible and functional, for any given input it must result in one and only one output or transformation. Thus, an `operator`

may have no side effects and no dependencies on random numbers (including quantum measurements) or global variables. In QCL, these subroutines may be reversed by preceeding a call with the `!` operator. eg, `!foo(myQuantumBit)`.

- `functions` restrict the user to functional manipulations of classical data types.
- `qufunct` type subroutines are pseudo-classical quantum operators in that they may be used to defined quantum operations of the type

$$\hat{U}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

I spent a good portion of the research period in familiarizing myself with QCL. I became familiar with QCL by implementing various, simple pseudo-classical arithmetic and logical operators. When QCL had become comfortable, I began brainstorming various ways of extending the already implemented examples of Grover's algorithm and Deutsch-Jozsa's algorithm, for which the mathematical escapades are documented below.

Some issues with QCL arose which I also considered confronting. The zero-failure Grover algorithm is most easily implemented with a controlled-rotation operator such as

$$\hat{R}_c(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\phi) & -\sin(\phi) \\ 0 & 0 & \sin(\phi) & \cos(\phi) \end{pmatrix}$$

which rotates the second bit by ϕ on the condition that the first bit is $|1\rangle$. Unfortunately, the given `Rot` operator in QCL is not conditional by default. That is, it cannot be contained within a (quantum) `if` statement. The subroutine hierarchy prevents any work-around to easily make it conditional. Attempting to simply hard code the operator as a matrix failed due to round off error causing a "non unitary operator application" error to be thrown when the matrix operator was checked for orthonormality.

Furthermore, the built in `Swap` function in the most recent version of QCL was broken when used in a conditional context. The problem was found and fixed in `extern.cc` which defines all built-in operators and Bernhard Ömar, the author of QCL, was contacted about the problem and informed of the solution.

1.2 Q-Language

Q-language is a quantum computation language written as a C++ library. It is much less readable than QCL, but also seems to be implemented much more independently from a quantum simulator.

2 Algorithms

2.1 Deutsch-Jozsa

Consider a function $f\colon \mathcal{B}^n \mapsto \mathcal{B}$ such that $\sum_{x\in\mathcal{B}^n} f(x) \in \{0, 2^{n-1}, 2^n\}$. That is, the function is either constant and always returns zero or one, or balanced and returns an equal number of zeroes and ones. The Deutsch-Jozsa algorithm uses quantum computing to determine which class of functions f belongs to.

First, we begin with the machine in the quantum state:

$$|\Psi_1\rangle|aux_1\rangle = |0\rangle^{\otimes n}|1\rangle$$

where $|aux\rangle$ is the auxiliary qubit used in the application of the oracle to provide a phase inversion. It will not be changing after the next step

Next, We apply the Hadamard Transform $H^{\otimes n} \otimes H$ to the machine state to produce the state

$$|\Psi_2\rangle|aux_2\rangle = \sum_{x\in\mathcal{B}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

We now apply the oracle operator U_f. This operator has the action $\hat{U_f}|x\rangle|y\rangle = |x\rangle|y \oplus x\rangle$. Thus, applying this operator will yield

$$\hat{U_f} \frac{1}{\sqrt{2}\sqrt{2^n}} \sum_{x\in\mathcal{B}^n} |x\rangle|0\rangle - |x\rangle|1\rangle$$

$$= \frac{1}{\sqrt{2}\sqrt{2^n}} \sum_{x\in\mathcal{B}^n} |x\rangle|f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle$$

$$= \frac{1}{\sqrt{2}\sqrt{2^n}} \sum_{x\in\mathcal{B}^n} (-1)^{f(x)} \left(|x\rangle|0\rangle - |x\rangle|1\rangle\right)$$

$$|\Psi_3\rangle|aux_2\rangle = \sum_{x\in\mathcal{B}^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

Finally, applying the Hadamard transform $H^{\otimes n}$ to the first register yields

$$|\Psi_4\rangle|aux_2\rangle = \frac{1}{2^n} \sum_{z\in\mathcal{B}^n} \sum_{x\in\mathcal{B}^n} (-1)^{x\cdot z + f(x)}|z\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

where $x \cdot z$ is a bit-wise dot-product of x and z.

Consider the $z = 0$ state. The amplitude for this state is:

$$\langle 0|^{\otimes n}|\Psi_4\rangle = \frac{1}{2^n} \sum_{x\in\mathcal{B}^n} (-1)^{f(x)} = \begin{cases} -1 & \text{f(x) = 1 for all x} \\ 0 & \text{f(x) is balanced} \\ 1 & \text{f(x) = 0 for all x} \end{cases}$$

Thus, we can measure the first register to determine the class of f. If we obtain the state $|0\rangle^{\otimes n}$, f must be balanced, otherwise it is constant.

2.2 Generalization of Deutsch-Jozsa

Here, we asked what classes of functions the Deutsch-Jozsa algorithm would be able to detect in the case that f was a function $f\colon \mathcal{B}^n \mapsto \mathcal{B}^m$. The most straight-forward generalization would be the following.

First, we begin with the machine in the quantum state:

$$|\Psi_1\rangle|aux_1\rangle = |0\rangle^{\otimes n}|1\rangle^{\otimes m}$$

where $|aux\rangle$ is the auxiliary qubit used in the application of the oracle to provide a phase inversion. It will not be changing after the next step

Next, We apply the Hadamard Transform $H^{\otimes n} \otimes H^{\otimes m}$ to the machine state to produce the state

$$|\Psi_2\rangle|aux_2\rangle = \left[\sum_{x\in\mathcal{B}^n} \frac{|x\rangle}{\sqrt{2^n}}\right]\left[\sum_{y\in\mathcal{B}^n} \frac{(-1)^{y\cdot\{1\}^m}|y\rangle}{\sqrt{2^m}}\right]$$

We now apply the oracle operator U_f. This operator has the action $\hat{U_f}|x\rangle|y\rangle = |x\rangle|y \oplus_m x\rangle$. Thus, applying this operator will yield

$$|\Psi_3\rangle|aux_2\rangle = \sum_{x\in\mathcal{B}^n}\left[\frac{|x\rangle}{\sqrt{2^n}} \sum_{y\in\mathcal{B}^n} \frac{(-1)^{(y\oplus f(x))\cdot\{1\}^m}|y\rangle}{\sqrt{2^m}}\right]$$

Finally, applying the Hadamard transform $H^{\otimes n}$ to the first register yields

$$|\Psi_4\rangle|aux_2\rangle = \frac{1}{\sqrt{2^n}\sqrt{2^m}} \sum_{z\in\mathcal{B}^n}\sum_{x\in\mathcal{B}^n}\sum_{y\in\mathcal{B}^m} (-1)^{x\cdot z+(y\oplus_m f(x))\cdot\{1\}^m}|z\rangle|y\rangle$$

So that the amplitude for measuring $|z\rangle = |0\rangle^{\otimes n}$ is

$$\frac{1}{\sqrt{2^n}\sqrt{2^m}} \sum_{x\in\mathcal{B}^n}\sum_{y\in\mathcal{B}^m} (-1)^{(y\oplus_m f(x))\cdot\{1\}^m}|y\rangle$$

We note that this amplitude is zero if $f(x)$ is constant with respect to parity and is one if it is balanced.

It was pointed out that by changing the intermittent transforms (the Hadamards), the balance condition could be manipulated. However, this, and all other extensions of this algorithm is simply equivalent to defining a new function $g_f\colon \mathcal{B}^n \mapsto \mathcal{B}$ which maps the interested subset from the image of f to the binary domain. Thus, the original Deutsch-Jozsa algorithm can now be used to determine if this is constant or balanced and nothing is gained by running this revamped version on f itself, even with an alternate intermittent transformation.

2.3 Grover's Algorithm

Consider a function $f\colon \mathcal{B}^n \mapsto \mathcal{B}$ such that $f(x_0) = 1$ and $\sum_x f(x) = 1$. Using the same initialization and execution of the oracle $\hat{U_f}$ as in the Deutsch-Jozsa algorithm, we obtain the state

$$|\Psi_3\rangle|aux_2\rangle = \sum_{x\in\mathcal{B}^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

so that only $|x\rangle = |x_0\rangle$'s sign is flipped.

We define the mean amplitude $\langle\alpha\rangle$ as

$$\langle\alpha\rangle = \frac{1}{2^n}\sum_k \alpha_k$$

where the α_k define the general quantum vector

$$|\Psi\rangle = \sum_k \alpha_k|k\rangle$$

Consider the operator

$$\hat{M} = \hat{H}^{\otimes n}\left(2|0\rangle\langle 0| - \sum_i |i\rangle\langle i|\right)\hat{H}^{\otimes n} = \frac{2}{2^n}\left(\sum_i\sum_j |i\rangle\langle k|\right) - \hat{I}$$

where $\hat{I}$ is the identity operator.

This operator, when applied to the general quantum state $|\Psi\rangle$ has the action

$$\begin{aligned}
\hat{M}|\Psi\rangle &= \left(\frac{2}{N}\sum_{ij}|i\rangle\langle j| - I\right)\left(\sum_k \alpha_k|k\rangle\right) \\
&= \frac{2}{N}\left(\sum_{ik}\alpha_k|i\rangle\right) - \sum_k \alpha_k|k\rangle \\
&= \frac{2\sum_k \alpha_k}{N}\sum_k |k\rangle - \sum_k \alpha_k|k\rangle \\
&= \sum_k \left(2\langle\alpha\rangle - \alpha_k\right)|k\rangle
\end{aligned}$$

That is, it inverts each state about $\langle\alpha\rangle$.

2.4 Incomplete Mean Inversion

We thought it reasonable for an operator to exist which would only take into account a subset of all the possble states. That is, zeroed amplitudes would be ignored in the inversion. This would allow the amplification of certain states while keeping the entire state of the machine "pure" in that new states would

not be introduced. A simple arguement shows that no such operator can exist. Consider the state $\{\alpha_1, \alpha_2, \ldots, \alpha_N\} = \{.5, .5, .5, -.5\}$ which has an active average $\langle\alpha\rangle = .25$ and would evolve under this proposed operator $\hat{D}$ to $\{0, 0, 0, 1\}$. This evolved state has an active averege of $\langle\alpha\rangle = 1$ and would evolve under $\hat{D}$ to $\{0, 0, 0, 1\}$. Thus $\hat{D}\{0, 0, 0, 1\} = \{0, 0, 0, 1\} = \hat{D}^\dagger\{0, 0, 0, 1\}$. Thus, this state is a fixed point of the operator so neither the operator or it's inverse will be able to produce the original state. Therefore, the operation is not invertable and cannot be a realizable quantum unitary operation.

It is interesting to note that the operator which applies the inversion to a subset of the possible quantum states does exist given that subset. Consider the set $A \subseteq \mathcal{B}^n$. The operator $\hat{M}_A = \frac{2}{|A|}\sum_{i,j\in A}|i\rangle\langle j| - I$ realizes this subset inversion:

$$\begin{aligned}
\hat{M}_A|\Psi\rangle &= \left(\frac{2}{|A|}\sum_{i,j\in A}|i\rangle\langle j| - I\right)\left(\sum_k \alpha_k|k\rangle\right) \\
&= \frac{2}{|A|}\left(\sum_{i,j\in A}\alpha_j|i\rangle\right) - \sum_k \alpha_k|k\rangle \\
&= \frac{2\sum_{i\in A}\alpha_i}{|A|}\sum_{j\in A}|j\rangle - \sum_k \alpha_k|k\rangle \\
&= \sum_{i\in A}\left(2\frac{\sum_{i\in A}\alpha_i}{|A|} - \alpha_i\right)|i\rangle + \sum_{i\notin A}\alpha_i|i\rangle
\end{aligned}$$

By letting $A \mapsto \mathcal{B}^n$, we see that $\frac{\sum_{i\in A}\alpha_i}{|A|} \mapsto \langle\alpha\rangle$ and we retrieve the Grover operator.

2.5 Zero Failure Grover Algorithm

The generalized Grover Operator $\hat{G}(\phi, \varphi)$ is given by

$$\hat{G}(\phi, \varphi) = \hat{H}\hat{S}_{\delta_{x,0}}(\phi)\hat{H}^{-1}\hat{S}_f(\varphi) = \hat{H}\left((1 - e^{i\phi})|0\rangle\langle 0| - I\right)\hat{H}^{-1}\hat{S}_f(\varphi) \quad (1)$$

Here, the $\hat{S}$ is an oracle operator which uses a function $f\colon \mathcal{B}^n \mapsto \mathcal{B}$ and operates on an arbitrary state by

$$\hat{S}_f(\phi)|\Psi\rangle = \hat{S}_f(\phi)\left(\sum_k \alpha_k|k\rangle\right) = \sum_k \alpha_k e^{i\phi\delta_{f(k),1}}|k\rangle \quad (2)$$

Note that the well known Grover Operator used previously:

$$\hat{G} = \hat{H}\left(2|0\rangle\langle 0| - I\right)\hat{H}\hat{\Theta} = \left(\frac{2}{N}\sum_{ij}|i\rangle\langle j| - I\right)\hat{\Theta} \quad (3)$$

with $\hat{\Theta}$ being the phase-inversion oracle is simply a special case with $\phi = \varphi = \pi$, $\hat{G} = \hat{G}(\pi, \pi)$. (Note that $\hat{G}$ can be further generalized to act with an arbitrary quantum unitary operator $\mathcal{A}$ in place of the Walsh-Hadamard Transform $\hat{H}$, but is beyond the scope and purpose of this paper.)

First, we use the Hadamard Transform to create an equal superposition of states. Let $N = 2^n$ be the number of states in this superposition. Also, assume we know the value of $a = \frac{1}{N}\sum_x f(x) = t/N$. That is, we know the number of target states, t, the oracle will be nontrivially operating on in each application. Let $|\Psi\rangle_0 = H|0\rangle = \frac{1}{\sqrt{N}}\sum_k |k\rangle$ be our initial state. It is clear that our initial state can be written

$$|\Psi\rangle_0 = \sqrt{\frac{t}{N}}|\Psi_1\rangle + \sqrt{\frac{N-t}{N}}|\Psi_0\rangle = \sqrt{\frac{t}{N}}\sum_{k|f(k)=1}|k\rangle + \sqrt{\frac{N-t}{N}}\sum_{k|f(k)=0}|k\rangle \quad (4)$$

and therefore we can use $|\Psi_1\rangle = \sum_{k|f(k)=1}|k\rangle$ and $|\Psi_0\rangle = \sum_{k|f(k)=0}|k\rangle$ to construct the orthonormal basis $|\vec{f}\rangle = (\sqrt{\frac{1}{t}}|\Psi_1\rangle, \sqrt{\frac{1}{N-t}}|\Psi_0\rangle)$. We must now determine the actions of the operators in this basis to determine the general form of the Grover Operator.

$$\text{Clearly,} \quad \hat{S}_f(\varphi) \hookrightarrow_{|\vec{f}\rangle} \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & 1 \end{pmatrix} \quad (5)$$

and to determine the matrix representation of $\hat{H}\hat{S}_{\delta_{x,0}}(\phi)\hat{H}^{-1}$, we will determine its action on the basis states.

$$\begin{aligned}
\hat{H}\hat{S}_{\delta_{x,0}}(\phi)\hat{H}^{-1}\begin{pmatrix} 1 \\ 0 \end{pmatrix}_{|\vec{f}\rangle} &= \left(\frac{(1-e^{i\phi})}{N}\sum_{i,j}|i\rangle\langle j| - I\right)\left(\frac{1}{\sqrt{t}}\sum_{k|f(k)=1}|k\rangle\right) \\
&= \frac{(1-e^{i\phi})}{N}\sqrt{t}\sum_i |i\rangle - \frac{1}{\sqrt{t}}\sum_{k|f(k)=1}|k\rangle \\
&= \frac{(1-e^{i\phi})\sqrt{t}}{N}\left(\sum_{i|f(i)=1}|i\rangle + \sum_{i|f(i)=0}|i\rangle\right) - \frac{1}{\sqrt{t}}\sum_{k|f(k)=1}|k\rangle \\
&= \frac{(1-e^{i\phi})t - N}{N}\frac{1}{\sqrt{t}}\sum_{i|f(i)=1}|i\rangle + \frac{(1-e^{i\phi})\sqrt{t}\sqrt{N-t}}{N}\frac{1}{\sqrt{N-t}}\sum_{i|f(i)=0}|i\rangle \\
&= ((1-e^{i\phi})\frac{t}{N} - 1)\begin{pmatrix} 1 \\ 0 \end{pmatrix}_{|\vec{f}\rangle} + (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}}\begin{pmatrix} 0 \\ 1 \end{pmatrix}_{|\vec{f}\rangle}
\end{aligned} \quad (6)$$

$$
\begin{aligned}
\hat{H}\hat{S}_{\delta_{x,0}}(\phi)\hat{H}^{-1}\begin{pmatrix}0\\1\end{pmatrix}_{|\vec{f}\rangle} &= \left(\frac{(1-e^{i\phi})}{N}\sum_{i,j}|i\rangle\langle j| - I\right)\left(\frac{1}{\sqrt{N-t}}\sum_{k|f(k)=0}|k\rangle\right)\\
&= \frac{(1-e^{i\phi})}{N}\sqrt{N-t}\sum_{i}|i\rangle - \frac{1}{\sqrt{N-t}}\sum_{k|f(k)=0}|k\rangle\\
&= \frac{(1-e^{i\phi})\sqrt{N-t}}{N}\left(\sum_{i|f(i)=1}|i\rangle + \sum_{i|f(i)=0}|i\rangle\right) - \frac{1}{\sqrt{N-t}}\sum_{k|f(k)=0}|k\rangle\\
&= \frac{(1-e^{i\phi})\sqrt{t}\sqrt{N-t}}{N}\frac{1}{\sqrt{t}}\sum_{i|f(i)=1}|i\rangle + \frac{(1-e^{i\phi})(N-t)-N}{N}\frac{1}{\sqrt{N-t}}\sum_{i|f(i)=0}|i\rangle\\
&= (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}}\begin{pmatrix}1\\0\end{pmatrix}_{|\vec{f}\rangle} - ((1-e^{i\phi})\frac{t}{N}+e^{i\phi})\begin{pmatrix}0\\1\end{pmatrix}_{|\vec{f}\rangle}
\end{aligned}
\tag{7}
$$

Thus, our operator has the matrix representation

$$
\begin{aligned}
\hat{G}(\phi,\varphi) &= \hat{H}\hat{S}_{\delta_{x,0}}(\phi)\hat{H}^{-1}\hat{S}_f(\varphi)\\
&\hookrightarrow_{|\vec{f}\rangle}\begin{pmatrix}(1-e^{i\phi})\frac{t}{N}-1 & (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}}\\ (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}} & (-(1-e^{i\phi})\frac{t}{N}-e^{i\phi})\end{pmatrix}\begin{pmatrix}e^{i\varphi}&0\\0&1\end{pmatrix}\\
&= \begin{pmatrix}((1-e^{i\phi})\frac{t}{N}-1)e^{i\varphi} & (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}}\\ (1-e^{i\phi})\sqrt{\frac{t}{N}}\sqrt{1-\frac{t}{N}}e^{i\varphi} & -((1-e^{i\phi})\frac{t}{N}+e^{i\phi})\end{pmatrix}
\end{aligned}
\tag{8}
$$

An interesting note is that when $\phi=\varphi=\pi$ and we let $\frac{t}{N}=\sin^2\theta$, this operator becomes

$$
\hat{G}(\pi,\pi)\hookrightarrow_{|\vec{f}\rangle}\begin{pmatrix}\cos 2\theta & \sin 2\theta\\ -\sin 2\theta & \cos 2\theta\end{pmatrix} \tag{9}
$$

When applied m times, this operator has the total effect

$$
\hat{G}^m(\pi,\pi)\hookrightarrow_{|\vec{f}\rangle}\begin{pmatrix}\cos 2m\theta & \sin 2m\theta\\ -\sin 2m\theta & \cos 2m\theta\end{pmatrix} \tag{10}
$$

Thus, we should determine an operator $\hat{Q}$ such that

$$
\hat{G}^m(\pi,\pi)\hat{Q}|\Psi\rangle_0 = \hat{G}^m(\pi,\pi)\hat{Q}\begin{pmatrix}\sqrt{\frac{t}{N}}\\ \sqrt{1-\frac{t}{N}}\end{pmatrix}_{|\vec{f}\rangle} = \begin{pmatrix}1\\0\end{pmatrix}_{|\vec{f}\rangle} \tag{11}
$$

So

$$
\hat{Q}\begin{pmatrix}\sin\theta\\ \cos\theta\end{pmatrix}_{|\vec{f}\rangle} = \hat{G}^{-m}(\pi,\pi)\begin{pmatrix}1\\0\end{pmatrix}_{|\vec{f}\rangle} = \begin{pmatrix}\cos 2m\theta\\ \sin 2m\theta\end{pmatrix}_{|\vec{f}\rangle} \tag{12}
$$

which can be realized as

$$
\hat{Q}\hookrightarrow_{|\vec{f}\rangle}\begin{pmatrix}\cos\omega & \sin\omega\\ -\sin\omega & \cos\omega\end{pmatrix} \tag{13}
$$

since

$$\begin{pmatrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{pmatrix} \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix} = \begin{pmatrix} \sin\theta\cos\omega + \cos\theta\sin\omega \\ \cos\theta\cos\omega - \sin\theta\sin\omega \end{pmatrix} = \begin{pmatrix} \sin(\theta+\omega) \\ \cos(\theta+\omega) \end{pmatrix} \tag{14}$$

and when $\omega = \pi/2 - (2m+1)\theta$, this is satisfied.

2.6 Matrix Decomposition and the Deutsch-Jozsa Problem

In this section, I wanted to investigate the theoretical operator

$$\hat{D}_f|00\rangle = |f(0)f(1)\rangle$$

which could essentially act as a direct extension to the Deutsch-Jozsa solution, providing a full description of $f\colon \mathcal{B} \mapsto \mathcal{B}$ rather than just determining whether it is in the balanced or constant class.

Logically and philosophically, we would expect that, given the typical oracle operator

$$\hat{O}_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

or, choosing $\hat{H}|1\rangle$ as a scratch qubit as in the Deutsch-Jozsa examples, this is effectively

$$\hat{O}_f|x\rangle = (-1)^{f(x)}|x\rangle$$

so that one representation, taking advantage of an implicit scratch register, would be

$$\hat{O}_f \hookrightarrow_{\mathbf{Q}_1} \begin{pmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{pmatrix}$$

Our goal is to now show that at least two of these oracle operators would be required to implement the theoretical $\hat{D}_f$. To do this, we must, as generally as possible, show that any decomposition of $\hat{D}_f$ must involve two applications of $\hat{O}_f$, no better than the classical result. However, on possibility is that, based on the parallelization of quantum operators (which would be dependent on the hardware), a decomposition may lend itself to natural parallelization.

We would like to define and decompose the $\hat{D}_f$ operator. To do this, it is not difficult to determine what the first column of the matrix representation should be to implement the desired operation above.

$$\hat{D}_f \hookrightarrow_{\mathbf{Q}_2} \begin{pmatrix} \delta_{f(0),0}\delta_{f(1),0} & \cdots & \cdots & \cdots \\ \delta_{f(0),0}\delta_{f(1),1} & \cdots & \cdots & \cdots \\ \delta_{f(0),1}\delta_{f(1),0} & \cdots & \cdots & \cdots \\ \delta_{f(0),1}\delta_{f(1),1} & \cdots & \cdots & \cdots \end{pmatrix} = \begin{pmatrix} \overrightarrow{\delta_f} & \cdots & \cdots & \cdots \end{pmatrix}$$

The ellipses are "Don't Care"s as long as they are chosen appropriately such that the operator is unitary. Without loss of generality we may choose the last

three columns to simply be permutations of the delta vector. As we will see later, this choice for the operator is actually more general than it first seems since any operator that attempts to project to one of four orthogonal spaces based on f and using some input is simply this operator with a unitary change of basis:

$$\hat{A}_f = \hat{T}^{-1}\hat{D}_f\hat{T}$$

Now, we have reduced the problem to finding an algebraic representation for the delta vector $\overrightarrow{\delta_f}$. To do this, we require a representation algebra. We would like this to relate to the terms within the original oracle operator. The following algebraic basis was found.

$$\mathbf{F} = \{(-1)^{f(0)+f(1)}, (-1)^{f(1)}, (-1)^{f(0)}, 1\}$$

We see that these generators (basis vectors) are orthogonal with respect to the set of binary functions $f\colon \mathcal{B} \mapsto \mathcal{B}$, $\{(f(0), f(1))\} = \{(0,0),(0,1),(1,0),(1,1)\}$

$$\overrightarrow{(-1)^{f(0)+f(1)}} \hookrightarrow_f \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad \overrightarrow{(-1)^{f(0)}} \hookrightarrow_f \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \quad \overrightarrow{(-1)^{f(1)}} \hookrightarrow_f \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \quad \overrightarrow{1} \hookrightarrow_f \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

From these vector representations over the set of binary functions, we can create a transformation matrix to translate a linear combination of "vectors" in $\mathbf{F}$ to their values over the set of binary functions. This is given by

$$\mathbf{T}_{\mathbf{F}\mapsto\mathbf{V}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \mathbf{H}_4$$

where $\mathbf{H}_4$ is interestingly the 4th-order Hadamard Matrix.

We should now be able to express our delta vector in terms of these algebraic primitives and find how the result relates to the original oracle. To do this, we must solve the linear equation

$$\hat{T}_{\mathbf{F}\mapsto\mathbf{V}}\overrightarrow{x} = \overrightarrow{\delta_f}$$

Since the delta vector becomes four orthogonal vectors when evaluated over the set of binary functions and the Transformation matrix is easily invertible, we can simply solve each case and take the sum of the solutions. In compact matrix notation, this comes out to

$$\overrightarrow{\delta_f} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} (-1)^{f(0)+f(1)} \\ (-1)^{f(0)} \\ (-1)^{f(1)} \\ 1 \end{pmatrix}$$

This now defines the column of $\hat{D}_f$ that we were interested in and we can fill in the rest of the operator with other orthonormal vectors. One way of doing this is

$$\hat{D}_f = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} (-1)^{f(0)+f(1)} & (-1)^{f(0)+f(1)} & (-1)^{f(0)+f(1)} & (-1)^{f(0)+f(1)} \\ (-1)^{f(0)} & -(-1)^{f(0)} & (-1)^{f(0)} & -(-1)^{f(0)} \\ (-1)^{f(1)} & (-1)^{f(1)} & -(-1)^{f(1)} & -(-1)^{f(1)} \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$= \frac{1}{4}\mathbf{H}_4\hat{A} = \frac{1}{2}\hat{H}^{\otimes 2}\hat{A}$$

Attempts to further decompose $\hat{A}$ were made, but were largely unsuccessful. Two interesting alternatives popped up however. First, since the set of binary functions is of cardinality four, it would be possible to do a zero-failure Grover search for the correct solution in one iteration. This would require the construction of a grover oracle from the original functional oracle. That is, we would need

$$\hat{G} = \begin{pmatrix} (-1)^{\delta_{f(0),0}\delta_{f(1),0}} & 0 & 0 & 0 \\ 0 & (-1)^{\delta_{f(0),0}\delta_{f(1),1}} & 0 & 0 \\ 0 & 0 & (-1)^{\delta_{f(0),1}\delta_{f(1),0}} & 0 \\ 0 & 0 & 0 & (-1)^{\delta_{f(0),1}\delta_{f(1),1}} \end{pmatrix}$$

$$= \hat{I} - \text{diag}\overrightarrow{\delta_f} = \hat{I} - \text{diag}\left(\frac{1}{2}\hat{H}^{\otimes 2}\begin{pmatrix} (-1)^{f(0)+f(1)} \\ (-1)^{f(0)} \\ (-1)^{f(1)} \\ 1 \end{pmatrix}\right)$$

which we can see reduces nearly immediately to the previous decomposition problem.

The final approach considered related to the concept of superdense coding. In this approach we shall use an input of

$$|\Psi_0\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

and based on the function f, apply one of the four gates:

$$00 : \hat{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad 01 : \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad 10 : \hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad 11 : i\hat{Y} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Based on the which gate is applied, the input will be sent to one of the four states which make up the Bell Basis:

$$00 : \frac{|00\rangle + |11\rangle}{\sqrt{2}} \qquad 01 : \frac{|00\rangle - |11\rangle}{\sqrt{2}} \qquad 10 : \frac{|10\rangle + |01\rangle}{\sqrt{2}} \qquad 11 : \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

Using the above algebra, we can generalize this to applying the gate

$$\frac{1}{2}\begin{pmatrix} 1 + (-1)^{f(0)} & 1 - (-1)^{f(0)} \\ (-1)^{f(1)} - (-1)^{f(0)+f(1)} & (-1)^{f(1)} + (-1)^{f(0)+f(1)} \end{pmatrix}$$

which, again, we can see is the same decomposition problem, but in a very nice basis. In this case, significant progress was made in finding a decomposition of the operator:

$$
\begin{aligned}
&= \begin{pmatrix} 1 & -(-1)^{f(0)} \\ (-1)^{f(1)} & (-1)^{f(0)+f(1)} \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \\
&= \begin{pmatrix} -(-1)^{f(0)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -(-1)^{f(0)} & 1 \\ (-1)^{f(1)} & (-1)^{f(0)+f(1)} \end{pmatrix} \hat{H} \\
&= \begin{pmatrix} -(-1)^{f(0)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -(-1)^{f(0)} & (-1)^{f(0)} \\ (-1)^{f(1)} & (-1)^{f(1)} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & (-1)^{f(0)} \end{pmatrix} \frac{1}{\sqrt{2}} \hat{H} \\
&= \begin{pmatrix} -(-1)^{f(0)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & (-1)^{f(0)} \end{pmatrix} \frac{1}{\sqrt{2}} \hat{H} \\
&= \begin{pmatrix} -(-1)^{f(0)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & (-1)^{f(0)} \end{pmatrix} \frac{1}{\sqrt{2}} \hat{H} \\
&= \begin{pmatrix} -(-1)^{f(0)} & 0 \\ 0 & 1 \end{pmatrix} \hat{O}_f \hat{H} (-\hat{Z}) \begin{pmatrix} 1 & 0 \\ 0 & (-1)^{f(0)} \end{pmatrix} \hat{H}
\end{aligned}
$$

This is an interesting result for a couple of reasons. First, there is a higher dependence on $f(0)$ than on $f(1)$, although in a similar fashion this could be reversed. A deeper investigation into this decomposition may be able to prove that a decomposition into a sequence of operators containing only one oracle call is, as we suspected, impossible. This investigation would probably have to generalize the expression algebra, generalize the operational basis, and generalize the decomposition procedure.

An argumentative approach to this problem would point out that the information we are attempting to extract from the oracle, both function evaluations, are encoded orthogonally. Thus, through no unitary transformations may the information encoded in one oracle call be manipulated to yield all the information about f. Furthermore, two oracle calls should easily be able to determine which of four possible functions f is simply by projecting o

3 Other Musings

3.1 Basis Transformations

As we saw in the discussion of the Zero-Failure Grover algorithm, a change of basis can simplify an argument greatly. No programming we investigated had an easy, intuitive way to transform a basis in a non-unitary way, operate on it in that basis, then transform back. In principle this should be possible and could potentially make some quantum algorithms (such as the zero-failure Grover algorithm) easier to implement.

Unfortunately, that not the whole story and the actual usefulness of this feature is in question. We'll see why in a moment.

Consider Grover's algorithm. We wish to make a transformation to the "Grover Basis" with the transformation

$$\hat{T}_G = \frac{1}{\sqrt{t}} \sum_{i \in \{t\}} |\alpha\rangle\langle i| + \frac{1}{\sqrt{N-t}} \sum_{i \notin \{t\}} |\beta\rangle\langle i|$$

where t is the number of target states, $\{t\}$ is the set of target states, and $(|\alpha\rangle, |\beta\rangle)$ is our new basis. Thus, this operator would transform a general state into a basis which described the amplitude that that state had in the projection onto the space of all target states and the amplitude that that state had in the projection onto the space of all non-target states. We could then transform back using the operator

$$\hat{T}_G^{-1} = \sqrt{t} \sum_{i \in \{t\}} |i\rangle\langle\alpha| + \sqrt{N-t} \sum_{i \notin \{t\}} |i\rangle\langle\beta|$$

right? It's easily verifiable that

$$\hat{T}_G^{-1} \hat{T}_G = \hat{I}$$

where $\hat{I}$ is the identity operator. So what's wrong with this?

Well, first we note that

$$\hat{T}_G^{-1} \hat{T}_G \neq \hat{T}_G \hat{T}_G^{-1}$$

which is a result of the transformation being non-square. In the standard basis, $\hat{T}_G$ is a $2xN$ matrix, where $N = 2^n$ is the dimension of the state in the standard basis. Thus, although both these sides are identity matrices, they are difference identity matrices, one is $2x2$, the other NxN. Of course, this also implies that $\hat{T}_G$ is not perfectly invertible, something we also could have expected because there must be some loss of information from an N dimensional basis to a 2 dimensional basis.

Ok, but in principle this should still work, we used it in the proof. We must have made some assumptions. The assumption that makes this all work is an understanding that in initially transforming to the Grover Basis all target states have the same amplitude and all non-target states have the same amplitude. Thus, we're already working in a 2 dimensional subspace of the state space and there is no loss of information:

$$|\Psi\rangle = \alpha \sum_{i \in \{t\}} |i\rangle + \beta \sum_{i \notin \{t\}} |i\rangle$$

is a general Grover state where α is the amplitude of the target states and β is the amplitude of the non-target states.

This should, in principle work, then. It's easy, we transform to the Grover basis, operate on it (rotate it towards the target), then transform it back to achieve an equal superposition of all the target states. If we wanted this in a programming language, the language would have to verify the following.

- The current state is in the space spanned by the transformed basis. Since we can't determine the actual state of the quantum register, we'll just have to take the word of the user and assume he knows what he's doing.
- The operators following the transformation may only operate in the transformed basis.
- Inverse transformation - should be automated so the user doesn't have to specify the inverse transformation operator.

This would have to be implemented by constructing unitary matrices from the transformation operators. For example:

$$\hat{T}_G \longmapsto \begin{pmatrix} [T_G]_{2xN} \\ [A]_{(N-2)xN} \end{pmatrix}$$

$$\hat{T}_G^{-1} \longmapsto \begin{pmatrix} \left[T^{-1}\right]_{Nx2} & [B]_{Nx(N-2)} \end{pmatrix}$$

where A and B are matrices of the specified dimensions. These "filler" rows and columns can be constructed with a process such as Graham-Schmidt orthonormalization. If we do assume the state to be transformed is within the space spanned by the transformation rows, the orthonormal rows in A are guaranteed to yield zero amplitudes for states outside of the transformation space, as desired.

Unfortunately, there's one more caveat. In our example, we defined a transformation to make operating within the Grover algorithm more simple. However, to make the transformation operator we need, we must know which states are target states and which are not. But if we knew this, we would be done already. Basis transformations can't be used to make Grover's algorithm simpler in practice. Although this would be an interesting and difficult feature to implement in a programming language, its actual usefulness is questionable.

3.2 Quantum Set Representations?

3.3 Quantum Predicate Logic and Satisfiability

3.4 Quantum Data Structures

References

[1] M. Nielson, I. Chuang, *Quantum Computation and Quantum Information* (Cambridge Univ. Press, Cambridge, 2000)

[2] D'Hondt, Ellie and Prakash Panangaden. *Quantum weakest preconditions.* Proc. QPL 2004, pp. 75-90

[3] Brassard, Gilles and Peter Hoyer, Michele Mosca, Alain Tapp. *Quantum Aplitude Amplification and Estimation.* May 2, 2000.

[4] Tuylsi, Tathagat and Lov Grover, Apoorva Patel. *A new algorithm for directed quantum search.* arXiv:quant-ph/0505007 v2. May 2, 2005.

[5] Hsieh, Jin-Yuan and Che-Ming Li, Der-San Chuu. *An Improved Phase Error Tolerance in a Quantum Search Algorithm.* Chinese Journal of Physics, 42(5). Oct 2004.

[6] Chen, Goong and Zijian Diao. *An exact quantum search algorithm.* Dec 11, 2004.

[7] Clark, Thomas L. and D.J. Kaup. *Quantum Algorithms for Modeling and Simulation: A Grand Challenge for Modeling and Simulation.*

[8] de Vries, Andreas. *Towards fast quantum search algorithms by qubit comparisons exploiting global phase interference.* arXiv:quant-ph/0506137 v2. Jul 3, 2005.

[9] Grover, Lov K. *A fast quantum mechanical algorithm for database search.* Bell Labs.

[10] Hoyer, Peter. *Conjugated operators in quantum algorithms.*

[11] Hoyer, Peter. *Arbitrary phases in quantum amplitude amplification.* Physical Review A, Vol 62. Oct 11, 2000.

[12] Omer, Bernard. *A Procedural Formalism for Quantum Computing.* Jul 23, 1998.

[13] S.Bettelli, T.Calarco, and L.Serafini. *Toward an architecture for quantum programming.* arXiv:cs.PL/0103009 v3. Mar 27, 2003