Quantum Computing and λ_q

Jesse A. Tov

November 30, 2006

1 The Qubit

Quantum mechanics is concerned with states in a Hilbert (vector, inner product) space. Consider bits **0** and **1** in a register. These are orthogonal "basis states". Actual states are superpositions $c_0|\mathbf{0}\rangle + c_1|\mathbf{1}\rangle$ where $\sum_i |c_i|^2 = 1$. We call these quantum states "qubits."

Examples:

$$\begin{aligned}
\psi &= |\mathbf{0}\rangle \\
\psi &= \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) \\
\psi &= \frac{\sqrt{3}}{2}|\mathbf{0}\rangle - \frac{i}{2}|\mathbf{1}\rangle
\end{aligned}$$

The squares of the magnitudes values of the coefficients form a probability distribution, which is what we see when we measure the system.

What space is generated by $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$? $\mathbb{C}^2$, but normalized (a Bloch sphere). The $|\cdot\rangle$ notation is Dirac's. It's nice, but sometimes it helps to think of our space this way:

$$|\mathbf{0}\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |\mathbf{1}\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \qquad \frac{\sqrt{3}}{2}|\mathbf{0}\rangle - \frac{i}{2}|\mathbf{1}\rangle = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{-i}{2} \end{bmatrix}$$

We probably want registers with more qubits in them. A register with n qubits is in the *quantized* space $H_{\mathrm{QB}(n)} = \ell^2(\mathbb{B}^n)$. Each classical state of n bits is one of our computational basis states:

$$|\mathbf{0}\rangle \otimes |\mathbf{0}\rangle = |\mathbf{00}\rangle = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T \qquad |\mathbf{0}\rangle \otimes |\mathbf{1}\rangle = |\mathbf{01}\rangle = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^T$$
$$|\mathbf{1}\rangle \otimes |\mathbf{0}\rangle = |\mathbf{10}\rangle = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}^T \qquad |\mathbf{1}\rangle \otimes |\mathbf{1}\rangle = |\mathbf{11}\rangle = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$$

2 Quantum Computation

To do quantum computation, we evolve this state through the application of unitary, reversible operators. Unitary: Preserves normalization. Reversible: zero entropy. If we throw away information, we generate heat, which causes decoherence. We also can't *duplicate* state, in the sense that there is no meaningful function $\psi \to \psi \otimes \psi$—anything we copy will be coupled.

2.1 Quantum Circuits

By analogy to classical circuits, we can build quantum circuits out of gates. The gates have to be unitary and reversible, of course.

2.2 Classical Gates

Must be reversable. Consider *and*:

$$\mathit{and} = \begin{bmatrix}1 & 1 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

So *and* is no good, because it loses information. (What is $\mathit{and}^{-1}\,0$?) Must not duplicate information. By the Pigeonhole principle, gates are invertible transformations $\mathbb{B}^n \to \mathbb{B}^n$.

Reversible classical logic has been shown to be complete. Some useful classical gates:

2.3 Identity

$$I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

2.3.1 Not

$$not = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Note that $not^{-1} = I$.

2.3.2 Cnot

Also known as invertible *xor*:

$$cnot = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Note that $cnot^{-1} = I$.

2.3.3 Toffoli Gate

Controlled-Controlled-Not. In general,

$$C_U = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$$

The Toffoli gate is a basis for classical (reversible) logic circuits.

2.4 Uncontrolled gates:

We'd like to apply, say, a one-bit gate to a particular bit of a two-bit register. E.g.:

$$(I \otimes U)|\psi_0, \psi_1\rangle = (I \otimes U)(|\psi_0\rangle \otimes |\psi_1\rangle) = |\psi_0\rangle \otimes U|\psi_1\rangle$$

Of course there are permutation gates, too.

2.5 Quantum Gates

To do quantum computations, though, we need some gates that create superpositions and take advantage of the phase space.

2.5.1 Hadamard

The Hadamard gate produces a uniform distribution, but is reversible.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$H|\mathbf{0}\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$$
$$H|\mathbf{1}\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

Note that $H^2 = I$.

2.5.2 Phase

$$R_n = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^n} \end{bmatrix}$$

The gates *cnot*, H, and R_3, along with composition and tensor product, form a basis for the space of unitary, reversible functions $H_{\mathrm{QB}(n)} \to H_{\mathrm{QB}(n)}$.

2.5.3 Pauli Gates

It may also be convenient to have the three Pauli gates:

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Some properties:

- $\sigma_n^2 = I$
- $\sigma_n \sigma_{n+1} = i\sigma_{n+2} \pmod 3$
- $\sigma_n \sigma_m = -\sigma_m \sigma_n \quad (i \neq j)$

2.6 Some Quantum Algorithms

What's quantum computation good for? Factorization, discrete Fourier transform, function inversion (!), coin flips, "teleportation"...

2.6.1 A Coin Flip

$H|\mathbf{0}\rangle$.

2.6.2 Deutsch's Algorithm

For a one-bit function f, $(H \otimes I)U_f(H \otimes H)|\mathbf{01}\rangle$ where $U_f : (x, y) \mapsto (x, y \oplus f(x))$. Why U_f.

The first bit of the result is $\mathbf{1}$ if f is balanced and $\mathbf{0}$ otherwise. This algorithm is *deterministic*.

2.6.3 Einstein, Podolsky, Rosen

The EPR paradox:

$$\begin{aligned}\mathbf{epr} &\equiv \mathit{cnot}(H \otimes I)|\mathbf{00}\rangle \\ &= \mathit{cnot}\,\frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{10}\rangle) \\ &= \frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{11}\rangle)\end{aligned}$$

"Spooky action at a distance." John Bell: No hidden variable (1964).

3 The Quantum Lambda Calculus λ_q

3.1 Syntax

Wheh. Now for λ_q:

$$\begin{aligned} t &::= x \mid \lambda x.t \mid t\ t \mid c \mid\ !t \mid \lambda !x.t && \text{terms} \\ c &::= 0 \mid 1 \mid H \mid R_n \mid \mathit{cnot} \mid T \mid \sigma_1 \mid \sigma_2 \mid \sigma_3 \mid \cdots && \text{constants} \end{aligned}$$

We need to be careful not to throw away or duplicate *non-definite* state. Well-formedness (for preserving linearity):

$$\frac{}{\vdash c}\ \textsc{Const} \qquad \frac{}{x \vdash x}\ \textsc{Id} \qquad \frac{!x_1, \ldots, !x_n \vdash t}{!x_1, \ldots, !x_n \vdash !t}\ \textsc{Promotion} \qquad \frac{\Gamma, x \vdash t}{\Gamma, !x \vdash t}\ \textsc{Dereliction}$$

$$\frac{\Gamma, !x, !y \vdash t}{\Gamma, !z \vdash t[z/x, z/y]}\ \textsc{Contraction} \qquad \frac{\Gamma \vdash t}{\Gamma, !x \vdash t}\ \textsc{Weakening} \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\ \multimap\text{-I}$$

$$\frac{\Gamma, !x \vdash t}{\Gamma \vdash \lambda !x.t}\ \to\text{-I} \qquad \frac{\Gamma \vdash t_1 \qquad \Delta \vdash t_2}{\Gamma, \Delta \vdash t_1\ t_2}\ \multimap\text{-E}$$

3.2 Dynamics

Call-by-value:

$$\begin{aligned} v &::= c \mid \lambda x.t \mid \lambda !x.t \mid !t && \text{values} \\ E &::= [\,] \mid (E\ t) \mid (v\ E) && \text{evaluation contexts} \end{aligned}$$

Reduction rules:

$$\begin{aligned} E[(\lambda x.t)\ v] &\to E[t[v/x]] && (\beta) \\ E[(\lambda !x.t)\ !t'] &\to E[t[t'/x]] && (!\beta) \\ E[|c_U\ \phi\rangle] &\to E[U|\phi\rangle] && (U) \end{aligned}$$

History-preserving rules…?

3.3 Useful Stuff

We need a fixed pointer operator, so let

$$\text{fix} \equiv (\lambda !u.\lambda !f.f\ !(u\ !u\ !f))\ !(\lambda !u.\lambda !f.f\ !(u\ !u\ !f))$$

Then we could verify that fix $!t \to^* t\ !(\text{fix}\ !t)$.

Sugar for pairs:

$$\begin{aligned} () &\equiv \lambda !x.\lambda !y.x\ \text{id} \\ \text{cons} &\equiv \lambda h.\lambda t.\lambda !x.\lambda !y.y\ h\ t \\ \textbf{case}\ t_1\ \textbf{of}\ (() \to t_2, h : t \to t_3) &\equiv t_1\ !(\lambda !z.t_2)\ !(\lambda h.\lambda t.t_3) \end{aligned}$$

Let's assume we also have **let**.

3.4 Nice Properties

Two terms are said to be "congruent" iff they differ only in **1**s and **0**s. Theorem: All superposed terms generated by λ_q are congruent.

If the initial state is definite, all !-suspended terms are definite.

We can embed the classical cbv untyped lambda calculus as follows:

$$\begin{aligned}(t_1\ t_2)^* &= (\lambda !z.z)\ t_1^*\ t_2^* \\ x^* &= !x \\ (\lambda x.t)^* &= !(\lambda !x.t^*)\end{aligned}$$

4 Some Algorithms in λ_q

4.1 Deutsch Revisited

$$\text{deutsch } U_f \to \textbf{let } (x,y) = U_f\ ((H\ 0),(H\ 1))\ \textbf{in } ((H\ x),y)$$

Example: if $f = \mathrm{id}$ then $U_f = \mathit{cnot}$.

$$\begin{aligned}
\text{deutsch } \mathit{cnot} &\to \textbf{let } (x,y) = \mathit{cnot}\ ((H\ 0),(H\ 1))\ \textbf{in } ((H\ x),y) \\
&\to \textbf{let } (x,y) = \mathit{cnot}\ \frac{1}{2}\Big(|(0,0)\rangle + |(1,0)\rangle - |(0,1)\rangle - |(1,1)\rangle\Big)\ \textbf{in } ((H\ x),y) \\
&\to \textbf{let } (x,y) = \frac{1}{2}\Big(|(0,0)\rangle + |(1,1)\rangle - |(0,1)\rangle - |(1,0)\rangle\Big)\ \textbf{in } ((H\ x),y) \\
&\to \frac{1}{2}\Big(|((H\ 0),0)\rangle + |((H\ 1),1)\rangle - |((H\ 0),1)\rangle - |((H\ 1),0)\rangle\Big) \\
&\to \frac{1}{2\sqrt{2}}\Big(|(0,0)\rangle + |(1,0)\rangle + |(0,1)\rangle - |(1,1)\rangle \\
&\qquad - |(0,1)\rangle - |(1,1)\rangle - |(0,0)\rangle + |(1,0)\rangle\Big) \\
&= \frac{1}{2\sqrt{2}}\Big(2|(1,0)\rangle - 2|(1,1)\rangle\Big) \\
&= |1\rangle \otimes \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big)
\end{aligned}$$

Now, what if $f = \lambda x.0$? Then $U_f(x, y) = (x, y \oplus f(x)) = (x, y)$. So,

$$\begin{aligned}
\text{deutsch id} &\to \textbf{let } (x, y) = \mathrm{id}_2\ ((H\ 0), (H\ 1))\ \textbf{in } ((H\ x), y) \\
&\to \textbf{let } (x, y) = \mathrm{id}_2\ \frac{1}{2}\Big(|(0,0)\rangle + |(1,0)\rangle - |(0,1)\rangle - |(1,1)\rangle\Big)\ \textbf{in } ((H\ x), y) \\
&\to \textbf{let } (x, y) = \frac{1}{2}\Big(|(0,0)\rangle + |(1,0)\rangle - |(0,1)\rangle - |(1,1)\rangle\Big)\ \textbf{in } ((H\ x), y) \\
&\to \frac{1}{2}\Big(|((H\ 0),0)\rangle + |((H\ 1),0)\rangle - |((H\ 0),1)\rangle - |((H\ 1),1)\rangle\Big) \\
&\to \frac{1}{2\sqrt{2}}\Big(|(0,0)\rangle + |(1,0)\rangle + |(0,0)\rangle - |(1,0)\rangle \\
&\qquad - |(0,1)\rangle - |(1,1)\rangle - |(0,1)\rangle + |(1,1)\rangle\Big) \\
&= \frac{1}{2\sqrt{2}}\Big(2|(0,0)\rangle - 2|(0,1)\rangle\Big) \\
&= |0\rangle \otimes \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big)
\end{aligned}$$

4.2 Spooky Action

Now we have epr $\equiv$ *cnot* $((H\ 0), 0)$. See teleportation from van Tonder, 29.

4.3 Controlled U

Suppose we have a one-bit function $U : H_{\mathrm{QB}(1)} \to H_{\mathrm{QB}(1)}$. We want to build a two-bit function C_U such that

$$\begin{aligned}
C_U|00\rangle &\mapsto |00\rangle \\
C_U|01\rangle &\mapsto |01\rangle \\
C_U|10\rangle &\mapsto |1\rangle U|0\rangle \\
C_U|11\rangle &\mapsto |1\rangle U|1\rangle
\end{aligned}$$

We have *cnot* $= C_{\mathrm{not}}$. Recall, in general, we can build $C_U = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$. How do we do it in λ_q?

$$
\begin{aligned}
C \equiv \lambda U.\, &\lambda(c, x). \\
&\quad \textbf{let } (x_1, x_2) = \mathit{cnot}(x, 0) \textbf{ in} \\
&\quad \textbf{let } (c', x_1', d_1) = T(\sigma_1 c, x_1, 0) \textbf{ in} \\
&\quad \textbf{let } (c'', x_2', d_2) = T(\sigma_1 c', U x_2, 0) \textbf{ in} \\
&\quad \textbf{let } (d_1', d_2', r) = T(\sigma_1 d_1, \sigma_1 d_2, 1) \textbf{ in} \\
&\qquad (c'', r, x_1', x_2', d_1', d_2')
\end{aligned}
$$

4.4 Fourier Transform

We have lists. We can also write (linear) *map* and *reverse*. See van Tonder, 31.

5 Related Work

5.1 Quantum Turing Machine

Equivalent to the quantum Turing machine. The basis states of $\mathcal{Q}$ are $|x; \mathbf{n}; \mathbf{m}\rangle$ where x is head position, n is a finite state, and m is an infinite memory (which always has only a finite number of $\mathbf{1}$s).

Transitions are represented by a reversible, unitary operator U, so $\psi(t) = U^t|x; \mathbf{0}; \mathbf{n}_0\rangle$. U has some restrictions: it can affect only the bit n_x at any step, and x can change only by 1. There's a universal QTM that takes an encoding of U as part of its input.

Theorem: For any *particular* "finite" QTM, there's a quantum circuit.

Theorem: QTM and λ_q are equally powerful.

5.2 Lambda-q

Another quantum lambda calculus which is strictly more powerful. Can compute NP efficiently. Does this using non-linear operators, so we don't know if it's implementable.

References

[1] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, V22(5):563–591, May 1980.

[2] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[3] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[4] Philip Maymin. The lambda-q calculus can efficiently simulate quantum computers, Feb 1997.

[5] Peter Selinger and Benoit Valiron. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science*, 16(3):527–552, June 2006.

[6] André van Tonder. A lambda calculus for quantum computation, April 2004.

[7] André van Tonder. Quantum computation, categorical semantics and linear logic, October 2004.

[8] Wikipedia. Various articles: "Deutsch-Jozsa algorithm", "Grover's algorithm", "Shor's algorithm", "Quantum gate", "Toffoli gate", "Hadamard transform", "Pauli matrices", "Quantum computer", "Quantum circuit".