April 10

Reading J Tolar, "On Clifford groups in quantum computing"

An N state system corresponds to a hilbert space $\mathbb{C}^N$.

"Generalized Pauli Matrices" generate a group, "Weyl-Heisenberg group", semantics.

The "normalizer" of this is called the Clifford group. So I guess WH is not normal in U(N), but in Clifford group it is. That's surprising to me, but I guess it makes sense given the normal property becomes weaker the less group elements you are conjugating against. So Clifford group is the set $\{g \mid g^{-1}Xg \in WH, \forall x \in WH\}$.

"Clifford quotient group" sounds like Clifford group without scalar multiplication, which sounds good to me. $U(N)$ seems so redundant/free I will take every quotient I can get.

"Symmetries of Pauli gradings" of an algebra apparently describe some detail of clifford quotient groups, and this paper will describe something more detailed than that? No idea what a Pauli grading is.

$Q_N|j\rangle = \omega_N^j|j\rangle$, $P_N|j\rangle = |j+1\rangle$, so in 2d:

$$Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = Z$$
$$P_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = X$$

These elements along with ω_N are order N, and are nearly commutative.

$$\Pi_N = \{\omega_N^i P_N^j Q_N^k\}$$

This is not $H(N)$ apparently? Do we need a generalized version of Y before this becomes the Weyl Heisenberg group? Or am I missing something.

ω_N and Q_N clearly have computational basis as their eigenvectors, being diagonal, and P_N will have $|v_i\rangle = \sum_j \omega_N^{ij}|j\rangle$ as eigenvectors, eigenvalues ω_N^i.... transforming into this basis is the discrete fourier transform! Aha! Ok back to the text. I don't know what a configuration space is or what "eigenvector of position means".

Ah yes $\tau_N = \omega_N^{\frac{1}{2}}$ lets us define Y.

$$\tau_2 P_2 Q_2 = i \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = Y$$

Then $H(N) = \{\tau_N^h \omega_N^i Q_N^j P_N^k\}$, good. $|H(N)| = 2\,|\Pi_N| = 2N^3$.

Oh this phase factor is just for even N. Fascinating. Naively that sounds like a novel thing to attack in a mixed level system?

Apparently $\tau_2 = -i$, so the equation is actually $Y = \tau_2 Q_2 P_2$

Then the centralizer is just the set of scalars $\{\tau^i\}$, and since Q_N and P_N commute, up to phase shift ω_N, quotienting by the centralizer gives the abelian group $\mathbb{Z}_N^2$. Easy.

Next we move on to the clifford group. Indeed the clifford group is the set of terms against which $H(N)$ is closed under conjugation. Since $H(N)$ is finitely generated, and τ_N, ω_N are scalar, we can simply check $XQ_NX^{-1} \in H(N)$ and $XP_NX^{-1} \in H(N)$.

Apparently these "Clifford operations" are one-step evolutions of "Clifford Gates", which makes sense.

I don't follow what the $1 \to$ and $\to 1$ have to do with the statement of how $H(N)$ maps into the clifford group and quotient group, seems like it has significance in generalized abstract nonsense world. (not meant in a derogatory way)

We don't quotient clifford by $H(N)$, however, though we could. We quotient by $U(1)$ to get a simpler space without phase factors.

Lemma: $XAX^{-1} = YAY^{-1} \iff X \propto Y$

The proof is an application of "Schur's Lemma" which I will intuit as related to the observation before that the centralizer of $H(N)$ is exactly the set of scalars. Here $Y^{-1}X$ is in the centralizer of the clifford group, and turns out to be a scalar, so $X \propto Y$.

The next paragraph at least, is very representation heavy, so I will try to understand the significance of this in my own terms first.

Two matrices conjugate any element of $H(N)$ the same way if and only if they are proportional to eachother. Since the clifford algebra is exactly the set of actions that conjugate $H(N)$ to other elements of $H(N)$, this statement can be refined to the statement that the conjugation action on $H(N)$, that is the automorphism $A \mapsto XAX^{-1}$, is equal only to the actions of scalar multiples of X. So then if we quotient the clifford group, we will end up with some group of automorphisms on $H(N)$. Wonderful.

May 4

Reading "Quantum Algorithm Implementations for Beginners"

- Basic notation, probabilities from modulus of computational coordinates,

column vector notation, Dirac notation.

- Tensor product in matrix (Kronecker) and Dirac form.
- Linear combinations of basis states, entanglement defined as states that aren't a tensor product of N-level (in this case 2-level) states.
- inner product or 'overlap', row matrices/bra states, (not described as covectors)
- outer product, defined as matrix product of column with row, spanned to give GL($\mathbb{C}$).
- "Measurement corresponds to transforming the quantum information into classical information."
- measurement of a single bit as the sum of all measurements for all values with that value (rather than as a corresponding projection matrix)
- {H, T, CNOT} and {CCNOT} are universal
- observable is an operator that maps a hilbert space to the expected/mean value of measuring a quantum computation.
- observables are Hermitian since the measurements they represent will be exactly their real eigenvalues

ibmqx4 offers a user interface with a large quantity of operators, that get decomposed into a simpler generator set, 'essentially' the following:

$$\{U_1(\lambda), R_X(\pi/2), \text{CNOT}\}$$

i.e.

$$\{Z^{\lambda/\pi}, (I+Y)/\sqrt{2}, \text{CNOT}\}$$

ibmqx4 lacks full connectivity, with higher index bits only acting as conditions for lower index bits, (or it would be this way if 3 and 4 were swapped) and no direct connections between 1 or 2 and 3 or 4.

CNOT can be reversed by conjugating with $H \otimes H$, and extended to indirectly connected bits using a series of 4 CNOTs.

Errors come from gate infidelity and state decoherence.

Have read the discussion of Grover's algorithm, but will return to it tomorrow and summarize it.

May 5

Overview of Grover

In computer science many problems require a certain time complexity to solve, but a significantly smaller time complexity to verify. E.g. problems that can be solved in non-deterministic polynomial time can be verified in deterministic polynomial time, but on a deterministic computer currently require exponential time to execute.

Grover's algorithm allows one to leverage superposition in a very direct manner in order to solve in probabilistic quantum square-root time, any single-solution problem that can be verified in quantum linear time.

The algorithm presupposes an oracle which performs the verification on qbits, which is then applied to a superposition of all possible inputs, followed by a reflection operation which based on this increases the magnitude of the valid inputs, and decreases the magnitude of the invalid inputs.

This sequence, oracle then reflection, is called the Grover Operator, which when applied repeatedly to an initial uniform state ψ yields a solution to the desired problem with probability ¿ 1/2.

Details

Specifically the oracle O is defined to map an input $|x\rangle|y\rangle$ to $|x\rangle|y+f(x)\rangle$, where f is some function with a unique solution $f(x^*) = 1$.

$|y\rangle$ is then set to $|-\rangle$ so that $|y+1\rangle = -|y\rangle$, i.e. $O|x^*\rangle|y\rangle = -|x^*\rangle|y\rangle$.

The reflection operator is applied to all bits except the ancillary bit after this, defined as $2|\psi\rangle\langle\psi| - I$. Repeatedly applying these solves the fixed point algorithm:

$$(\alpha|\psi\rangle + \beta|x^*\rangle)|-\rangle = G(\alpha|\psi\rangle + \beta|x^*\rangle)|-\rangle$$

May 11

Bernstein-Vazirani Algorithm.

This algorithm is fairly simple, taking a bitstring-dot-product oracle and inferring the string represented by the oracle in a single super-positioned query.

Unsurprisingly the oracle is applied to a state $|+\rangle^{\otimes n}$ with an ancilla $|-\rangle$, and then the result is transformed into the computational basis via $H^{\otimes n}$ to get the exact state $|s\rangle$ represented by the oracle.

Although I say this is unsurprising, given that it has the same "superimpose, apply oracle, transform, measure" structure as Grover and as eigenvalue measurement, the specifics of why it works escape me a little.

If the oracle is the map:

$$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus \langle s, x\rangle\rangle$$

and specifically setting $|y\rangle = |-\rangle$

$$|x\rangle|-\rangle \mapsto (-1)^{\langle s,x\rangle}|x\rangle|-\rangle$$

and finally setting $|x\rangle = |+\rangle^{\otimes n}$ we have

$$\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|-\rangle \mapsto \frac{1}{\sqrt{2^n}}\sum_x (-1)^{\langle s,x\rangle}|x\rangle|-\rangle$$

Then $H^{\otimes n}$ is the map:

$$|x\rangle \mapsto \frac{1}{\sqrt{2^n}}\sum_y (-1)^{\langle x,y\rangle}|y\rangle$$

Composing these and ignoring the ancilla gives

$$\frac{1}{2^n}\sum_{x,y}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle}|y\rangle$$

and apparently this is precisely the state $|s\rangle$

It is pretty clear that the $|s\rangle$ component of this state is exactly:

$$\begin{aligned}\frac{1}{2^n}\sum_x (-1)^{2\langle s,x\rangle} =& \frac{1}{2^n}\sum_x 1\\ =& \frac{1}{2^n}2^n\\ =& 1\end{aligned}$$

Then if we consider $s_i \neq y_i$ at least 1 bit difference, the corresponding component will be:

$$\begin{aligned}\frac{1}{2^n}\sum_x (-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle} =& \frac{1}{2^n}\sum_{x_i=s_i}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle} + \frac{1}{2^n}\sum_{x_i=y_i}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle}\\ =& \frac{1}{2^n}\sum_{x_i=s_i}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle} + \frac{1}{2^n}\sum_{x_i=s_i}(-1)^{s_i}(-1)^{\langle s,x\rangle}(-1)^{y_i}(-1)^{\langle x,y\rangle}\\ =& \frac{1}{2^n}\sum_{x_i=s_i}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle} - \frac{1}{2^n}\sum_{x_i=s_i}(-1)^{\langle s,x\rangle}(-1)^{\langle x,y\rangle}\\ =& 0\end{aligned}$$

Alternatively by unitarity these maps must map a unit vector to a unit vector, so since we already have a component of length 1, all other components must be 0.

This is a really direct demonstration of quantum superposition in computation! Compute on a superposition, and rotate so that the failures interfere destructively, and the successes constructively.

Testing on IBM

Although this algorithm is simple, in practice the oracle even for a constant bit-string will be quite expensive.

The paper references a program that can decompose unitary matrices including those representing these bit-string oracles into fairly simple quantum circuits for practical use, and in doing so yielded gates that require around 38 time steps.

Executing the full algorithm on ibmqx4 gave a higher probability of getting the correct string than any other, but in absolute terms the probabilities weren't that much better than uniform, which speaks to the drastic cost of this oracle and of not using error correction!