So far: circuit-level fault-tolerant quantum memory
protects stored logical qubit

Need: " computation
manipulates logical qubit

Can't do this by unencoding/doing logical gate/re-encoding. Then limited by gate errors during unencoded step. Have to do logical gates directly on encoded info.

First step: encode logical info
In general non-trivial... but easy for simple states in CSS codes:

Prop: fault-tolerant preparation of $|0_L\rangle$ in LDPC CSS code:

Proof: Physical qubits start in product state $|\psi_0\rangle = |0_1 0_2 \cdots 0_n\rangle$
$\Rightarrow$ eigenstate of all Z-checks & Z-logicals

Now repeat: - measure X-checks (& Z-checks)
- apply likely recovery (or Pauli frame track...)

LARGE "Z-error" in first round... but it simply sets Pauli frame

similar to how we achieved fault-tolerant memory before ▨

Prop: fault-tolerant extraction of all Z-logicals in LDPC CSS

Proof: Measure all Z_i.

- faulty measurement $f_Z \Rightarrow$ final X-error: $\tilde{e}_X = e_X + f_Z$ (in $\mathbb{F}_2$; e_X = original X-error)
- MAP decoder $\Rightarrow$ most likely error class

$$\underset{l_X \in \mathrm{im}(G_X^T)}{\operatorname{argmax}} \mathbb{P}\left[\tilde{e}_X + \mathrm{im}(H_X^T) + l_X\right] =: \tilde{l}_X$$

($\tilde{e}_X + \mathrm{im}(H_X^T)$: pick some m_X; l_X: most likely X-logical (from $|0_L\rangle$))

- correct measurement to $m_{corr} = \tilde{e}_X + m_X + \tilde{l}_X$

(don't need to distinguish btwn X-errors related by X-checks)

Decoding successful $\Rightarrow$ Z-logical outcomes match intended state

lack of error cluster propagation again

make this fault-tolerant

Next step is to apply logical gates to encoded information.

Def: **Logical gate** = unitary U s.t.

$$U \cdot \mathcal{C} = \mathcal{C}$$

($\mathcal{C}$: codespace)

it takes codewords to codewords.

b/c identity is a logical gate...

Want: $|\psi_a\rangle, |\psi_b\rangle \in \mathcal{C}$ w/ $\langle \psi_a | \psi_b \rangle = 0$ and $\langle \psi_a | U | \psi_b \rangle \neq 0$

Prop: U is Clifford + logical gate $\Rightarrow U S U^{-1} = S$.

Proof: If $|\psi\rangle \in \mathcal{C}$ and $S \in \mathcal{S}$, $S U|\psi\rangle = U|\psi\rangle \Rightarrow \underbrace{U^\dagger S U}_{\text{Pauli string}} |\psi\rangle = |\psi\rangle$

$U^\dagger S U \in \mathcal{S}$ $\Leftarrow$

But generic non-Clifford logicals may not leave Pauli stabilizer group $\mathcal{S}$ invariant.
Next question: how do we find logical gates, especially ones that are useful?

Def: Divide qubits $\{1, \ldots, n\}$ into disjoint subsets:

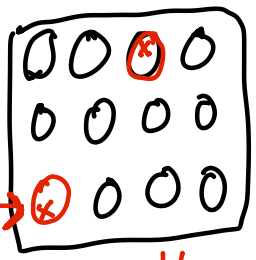
$\{1, \ldots, n\} = R_1 \cup \cdots \cup R_\ell$ w/ each $|R_i| \leq q$ (O(1))

Gate U is **transversal** if $U = \prod_{i=1}^{\ell} U_{R_i}$ (U_{R_i} acts only on R_i)

some people want each R_i to be as small as possible $\Rightarrow$ strictly transversal

Prop: In LDPC CSS code w/ $d \geq \log n$, transversal gates are fault-tolerant [if error rates sufficiently low]

Proof: analogous to our earlier arguments about quantum memory... large error cluster can't grow...



independent errors can't → propagate outside R_i b/c transversal!

How do we find these transversal gates?

Thm: Take two copies of any CSS code:

$$H_X = \begin{pmatrix} h_X & 0 \\ 0 & h_X \end{pmatrix} \qquad H_Z = \begin{pmatrix} h_Z & 0 \\ 0 & h_Z \end{pmatrix}$$

$$G_X = \begin{pmatrix} g_X & 0 \\ 0 & g_X \end{pmatrix} \qquad G_Z = \begin{pmatrix} g_Z & 0 \\ 0 & g_Z \end{pmatrix}$$

$\exists$ transversal logical $\overline{CNOT} = \prod_{i=1}^{n} CNOT_{i^{(1)} \to i^{(2)}}$ btwn the two copies of the code

Proof:

$$M_{\overline{CNOT}} \cdot \begin{pmatrix} H_X^T & 0 \\ 0 & H_Z^T \end{pmatrix} = \left(\begin{array}{cc|cc} h_X^T & 0 & 0 & 0 \\ h_X^T & h_X^T & 0 & 0 \\ \hline 0 & 0 & h_Z^T & h_Z^T \\ 0 & 0 & 0 & h_Z^T \end{array}\right)$$

So after applying $\overline{CNOT}$ new stabilizers are

$$H_X' = \begin{pmatrix} h_X & h_X \\ 0 & h_X \end{pmatrix} \qquad H_Z' = \begin{pmatrix} h_Z & 0 \\ h_Z & h_Z \end{pmatrix}$$

↳ add row 2 to row 1: equivalent to H_X (same idea for H_Z)

$\Rightarrow$ codespace $\mathcal{C}$ invariant.

Same calculation for logicals: $G_X' = \begin{pmatrix} g_X & g_X \\ 0 & g_X \end{pmatrix}$ $G_Z' = \begin{pmatrix} g_Z & 0 \\ g_Z & g_Z \end{pmatrix}$

Key: stored info transformed!

Easiest to see that a logical gate was implemented via Gottesman-Knill type stabilizer tracking!

$$X_L^{(1)} \to X_L^{(1)} X_L^{(2)}$$
$$X_L^{(2)} \to X_L^{(2)}$$
$$Z_L^{(1)} \to Z_L^{(1)}$$
$$Z_L^{(2)} \to Z_L^{(1)} Z_L^{(2)}$$

} logical CNOT implemented ▨

These CNOTs are done pairwise btwn each logical pair, as shown above

---

There's a sufficient condition to get the rest of the Clifford group transversally, which is very useful! (At least for k=1...)

Thm: CSS code w/ $H_X = H_Z = H_0$, parameters $[[n, 1, d]]$ and all check weights $= 0 \pmod 4$ has:

1. Transversal H: $\bar{H} = \prod_{i=1}^{n} H_i$
2. Transversal S or $S^\dagger$: $\bar{S} = \prod_{i=1}^{n} S_i$
3. Transversal CNOT on 2 code blocks ← from before!

Proof: ①: $\bar{H} X_1 X_2 X_3 X_4 \bar{H}^\dagger = Z_1 Z_2 Z_3 Z_4$

operation exchanges X & Z checks

i.e. $M_{\bar{H}} \cdot \begin{pmatrix} H_X^T & 0 \\ 0 & H_Z^T \end{pmatrix} = \begin{pmatrix} 0 & H_0^T \\ H_0^T & 0 \end{pmatrix}$

columns (stabilizers) permuted $\Rightarrow \bar{H} S \bar{H}^\dagger = S$

$M_{\bar{H}} \cdot \begin{pmatrix} G_X^T & 0 \\ 0 & G_Z^T \end{pmatrix} = \begin{pmatrix} 0 & G_0^T \\ G_0^T & 0 \end{pmatrix}$ ← if $k=1$, $G_0 =$ $1 \times n$ matrix...

$\Rightarrow$ $\bar{H}$ implemented logical Hadamard: $\bar{H} X_L \bar{H}^\dagger = Z_L$ $\quad \bar{H} Z_L \bar{H}^\dagger = X_L$

② $M_{\bar{S}} \cdot \begin{pmatrix} H_0^T & 0 \\ 0 & H_0^T \end{pmatrix} = \begin{pmatrix} H_0^T & 0 \\ H_0^T & H_0^T \end{pmatrix}$ looks good but have we changed stabilizer eigenvalues?

$$\bar{S}(X_1X_2X_3X_4)\bar{S}^\dagger = Y_1Y_2Y_3Y_4 = \prod_{j=1}^{4}(iX_jZ_j) = (X_1X_2X_3X_4)(Z_1Z_2Z_3Z_4)$$

↳ needed multiple of 4 check weight so $i^4 = 1$

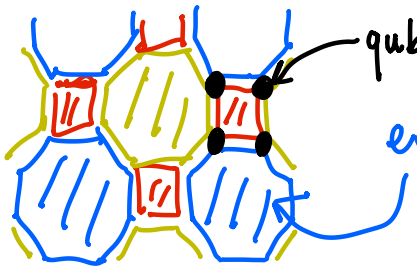
Analogous calculation involving check weight shows that

$$\bar{S} X_L \bar{S}^\dagger = \begin{cases} Y_L & \leftarrow \text{logical } S \\ -Y_L & \leftarrow \text{logical } S^\dagger \end{cases} \qquad \text{weight}(X_L) \equiv \begin{cases} 1 \pmod 4 \\ 3 \pmod 4 \end{cases}$$

must be odd so that $X_L Z_L = -Z_L X_L$ ▨

If we have $k>1$ code obeying same conditions of theorem, transversal H & S are logical gates but may not simply be elementary logical H & S... they can also entangle other logical qubits.

Def: **2d color code**: draw trivalent 3-colorable planar graph



every face f has $\prod_{v \in f} X_v$ & $\prod_{v \in f} Z_f$ checks and an even # of vertices

Prop: 2d color code is a CSS code.

Proof: planar graph $\Rightarrow$ 2 distinct faces share 0 or 2 vertices

$\Rightarrow H_X H_X^T = H_X H_Z^T = 0$

Same face alternates btwn 2 colors $\Rightarrow$ even # of edges $\Rightarrow$ even # of vertices

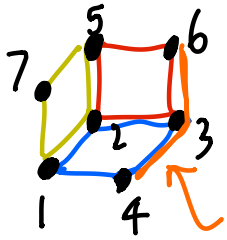
so X & Z checks on same face also commute ▨

Prop: Square-octagon color code has full transversal Clifford

Proof: follows from above Thm & 4/8 sided polygons

Color codes are 2d topological codes like surface codes & subject to same bounds... but have more transversal gates and other desirable properties. We'll revisit in more detail later...

Example: $[[7, 1, 3]]$ Steane code is a color code w/ transversal Clifford gate set.



$$H_X = H_Z = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

X & Z logical representative: $G_X = G_Z = (0\ 0\ 1\ 1\ 0\ 1\ 0)$

Prop: Steane code is smallest CSS code w/ $d=3$.