1 Last Time

Last lecture we covered the following topics:

- Rough sketch for Threshold Theorem based upon recursive fault-tolerant simulation.
- Assumptions required by the Threshold Theorem.

To elaborate, recall that if for each gate in a given circuit we can perform a fault-tolerant simulation at level k (encoded) with error probability $\mathcal{O}(p^t)$ using level $k-1$ primitives with error probability p and that the residual noise at level k is similar to that in level $k-1$ then we can recursively simulate and bring the error to arbitrarily small values.

Let us spell out the assumptions of the Threshold Theorem. Necessary:

- Can bring in freshly prepared known states
- Can make measurements duing computations
- disjoint set of qubits can be acted upon by different gates in parallel

Unnecessary, but simplifying and beautifying:

- Each location is either perfect (with probability $1-p$) or faulty (with probability p). The result set of faulty locations is called a fault path, and it is a random variable. Once a fault path is realized, the actual error on the fault path can be any adversarially chosen TCP map.

  Consider all possible fault paths. The probability that computation succeeds is at least the probability of fault paths on which any error is correctible by our fault-tolerant simulation.
- Long range nonlocal gates are permitted
- Same code is used at each level of concatenation

It remains to construct these fault-tolerant simulations.

2 Fault-Tolerant Simulations

Fault-tolerant simulations are composed of the following ingredients:

1. Discretize elementary operations—use a discrete universal set of gates
2. Encode data—protect quantum information using quantum error correcting codes
3. Encode gates—compute with encoded qubits without decoding them

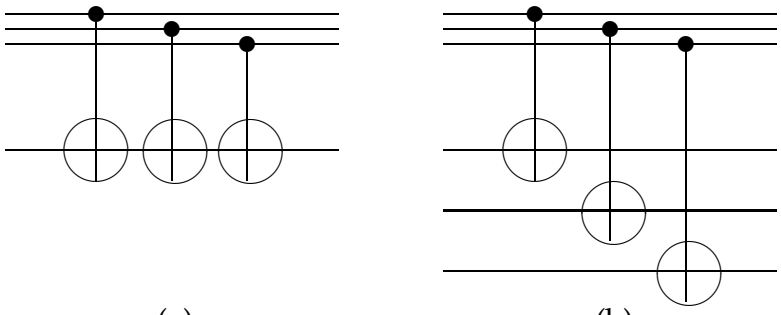

Figure 1: Examples of good and bad error propegation. Figure 1(a) is bad because a "sick" target qubit can "infect" three different control qubits. Figure 1(b) is good because any sick qubit can infect at most one other qubit.

4. Do not spread error. That is, in each simulation one error at any qubit should produce at most one error in any other code block. For example, *transversal* circuits do not spread error because each qubit in such a circuit only ever interacts with its corresponding qubit in another code block or ancilla. Figure 1 depicts examples of good and bad error propagation.
5. Error correction after each gate
6. Verify known states. Measurements should be repeated.

For the rest of this lecture we will focus on ingredient 3. For a more detailed summary of encoded quantum gates, the reader is referred to Sections 3.3 and 3.4 of

http://arxiv.org/abs/cs.CC/0012017

or Chapter 5 of

http://arxiv.org/abs/quant-ph/9705052

To facilitate our understanding of encoded quantum gates, we will illustrate the encoded gates of the 7-qubit code.

3 Example: Fault Tolerant Quantum Gates for the 7-Qubit Code

Before we proceed, let us review the foundations of the 7-qubit code.

3.1 Review of Stabilizer Codes

Recall that the n-qubit unitary matrix that describes an X gate on the ith qubit is denoted by $X^{(i)}$. For example, if $n = 7$ and $i = 3$ then

$$X^{(3)} = I \otimes I \otimes X \otimes I \otimes I \otimes I \otimes I,$$

which is often abbreviated to

$$X^{(3)} = IIXIIII.$$

We adopt a similar convention for the Z gate on the ith qubit.

Recall that the real Pauli group $\mathcal{G}_n$ is generated by the matrices $X^{(i)}$ and $Z^{(i)}$ for each $i = 1, \ldots, n$. Each element in $\mathcal{G}_n$ has eigenvalue ± 1 or $\pm i$ and any two elements of $\mathcal{G}_n$ either commute or anticommute. Any

Abelian subgroup S of $\mathcal{G}_n$ is a *stabilizer* (where *Abelian* means only that S is commutative). The eigenspace $\mathcal{C}$ corresponding to the eigenvalue $+1$ for any generator of S is called the *stabilizer code* with stabilizer S. By definition, any state $|\psi\rangle \in \mathcal{C}$ satisfies $M|\psi\rangle = |\psi\rangle$ for each $M \in S$—that is, $|\psi\rangle$ is stabilized by S.

Recall that the *centralizer* of S is the set of operations that commute with every element of S. For any stabilizer S, the centralizer is just the *normalizer* $N(S)$ of S, which is the set of all operations U such that $S = USU$ where

$$USU \stackrel{\text{def}}{=} \{UMU : M \in S\}.$$

Elements of $N(S)$ map codewords to codewords and hence these elements can be viewed as encoded operations on the encoded data.

3.2 Encoded Gates for the 7-Qubit Code

The stabilizer S for the 7-qubit code has generators

$$\begin{array}{ccccccc} I & I & I & Z & Z & Z & Z \\ I & Z & Z & I & I & Z & Z \\ Z & I & Z & I & Z & I & Z \\ I & I & I & X & X & X & X \\ I & X & X & I & I & X & X \\ X & I & X & I & X & I & X \end{array}$$

A nice property of this stabilizer is that the elements of $N(S)$ have a convenient and intuitive form, as depicted in the following table.

Logical Gate	Encoded 7-Qubit Gate
$\overline{X}$	$X^{\otimes 7}$
$\overline{Z}$	$Z^{\otimes 7}$
$\overline{H}$	$H^{\otimes 7}$
$\overline{\text{CNOT}}$	$\text{CNOT}^{\otimes 7}$
$\overline{P}$	$P^{\otimes 7}$

For each $U \in \left\{\overline{X}, \overline{Z}, \overline{H}, \overline{\text{CNOT}}, \overline{P}\right\}$ it is straightforward to verify that $U \in N(S)$. In particular, it suffices to note that

$$\begin{aligned} XXX^* &= X \\ XZX^* &= -Z \\ ZXZ^* &= -X \\ ZZZ^* &= Z \\ HXH^* &= Z \\ HZH^* &= X \\ (\text{CNOT})\, XI\, (\text{CNOT})^* &= XX \\ (\text{CNOT})\, XX\, (\text{CNOT})^* &= XI \\ (\text{CNOT})\, IZ\, (\text{CNOT})^* &= ZZ \\ (\text{CNOT})\, ZZ\, (\text{CNOT})^* &= IZ \\ PXP &= -Y \\ P(-Y)P &= -X. \end{aligned}$$

3.3 Encoded Gates for Other Codes

What about other codes? Under what conditions do we get $\{\overline{X}, \overline{Z}, \overline{H}, \overline{\mathrm{CNOT}}, \overline{P}\} \subset N(S)$?

In addition to being a CSS code, our code must also satisfy the following. To get $\overline{X}, \overline{Z} \in N(S)$ it must be the case that each generator of S has an even number of X's (or Z's)—that is, each generator has even *weight*. To get $\overline{H} \in N(S)$ it suffices to have $H_P = H_B$. To get $\overline{P} \in N(S)$ we require that the generators of S each have weight divisible by 4 and that $H_P = H_B$.