CS191 – Fall 2014
Lecture 18: Fault-tolerance and the threshold theorem

Mohan Sarovar

(Dated: October 29, 2014)

Now that we have looked at a few error correction codes and their implementation, we are ready to examine the notion of *fault tolerance*. The basic idea of fault tolerance is that if we perform error correction "well", then it is possible to execute an arbitrarily long quantum computation with "reasonable" resources. This may seem intuitive but it is a nontrivial statement because (i) we need to specify how "well" error correction must be performed, and (ii) what we count as "reasonable" resources, and then show that this can still be achieved. This is exactly what the theory of fault tolerance does.

The center piece of this theory is the *threshold theorem*, which we will sketch below. The theorem hinges on three assumptions:

1. Errors occur independently (on qubits, gates, measurements), with no spatial or temporal correlations [1].
2. It is possible to perform gates in parallel.
3. Each error correction circuit we execute will reduce the error probability from p to $cp^2 < p$, for some constant c. More generally, it is only necessary that the new error probability is cp^t for some $t > 1$, but we will restrict ourselves to $t = 2$. The constant c is proportional to the number of positions in the error correction circuit that an error can occur. We saw in the example of the bit-flip code how one (perfect) implementation of the error correction circuit can provide this reduction from an error probability of p to cp^2.

We will come back to these assumptions at the end, especially the third one. In fact, proving that these assumptions hold in a physically realistic implementation turns out to be the difficult part of proving fault tolerance.

I. THE THRESHOLD

The first step in moving to fault tolerance is noting that we can ratchet up the gain we get from one round of error correction (the reduction of error probability from p to cp^2) by *concatenation* of codes. This idea replaces each physical qubit at layer k by an encoded qubit. An example of concatenation using the bit-flip code is shown in figure 1 [2]. If we perform error correction at each layer of concatenation, then if the bare qubit experiences an error probability of p, then after one layer of error correction the logical error probability is cp^2, and after two layers, the logical error probability is $c(cp^2)^2$ (since two blocks have to fail and each block has failure probability cp^2). With each concatenation later, we get a gain in error probability. That is,

$$\begin{array}{ccccccc} \text{level 0} & & \text{level 1} & & \text{level 2} & \ldots & \text{level } k \\ p & \rightarrow & cp^2 & \rightarrow & c(cp^2)^2 & \ldots \rightarrow & c^{-1}(cp)^{2^k} \end{array}$$

And we are guaranteed that this will be a gain (*i.e.*, that the error probability will decrease with level of concatenation) if

$$p > cp^2 \tag{1}$$

We will come back to this condition, which tells us what we need in order to benefit from increasing the level of concatenation (it is the same as assumption 3 above) [3]. But before doing this let's look at the cost of doing this concatenated coding. If the size of the original (unencoded) circuit we want to execute is R gates, and the cost of

[1] Refinements to the original threshold theorem allow for limited spatial and temporal correlation in errors, but we will not cover these here.
[2] Of course the bit-flip code is insufficient in general since it doesn't correct phase errors. However, everything we shall demonstrate with the bit-flip code also holds for more general quantum codes.
[3] Concatenation is just one way of getting a code that can correct more errors. It is the simplest and the easiest one to analyze.

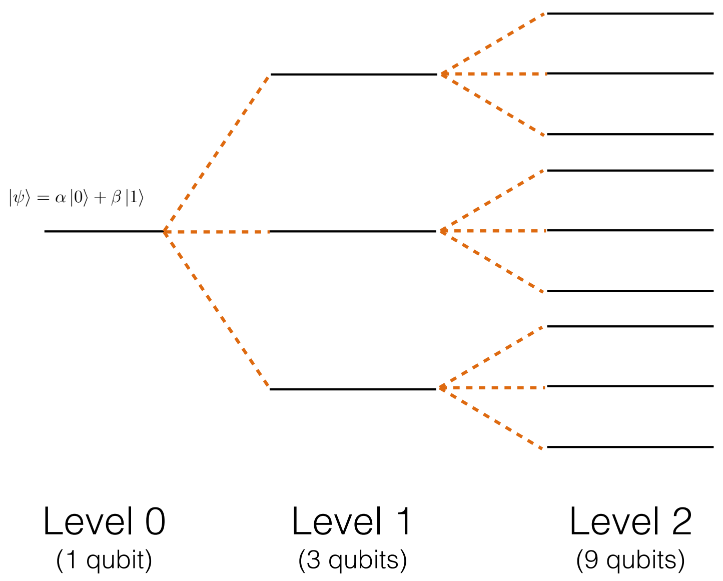


FIG. 1: An illustration of concatenation of the bit flip code. Each qubit at level k is encoded into three qubits at level $k+1$.

implementing each gate at the logical level (this depends on the code being used) is at most N physical gates, then the number of gates needed at level 1 of concatenation is at most NR. And for each level of increasing concatenation we must use N times as many gates:

$$\begin{array}{ccccccc} \text{level } 0 & & \text{level } 1 & & \text{level } 2 \;... & & \text{level } k \\ R & \rightarrow & NR & \rightarrow & N^2R \;... & \rightarrow & N^kR \end{array}$$

Note that the number of gates needed grows exponentially in k, but recall that the error probability reduced *doubly exponentially* in k. This suggests that we can win with this approach. But to see this convincingly, let us see how many gates we need to achieve a given error probability in the computation. Suppose we want an error probability of ϵ at the end of the encoded computation. Then we want

$$Rp_k = Rc^{-1}(cp)^{2^k} < \epsilon \tag{2}$$

for some concatenation level k. Here R is the number of logical quantum gates needed to perform the computation (which is the same as the number of gates necessary at the unencoded level), and p_k is the error probability per logical gate (or time step) at concatenation level k. We can solve for k to get

$$\begin{aligned} & (cp)^{2^k} < \frac{\epsilon c}{R} \\ \Rightarrow \quad & 2^k < \log_{cp}\left(\frac{\epsilon c}{R}\right) = \frac{\log\left(\frac{\epsilon c}{R}\right)}{\log(cp)} \\ \Rightarrow \quad & k < \log\left[\frac{\log\left(\frac{\epsilon c}{R}\right)}{\log(cp)}\right], \end{aligned} \tag{3}$$

where all logs are base 2 unless explicitly noted otherwise.

From this upper bound on the concatenation level we can also form an upper bound the number of physical gates we will need to achieve this level of logical error. We calculated above that at concatenation level k we need N^kR gates. Then,

$$N^kR < N^{\log\left(\frac{\log\left(\frac{\epsilon c}{R}\right)}{\log(cp)}\right)}R \tag{4}$$

But using $\log_a x = \frac{\log_b x}{\log_b a}$, we simplify

$$N^{\log(\cdot)} = \left(N^{\log_N(\cdot)}\right)^{\frac{1}{\log_N 2}} = (\cdot)^{\log N}. \tag{5}$$

Using this, the number of gates needed is upper bounded by

$$\left(\frac{\log\left(\frac{\epsilon c}{R}\right)}{\log(cp)}\right)^{\log N} R = \left(\frac{\log\left(\frac{R}{\epsilon c}\right)}{\log(\frac{1}{cp})}\right)^{\log N} R \tag{6}$$

The original R gates is scaled by the factor in braces. This factor is a polynomial in the log (is polylog) in $\frac{1}{\epsilon}$ (inverse error) and in R (the number of gates in the unencoded computation). Therefore the cost of achieving arbitrary error by concatenation scales very favorably, and this is the crux of the fault tolerance theorem: we can achieve an arbitrary error probability ϵ with resources that scale only polylogarithmically in the inverse of the desired error and the complexity of the computation to be performed (R).

Now, let us go back to Eq. (1), which defined the crucial condition for concatenation to be effective: $p > cp^2$. We can compute the value of p that is at the boundary defined by this condition, $p = cp^2$. This value of p is called the *threshold probability*, and is easily seen to be

$$p_{\text{th}} = c^{-1} \tag{7}$$

Thus if the physical error probability is below this threshold probability $p < p_{\text{th}}$, then concatenation and error correction is beneficial. Finally, writing the logical error probability at concatenation level k in terms of p_{th} gives us

$$p_k = c^{-1}(cp)^{2^k} = p_{\text{th}}\left(\frac{p}{p_{\text{th}}}\right)^{2^k} \tag{8}$$

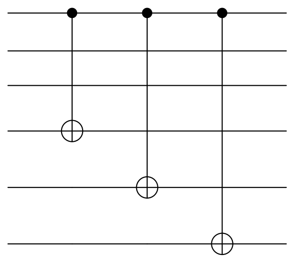
It is instructive to plot the behavior of this quantity as a function of p, for a fixed p_{th} and k. You will see that if p is a little below the threshold probability the logical error probability quickly goes to zero. In contrast, if p is a little above the threshold probability the logical error probability goes up very quickly. Thus, p_{th} defines a sharp boundary that specifies when reliable computation is possible and when it is not.

So what is the value of p_{th}? This depends heavily on the type of error correction code used, the type of constructions used for gates, measurements, state preparation, etc. However, a rough estimate for the threshold probability in standard circuit model QC implementations is $p_{\text{th}} \approx 10^{-4}$. Thus many QC architectures are striving to get their fundamental errors (in gates, measurements, state preparation, and idle periods) down to this level. In addition, it is very much an active area of research to find constructions and codes that will increase p_{th} to larger values.

II. REQUIREMENTS FOR FAULT TOLERANCE

Now let's go back to the assumptions we needed for the threshold theorem. The first two assumptions are pretty straightforward and are about the physical error model and architecture capabilities. The last one is more interesting; it assumes that performing error correction can actually reduces the effective error probability, *i.e.*, the probability goes from p to cp^2. Now, this might seem obvious given our analysis of the bit flip code from last class. But remember that that analysis assumed that everything (*e.g.*, gates, ancilla preparation, measurement) worked perfectly. In reality there will be errors in each of these elements, and so how can we be sure that we don't introduce more errors by trying to correct existing errors? The art of fault tolerance is designing error correction and logic circuits such that the condition $p > cp^2$ is indeed true, and the key idea here is to control errors from propagating and multiplying between each error correction cycle. This is most easily illustrated by some examples.

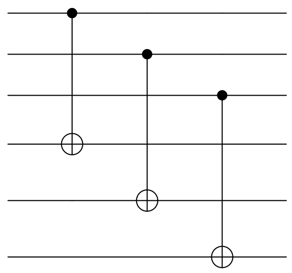
Consider encoding two qubits in the bit flip code, and let's say we want to perform a CNOT between the original unencoded qubits. One way to do this CNOT using the encoded qubits is to do the following circuit:



where the first three lines at the first encoded qubits and the second three lines are the second encoded qubit.

Exercise: Show that this circuit does perform a CNOT between two arbitrary encoded states.

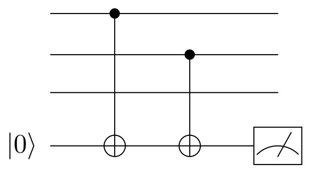
However, note that this circuit has poor error properties. In particular, if there is a bit flip error on the first line before the execution of the CNOTs then this error will propagate to *all* of the qubits in the second block (the last three lines). Therefore one error before the (perfect) CNOT can be turned into four errors afterwards. This is manifestly *not* fault tolerant. An alternative, fault tolerant version of this encoded, or logical, CNOT is



This is what is called a *transversal gate*, and in this case we see that an error on the first qubit will only turn into an additional error on the fourth qubit (which is in the second block). Transversal gates are constructed such that the ith qubit in an encoded block only interacts with the ith qubit in any other block. The particular for of a transversal gate will depend on the code being used and the operation being implemented (*e.g.*, any single qubit gate is automatically transversal). We know how to perform transversal gates for most gates and many common codes, however this is a subject of ongoing research. The basic desiderata of a transversal gate is that if the input to it contains an error in some block, then this error is not spread to many other blocks, or turned into multiple errors in the same block.

Exercise: Show that the transversal CNOT for the bit flip code written above performs a CNOT between two arbitrary encoded states.

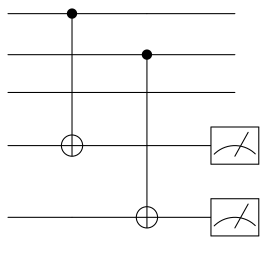
We saw in the last lecture that a circuit for implementing the parity measurement ZZI for the bit flip code was



where the first three qubits form the encoded state and the fourth qubit is one of the ancilla. This is not very good for fault tolerance since the ancilla interacts with two of the encoded qubits.

Exercise: Work out what happens if the ancilla in the above circuit is flipped to the state $|-\rangle$ before any of the CNOTs are executed, and the initial state of the encoded qubits is $\alpha|100\rangle + \beta|011\rangle$, *i.e.*, there is a bit flip error on the first encoded qubit. What encoded states corresponding to the two possible ancilla measurement results?

Instead, for a transversal implementation of this parity measurement we must execute the following circuit



where again, the first three qubits form the encoded state, and now we have two ancilla (for each parity measurement). But note that the ancilla can no longer start in the $|0\rangle$ state. If this were true the above circuit would destroy the quantum information stored in the encoded state.

Exercise: Work out what the state of all the qubits is after the measurement of the ancilla if the initial state of the encoded qubits is $\alpha|000\rangle + \beta|111\rangle$ and the ancillas are in $|0\rangle$ states.

For the transversal implementation we must initialize the ancilla in the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ before executing the above circuit. Given this initial state, the parity of the measurement outcomes gives the parity measurement ZZI on the encoded qubits, but at the same time does not destroy any phase information in the encoded state.

Example 1 Suppose the initial state of the encoded qubits is $\alpha|000\rangle + \beta|111\rangle$ (*i.e.*, no error). Then applying the above transversal circuit yields

$$\begin{aligned}
& (\alpha|000\rangle + \beta|111\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
\rightarrow\ & \frac{1}{\sqrt{2}}(\alpha|000\rangle|00\rangle + \alpha|000\rangle|11\rangle + \beta|111\rangle|11\rangle + \beta|111\rangle|00\rangle) \\
=\ & (\alpha|000\rangle + \beta|111\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),
\end{aligned}$$

where the second line is the state after the two CNOTs in the circuit (and we have omitted tensor products for ease). Therefore if there is no error then the original state of the ancilla is preserved, and the measurements of the ancilla yield either $|00\rangle$ or $|11\rangle$ with equal probability and both these states have even parity.

On the other hand, suppose there is an error on the first qubit and the state is $\alpha|100\rangle + \beta|011\rangle$. Now applying the above transversal circuit yields

$$\begin{aligned}
& (\alpha|100\rangle + \beta|011\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
\rightarrow\ & \frac{1}{\sqrt{2}}(\alpha|100\rangle|10\rangle + \alpha|100\rangle|01\rangle + \beta|011\rangle|01\rangle + \beta|011\rangle|10\rangle) \\
=\ & (\alpha|100\rangle + \beta|011\rangle) \otimes \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle).
\end{aligned}$$

So this time measuring the ancilla yields $|01\rangle$ or $|10\rangle$ with equal probability and both of these states have odd parity. Similarly, you should show that if the error is on the second qubit, then all measurement results on the ancilla (after executing the above circuit) will yield odd parity states. Therefore by looking at the parity of the ancilla qubits one can tell is there is a disagreement between qubit 1 and qubit 2. Then doing the same thing (with another two ancilla qubits) for qubits 1 and 3, one can narrow in on the location of the bit flip error.

The above are just two examples of how circuits can be modified in order to make them fault tolerant. There are many other constructions for gates, measurements, and state preparation that satisfy the fault tolerant requirement of not spreading errors too much. Put together, they enable us to guarantee that the execution of error correction using a concatenated code results in a decreasing error probability with each level of concatenation, even if each of the constituent physical operations are imperfect and have some probability of error. Then the threshold theorem tells us exactly how small these probabilities of error have to be before we can construct a circuit capable of reliable quantum computation.

III. REFERENCES AND FURTHER READING

1. A comprehensive but readable review of error correction and fault tolerance is *An introduction to quantum error correction and fault-tolerant quantum computation* by D. Gottesman.
`http://arxiv.org/abs/0904.2557`

2. Chapter 7 of John Preskill's notes on quantum information is a great introduction to quantum error correction and fault tolerance.
`http://www.theory.caltech.edu/people/preskill/ph229/notes/chap7.pdf`

3. Another good review is *Quantum error correction for beginners* by S. J. Devitt, W. J. Munro, and K. Nemeto.
`http://arxiv.org/pdf/0905.2794.pdf`