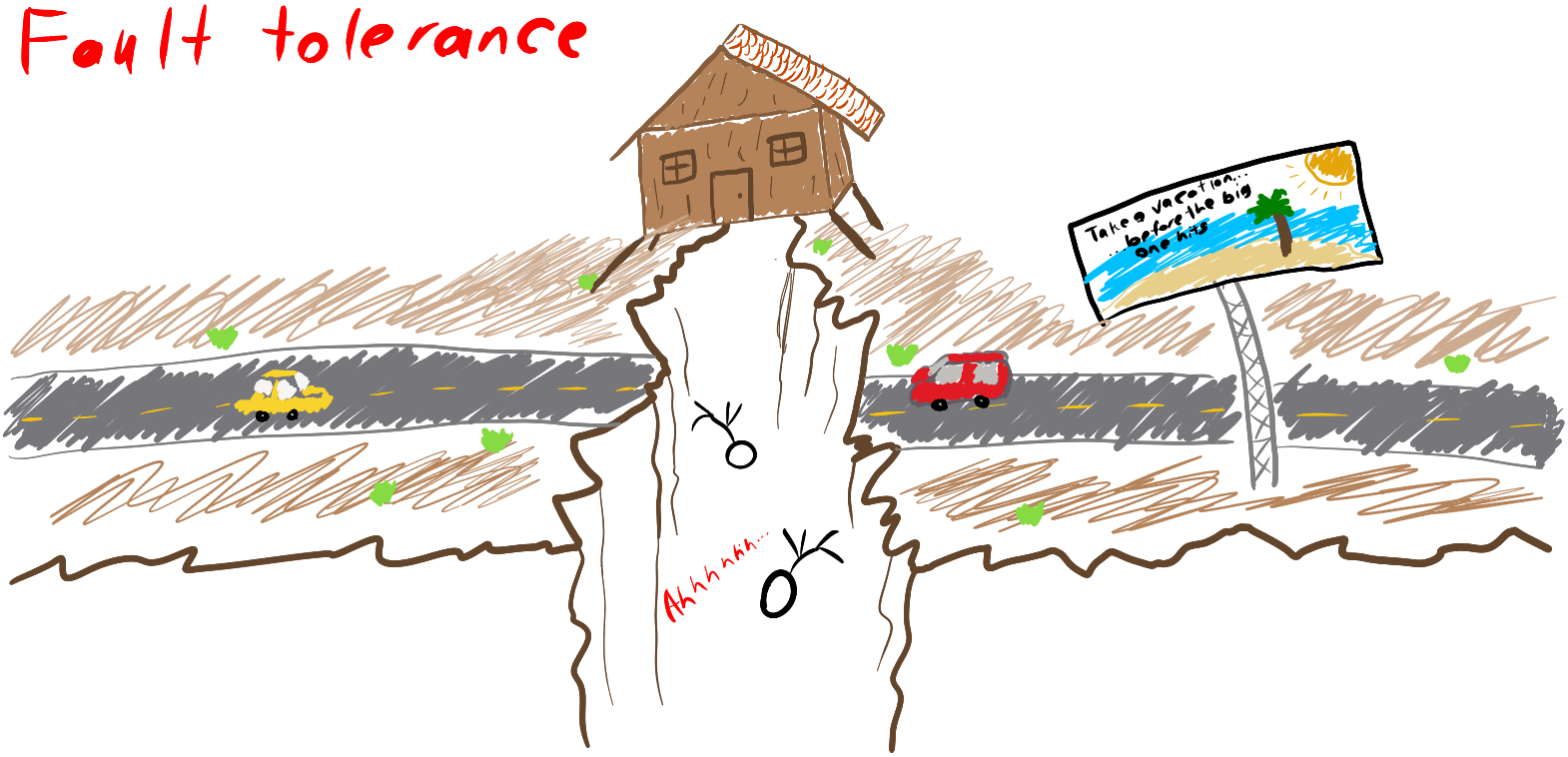
CMPT 476 Lecture 29

Fault tolerance



We now come to the final topic of our course - fault tolerant quantum error correction, or FTQEC, the holy grail of quantum computer engineering which, if implemented in a scalable system, should theoretically allow the practical implementation of useful quantum algorithms.

(Continuous error correction)

So far we've discussed methods of **encoding** states and **correcting** errors, but we haven't discussed what to do when we want to perform a **computation**.

Classically, we would just

decode → compute → re-encode

which is satisfactory because we typically only need correction against errors when **storing** or **transmitting** data.

However, in a quantum context errors occur with high probability **all the time**, so decoding immediately undoes all our hard work protecting from errors. It's like stepping out from under your umbrella to get into a car — you're going to get wet.

To maintain an error-free state through a computation, we hence need to be able to:

1. Compute directly on encoded states
2. Correct errors faster than they occur

Let's think about how we might do 1. first.

(Encoded gates)

Let U be a unitary transformation on k qubits and suppose we encode the state of k logical qubits into $n > k$ physical qubits. An **encoded** or **logical** U gate is an n qubit unitary U_L such that

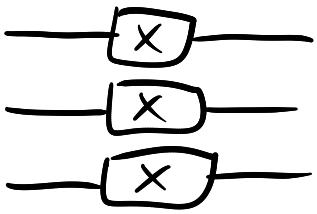
$$U_L |\psi\rangle_L = (U|\psi\rangle)_L$$

Ex.

Consider the 3-qubit bit flip code

$$|0\rangle_L = |000\rangle$$
$$|1\rangle_L = |111\rangle$$

The X gate maps $|0\rangle \longleftrightarrow |1\rangle$, so to devise an encoded X gate we need some circuit that maps $|0\rangle_L \longleftrightarrow |1\rangle_L$. Observe that the following circuit suffices



In particular, $X \otimes X \otimes X |000\rangle = |111\rangle$
$X \otimes X \otimes X |111\rangle = |000\rangle$. X is called transversal in the bit flip code, which means that $X_L = X \otimes X \otimes X$ or more generally that

$$U_L = U_1 \otimes U_2 \otimes \cdots \otimes U_n$$

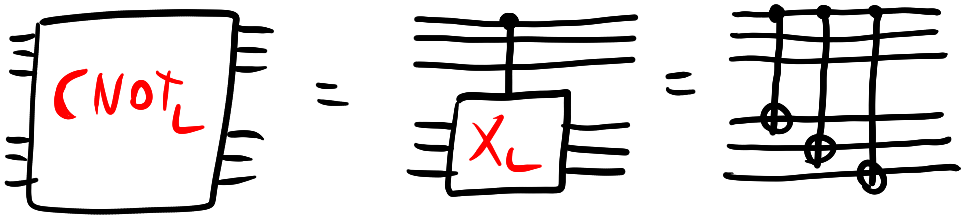
Ex.

How might we devise an encoded CNOT? We need

$$CNOT_L |0\rangle_L |\psi\rangle_L = |0\rangle_L |\psi\rangle_L$$

$$CNOT_L |1\rangle_L |\psi\rangle_L = |0\rangle_L (X_L |\psi\rangle_L)$$

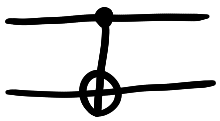
In the bit flip code, the first qubit of $|0\rangle_L$ is $|0\rangle$, and likewise for $|1\rangle_L$ the first qubit is $|1\rangle$, so we could implement $CNOT_L$ by an X_L gate controlled on the first physical bit



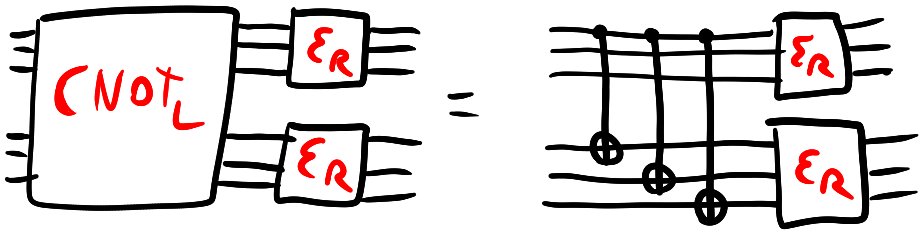
While this works in theory, there's a glaring problem: it propagates errors.

(Error propagation)

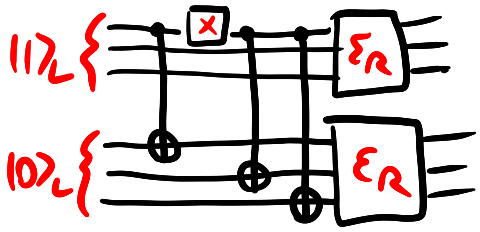
Suppose we want to run the following circuit with the three-bit code and we wish for it to be tolerant of a single bit flip on any code-block (i.e. a block of three physical bits encoding a logical qubit).



Intuitively, we could just encode each bit, run our $CNOT_L$ circuit, and then correct errors on each code block using our error correction process $\mathcal{E}_R$



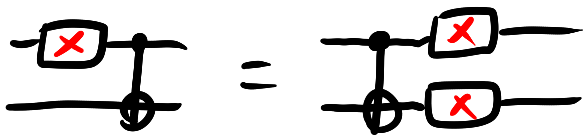
Theoretically, a single X error anywhere in the above circuit should be correctible. However, consider the following:



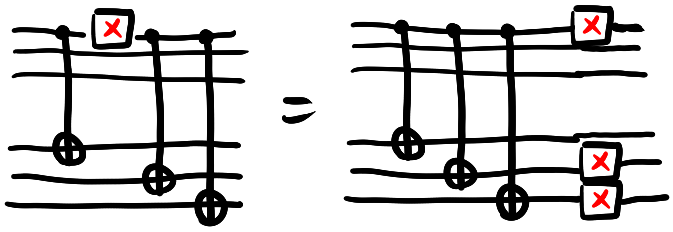
The resulting state should be $|1\rangle_L|1\rangle_L$ after correction. What is the state before correction?

$$|111\rangle|000\rangle \xrightarrow{CNOT} |111\rangle|100\rangle$$
$$\xrightarrow{X} |011\rangle|100\rangle$$
$$\xrightarrow{CNOT} |011\rangle|100\rangle$$
$$\xrightarrow{CNOT} |011\rangle|100\rangle$$

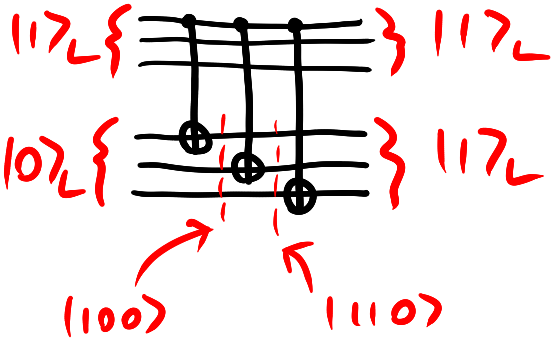
The state $|011\rangle|100\rangle$ gets corrected to $|1\rangle_L|0\rangle_L$, so we've failed to correct a single X error. The problem is error propagation: CNOT copies an X error on the control to an X error on the target, i.e.



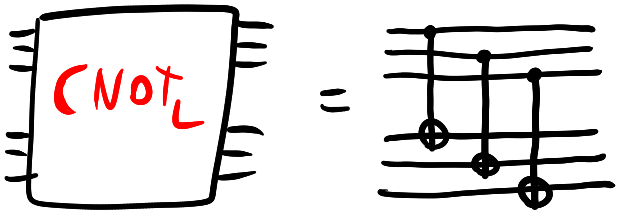
In the case above, one X error on the control block becomes an uncorrectable pair of X errors on the target block



How can we fix this? We could try interspersing error correction throughout the $CNOT_L$ circuit, but this doesn't work because the states of the logical qubits are **not** codewords in the middle of the circuit



Our only option is hence to design the encoded CNOT in such a way as to **not propagate errors**. The following circuit suffices:



This is again a **transversal** implementation.

Aside:

There is **no formally agreed upon** definition of **transversal**, but the most common is for a **k-qubit Unitary U** in a **block encoding** where each Logical qubit is encoded in n physical qubits, write the encoded state as

$$|\psi_1\rangle_L |\psi_2\rangle_L \cdots |\psi_k\rangle_L = \underbrace{|\psi_{1,1}\rangle |\psi_{2,1}\rangle \cdots |\psi_{k,1}\rangle}_{\text{first bit of all encoded } \psi\text{'s}} |\psi_{1,2}\rangle \cdots |\psi_{k,n}\rangle$$

Then the transversal U is

$$U_L = U \otimes U \otimes \cdots \otimes U$$

that is, each physical U acts only on the i^{th} bit of the encodings.

(Transversality)

Transversal gates are ideal in many ways:

1. They're efficient, and
2. They don't propagate errors

We may then wonder if we can get by with only transversal gates. The answer is NO, due to the Eastin-Knill theorem

(Eastin-Knill theorem, 2009)

No QECC can have an approximately universal set of transversal encoded gates.

(Clifford + T, the origin story)

So where do we go from here? Well, for technical reasons we won't go into, the Clifford group, consisting of (among other gates)

$$X, Y, Z, H, CNOT, S$$

is transversal in most useful codes. To get a universal set of gates, we only need to add the T gate

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

(Gate teleportation)

It turns out the T gate can be implemented using gate teleportation which you will explore in the last assignment. Gate teleportation uses a resource state

$$|A\rangle$$

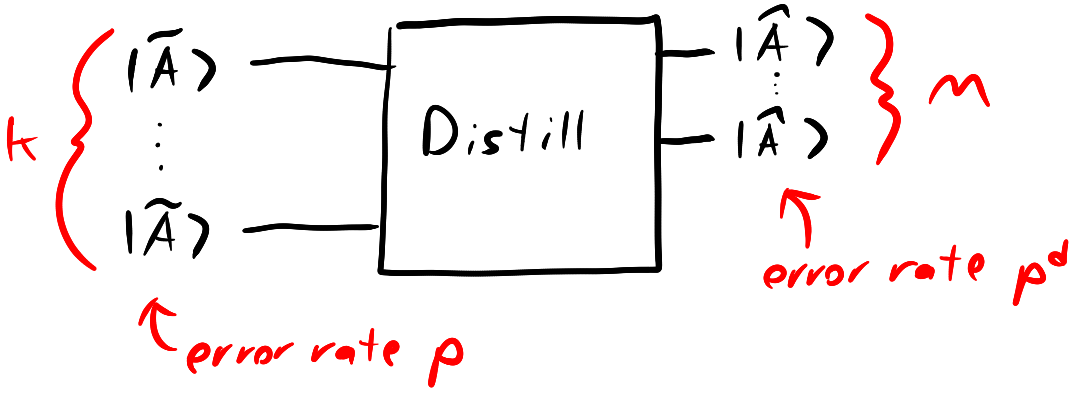
called a magic state which is entangled with the system and then measured to produce the effect of applying the intended state. In the case of the T gate,

$$|A\rangle = TH|0\rangle = \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}$$

However, the state itself needs to be encoded, so in particular we can't apply T directly to produce the state! The modern solution is to use magic state distillation.

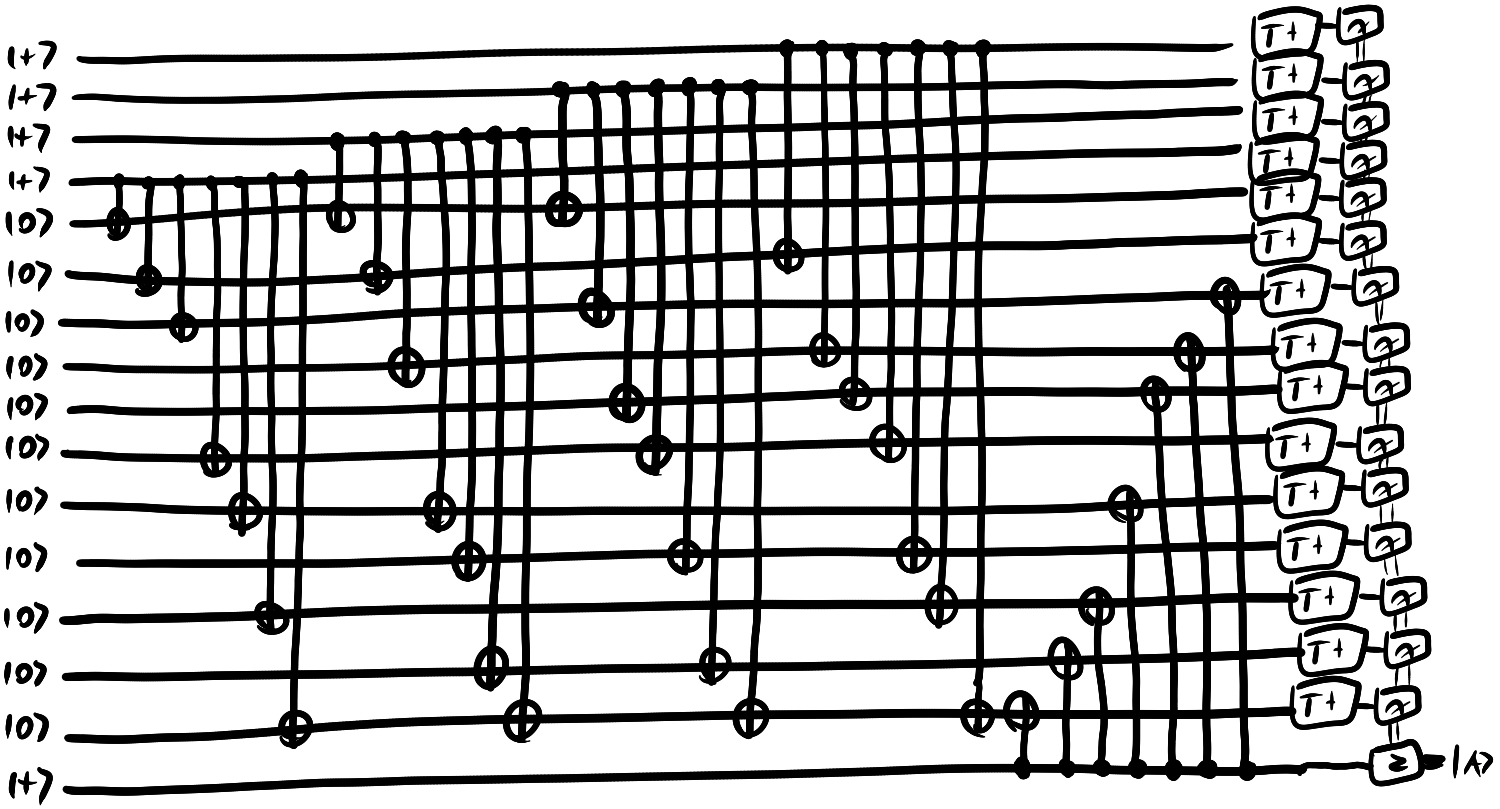
(Magic state distillation)

The basic idea of magic state distillation is to take many faulty copies of $|A\rangle$ and distill them into fewer better copies. A k-to-m protocol with $m < k$ generically looks like



(T-state distillation)

A classic magic state distillation protocol for T gates is the 15-to-1 protocol based on classical Reed-Muller codes. The protocol takes 15 T-states with error p to 1 T-state with error $O(p^3)$, or a cubic reduction. Just to understand the scale of magic state distillation, here is the circuit.



Now that's already pretty bad, but if we need to get down to $O(p^9)$ error, we would need to do this 16 times on a total of $15^2 = 125$ initial approximations. Not only that, but the final T gate is still faulty, unlike transversal gates! This is what leads us to the mantra that we repeat to ourselves at night (ok, maybe that's just me...)

T GATES ARE BAD

(Fault tolerance threshold)

So what's the upshot? Well, even though the costs (in terms of time and space) are horrendous, it can be shown that if we implement gates and measurements in a way that doesn't propagate errors, then if physical error rates are below some value p_{th} – called the threshold – then we can suppress the error rate arbitrarily small with enough error correction. This is what is known as a threshold theorem.

More formally, suppose we have a code that can correct a single qubit error of any kind, and assume that:

1. Single-qubit errors are independent, and
2. They occur with probability p per unit of time

An operation (Identity/waiting around, encoded gate, or measurement) is said to be Fault-tolerant if the probability of the operation introducing an uncorrectable error is at most

$$cp^2$$

for some constant c.

Observe that the identity operation satisfies this condition in the Shor code, since the probability that two or more errors occur in a single unit of time is at most

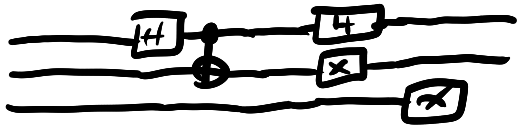
$$\binom{9}{2}p^2 + \binom{9}{3}p^3 + \cdots \leq cp^2$$

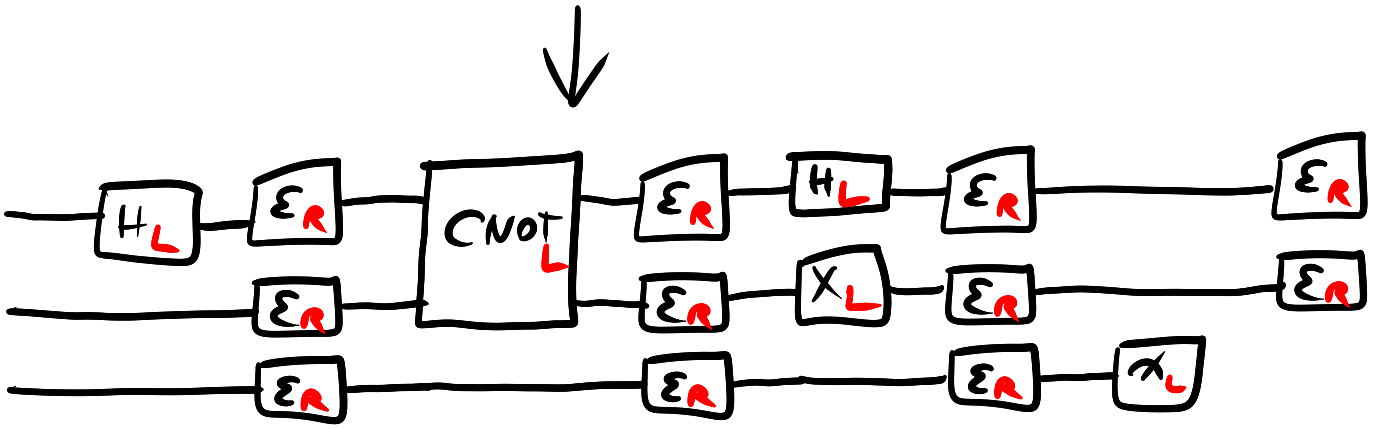
The threshold in this case is

$$p_{th} = \frac{1}{c}$$

as if $p < p_{th}$ then $cp^2 < p$, or in other words our error correction can suppress errors.

The general scheme of Fault-tolerant computation is then to, in each unit of time called a clock cycle, perform a (logical) operation and then apply a round of error correction





After every operation we have probability at most cp^2 of observing an incorrectible error, and all other errors the layer of error correction corrects, so given a circuit of depth S, the total probability of error is

$$\leq Scp^2$$

(Code concatenation)

The final piece of the puzzle is how to suppress errors further below cp^2. We can do this by concatenating codes, as we did when constructing the Shor code. For example, concatenating the three-bit code with itself amounts to applying two-layers of the code:

$$|0\rangle_0 \xrightarrow{\text{outer code}} |0\rangle_I |0\rangle_I |0\rangle_I \xrightarrow{\text{inner code}} |000\rangle|000\rangle|000\rangle$$

$$|1\rangle_0 \longrightarrow |1\rangle_I |1\rangle_I |1\rangle_I \longrightarrow |111\rangle|111\rangle|111\rangle$$

The idea is that the inner code produces an effective error rate of $p' \leq cp^2$, which the outer code then suppresses to

$$\leq c(p')^2 = c^3 p^4$$

If we apply k-levels of concatenation, the effective error rate is then

$$\leq \frac{(cp)^{2^k}}{c}$$

Or an exponential suppression of the hardware error rate p, so long as $p < p_{th}$.

<u>However</u>, concatenation also expands the gate count exponentially in k, so we also need to check that it doesn't wipe out any potential algorithmic improvement from quantum computation.

Without going deep into details, if d is the maximum cost of a quantum operation and we want to perform a circuit of depth S to error ε with k levels of concatenation, we want

$$S \cdot \frac{(cp)^{2^k}}{c} \leq \varepsilon$$

which we can show gives $2^k \leq \frac{\log\left(\frac{S}{c\varepsilon}\right)}{\log\left(\frac{1}{cp}\right)}$ or

$$d^k \leq \left(\frac{\log\left(\frac{S}{c\varepsilon}\right)}{\log\left(\frac{1}{cp}\right)}\right)^{\log_2 d} \in O\left(\log^m\left(\frac{S}{\varepsilon}\right)\right)$$

which leads to the generic threshold theorem

(Threshold theorem)

A circuit with depth S can be approximated to error ε on a fault-tolerant quantum computer with error rates at or below threshold in time

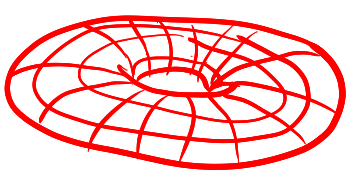
$$O\left(S\left(\log^m\left(\frac{S}{\varepsilon}\right)\right)\right)$$

(So what is the threshold?)

To the best of my knowledge, only one threshold value has been proven:

$p_{th} \approx 10^{-5}$ for FTQEC in the Steane code

By comparison, current hardware error rates hover around $10^{-1} - 10^{-2}$ depending on the operation. For this reason, most recent proposals use surface codes which have an estimated p_{th} of $\approx 10^{-3}$. These codes are fascinating, having been originally formulated on the surface of a torus by Kitaev



But alas we're out of time...