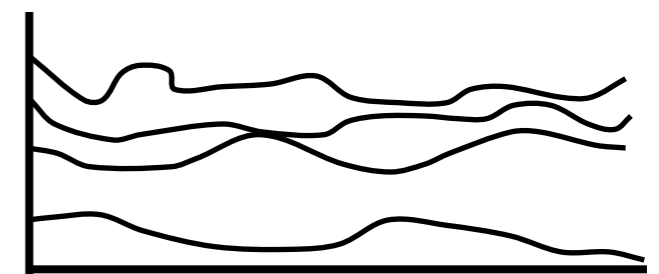
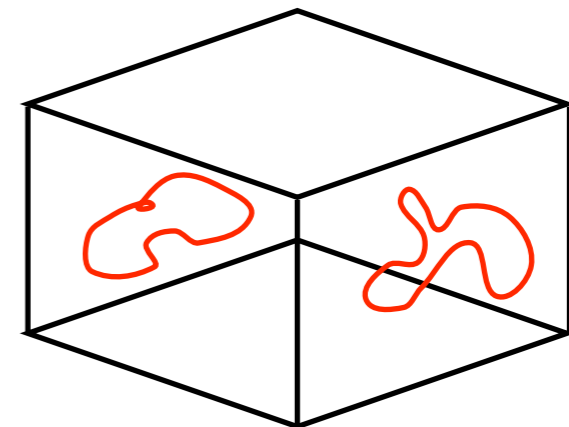
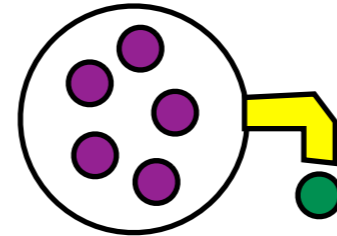
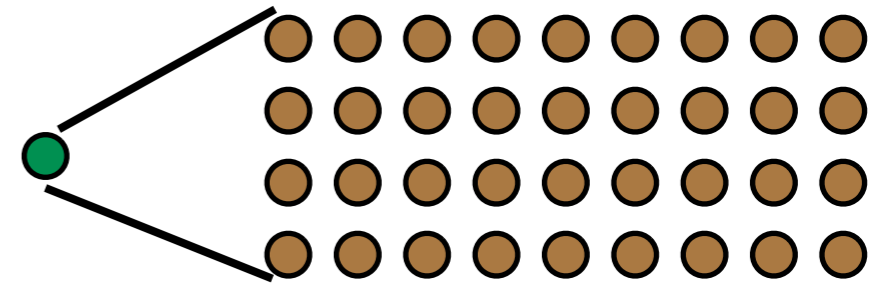


General Principles of Fault-Tolerance

Daniel Gottesman
Perimeter Institute

What's Left For Fault-Tolerance?

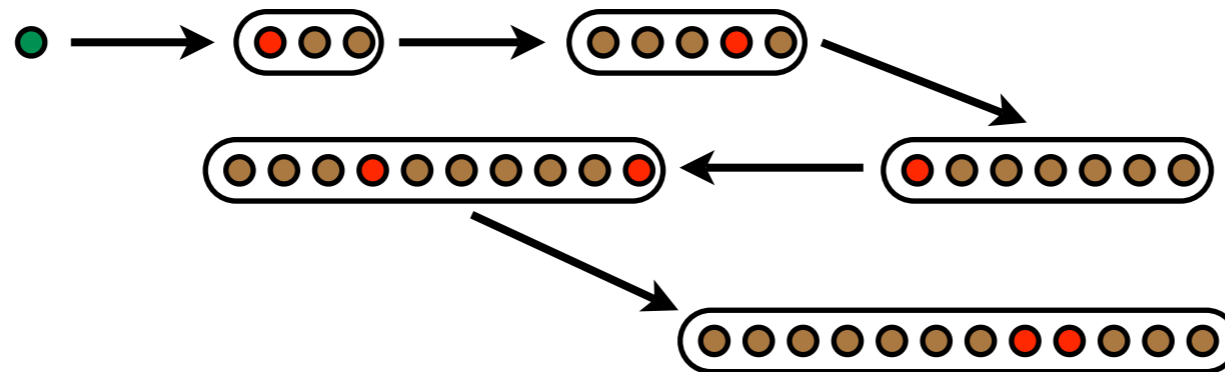
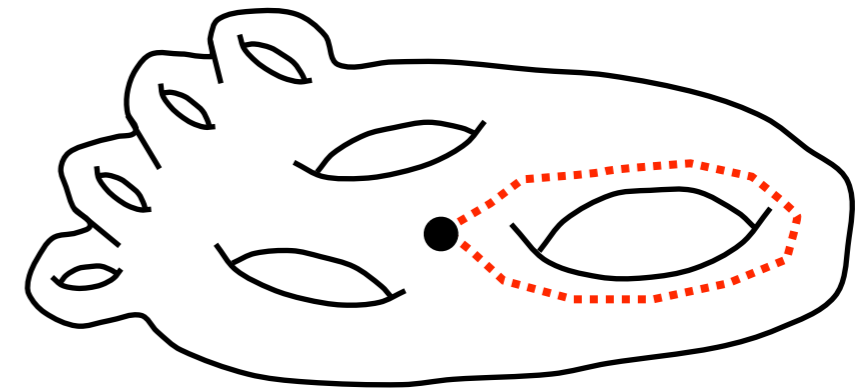
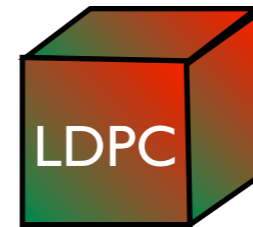
- Reduce the overhead (in space and in time) needed for fault-tolerance
- Better fault-tolerance in 1D?
- Better magic state protocols
- Fault tolerance for specified noise models
- Fault tolerance for specified gate sets
- Exponential self-correcting codes
- Fault-tolerant adiabatic quantum computation
- Improving the threshold could always help



~~10^{-6}~~ ~~10^{-5}~~ ~~10^{-3}~~ 0.03 ?

Outline

1. Unifying properties of fault-tolerant gates.
2. Properties needed for a code family to give a threshold.
3. Achievable overhead compatible with threshold existence.



Types of Fault-Tolerant Gates

There are three classes of constructions of FT gates:

- **Transversal gates**

Non-universal, but very efficient

Some codes have more than others

- **Path-based gates**

For topological codes

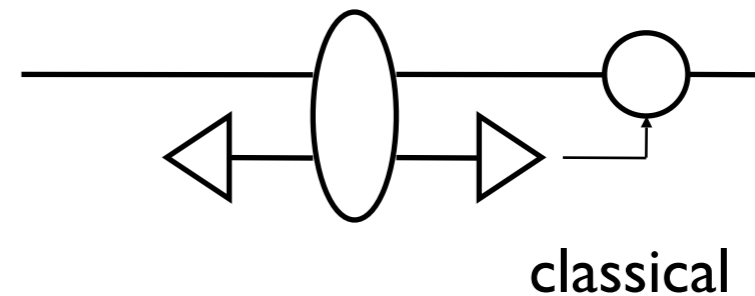
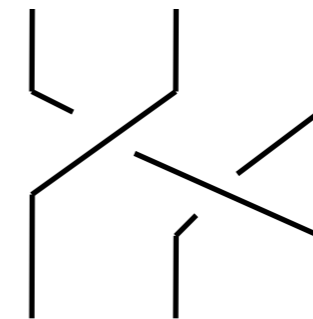
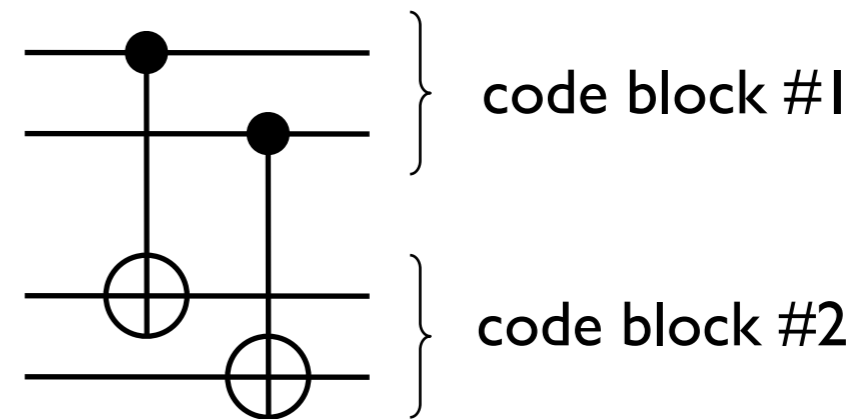
Low space overhead, but significant time overhead

- **Ancilla constructions**

E.g., magic states

Require much extra space for ancilla factories

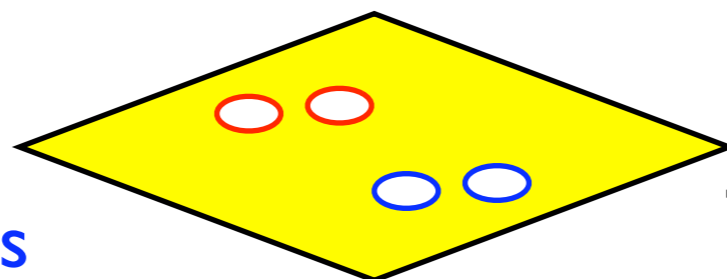
- **Other constructions**



Code Deformation

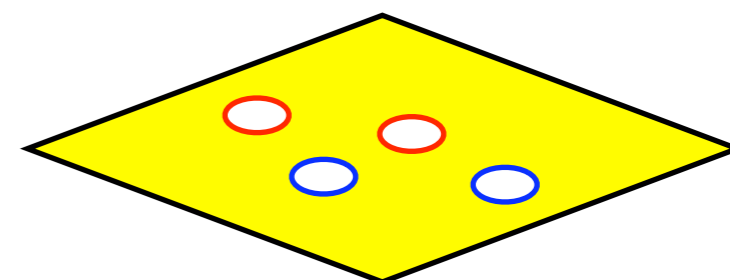
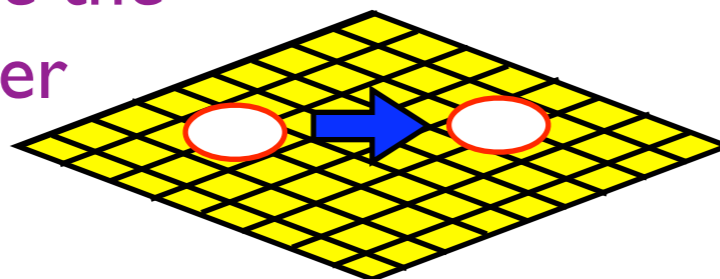
What does it mean, in circuit terms, to do a path-based gate for a topological code?

Qubits are encoded as lattice defects



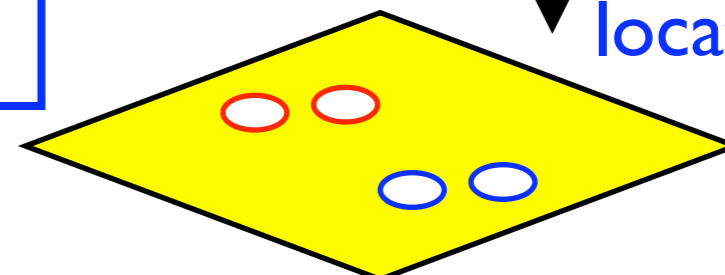
To perform gates, we move the defects

To move a defect, we rearrange the local stabilizer generators.



Eventually, the defects return to their starting location

That is, we deform through a series of topological codes, eventually returning to the original code.

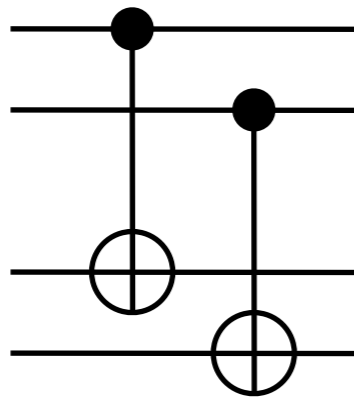


FT Gate Type and Error Models

Why do we use certain types of fault-tolerant gates with certain kinds of code?

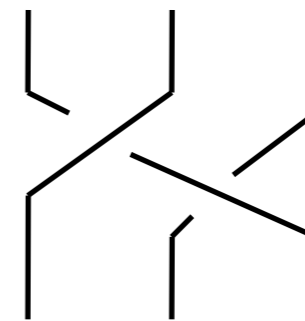
The gate type must match the error model handled by the code.

A standard block code corrects errors of weight t



Transversal gates preserve the weight of errors

Topological codes correct physically local errors



Small code distortion keeps local errors local

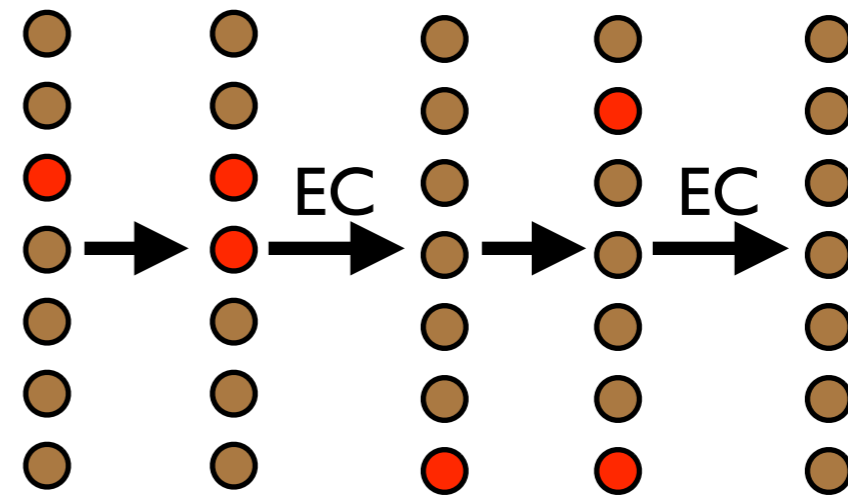
If we pick a code which corrects a different error model, we must use fault-tolerant gates that preserve that type of error. E.g., phase error correcting code needs diagonal transversal gates.

Gates Which Make Errors Worse

Transversal gates have a notable advantage over path-based gates: Small code distortion spreads out errors, whereas transversal gates leave the same number of errors per block.

Consequently, during code distortion, one must periodically stop to do error correction.

For block codes, we can also consider gates which **slightly increase the number of errors**. Aharonov and Ben-Or (quant-ph/9906129) called this property “**spread**.” (i.e., a gate with spread 2 could double the number of errors.)

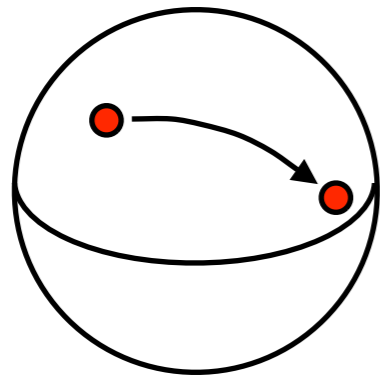


Disadvantage: Effective error-tolerance of the code is reduced.

Infinitesimal Generation of Gates

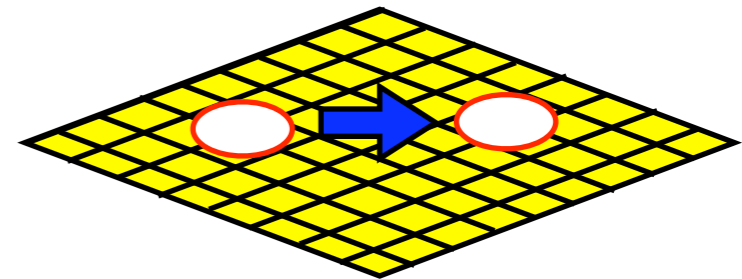
Unitary gates cannot be performed instantaneously. They must be generated from a series of smaller infinitesimal gates.

$$U = e^{iHt}$$

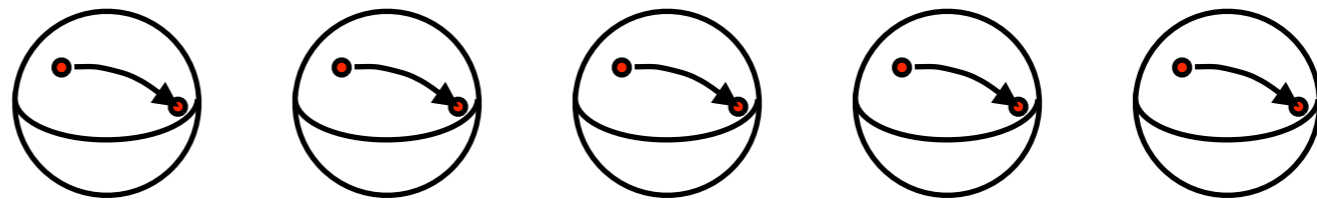


This is why fault-tolerance for phase-correcting codes requires diagonal gates (Aliferis & Preskill, 0701.1301, Peter Brooks talk), even though CNOT, for instance, maps phase errors to phase errors.

Topological gates are generated by infinitesimal code deformation.



Transversal gates can always be generated by infinitesimal transversal gates.

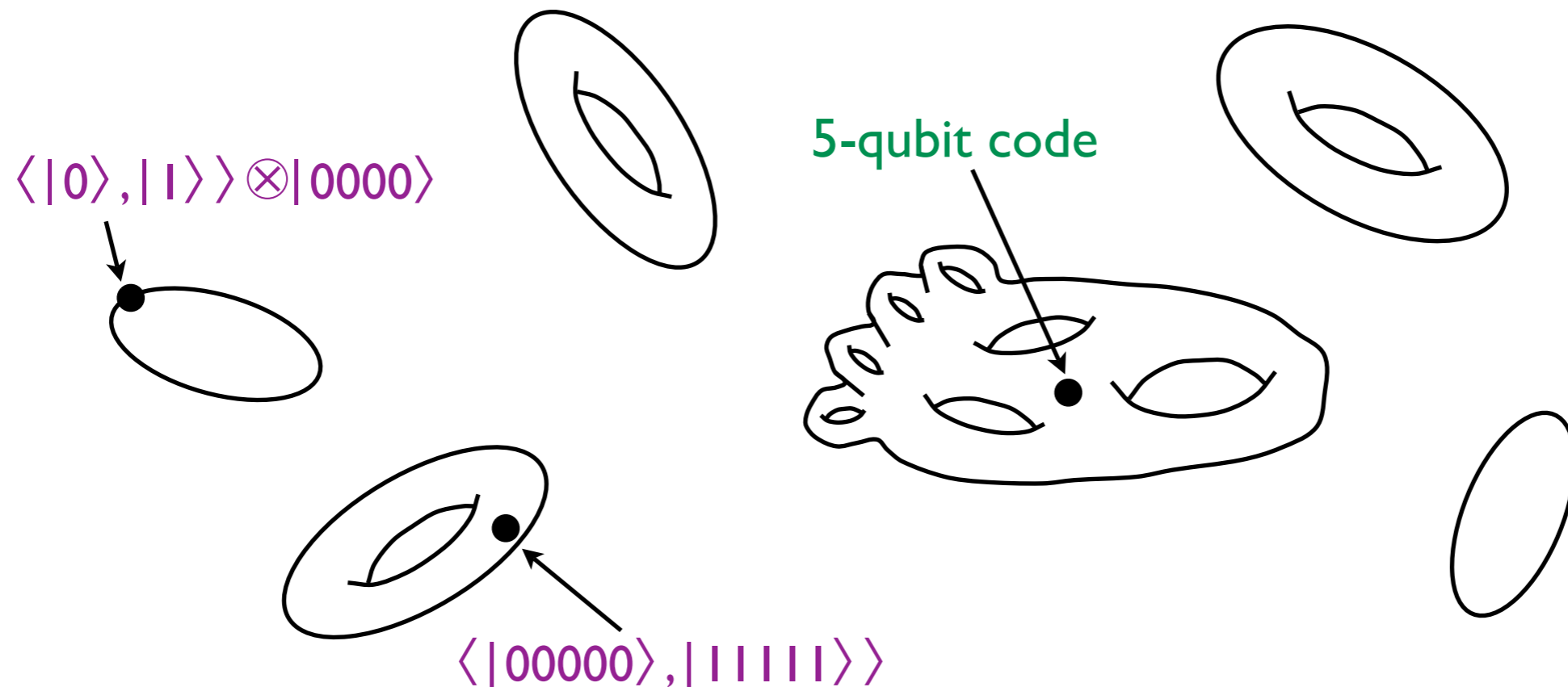


Codes Deformed by LU Operations

(Work with Lucy Liuxuan Zhang)

Infinitesimal transversal operations map $((n,K,d))$ QECCs to different $((n,K,d))$ codes.

Transversal gates are performed by code deformation under LU.

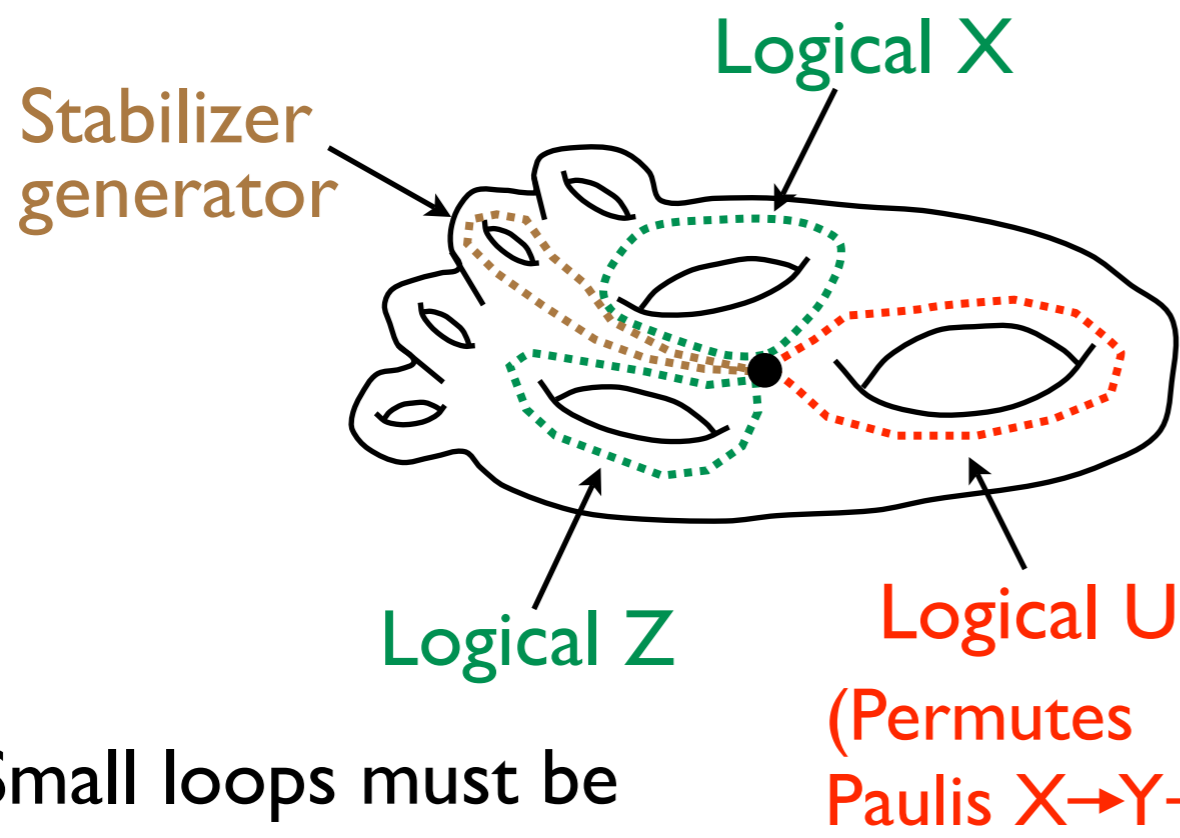


Classify K -dim subspaces by local unitary equivalence. They form connected components, corresponding to inequivalent codes.

Transversal Gates Are Topological

(Work with Lucy Liuxuan Zhang)

Each component under LU forms a manifold. To implement a transversal gate, we perform a loop on this manifold.



We can think of the manifold as a vector bundle. I.e., each point has a K -dimensional subspace attached to it.

LU gates map the subspace at one point of the manifold to the subspace at another point. This gives a **connection** (parallel transport).

Small loops must be trivial (see, e.g., Eastin & Knill, 08 | 1.4262). Therefore the connection is **flat**.

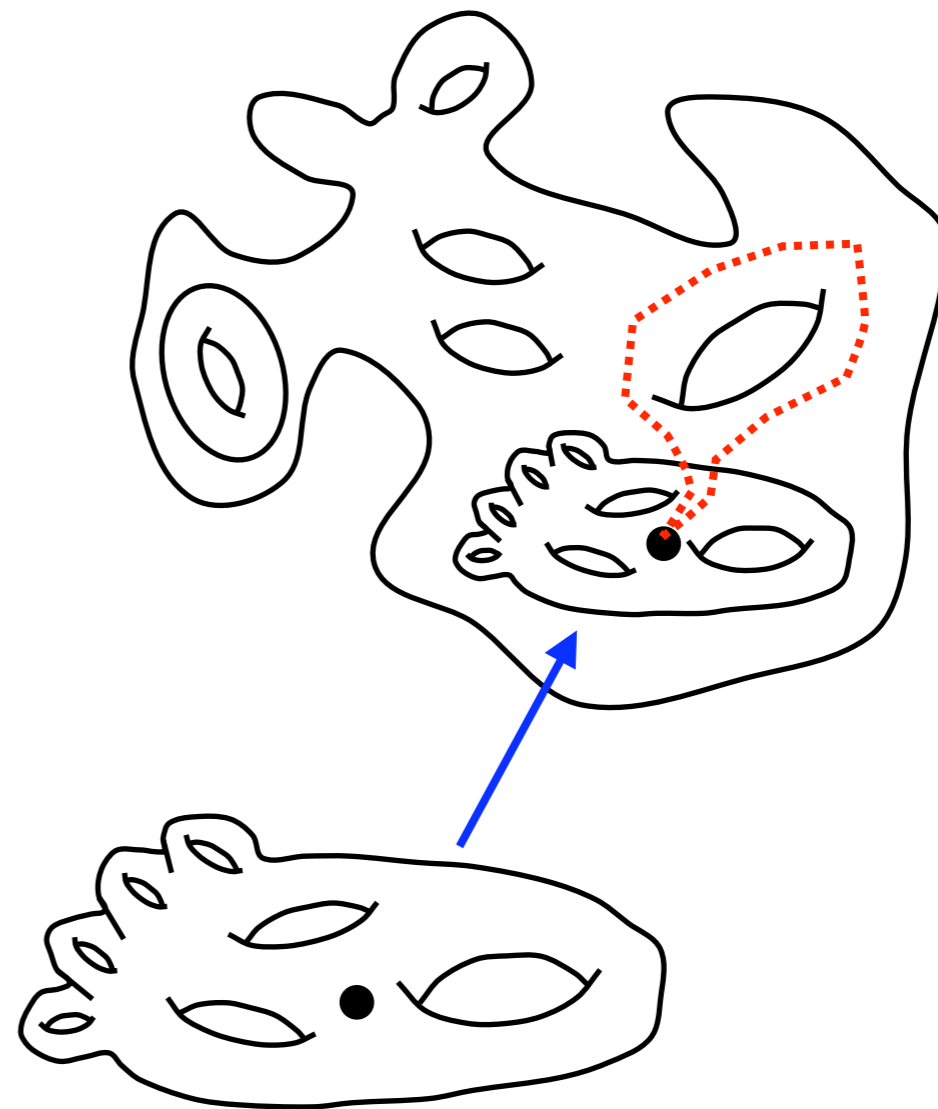
Transversal gates are associated with topologically non-trivial paths on the manifold.

Ancilla and Other Constructions

Magic state and other ancilla-based constructions can be understood in a similar way. They involve first imbedding the manifold in a higher-dimensional manifold, and then following a topologically non-trivial path.

A similar statement can be made of ancilla injection procedures for topological codes.

Some miscellaneous other fault-tolerant gate constructions are known (e.g., Toffoli gate for polynomial codes in Aharonov & Ben-Or, quant-ph/9906129). All involve leaving the code and then returning to it later. This is a universal property of fault-tolerant gates.

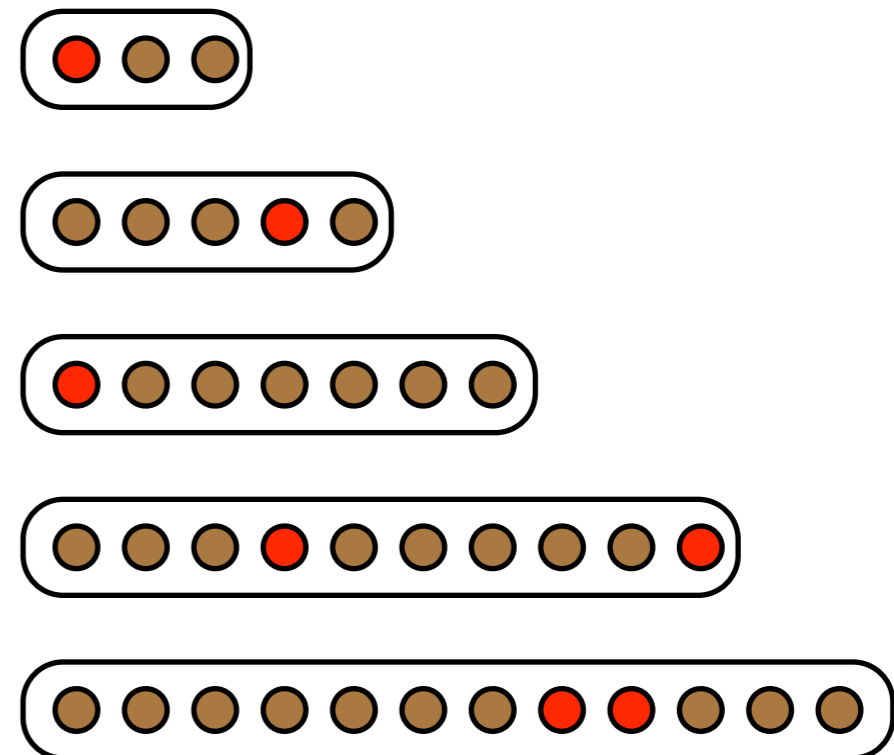


What Do We Need For a Threshold?

Thresholds have been derived for concatenated codes and for topological codes. Can we get a threshold using other codes? (In particular, can we get a threshold using more efficient codes?)

To have a threshold, the first thing we need is a **family of QECCs** using n physical qubits, with $n \rightarrow \infty$. The family should have the following properties:

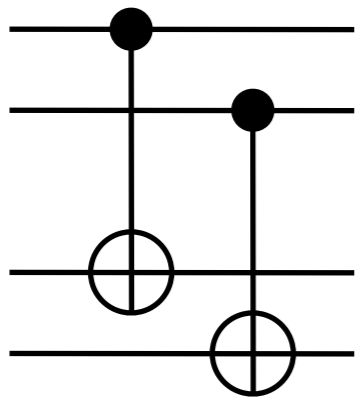
- Corrects a constant fraction of likely errors (prob. p per qubit).
- Probability of a logical error $\rightarrow 0$ as $n \rightarrow \infty$.
- Efficient decoding algorithm.



Not necessary to have a good distance d/n , nor is it necessary to have a vanishing rate k/n .

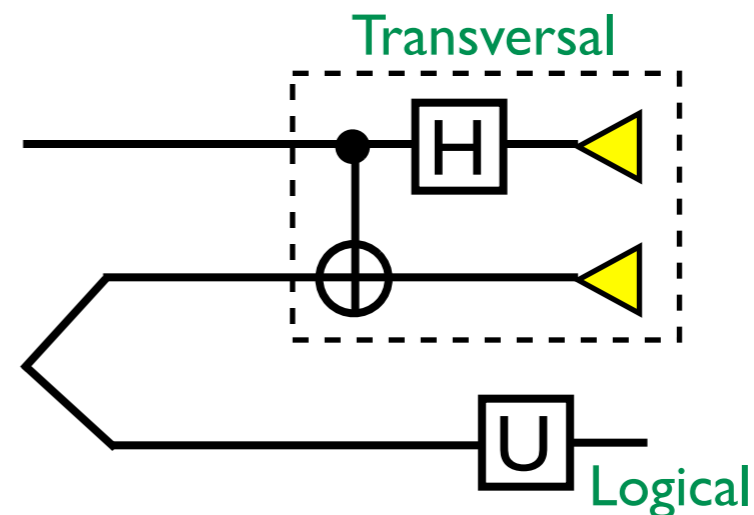
Threshold: Gates and Measurement

To get a threshold, we need a universal set of fault-tolerant gates and fault-tolerant measurements for the family of codes. It must work even when the block size gets large.



For CSS codes, CNOT gate and X or Z measurement can always be done transversally. We can then compose a universal set of gates given an appropriate set of magic states.

For a general stabilizer code, gates and measurements can be done using Knill error correction, which requires appropriate ancilla states.

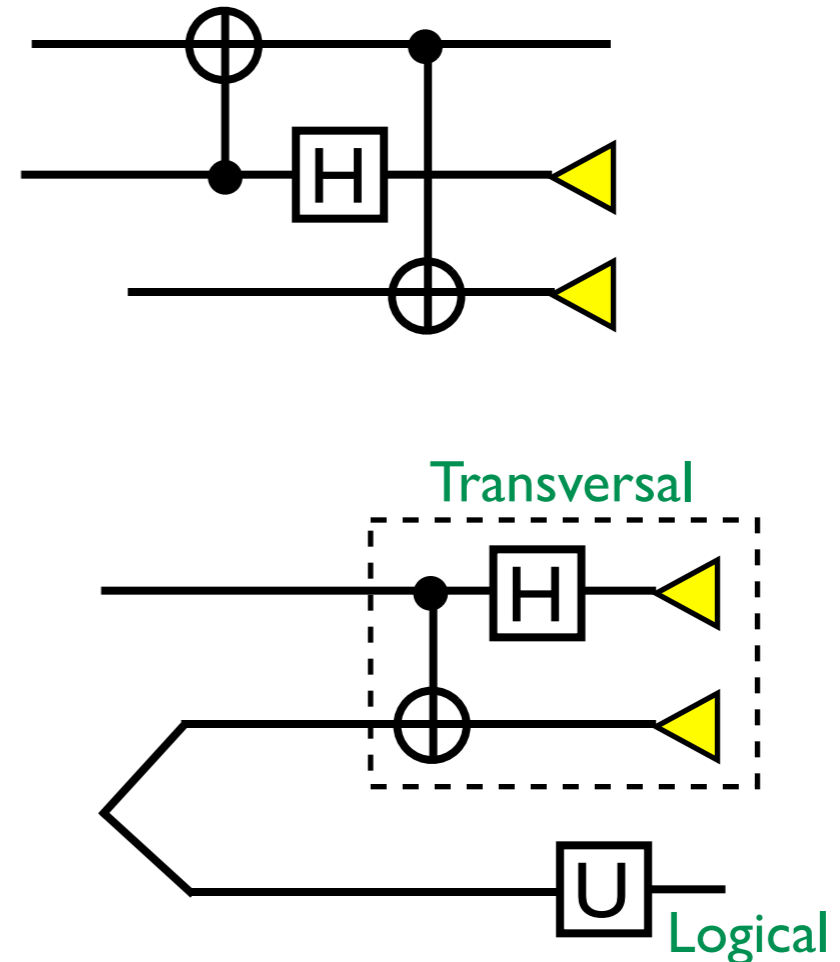


Given a reliable source of appropriate ancilla states, gates and measurements can be done safely even for large codes.

Threshold: Error Correction

There are three known methods of FT error correction:

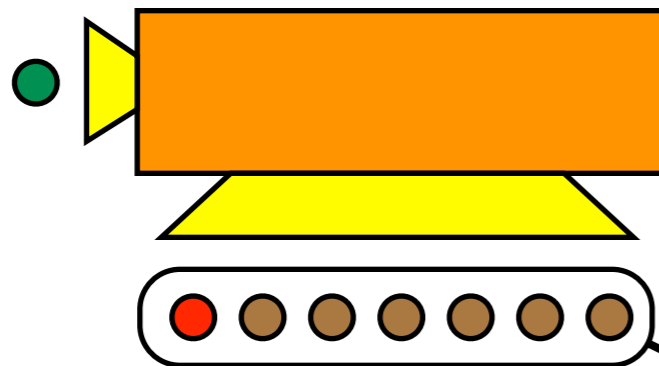
- **Shor:** For any stabilizer code. Requires cat states the size of the stabilizer generators and needs repetition of syndrome measurement.
- **Steane:** For CSS codes. Requires states encoded in block of the code. No syndrome repetition needed.
- **Knill:** For any stabilizer code. Requires EPR pairs encoded in block of the code. No syndrome repetition needed.



Steane and Knill EC can be done for big blocks whenever we can encode codewords. Shor EC is only viable for LDPC codes, as big cat states are inherently unstable.
(LDPC = low density parity check)

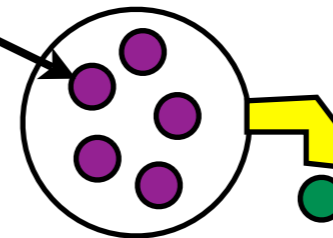
Threshold: State Preparation

State preparation is often done in two stages: **Encoding** & **Distillation**.



Encoding involves creating a fully-encoded state with a low but not very low logical error rate.

Then distillation usually involves comparing many copies of the poor encoded state to produce a smaller number of better encoded states.

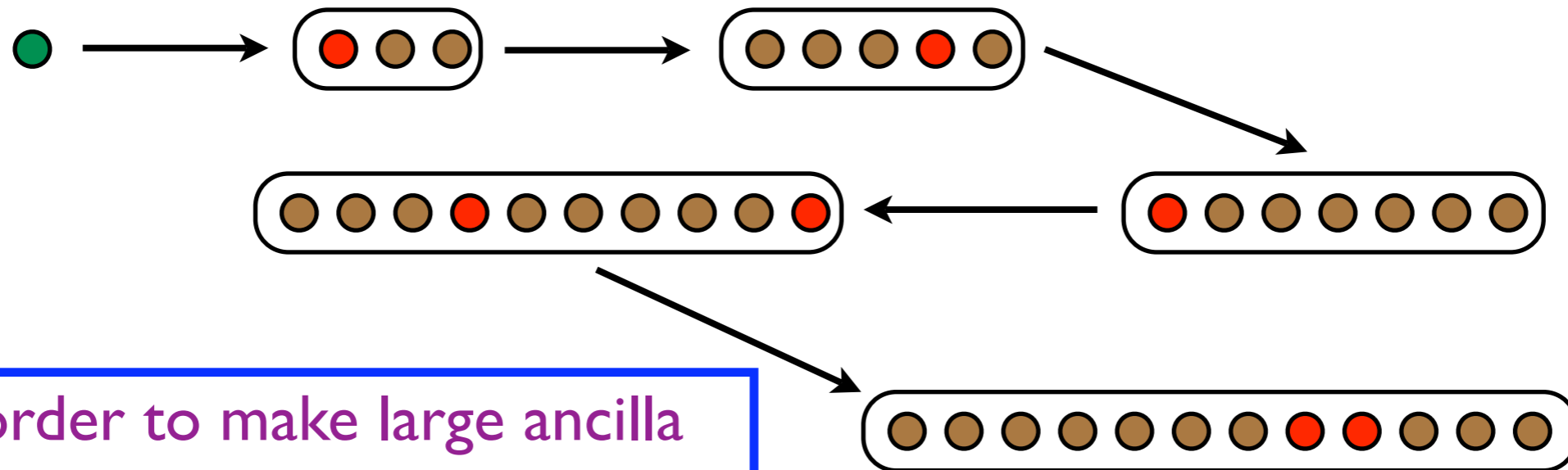


Distillation procedures work even for very big block sizes but require some fault-tolerant gates. It can definitely be made to work for CSS codes, and probably for any stabilizer code.

Thus, given a family of codes, we only need to figure out encoding.

Progressive Encoding

- For concatenated codes, we encode level-by-level, stopping at each level to perform error correction and state distillation.
- For topological codes, we encode by making first a small code and growing it, stopping periodically for EC and state distillation.

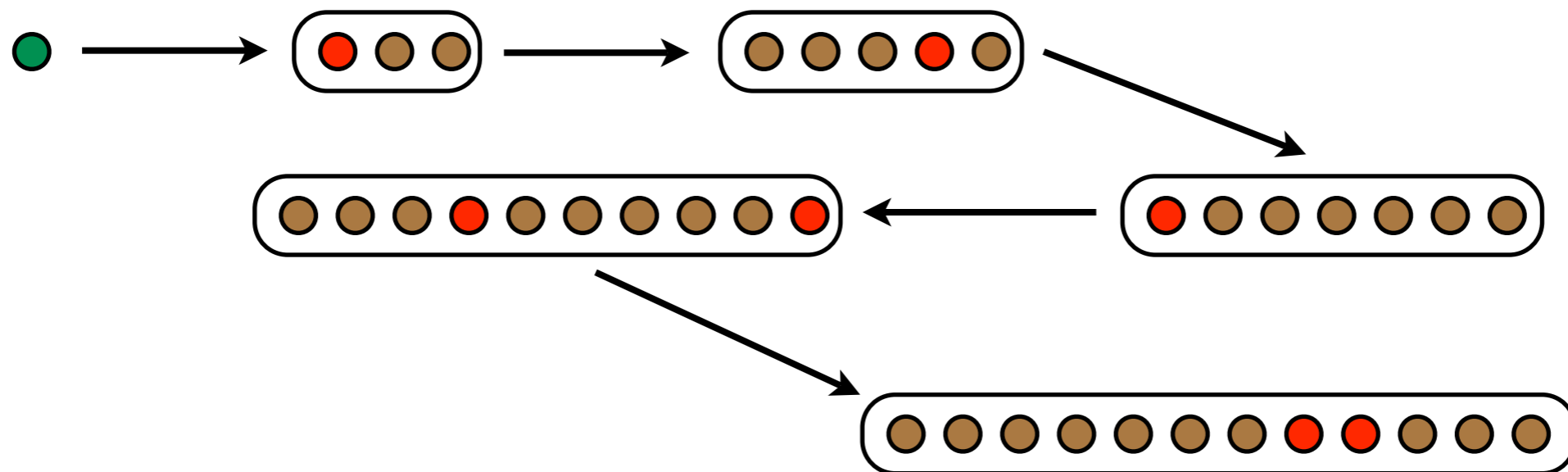


In order to make large ancilla blocks, we need **progressive encoding**: the ability to move to a larger member of the code family with a small circuit.

Threshold Existence: Summary

Summarizing, a code family with these properties gives a threshold:

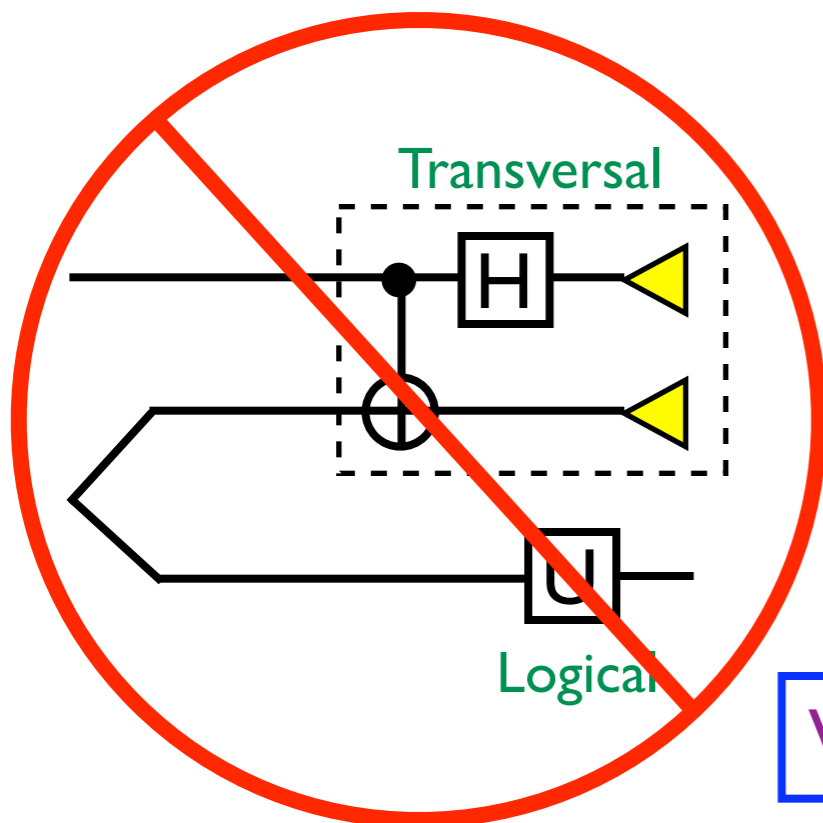
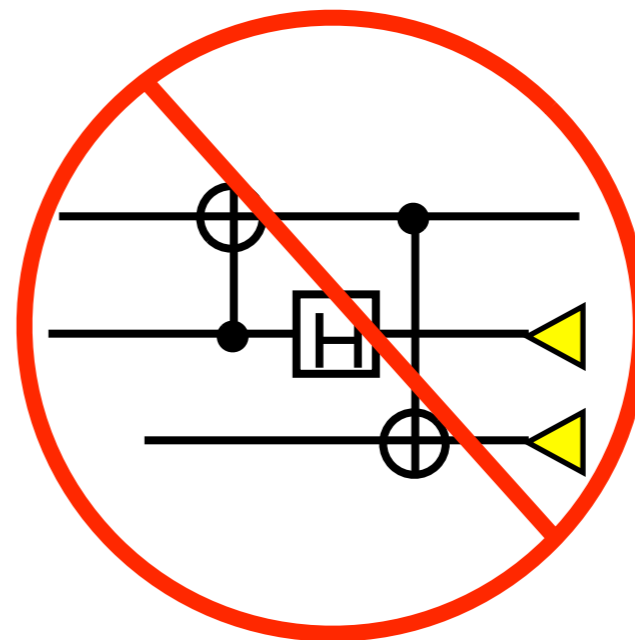
- CSS (or maybe stabilizer) codes.
- Corrects a constant fraction of likely errors (prob. p per qubit).
- Probability of a logical error $\rightarrow 0$ as $n \rightarrow \infty$.
- Efficient decoding algorithm.
- Progressive encoding



What is the Minimal Overhead?

What is the minimum **overhead** (ratio of physical qubits to logical qubits) required for a fault-tolerant threshold to exist?
For purposes of this discussion, assume free classical computation.

Obviously, we will want to use a family of codes that has a **good rate** (ratio logical qubits/ physical qubits).
Can we have overhead \approx rate?



However, Steane error correction will never give us an overhead better than 2, and Knill error correction cannot do better than an overhead of 3.

We need to use Shor EC and LDPC codes.

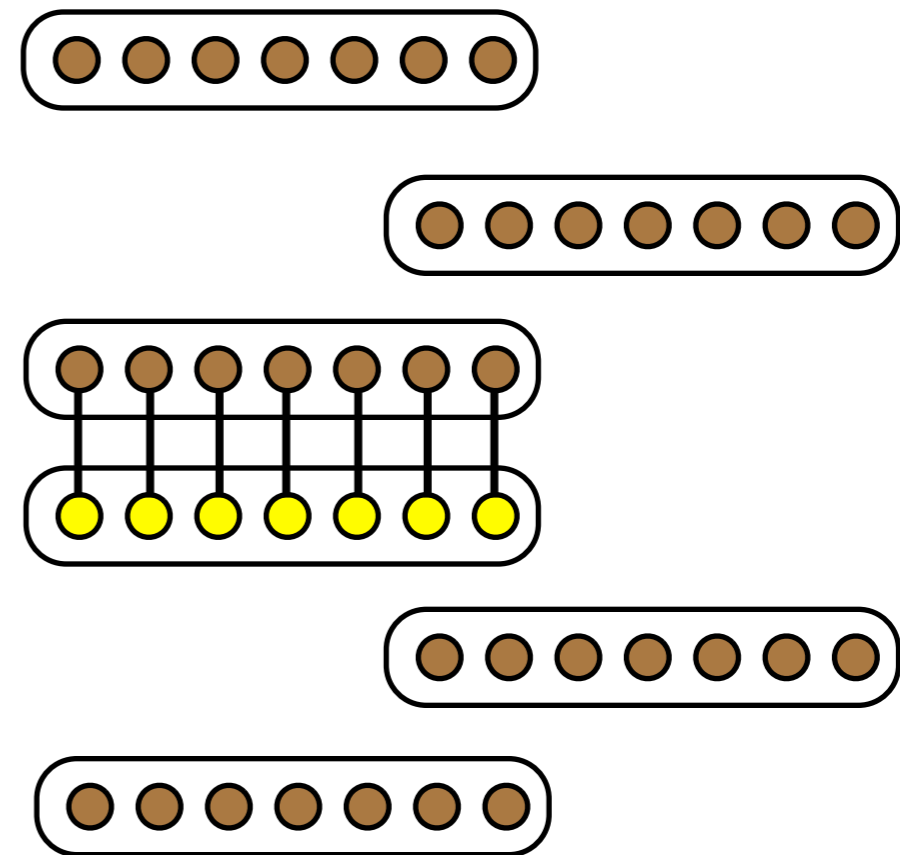
Keeping Gate Overhead Down

If we encode all qubits in one block, the ancilla blocks we need will drive the overhead much higher than the rate. The following tricks let us do gates with minimal overhead:

- Encode only $O(K/\text{polylog } K)$ logical qubits per block when there are K total logical qubits.
- Perform logical gates on only one block at a time.

Then, only one ancilla block is needed at a time, and it has size sublinear in the total system size.

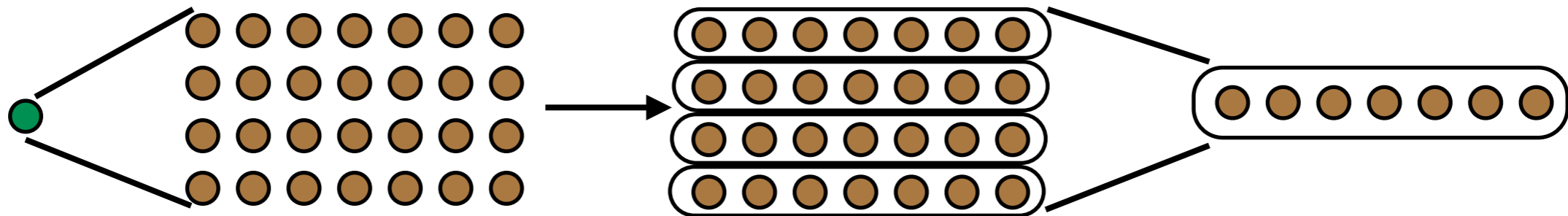
Note: block size cannot be too small or logical errors are not sufficiently suppressed.



EC and Preparation Overhead

Even without progressive encoding, we can encode by using another family of codes (e.g., concatenation) -- perform the LDPC encoding using FT concatenated gates, then decode the concatenated code.

This uses polylog overhead, so only encode one block at a time.

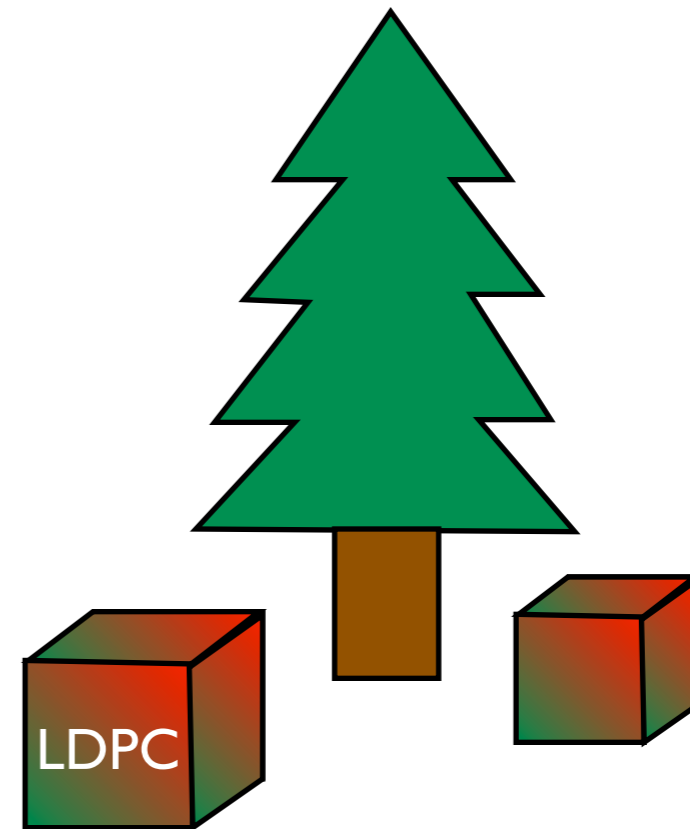


For error correction, every block must get attention, but it doesn't need attention every time step. Instead, perform EC on only a constant fraction of blocks at any given time. Then the extra overhead spent on cat states can be made arbitrarily small. The cost is that resistance to storage errors decreases.

Threshold with Constant Rate?

In short, if a family of QECCs exists with the following properties:

- Low density parity check codes.
- Corrects a constant fraction of likely errors (prob. p per qubit).
- Constant rate R as $n \rightarrow \infty$.
- Probability of a logical error $\rightarrow 0$ as $n \rightarrow \infty$.
- Efficient decoding algorithm.
- Robust against syndrome bit errors.*



Then:

There exists a threshold error rate for polynomial length computations, with overhead arbitrarily close to $1/R$.

* In Shor EC, syndrome bit errors can cause errors on multiple qubits.

Summary

- All fault-tolerant gates move through a family of codes to return to the starting code. The logical gate performed is a topological effect.
- To find a new family of codes with a threshold, the main property needed is progressive encoding.
- Based on what we know now, there seems no fundamental limit to overhead for a threshold. Indeed, if a good family of LDPC codes exist, a threshold exists with the same rate as the code family.

