

# Graphical Proofs for Fault-Tolerant Quantum Computation

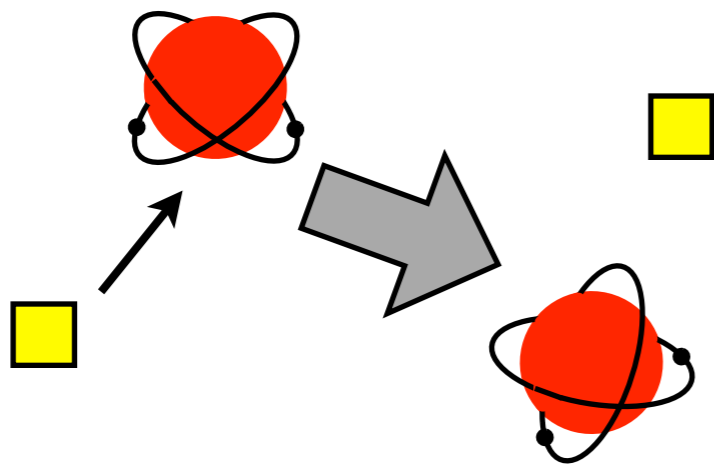
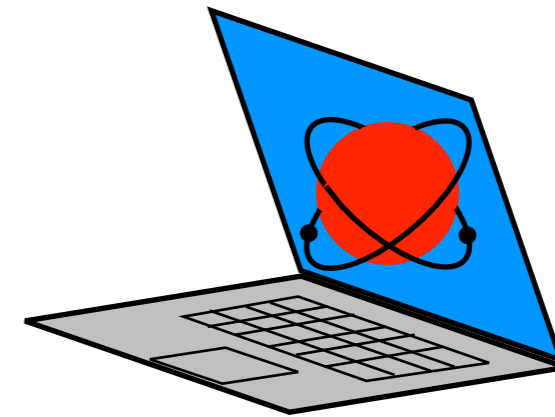
Daniel Gottesman  
Perimeter Institute  
Quantum Benchmark

Based on [quant-ph/0504218](https://arxiv.org/abs/quant-ph/0504218) with  
Panos Aliferis and John Preskill

# Quantum Computation

A quantum computer performs computations with single atoms or other quantum objects and can solve some problems exponentially faster than a classical computer.

- Quantum bits (“**qubits**”) have basis states labelled by 0 and 1 but can also be in **superpositions** or **entangled** states.
- A quantum computation is built up of individual components called **gates**, which include both quantum versions of classical gates and some new purely quantum gates.

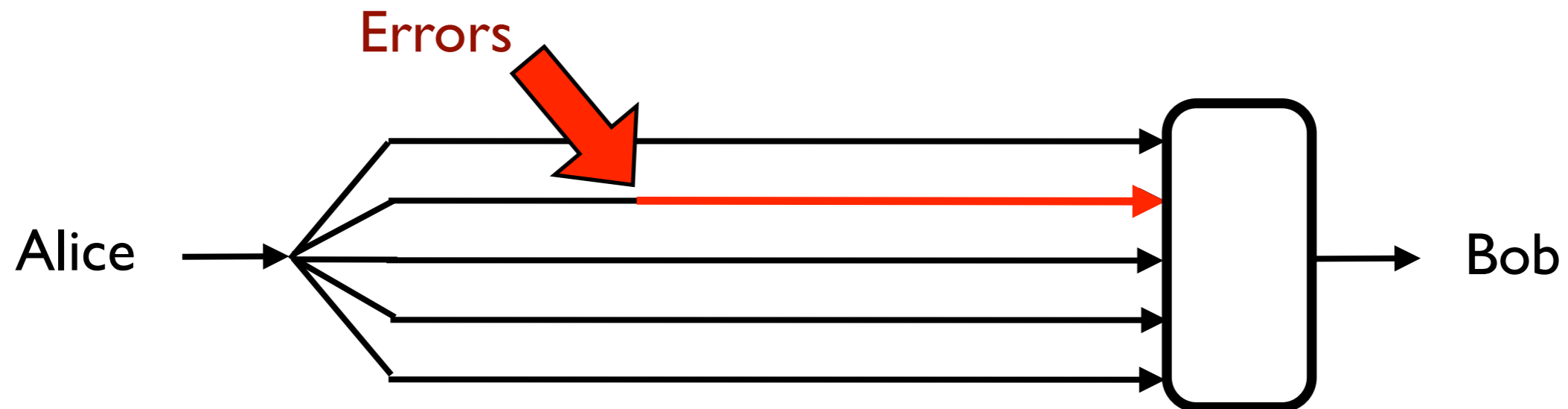


Qubits are inherently more sensitive to noise than classical bits:

- They are **very small**.
- Quantum states are susceptible to **decoherence** when the environment learns about the state of the system.

# Quantum Error Correction

Suppose Alice wants to send qubits to Bob, but errors in transmission are an issue. A solution is **quantum error correction**. Alice encodes her qubits using some extra qubits so that any errors can be identified and corrected.




Error correction is useful for transmission, but it assumes that Alice and Bob have perfect quantum computers. Also, we need to know how to perform gates on qubits while they are encoded. For this, we need **fault-tolerant quantum computation**.

# Some Definitions

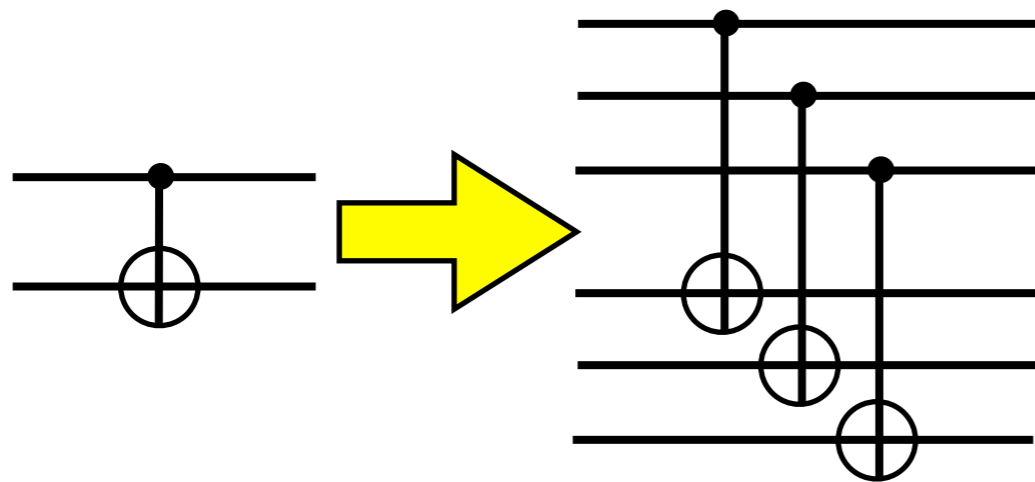
- **Qubit:** A system with a 2-dimensional Hilbert space. The Hilbert space of  $n$  qubits is  $2^n$ -dimensional.
- **Quantum error-correcting code (QECC):** A quantum error-correcting code is a subspace of the Hilbert space of  $n$  physical qubits. The subspace encodes  $k$  or more logical qubits. The code can correct errors on  $k$  or more physical qubits; a code that can correct  $t$  errors has distance  $2t+1$ .
- **Quantum gate:** A unitary operation that can be applied to  $k$  or more qubits.
- **State preparation:** A procedure that produces a qubit in a standard state (e.g.  $|0\rangle$ ) in a quantum computer.
- **Measurement:** A procedure that projects a qubit onto  $|0\rangle$  or  $|1\rangle$  with probability equal to the norm squared of the output state and produces a classical bit reporting the outcome.
- **Quantum circuit:** A sequence of state preparations, quantum gates, and measurements.

# What is a Fault-Tolerant Protocol?

A fault-tolerant protocol is a **circuit encoding**:

Ideal circuits  $C$   Fault-tolerant circuits  $C'$

The output bits of  $C'$  can be decoded to give an output distribution, which in the absence of error should be the same as the output of  $C$ . If we are working in an asymptotic context, the encoding and decoding maps should be **efficient**.



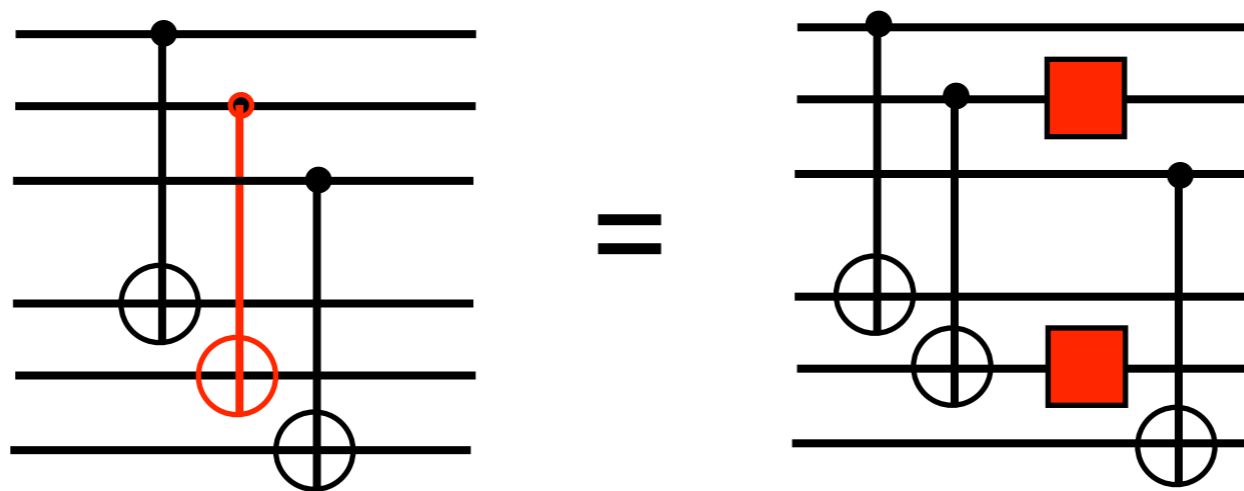
Usually each qubit (or group of qubits) is replaced by a QECC.

Each circuit element (state preparation, gates, measurement) is replaced by a **gadget** for that circuit element. Error correction gadgets are included.

But we also need some conditions reflecting the fault tolerance.

# Errors and Faults

The encoded circuit is made up of **physical gates**, each of which we assume may experience a **fault** with probability  $p$ . A gate with a fault does the correct unitary followed by an arbitrary quantum operation on the qubit(s) involved in the gate. Similarly for state preparations and measurements.

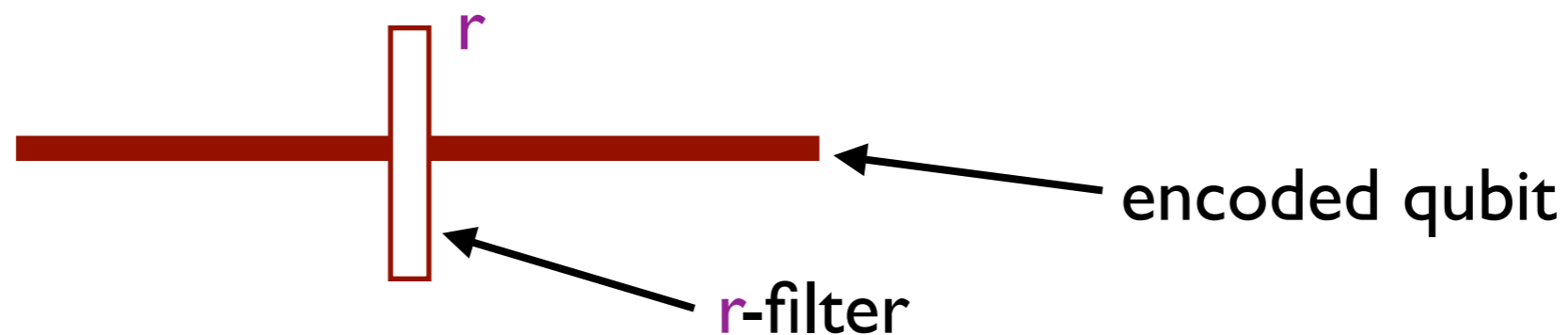


A general quantum operation is a **completely positive trace preserving (CPTP)** map on the density matrix, which can be written as  $\rho \mapsto \sum_k A_k \rho A_k^\dagger$ .

We can consider the state after the error to be  $A_k |\psi\rangle$ .

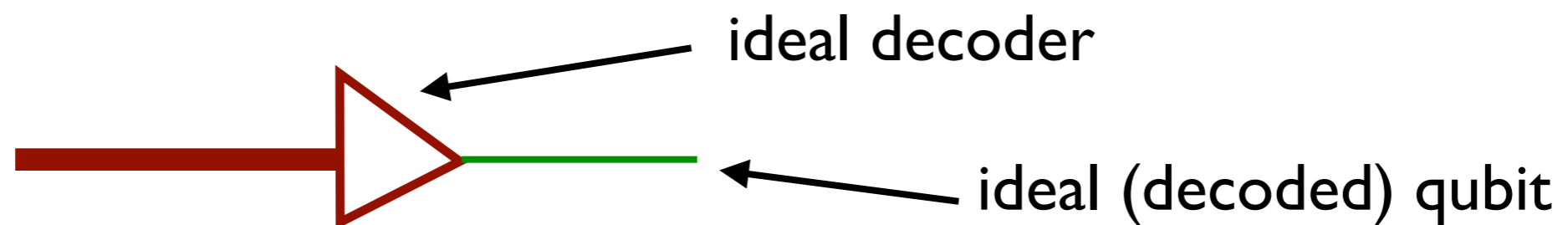
# Filters and Ideal Decoders

We want to say “this state has  $r$  (physical) errors on it.” We can do so via the  $r$ -filter, which is a projector onto the subspace spanned by the QECC with arbitrary  $r$ -qubit errors on it.



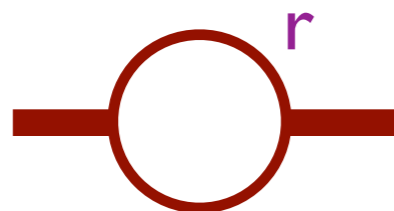
A state has  $r$  errors on it if an  $r$ -filter leaves it unchanged.

We also want to talk about the logical state of the computer even when errors are present. We do so using the *ideal decoder*, which corrects the errors and then decodes to the logical qubit using a fault-free decoding circuit.

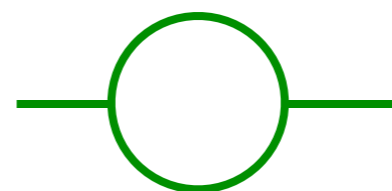


# Pictures for Gadgets

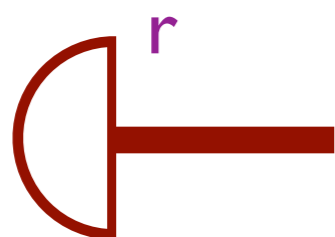
I will represent the different kinds of gadgets with the following pictures. Some represent gadgets with faults in the physical qubits acting on the QECC and some are for an unencoded instance of the circuit element on ideal unencoded qubits.



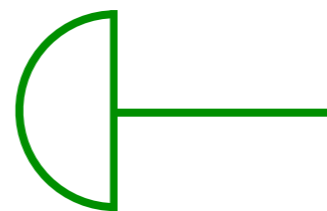
Gate gadget with  $r$  faults



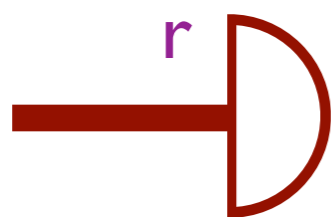
Ideal gate



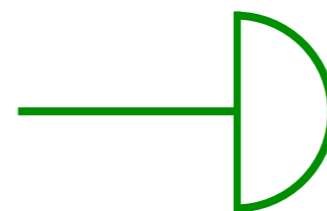
Preparation gadget with  $r$  faults



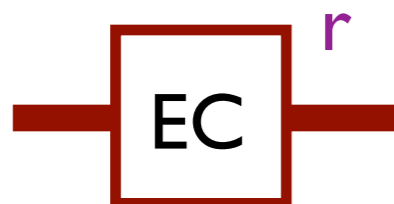
Ideal preparation



Measurement gadget with  $r$  faults



Ideal measurement



Error correction gadget with  $r$  faults

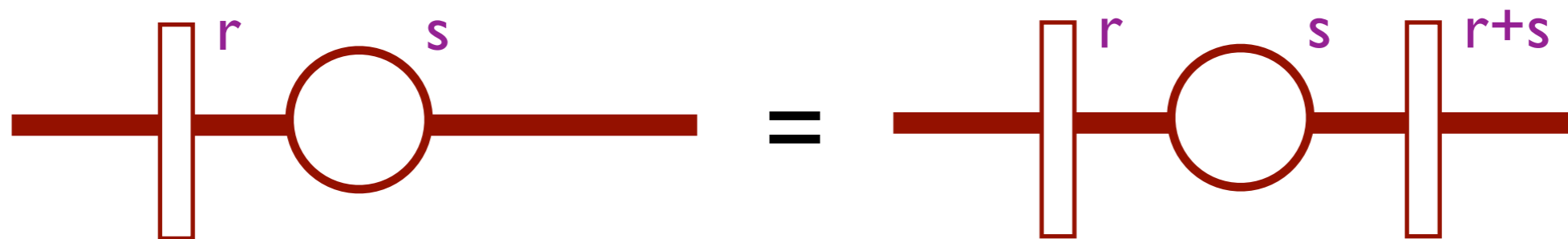
Each picture represents a CPTP map.



# Fault-Tolerant Gate Gadgets

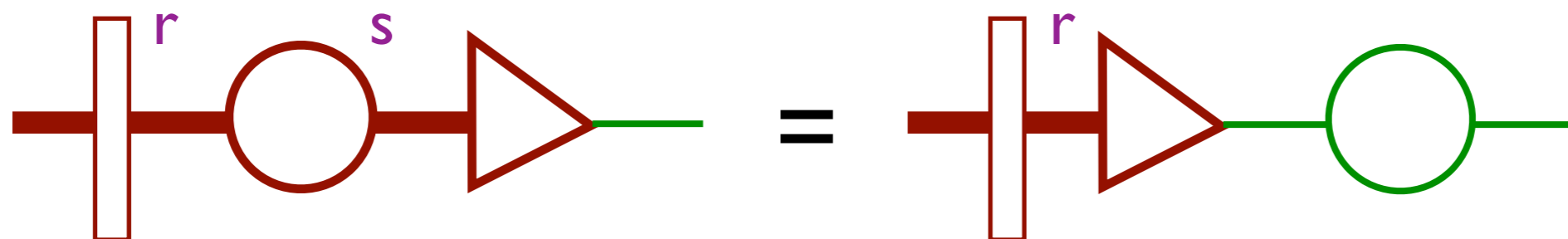
A gate gadget is fault tolerant if it satisfies two properties when  $r + s \leq t$ , where  $t$  is the number of errors the QECC can correct:

Gate Propagation Property (GPP):



This says errors don't propagate too badly: the number of errors after the gadget is the number before plus faults in the gadget.

Gate Correctness Property (GCP):

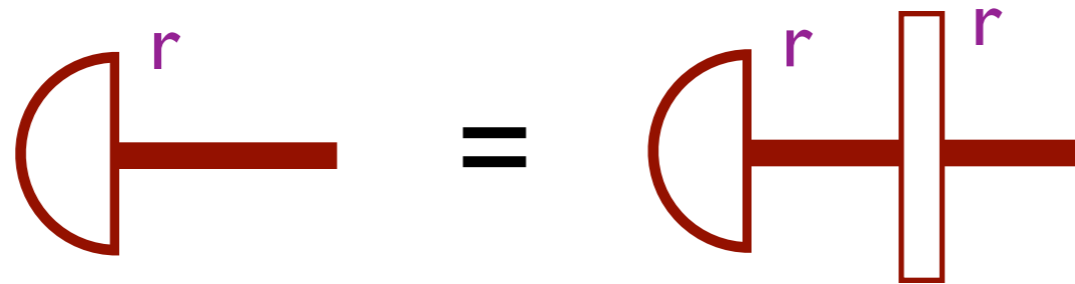


This says the encoded state changes according to the gate.

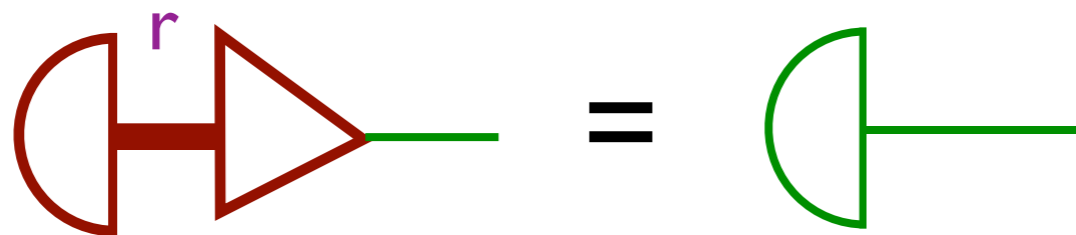
These properties must hold for arbitrary sets of  $s$  faults.

# FT for Preparation & Measurement

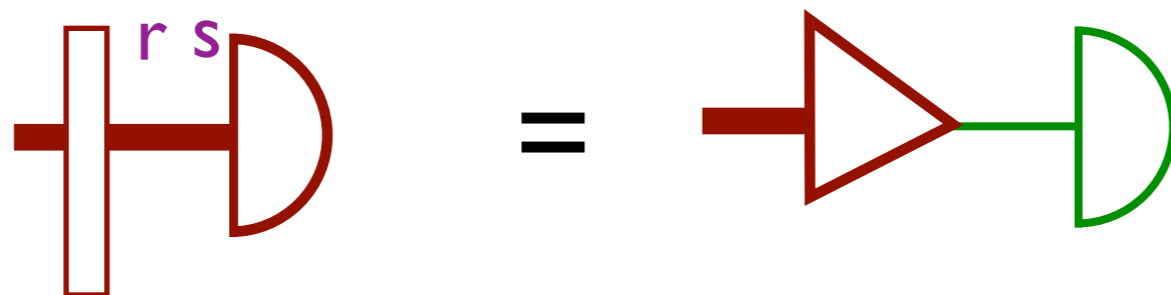
Preparation Propagation Property (PPP):



Preparation Correctness Property (PCP):



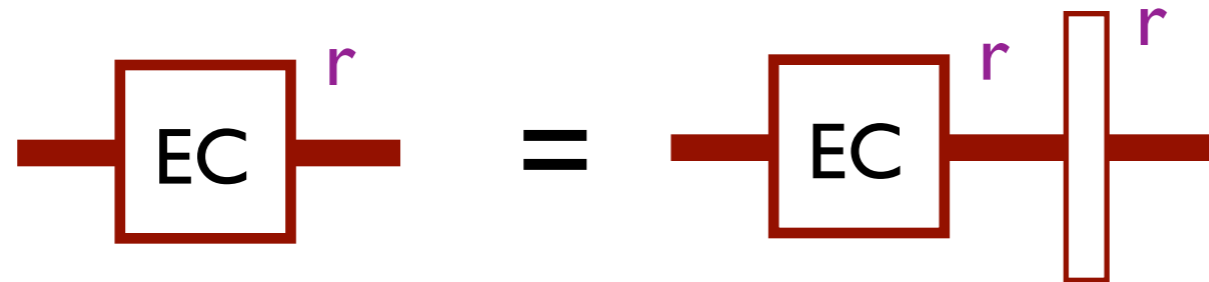
Measurement Correctness Property (MCP):



There is no measurement propagation property because the output of a measurement is a classical bit (assumed perfect).

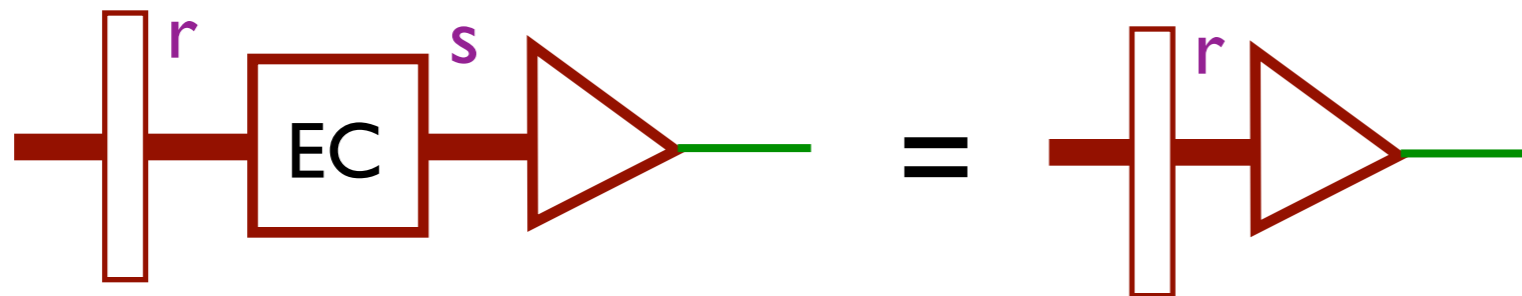
# FT Error Correction Gadgets

Error Correction Recovery Property (ECRP):



Note that the ECRP says an EC gadget outputs a state with at most  $r$  errors no matter what the input is. This is so that even if we have a bad error at some point, the remainder of the fault tolerant circuit can behave normally.

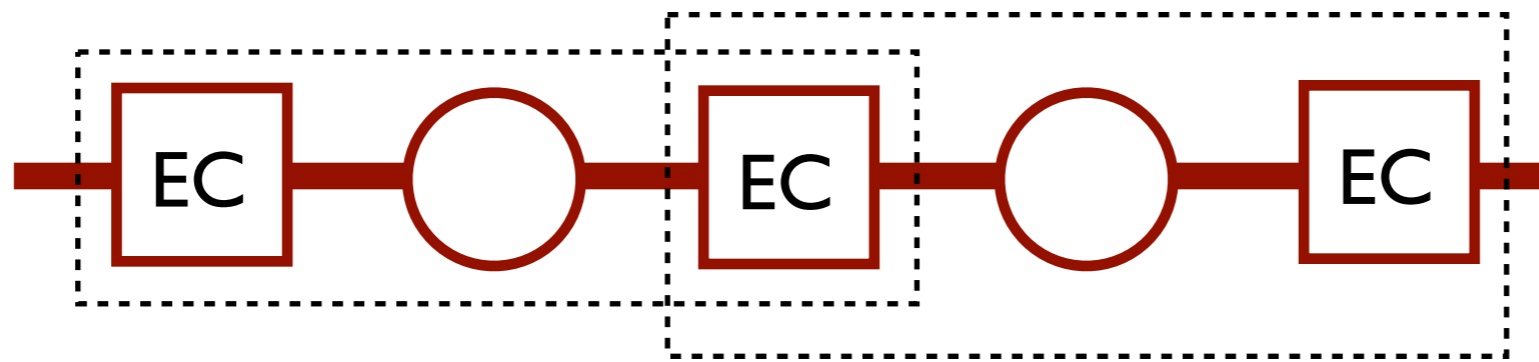
Error Correction Correctness Property (ECCP):



The EC step should not change the encoded state.

# Extended Rectangles

A fault-tolerant circuit alternates gate (or preparation or measurement) gadgets with EC gadgets. An **extended rectangle** consists of a gate gadget (or preparation or measurement) plus the EC gadgets before and after it.

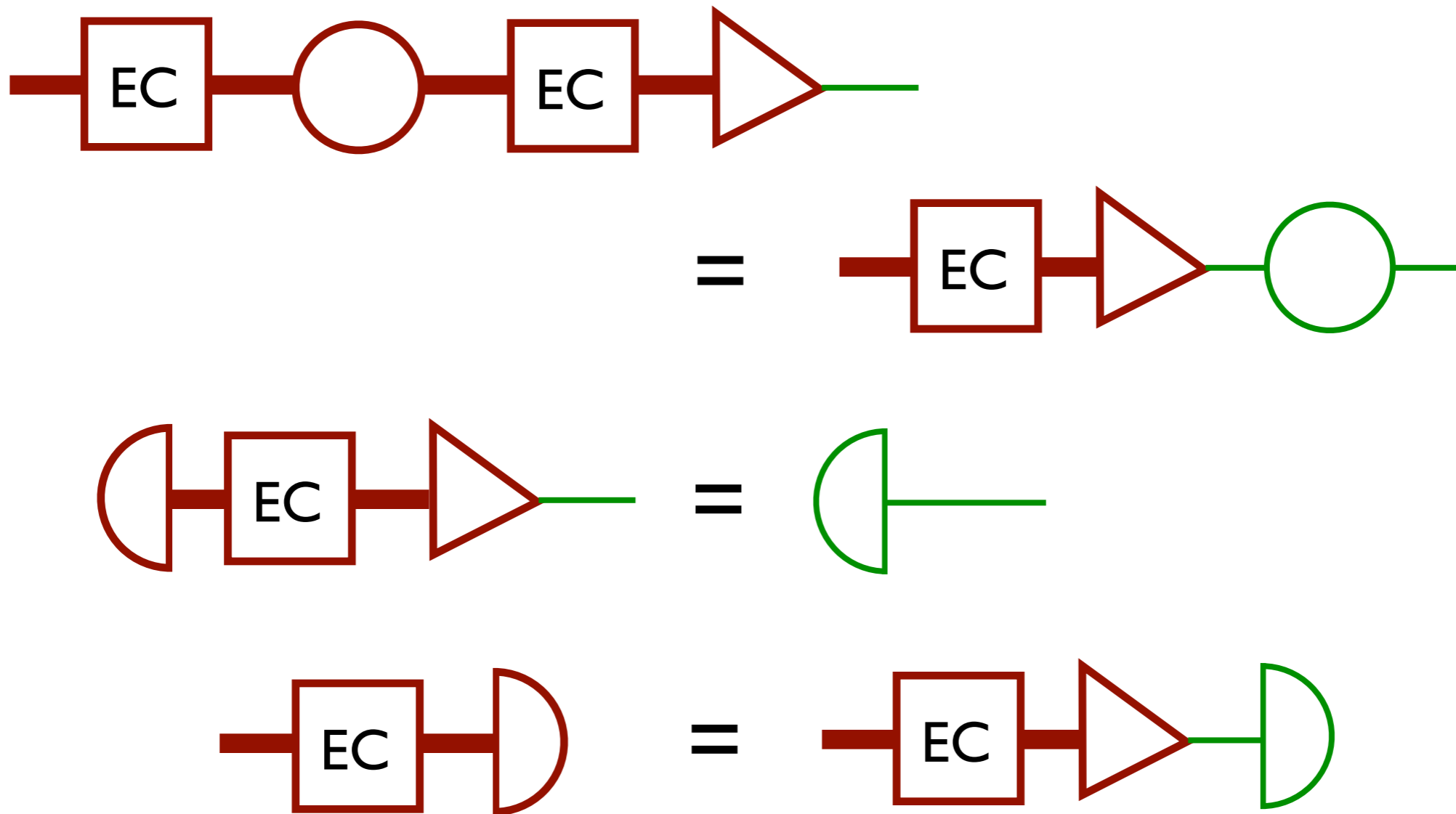


**Note:** Extended rectangles overlap.

An extended rectangle is **good** if it has at most  $t$  faults in it (where the QECC can correct  $t$  errors). Otherwise it is **bad**.

# Correctness

An extended rectangle is **correct** if the gadget does the correct action on the encoded state. Specifically, if:

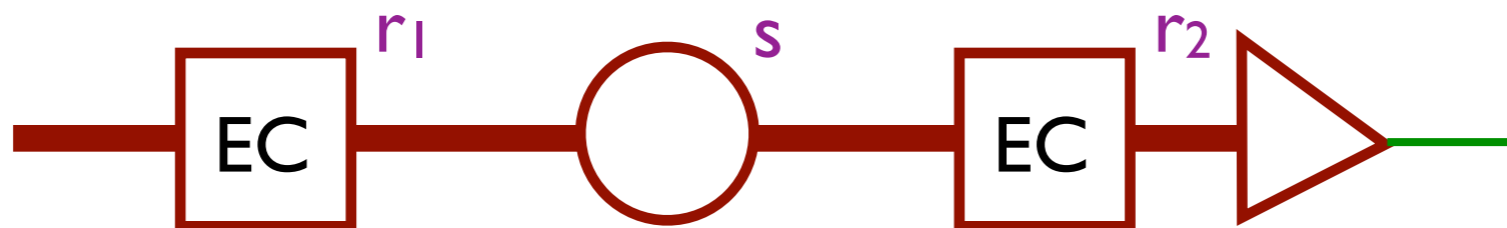


# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .

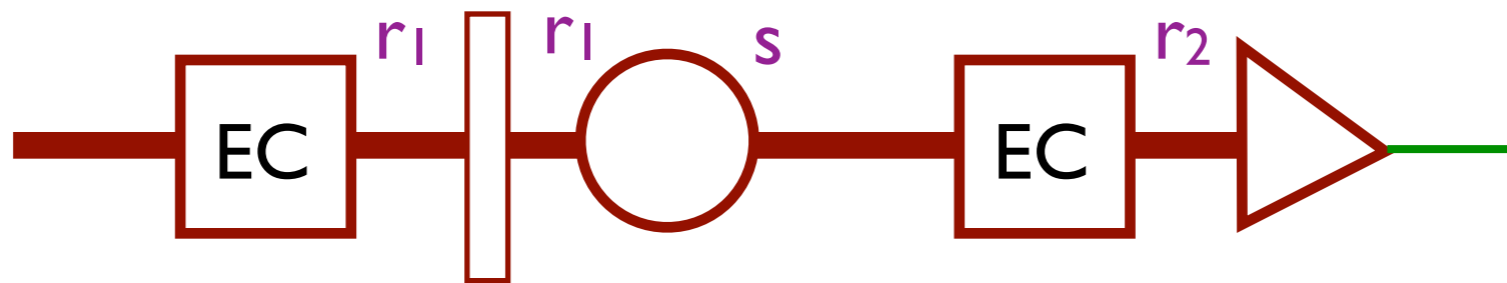


# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .



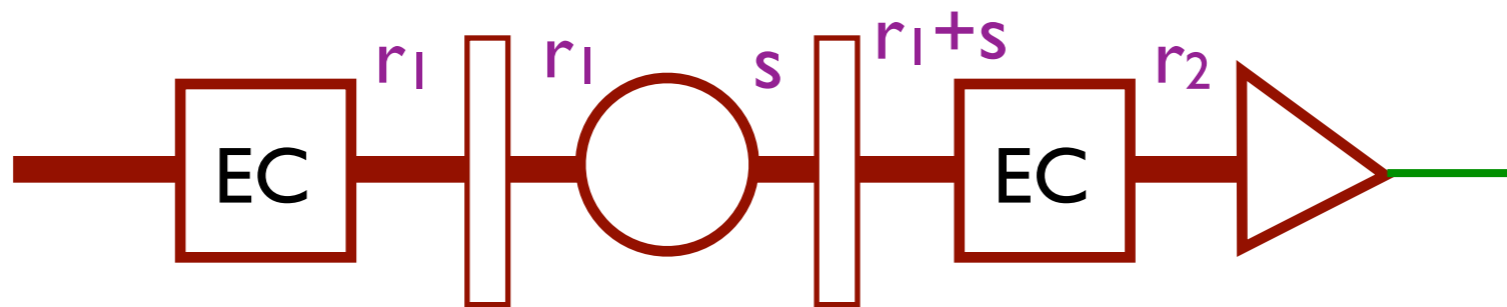
By ECRP

# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .



By GPP

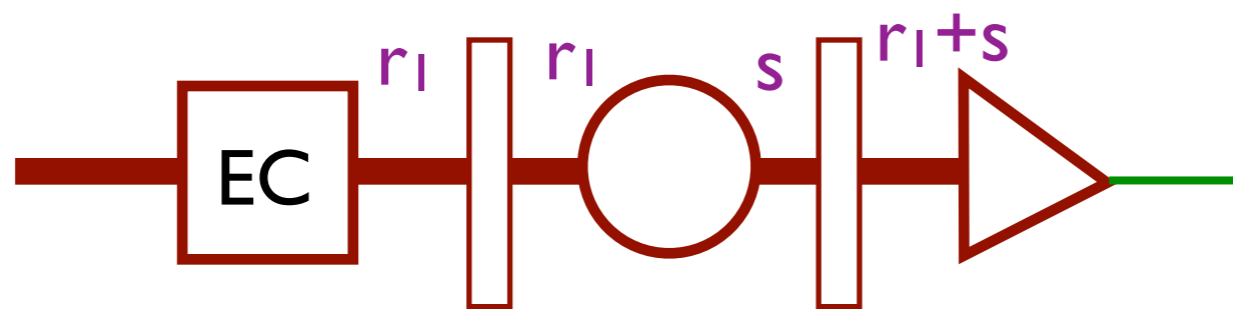


# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .



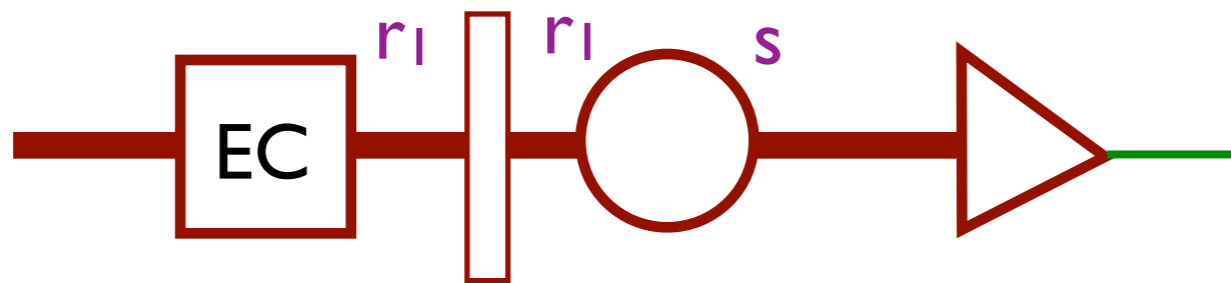
By ECCP

# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .



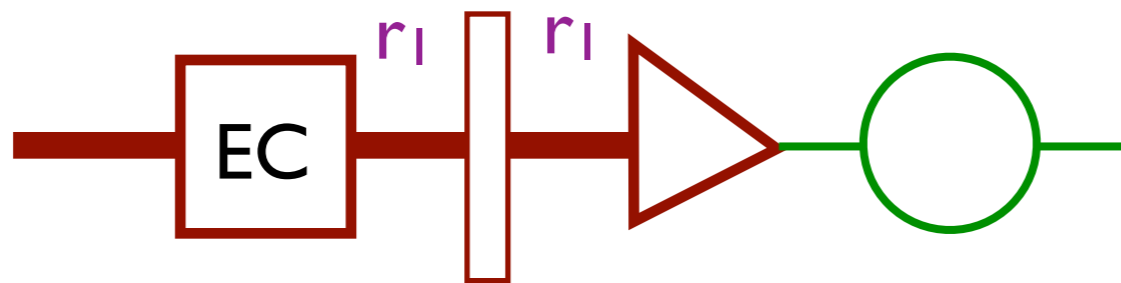
By GPP (reverse)

# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .



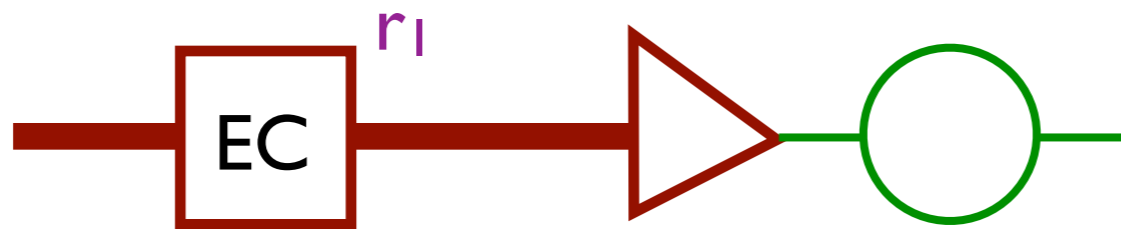
By GCP

# Good = Correct

**Proposition:** A good extended rectangle is correct.

**Proof:** (for the gate gadget case)

Suppose there are  $r_1$  faults in the first EC,  $s$  in the gate gadget, and  $r_2$  faults in the second EC. Then  $r_1 + r_2 + s \leq t$ .

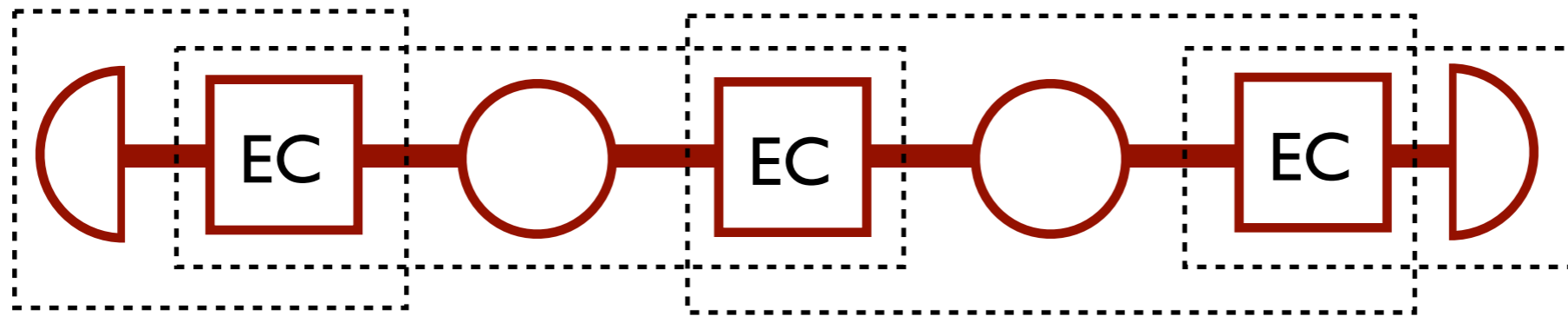


By ECRP (reverse)

# Correctness of Circuits

**Proposition:** If all the extended rectangles in a circuit are good, the fault-tolerant circuit is equivalent to an ideal unencoded circuit.

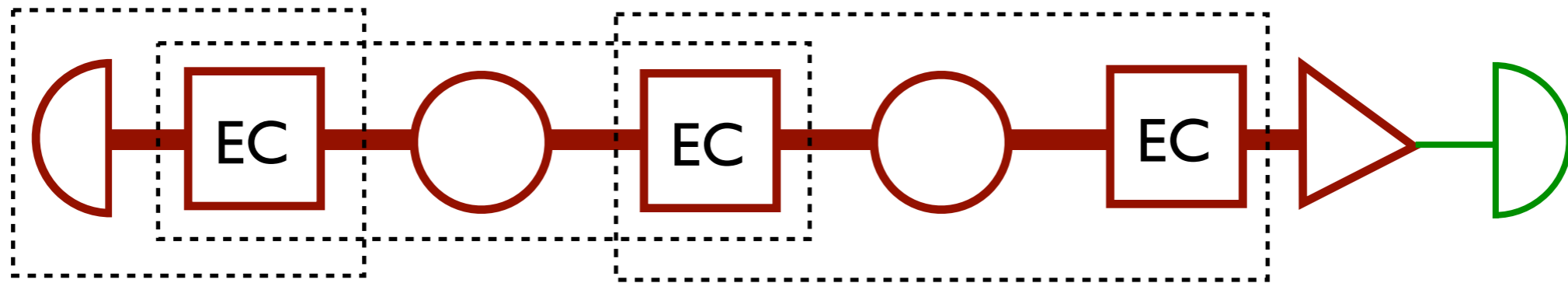
**Proof:**



# Correctness of Circuits

**Proposition:** If all the extended rectangles in a circuit are good, the fault-tolerant circuit is equivalent to an ideal unencoded circuit.

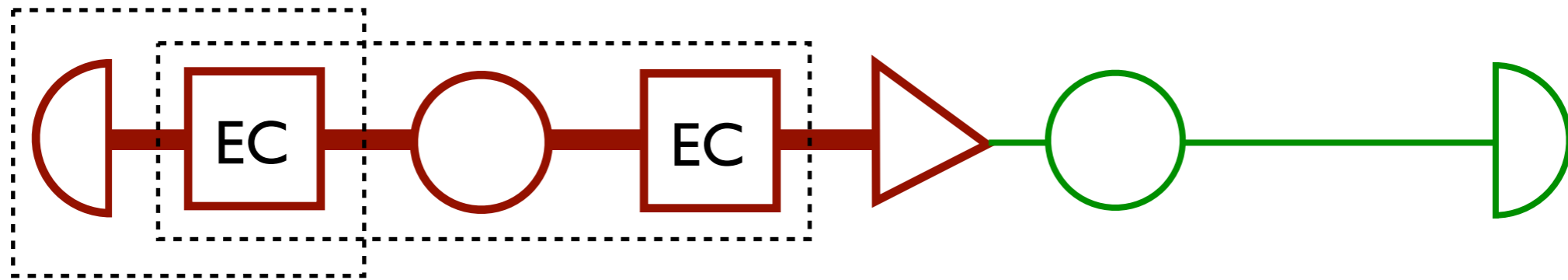
**Proof:**



# Correctness of Circuits

**Proposition:** If all the extended rectangles in a circuit are good, the fault-tolerant circuit is equivalent to an ideal unencoded circuit.

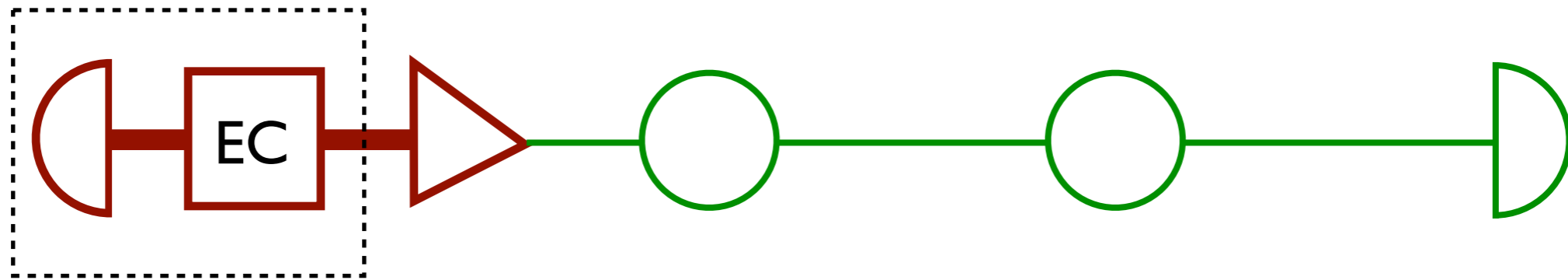
**Proof:**



# Correctness of Circuits

**Proposition:** If all the extended rectangles in a circuit are good, the fault-tolerant circuit is equivalent to an ideal unencoded circuit.

**Proof:**





# Correctness of Circuits

**Proposition:** If all the extended rectangles in a circuit are good, the fault-tolerant circuit is equivalent to an ideal unencoded circuit.

**Proof:**

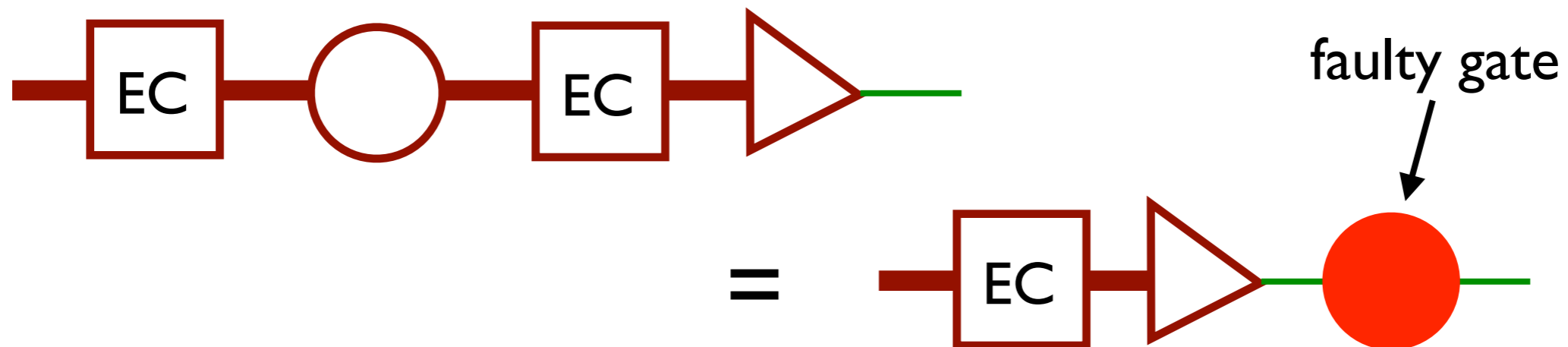


# Bad Extended Rectangles

If there are  $A$  circuit elements in an extended rectangle, the probability of the extended rectangle being bad is at most

$$p_{\text{bad}} \leq \binom{A}{t+1} p^{t+1}$$

If there is a bad extended rectangle, however, the previous argument does not work. We have no rule that lets us move an ideal decoder through a bad extended rectangle. We want to say

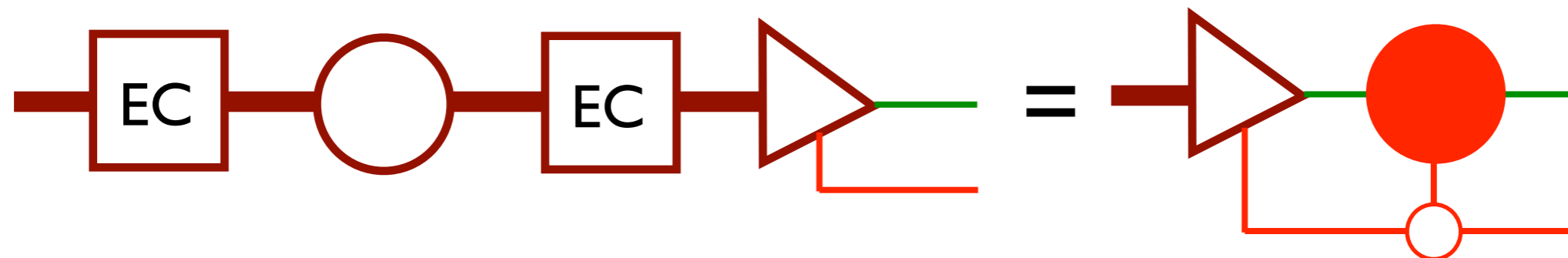


for a bad extended rectangle, but this is not true.

# \*-Decoders

The problem is that the type of fault that occurs on the RHS is a combination of faults on the LHS and errors on the qubits at the start of the extended rectangle. On the RHS, the picture implies that the fault is supposed to be independent of the errors entering the extended rectangle.

The solution is to use a \*-decoder that keeps track of the incoming error information:

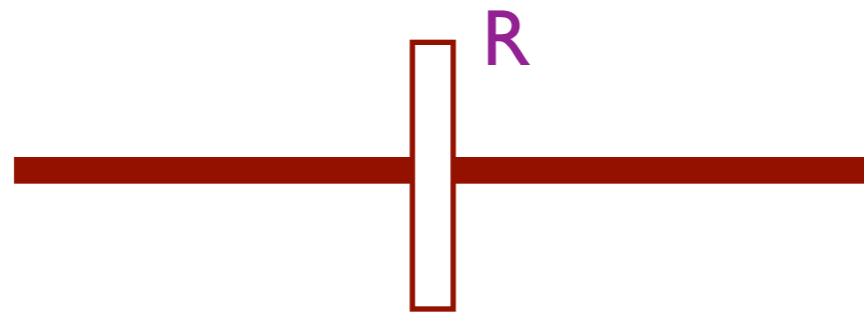


We push the \*-decoder back past the first EC in order to remove correlations causing overlapping extended rectangles to both be bad.

**Theorem:** (Rough) A FT circuit is equivalent to a circuit with error rate at most  $\binom{A}{t+1} p^{t+1}$ .

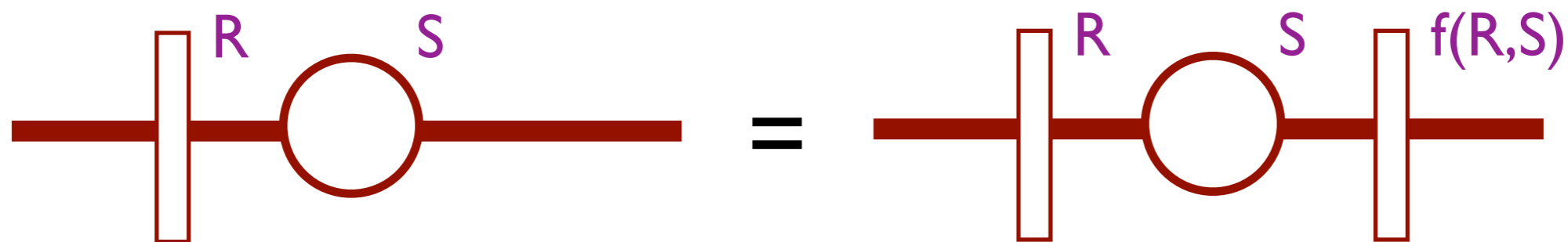
# More General Error Models

What if some types of errors are more common than others? The current approach treats all of them as equally bad, but with biased noise, we should use QECCs capable of correcting more of the most common sorts of errors.



Let's replace the  $r$ -filter with an  $R$ -filter, where  $R$  is some arbitrary set of errors.

The propagation properties now look like this:



There is also a composition property on the functions  $f$  which determines which sets of errors are bad.

(This part is work in progress and there is not yet a full theory.)

# Summary

- This approach enables us to isolate small segments of a fault-tolerant circuit and analyze fault tolerance in a single segment rather than having to analyze the whole circuit together.
- The picture language naturally captures the relationships between errors and faults and between physical and logical qubits.
- Note that little here is specific to quantum mechanics. The same language should work with classical fault tolerance or fault tolerance in some alternate physical theory.

## Open questions:

- The language described here does not work with all types of fault-tolerant protocols. Can it be extended to other kinds?
- How can we extend it to work with biased noise models?