Verifying Quantum Error Correction Codes with SAT Solvers

Finding Bugs in Surface Code Implementations

Pengyu Liu, Mengdi Wu

December 1, 2025

Outline

What is Quantum Error Correction?

- Quantum computers are susceptible to errors from decoherence and noise
- Quantum error correction codes protect quantum information
- Key question: **How do we evaluate whether a quantum error correction code is good?**
  - Can it correct a certain number of errors?
  - What is the maximum number of correctable errors?

What is the Surface Code?

- One of the most promising quantum error correction codes
- Based on topological properties
- Qubits arranged in a 2D lattice
- Error correction performed through stabilizer measurements
- Widely studied and implemented in quantum computing platforms

SAT Encoding Techniques

Goal: Encode quantum error correction verification as a SAT problem

Key Encodings:

- **Totalizer encoding** for cardinality constraints
  - Efficiently encode constraints like "at most k errors occur"
- **Tree encoding** for XOR constraints
  - Handle parity checks efficiently

Verification Strategy

Two types of problems:

Can the code correct k errors? (UNSAT Problem)

- Try to find a counterexample with $\leq k$ errors that cannot be corrected
- If UNSAT, the code can correct k errors

Can the code fail with k errors? (SAT Problem)

- Try to find an example with $\leq k$ errors that leads to failure
- If SAT, the code cannot correct all k-error cases

Bug Discovery in Nature Paper

Major Achievement: We successfully identified and verified a bug in a recently published Nature paper!

Distance	Actual Correctable Errors	Claimed
3	0	0
5	1	1
7	2	2
9	3	3
11	**3**	4
13	**4**	5

The code fails to correct the claimed number of errors for distances 11 and 13!

Performance Results

Bug Detection: Pretty Fast! ✓

- SAT solver quickly finds counterexamples
- Verification of bug completed in reasonable time
- Demonstrates effectiveness of SAT-based approach

Runtime details: [Include specific timing results]

The Verification Challenge

The Problem

We can propose a fix for the bug, but we **cannot verify** whether it works using SAT solvers alone.

Verification is very slow...

- Proving correctness requires solving UNSAT problems
- Much harder than finding bugs (SAT problems)

Runtime details:

Why is This So Hard?

Combination of SAT solver weaknesses:

1. **UNSAT problems**
   - Proving non-existence is inherently harder than finding examples
2. **XOR encodings**
   - SAT solvers struggle with parity constraints
3. **Cardinality constraints**
   - "At most k errors" constraints are challenging
4. **Pigeonhole principle**
   - Notoriously hard for SAT solvers

Each alone is challenging; together they are formidable!

A Hybrid Approach

Leverage the strengths of different tools:

SAT Solvers: Fast Pruning

- Quickly find bugs and counterexamples
- Prune the search space efficiently
- Identify promising candidates

Lean Theorem Prover: Formal Verification

- Formally verify correctness of proposed fixes
- Provide mathematical proof of error correction properties
- Guarantee correctness where SAT solvers struggle

Combine SAT and Lean for comprehensive verification!

Summary

- **Problem:** Verifying quantum error correction codes
- **Approach:** SAT solver with specialized encodings
- **Success:** Found bugs in published Nature paper
- **Challenge:** Verifying fixes is computationally hard
- **Future:** Hybrid SAT + Lean approach

Thank you!

Questions?