Verifying Quantum Error Correction Codes with SAT Solvers

Finding Bugs in Surface Code Implementations

Pengyu Liu, Mengdi Wu

December 1, 2025

Outline

Why Quantum Error Correction is Different

Classical Error Correction:

- Errors mainly occur during transmission/storage
- Can copy data freely for redundancy
- Can measure directly without disturbing data

Quantum Error Correction:

- **Errors occur continuously during computation**
- No-cloning theorem: cannot copy quantum states
- Measurement inevitably destroys quantum states

Detectors: Measuring Errors Indirectly

- Through measurements, we convert quantum states into classical bits.
- A linear combination of measurements that is deterministic under no error is a detector.
- The decoder, which is a classical algorithm, rely on the detectors to infer the error pattern.

Key Idea

Different error patterns $\rightarrow$ different detectors $\rightarrow$ error correction possible

Key question: How do we evaluate whether a quantum error correction code is good?

- What is the maximum number of correctable errors?

The Distance d of a Code

A code with distance d can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.

- Distance 3 code: corrects 1 error
- Distance 5 code: corrects 2 errors

Our Approach: Assuming Optimal Decoder

What does optimal mean?

- Decoder uses detector detectors to infer which errors occurred
- **Optimal decoder:** Always chooses the minimum-weight error pattern consistent with the detector

For linear codes:

- If distance is d, then any two error patterns with $< d$ errors produce different detectors OR same logical outcome
- **Simplified verification:** Just check if errors with *zero detector* (no detectors fired) can corrupt the logical qubit!

Why Zero-detector Check is Sufficient

General problem: Two different error patterns E_1 and E_2 might produce the same detector but different logical outcomes

- This would fool the decoder!
- Checking all pairs would be exponential

Our simplification (for linear codes):

- Due to linearity: $E_1 \oplus E_2$ produces zero detector
- If E_1 and E_2 differ in logical outcome, then $E_1 \oplus E_2$ is a zero-detector logical error
- **So we only need to find zero-detector errors that flip the logical!**

Result

Distance d = minimum weight of zero-detector logical error $\Rightarrow$ can correct $\lfloor \frac{d-1}{2} \rfloor$ errors

SAT Encoding: The Big Picture

Goal: Encode quantum error correction verification as a SAT problem

What we need to encode:

1. Each possible error mechanism: boolean variable e_i
2. Detector constraints: each detector measures XOR of certain errors
3. Observable constraints: logical qubits are corrupted by XOR of errors
4. Cardinality constraint: at most k errors occur

Challenge: SAT solvers work with AND/OR/NOT, but quantum error correction uses XOR extensively!

The Input: Detector Error Model (DEM)

Input format: Stim's DEM file describes error mechanisms

Example DEM Entries

```
error D0 D2 L0
error D1 D3
error D0 D1
```

Interpretation:

- Each `error` line is one error mechanism
- `D#`: This error triggers detector #
- `L#`: This error flips logical observable #

Encoding Step 1: Boolean Variables

Create one boolean variable per error mechanism

- Parse DEM file to count n error mechanisms
- Create variables: $e_1, e_2, \ldots, e_n$
- $e_i =$ True means error i occurs
- $e_i =$ False means error i does not occur

Example

If DEM has 100 error lines, we create variables $e_1, \ldots, e_{100}$

Encoding Step 2: XOR Constraints (The Hard Part)

Problem: Detectors compute XOR, but SAT uses AND/OR/NOT

Example: Detector D0 fires iff $e_1 \oplus e_3 \oplus e_7 = 1$

For verification, we want detectors to NOT fire:

$$e_1 \oplus e_3 \oplus e_7 = 0$$

Solution: Tseitin Transformation

- Introduce auxiliary variables
- Convert XOR into CNF clauses using helper variables
- Two methods: *chain* and *tree* encoding

XOR Encoding: Chain Method

Encode $e_1 \oplus e_2 \oplus e_3 \oplus e_4 = 0$

Chain approach:

1. Create auxiliary variable $a_1 = e_1 \oplus e_2$
2. Create auxiliary variable $a_2 = a_1 \oplus e_3$
3. Create auxiliary variable $a_3 = a_2 \oplus e_4$
4. Assert $a_3 = 0$

Binary XOR encoding: $c = a \oplus b$ becomes 4 CNF clauses:

- $\neg a \vee \neg b \vee \neg c$
- $a \vee b \vee \neg c$
- $a \vee \neg b \vee c$
- $\neg a \vee b \vee c$

XOR Encoding: Tree Method

Encode $e_1 \oplus e_2 \oplus e_3 \oplus e_4 = 0$

Tree approach (more parallel):

1. Level 1: $a_1 = e_1 \oplus e_2, \quad a_2 = e_3 \oplus e_4$
2. Level 2: $a_3 = a_1 \oplus a_2$
3. Assert $a_3 = 0$

Trade-off

Chain: Linear depth, can be slow for SAT propagation
Tree: Logarithmic depth, better propagation, more variables

Encoding Step 3: Cardinality Constraints

Constraint: At most k errors occur

$$\sum_{i=1}^{n} e_i \leq k$$

Naive encoding: Forbid all $\binom{n}{k+1}$ combinations $\rightarrow$ exponential!

Totalizer encoding: Efficient polynomial-size encoding

- Builds a circuit that counts the number of true variables
- Uses auxiliary variables to represent partial sums
- Results in $O(nk)$ clauses and variables
- Provided by PySAT

Encoding Step 4: Observable Constraints

Goal: Find errors that corrupt the logical qubit

Constraint: At least one logical observable is flipped

1. For each observable L_j, encode: $r_j = \bigoplus_i e_i$ where error i affects L_j
2. Assert: $r_1 \vee r_2 \vee \cdots \vee r_m$ (at least one observable flipped)

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 [1]!

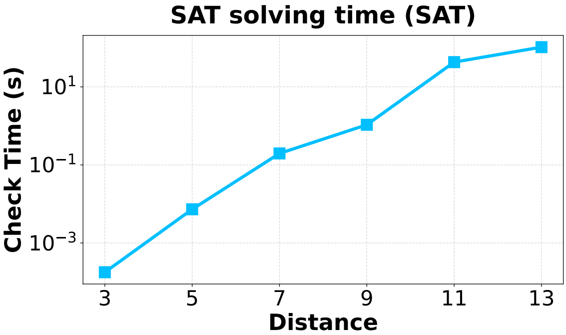
Distance	Actual Correctable Errors	Claimed
3	0	0
5	1	1
7	2	2
9	3	3
11	**3**	4
13	**4**	5

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



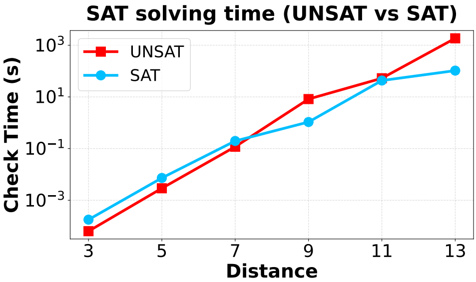
The Verification Challenge

The Problem

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

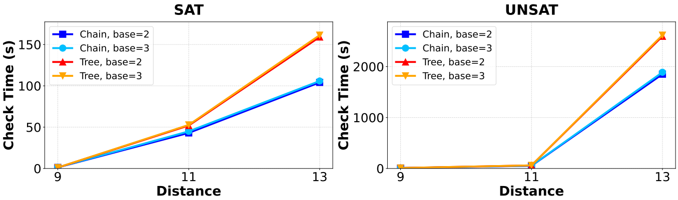
Verification is very slow...

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



Comparison of XOR Encoding Strategies

1. **Chain Encoding**: sequentially combine variables using XOR
2. **Tree Encoding**: parallel, balanced reduction of XOR terms
3. **Base-2 Encoding**: use $c = a \oplus b$ as the atomic building block
   $\Rightarrow$ introduces more auxiliary variables but simpler constraints
4. **Base-3 Encoding**: use $d = a \oplus b \oplus c$ as the atomic block
   $\Rightarrow$ reduces auxiliary variables but increases clause complexity



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]
3. **Cardinality constraints**
   - "At most k errors", is similar to Pigeonhole principle, which is challenging [2]

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

Summary

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

Thank you!

Questions?

Dolev Bluvstein, Alexandra A Geim, Sophie H Li, Simon J Evered, J Pablo Bonilla Ataides, Gefen Baranes, Andi Gu, Tom Manovitz, Muqing Xu, Marcin Kalinowski, et al.
A fault-tolerant neutral-atom architecture for universal quantum computation.
Nature, pages 1–3, 2025.

Armin Haken.
The intractability of resolution.
Theoretical computer science, 39:297–308, 1985.

Alasdair Urquhart.
Hard examples for resolution.
Journal of the ACM (JACM), 34(1):209–219, 1987.