Abstractions for Error Masking

Motivation

Verifying concrete programs is hard!

Can we simplify our programs while preserving correctness guaranties?

Sound Abstractions

Correct Abstract Program ⇒ Correct Concrete Program

- No false negatives.
- There might be false positives.

Different Levels of Abstraction

Verification Time VS # False Positives

Supported Operations

Abstraction Level 1: Error Weights

Let d_{min} be the minimal distance between two codewords.

Track only the number of flips in each variable.

Concrete:

$z = x + y$

Abstract:

$z_{errw} = x_{errw} + y_{errw}$

Assertion: $z_{errw} < d_{min}$

Abstraction Level 2: Concrete Errors

$$z_{err} = x_{err} + y_{err}$$

0000 1000 = 0000 0100 + 0000 0100

0000 0110 = 0000 0010 + 0000 0100

Assertion: $w(z_{err}) < d_{min}$

Abstraction Level 3: Concrete Errors (refined)

Not all errors with d_{min} or more flips are undetectable.

If z_{err} is not a valid codeword we can still detect the error!

Assertion: $\neg isValid(z_{err})$

Abstraction Level 4: Concrete Values

Use the concrete values as well as errors.

This allows us to verify multiplication and division.

Assertion: $\neg isValid(z + z_{err}) \wedge (z_{err} \neq 0)$

Conclusion

Different abstractions can verify different programs.

Tracking more information, takes more time to verify.

Automatic checking with CPAchecker (a model checker).