Debugging the Undebuggable: Why Multi-Fault Programs Break Debugging and Repair Tools

Omar I. Al-Bataineh

ASE NIER 2025

Gran Sasso Science Institute (GSSI), Italy

1. Motivation

Most debugging and repair tools assume

“one fault at a time.”

1. Motivation

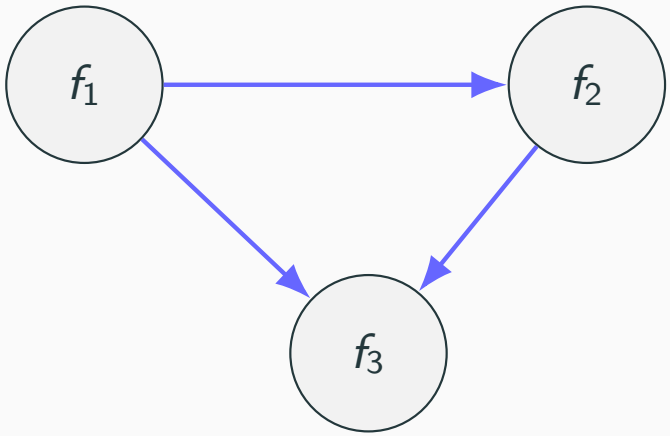
Most debugging and repair tools assume

“one fault at a time.”

In real programs, faults **interact**—masking, amplifying, or triggering each other.

$\rightarrow$ *Debugging and APR break down.*

2. Problem: Fault Interactions



Masking: one hides another

Synergy: two combine into new failure

Cascading: one activates others

3. Research Question

Why do debugging and repair tools fail on multi-fault programs?

Goal: Build a formal model that explains and predicts how faults interact and mislead tools.

4. Formal Model of Interactions

Each fault f_i affects program behavior b and visibility v.

$$\mathcal{I}(f_i, f_j, P, T) = \langle b_i, v_i, b_j, v_j \rangle \Rightarrow \text{interaction type}$$

Type	**Definition**
Independence	b_i, b_j unaffected
Masking	v_j hidden by f_i
Synergy	$b_i + b_j$ cause new failure
Cascading	f_i enables f_j

4. Formal Model of Interactions

Each fault f_i affects program behavior b and visibility v.

$$\mathcal{I}(f_i, f_j, P, T) = \langle b_i, v_i, b_j, v_j \rangle \Rightarrow \text{interaction type}$$

Type	**Definition**
Independence	b_i, b_j unaffected
Masking	v_j hidden by f_i
Synergy	$b_i + b_j$ cause new failure
Cascading	f_i enables f_j

A behavioral–visibility model for reasoning about fault interactions.

5. Key Insight

Multi-fault programs are "undebuggable"

because tools assume fault independence.

Our model exposes interaction patterns that
explain failures of fault localization and APR.

6. Implications

- Multi-Fault Localization (MFL): detect masking and synergy through visibility and behavioral shifts, leveraging mutation-based analysis.
- **Multi-Fault APR:** orchestrate repairs respecting interaction types.
- **Beyond single faults:** reason over fault networks, not isolated faults.

Bridging formal models and AI-driven debugging.

7. Takeaway

We model how faults interact.

We explain why tools fail.

We guide multi-fault debugging and repair.

Towards making the undebuggable—debuggable.