The Undecidability of Overfitting in Automated Program Repair

Omar I. Al-Bataineh

Gran Sasso Science Institute, L'Aquila, Italy

ICSE 2026 — NIER Track April 2026, Rio de Janeiro

The Persistent Problem: Overfitting in APR

Automated Program Repair (APR) generates patches automatically.

Yet one problem persists despite years of progress:

Overfitting

A patch **passes all tests** but **fails on unseen inputs**.

$$\text{Plausible} \neq \text{Correct}$$

Standard explanations:

- Weak or incomplete test suites
- Poor repair strategies
- Limited oracle quality

Common Assumption

"Better engineering can fix overfitting"

Our Claim

Overfitting is **not** an engineering problem.

It is a **fundamental** theoretical boundary.

The Key Question

Years of progress. Stronger tools. Better oracles.

Yet overfitting persists.

Is overfitting a solvable engineering challenge, or an inherent theoretical limit?

This paper argues: it is a consequence of undecidability.

Not an indictment of APR tools — a call for theory-aware evaluation.

Why Correctness Cannot Be Fully Verified

Patch validation requires deciding:

Five Undecidable Problems

1. **Termination:** Does $P'(x)$ halt for all x?
2. **Correctness:** $P' \models \varphi$ for all inputs?
3. **Equivalence:** $\forall x,\ P'(x) = P_{\text{ref}}(x)$?
4. **Formal Verification:** Prove $P' \models \varphi$?
5. **Synthesis:** Generate P' s.t. $P' \models \varphi$?

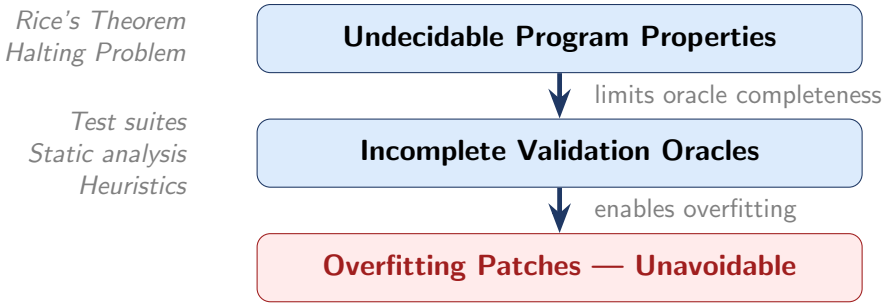
Consequence for APR

Each is **undecidable** in general (Halting Problem, Rice's Theorem).

What This Means

Any patch validation oracle — test suites, static analysis, heuristics, learned models — provides only **partial**, **approximate** guarantees.

The Undecidability–Overfitting Chain



This is not a new engineering gap. It is the same chain as in any system reasoning about arbitrary programs.

Formal Basis: Definitions and Key Result

Definitions

Plausible Patch P':

$$\text{passes}(P', T) = \text{true}$$

Correct Patch P':

$$\forall x \in \text{Dom}(\varphi),\ P'(x) \models \varphi(x)$$

Overfitting:

P' plausible but

$$\exists\, x \notin T : P'(x) \not\models \varphi(x)$$

Proposition

Determining whether $P' \models \varphi$ for all inputs is **undecidable**.

Corollary

Because correctness and equivalence are undecidable, no finite test-based oracle can establish full patch correctness.

Overfitting is an inherent risk.

Proof: Rice's Theorem — correctness is a non-trivial semantic property; all such properties are undecidable.

Illustrative Example: Two Faults, Two Barriers

Listing 1: Buggy Program P

```
1  int P(int input) {
2    // Fault A: wrong input check
3    if (input <= 0)
4      return 0; // should reject
5
6    // Fault B: Collatz-like loop
7    int n = input;
8    while (n > 1) {
9      if (n % 2 == 0) n = n/2;
10     else n = 3*n+1;
11   }
12   return 1;
13 }
```

Test suite: $T = \{P(4) \to 1,\ P(-5) \to 0\}$

Fault A — Undecidable Correctness

Intended behavior is implicit.
Rice's Theorem: no oracle can infer true intent from tests.

$\Rightarrow$ **Semantic overfitting**

Fault B — Undecidable Termination

Collatz-like loop.
2LS termination prover: returns `UNKNOWN` ($\perp$).

$\Rightarrow$ **Termination overfitting**

Patch passes T but fails on unseen inputs.

Revisiting APR Correctness Claims

Claim in APR Literature	Theoretical Caveat	Problem
"Generates correct patches"	Correctness is undecidable	Correctness
"Patch equivalent to developer fix"	Equivalence is undecidable	Equivalence
"Fixed X% of benchmarks"	Fix validity unverifiable	Correctness
"Verifiable with formal spec"	Only for decidable specs	Verification
"Labelled overfit/correct"	Labels rely on partial oracles	Correctness

Angelix, Prophet, CapGen, GenProg — all rely on partial oracles.
Empirically valuable, but theoretically bounded.

When Is APR Decidable? Overfitting Boundaries

Decidable Cases — No Overfitting

- **Finite input domain D:**
  Enumerate all $x \in D$, verify $P'(x)$
- **Bounded verification:**
  BMC, bounded symbolic execution
- **Loop-free fragments:**
  Decidable correctness here

Overfitting can be **eliminated** *in these cases.*

General Case — Unavoidable

- Unbounded input domains
- Arbitrary loops and recursion
- Implicit or partial specifications

No sound and complete oracle exists.

Analogy: Bounded Model Checking

In verification: exploit decidable fragments, be honest about limits.
APR should do the same.

Compounding the Problem: Multi-Fault Repair

Single fault: undecidable correctness.

Multiple interacting faults: **exponentially worse**.

Proposition: Multi-Fault Undecidability

Let P have interacting faults $F = \{f_1, \ldots, f_n\}$.

Determining whether P' simultaneously repairs all faults in F for all inputs is **undecidable**.

Why: Fault masking, cascading, and synergistic dependencies create non-monotonic repair spaces.

Interaction Effects

- **Masking:** f_i hides f_j
- **Cascading:** fixing f_i exposes f_j
- **Synergy:** neither alone violates correctness

Test-based oracles cannot distinguish intentional unmasking from genuine regression.

Rethinking Evaluation: From Binary to Risk Spectrum

Current evaluation:

$$\text{Patch} \xrightarrow{\text{test suite}} \textbf{Correct} \;/\; \textbf{Overfit}$$

Binary. Ignores unverified input space.

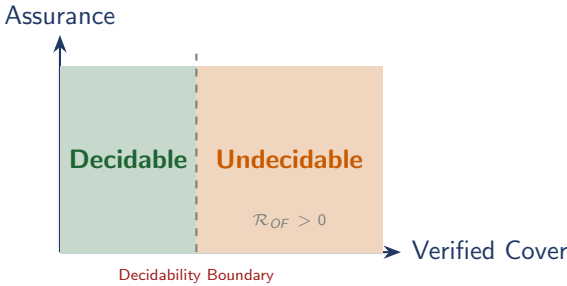
Proposed shift:

Quantified Overfitting Risk $\mathcal{R}_{OF}$

Proportion of the input domain **left unverified**.

$\mathcal{R}_{OF} = 0$: no overfitting risk
$\mathcal{R}_{OF} > 0$: theoretical uncertainty

When termination prover returns `UNKNOWN`*,*
$\mathcal{R}_{OF} > 0$ *necessarily.*



Implications for the APR Community

For Tool Builders

- Target decidable fragments explicitly
- Report $\mathcal{R}_{OF}$ alongside pass rates
- Use termination provers for decidable subclasses
- Hybrid: test-based search + formal analysis

For Evaluators

- Separate decidable from undecidable tasks
- Move beyond fix-rate as primary metric
- Design benchmarks with decidability

What This Paper Does NOT Say

- APR tools are useless
- Overfitting cannot be *reduced*
- Formal specs always exist

What It DOES Say

- Overfitting *cannot be eliminated* generally
- Community should be **explicit about limits**
- Progress means advancing the decidability boundary

Research Agenda: Three Directions

#1 Overfitting–Correctness Frontier

Design benchmarks separating decidable from undecidable repair tasks.

Measure how far tools advance correctness before hitting the undecidable boundary.

Move beyond pass-rate reporting.

#2 Multi-Fault Scenarios

Develop multi-fault benchmarks with interaction annotations.

Hybrid pipelines: bounded reasoning + automated synthesis + human guidance.

Reveal theoretical uncertainty to developers.

#3 Decidable Repair Resources

Curated datasets and taxonomies of decidable repair fragments.

Reference implementations with formal guarantees.

Community-wide shift toward theory-led APR.

Summary

Main contributions:

1. **Theoretical inevitability** of overfitting under standard APR assumptions
2. **Five undecidable problems** linked to limits of patch validation
3. **Decidable fragments** where APR provides formal guarantees
4. **Quantified Overfitting Risk** $\mathcal{R}_{OF}$ as a theory-aware metric

The Takeaway

We cannot **eliminate** overfitting.

We must **reason about it**.

Decidable fragments.
Risk-aware metrics.
Theory-honest evaluation.

Omar I. Al-Bataineh · Gran Sasso Science Institute · omar.albataineh@gssi.it

Questions?

The Undecidability of Overfitting
in Automated Program Repair

Omar I. Al-Bataineh
Gran Sasso Science Institute
omar.albataineh@gssi.it

ICSE 2026 NIER · April 2026, Rio de Janeiro