

Repair Evidence Resolution: Separating Prediction, Structural Resolution, and Transfer in Automated Patch Correctness Assessment

Mohammad Amir Khusru Akhtar 1

akakhtar.2024@gmail.com

Usha Martin University

Research Article

Keywords: automated program repair, patch correctness assessment, evidence resolution, overfitting patches, benchmark transfer, software testing

Posted Date: October 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11166234/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Repair Evidence Resolution: Separating Prediction, Structural Resolution, and Transfer in Automated Patch Correctness Assessment

Mohammad Amir Khusru Akhtar
Usha Martin University, Ranchi – 834001, India
akakhtar.2024@gmail.com

Abstract

Automated patch correctness assessment (APCA) is usually evaluated by predictive performance: accuracy, precision, recall, F1, ranking quality, or related measures. These metrics answer whether an assessor tends to make the right decision, but they do not determine whether the evidence visible to the assessor is structurally capable of separating repairs that require opposite correctness decisions. We introduce *Repair Evidence Resolution* (RER), a claim-relative framework for studying that prior question. RER models assessor-visible evidence as a partition of a declared repair population and asks whether every evidence cell is homogeneous with respect to a repair claim. It yields repair-claim resolution (RCR), concrete claim-conflict witnesses (CCWs), resolution deficit (RD), claim-resolution profiles, and claim-boundary completeness (CBC) for transferring resolution conclusions from evaluated repair subsets to larger target populations. We prove basic separation results and explicitly reduce exact-identification special cases to established teaching/certificate-complexity machinery. We then implement a deterministic audit pipeline and evaluate a publisher-verified ASE 2020 APCA artifact. Across 1,762 generator-aware records, one released evidence channel produces nine mixed correctness cells covering 43 patches (RD=2.44%), whereas a second channel and the combined evidence are exact-cell claim-resolving on the observed corpus. We also derive the exact probability that restricted evaluation subsets falsely appear resolved: for a sample of 100 records it is 91.29%. The results motivate a three-layer evaluation view: *prediction*, *resolution*, and *transfer*.

Keywords: automated program repair; patch correctness assessment; evidence resolution; overfitting patches; benchmark transfer; software testing

1 Introduction

Automated program repair (APR) has evolved from generate-and-validate search and redundancy-based mutation to template-based, synthesis-based, representation-learning, and large-language-model approaches (Dikici & Bilgin, 2025; Gazzola et al., 2019; Le Goues et al., 2012; Li et al., 2020; Liu et al., 2019; Lutellier et al., 2020; Monperrus, 2018). Yet a persistent difficulty is that a patch can satisfy the available test suite while still being semantically wrong, a phenomenon usually described as test-suite overfitting (Le et al., 2018; Long & Rinard, 2016a; Martinez et al., 2017; Qi et al., 2015; Smith et al., 2015). Automated patch correctness assessment (APCA) responds to this problem by ranking or classifying plausible patches using static, dynamic, learned, semantic, and hybrid signals (Fuster-Pena et al., 2026; Ghanbari & Marcus, 2022; Le-Cong et al., 2023; Martinez et al., 2024; Tian et al., 2020; Wang et al., 2020; Xiong et al., 2017).

The dominant evaluation question in APCA is predictive: how accurately does an assessor label or rank patches? That question is necessary, but not sufficient for understanding the evidential basis of an assessment. Suppose two candidate repairs expose exactly the same

assessor-visible representation while independent ground truth assigns opposite correctness claims. No deterministic downstream classifier restricted to that representation can distinguish them. Conversely, a representation may be structurally resolving even when a particular learned classifier performs poorly. Prediction and evidence resolution are therefore related but logically distinct.

This paper develops *Repair Evidence Resolution* (RER), an APR-specific operational framework for this distinction. The intended contribution is not the generic mathematics of partitions, observational equivalence, version spaces, teaching dimension, certificate complexity, uncertainty, or test completeness; those areas are well established (Barr et al., 2015; Geifman & El-Yaniv, 2017; Goldman & Kearns, 1995; Gulwani, 2011; Jia & Harman, 2011; Lau et al., 2003; Offutt, 1992; Vovk et al., 2005). Instead, RER organizes these ideas around a concrete APCA audit question: *does the evidence actually used by an assessor resolve the repair claim it emits over the declared repair population, and does that conclusion transfer beyond the evaluated subset?*

Figure 1 summarizes the resulting three-layer view. Prediction measures empirical decision quality. Resolution asks whether the visible evidence merges repairs lying on opposite sides of the declared claim. Transfer asks whether apparent resolution on an evaluated surrogate persists in the larger repair population of interest.



Figure 1: Three distinct evaluation layers for automated patch correctness assessment.

The paper makes four contributions. First, it formalizes repair-claim resolution (RCR), claim-conflict witnesses (CCWs), resolution deficit (RD), claim-resolution profiles (CRP), and claim-boundary completeness (CBC). Second, it proves elementary but useful separation statements while explicitly identifying special cases that reduce to classical teaching sets, certificate complexity, and conformance-style completeness. Third, it provides deterministic software for exact finite audits, structured calibration experiments, and real-artifact analysis. Fourth, it reports two empirical findings on a publisher-verified ASE 2020 APCA artifact: evidence-channel enrichment eliminates all observed exact mixed cells, and restricted evaluation subsets can appear resolved with high exact probability even when the parent observed corpus is not.

2 Background and Related Work

2.1 Automated program repair and overfitting

APR research spans search-based repair, redundancy assumptions, mutation and templates, semantic synthesis, code transformation learning, neural translation, and recent foundation-model-based systems (Dikici & Bilgin, 2025; Jiang et al., 2018; Le Goues et al., 2012; Li et al., 2020; Liu et al., 2019; Long & Rinard, 2016b; Lutellier et al., 2020; Nguyen et al., 2013; Weimer et al., 2009; Wen et al., 2018; Xia & Zhang, 2022; Xia et al., 2023; Xuan et al., 2017; Zhu et al., 2021). Benchmarks such as Defects4J have enabled controlled comparison but also make ground-truth, labeling, fault-localization, and benchmark-representativeness assumptions consequential (Just, Jalali, & Ernst, 2014; Just, Jalali, Inozemtseva, et al., 2014; Kochhar et al., 2016; Pearson et al., 2017; Ye et al., 2014).

Test-suite overfitting is now a standard threat in generate-and-validate APR. Empirical studies showed that plausible patches can preserve supplied tests while failing additional semantic obligations, and that developer/reference patches are useful but not necessarily unique semantic oracles (Le et al., 2018; Long & Rinard, 2016a; Martinez et al., 2017; Qi et al., 2015; Smith et al., 2015). The oracle problem in testing reinforces the broader point that observed executions and asserted outcomes can under-specify intended behavior (Barr et al., 2015; Fraser & Arcuri, 2011; Pacheco et al., 2007; Staats et al., 2012).

2.2 Patch correctness assessment

APCA methods include test-generation approaches, semantic similarity, dynamic behavior, invariant reasoning, static code features, learned representations, ranking models, and hybrid combinations (Csuvik et al., 2020; Fuster-Pena et al., 2026; Ghanbari & Marcus, 2022; Le-Cong et al., 2023; Martinez et al., 2024; Tian et al., 2020; Wang et al., 2020; Xiong et al., 2017). Wang et al. compared multiple static and dynamic APCA techniques on 902 patches from 21 APR tools and reported substantial complementarity across assessors (Wang et al., 2020). Shibboleth combines syntactic similarity, semantic similarity, and branch-coverage change (Ghanbari & Marcus, 2022). Invalidator combines semantic invariants with a learned syntactic representation (Le-Cong et al., 2023). xTestCluster clusters patches by behavior on generated tests (Martinez et al., 2024). Recent reasoning-model approaches continue to push static APCA toward foundation models (Fuster-Pena et al., 2026).

RER does not replace any of these assessors. It provides an orthogonal audit: for the representation actually supplied to an assessor, which correctness distinctions are structurally visible, which remain merged, and how sensitive is that conclusion to the evaluated patch population?

2.3 Equivalence, learning, completeness, and uncertainty

The underlying mathematical ancestors are broad. Version spaces and programming-by-example characterize candidate programs consistent with examples (Gulwani, 2011; Lau et al., 2003; Mitchell, 1982). Teaching dimension studies how many examples distinguish a target concept within a concept class (Goldman & Kearns, 1995; Zilles et al., 2011). Certificate complexity studies partial information sufficient to force a Boolean property (Buhrman & de Wolf, 2002). Observational equivalence and abstraction formalize when programs or states cannot be distinguished by an observation language (Cousot & Cousot, 1977; Milner, 1989). Mutation testing, coupling, and subsumption ask whether generated mutants provide useful surrogates for real faults and test obligations (Ammann et al., 2014; DeMillo et al., 1978; Jia & Harman, 2011; Offutt, 1992). Selective prediction and conformal prediction distinguish uncertainty/risk control from ordinary classification (Angelopoulos & Bates, 2023; Chow, 1970; Geifman & El-Yaniv, 2017; Vovk et al., 2005).

Recent software-engineering methodology also cautions against treating benchmark scores as self-interpreting evidence. Defect-prediction studies have shown that validation design, parameter tuning, researcher degrees of freedom, data leakage, class imbalance, and dataset shift can materially change reported performance (Ghotra et al., 2015; Hall et al., 2012; Herbold et al., 2018; Kapoor & Narayanan, 2023; Lessmann et al., 2008; Menzies & Shepperd, 2019; Saito & Rehmsmeier, 2015; Shepperd et al., 2014; Tantithamthavorn et al., 2017, 2019). Reproducibility work likewise emphasizes executable artifacts, transparent experimental protocols, and separation between code availability and successful independent reproduction (Collberg & Proebsting, 2016; Kitchenham et al., 2002; Pineau et al., 2021; Runeson & Höst, 2009; Wohlin et al., 2012). RER follows that discipline by freezing machine-readable results, checking primary-archive integrity, and stating explicitly which claims were reproduced from released outputs versus which remain future replications.

These literatures prevent overclaiming. Exact target identification under coordinate observations reduces to teaching-set ideas; minimum evidence for a binary property reduces to certificate-like structure; and test-only/full-specification special cases can reduce to relative test completeness. The APR-specific object retained here is the joint operationalization of *assessor-visible heterogeneous evidence*, a *declared repair claim*, and an *evaluated surrogate repair population*, with executable conflict witnesses and transfer diagnostics.

RER also connects naturally to robustness and distribution-shift research. Adversarial robustness asks how predictions change under controlled perturbations, while out-of-distribution and dataset-shift studies ask whether predictive behavior persists when the input distribution changes (Carlini & Wagner, 2017; Goodfellow et al., 2015; Madry et al., 2018; Northcutt et al., 2021; Ovadia et al., 2019; Quinonero-Candela et al., 2009; Recht et al., 2019; Szegedy et al., 2014; Torralba & Efros, 2011). Those questions become important for continuous APCA representations, where exact equality may be too fine to expose meaningful effective collisions. We therefore keep exact RER separate from future tolerance- or margin-based extensions rather than silently importing an arbitrary distance threshold.

3 Repair Evidence Resolution

Let P_0 be a known defective program, $\mathcal{R}(P_0)$ a declared admissible repair population, $O_E : \mathcal{R} \rightarrow \Omega_E$ the evidence representation visible to an assessor, and $c : \mathcal{R} \rightarrow \mathcal{C}$ a declared repair claim.

Definition 1 (Evidence equivalence). *Two repairs are evidence-equivalent under E when*

$$r_i \sim_E r_j \iff O_E(r_i) = O_E(r_j).$$

The corresponding evidence cell of r is $[r]_E = \{r' \in \mathcal{R} : r' \sim_E r\}$.

Definition 2 (Repair-Claim Resolution). *Evidence E resolves claim c over $\mathcal{R}$ when every evidence cell is claim-homogeneous:*

$$\text{RCR}(\mathcal{R}, E, c) = 1 \iff \forall r_i, r_j \in \mathcal{R}, \quad r_i \sim_E r_j \Rightarrow c(r_i) = c(r_j).$$

Definition 3 (Claim-Conflict Witness). *A claim-conflict witness (CCW) is a pair (r_+, r_-) such that*

$$O_E(r_+) = O_E(r_-), \quad c(r_+) \neq c(r_-).$$

Any CCW proves $\text{RCR}(\mathcal{R}, E, c) = 0$.

Definition 4 (Resolution Deficit). *For a finite repair population,*

$$\text{RD}(E, c) = \frac{|\{r \in \mathcal{R} : [r]_E \text{ is claim-mixed}\}|}{|\mathcal{R}|}.$$

A cell-count analogue is

$$\text{RD}_{\text{cell}}(E, c) = \frac{|\{C \in \mathcal{R}/\sim_E : |c(C)| > 1\}|}{|\mathcal{R}/\sim_E|}.$$

Proposition 1 (Confidence does not imply structural resolution). *For every confidence threshold $\alpha < 1$, there exists a finite repair population and representation for which a deterministic classifier attains confidence greater than α and accuracy arbitrarily close to one while RCR fails.*

Proof. Place $N - 1$ positive repairs and one negative repair in the same evidence cell, predict positive, and assign confidence $1 - \epsilon > \alpha$. Accuracy is $1 - 1/N \rightarrow 1$, but the cell remains mixed for every finite $N \geq 2$. $\square$

Proposition 2 (Cell impurity lower bound). *For a finite evidence cell C_z with $n_0(z)$ negative and $n_1(z)$ positive repairs, every deterministic classifier based only on z makes at least $\min(n_0(z), n_1(z))$ errors under uniform counting on the cell.*

The proposition is elementary classification theory rather than a standalone novelty claim. Its role is to connect structural mixed cells to a lower bound that cannot be repaired by changing only the downstream decision rule.

3.1 Claim-resolution profiles

Different repair claims can induce different requirements on the same evidence. For claims $c_1, \dots, c_k$, define

$$\text{CRP}(E, \mathcal{R}) = (\text{RCR}_{c_1}, \dots, \text{RCR}_{c_k}).$$

A natural hierarchy includes passing supplied tests, repairing the reported failure, preserving declared regressions, satisfying a semantic specification, and behavioral equivalence to an accepted contract. RER is therefore claim-relative rather than a single scalar notion of evidence adequacy.

A second connection is to active information acquisition. Active learning, membership queries, and oracle-guided synthesis study which additional observations most efficiently shrink a candidate set (Alur et al., 2013; Angluin, 1988; Hanneke, 2014; Jha et al., 2010; Peleg et al., 2018; Settles, 2009). RER differs in emphasis: it first diagnoses whether the currently visible evidence already resolves a declared repair claim, and only then asks which evidence channel should be added. This distinction matters because evidence enrichment can be justified by the specific claim-conflict cells it splits rather than only by aggregate predictive gain.

4 Surrogate Transfer and Claim-Boundary Completeness

Empirical APCA studies evaluate a finite patch corpus G rather than an exhaustive semantic repair universe $\mathcal{R}$. A sample can be homogeneous even when a larger evidence cell contains repairs with the opposite claim.

Definition 5 (Claim-Boundary Completeness). *Let $G \subseteq \mathcal{R}$. G is claim-boundary complete (CBC) for $(E, c, \mathcal{R})$ when*

$$\forall C \in \mathcal{R}/\sim_E, \quad |c(C)| > 1 \Rightarrow |c(C \cap G)| > 1.$$

Theorem 1 (Exact transfer under CBC). *If G satisfies CBC, then*

$$\text{RCR}(G, E, c) = 1 \iff \text{RCR}(\mathcal{R}, E, c) = 1.$$

Proof. The reverse implication follows by restriction. For the forward implication, suppose the full population were unresolved. Then some evidence cell C would contain at least two claim values. CBC requires $C \cap G$ also to contain at least two claim values, contradicting $\text{RCR}(G, E, c) = 1$. $\square$

CBC is deliberately an exact, worst-case transfer condition. It is not a replacement for PAC-style statistical generalization or generic benchmark representativeness. Instead it isolates the precise boundary-preservation requirement needed for the specific conclusion “the evaluated evidence is claim-resolving.”

5 Algorithms

Algorithm 1 gives the exact finite RER audit. The key implementation discipline is to construct cells without using correctness labels and attach labels only after the evidence partition has been formed.

Algorithm 1: Exact RER audit for a finite repair population

Input: Repairs R , evidence map O_E , claim labels c
Output: RCR, RD, mixed cells, CCWs
 Create an empty map M from evidence signatures to repair lists;
foreach $r \in R$ **do**
 $z \leftarrow O_E(r)$;
 append r to $M[z]$;
 $mixed \leftarrow \emptyset$;
foreach cell C in M **do**
 if $|\{c(r) : r \in C\}| > 1$ **then**
 add C to $mixed$;
 $RCR \leftarrow (|mixed| = 0)$;
 $RD \leftarrow (\sum_{C \in mixed} |C|) / |R|$;
 Emit all opposite-claim pairs within each mixed cell as CCWs;
return $RCR, RD, mixed, CCWs$;

Algorithm 2 computes the exact false-resolution probability for fixed-size sampling from a finite observed parent population. For each mixed parent cell with a_i correct and b_i incorrect members, the generating polynomial counts sample choices that take members from at most one side of the claim boundary.

Algorithm 2: Exact false-resolution probability under fixed-size sampling

Input: Parent population R , mixed-cell counts (a_i, b_i) , sample size n
Output: $\Pr(RCR(G) = 1 \mid RCR(R) = 0)$
 Let u be the number of records outside all mixed parent cells;
 Initialize polynomial $P(x) \leftarrow 1$;
foreach mixed cell (a_i, b_i) **do**
 $Q_i(x) \leftarrow 1 + \sum_{k \geq 1} [\binom{a_i}{k} + \binom{b_i}{k}] x^k$;
 $P(x) \leftarrow P(x) Q_i(x)$;
 $F \leftarrow \sum_k [x^k] P(x) \binom{u}{n-k}$;
return $F / \binom{|R|}{n}$;

6 Software and Reproducibility

Following the mathematical development of RER, we implemented the definitions and exact finite procedures as deterministic software. The implementation enumerates finite evidence cells, emits CCWs, computes RCR and RD, evaluates claim-resolution profiles and CBC, and contains brute-force calibration tests and reproducibility scripts. We then applied the implementation to the released empirical artifact and used the resulting machine-readable outputs for the analyses reported below. Data, source code, reproducibility scripts, and frozen result artifacts are available at <https://github.com/Arithmetic-Power-Geometry/Repair-Evidence-Theory>. The ASE 2020 primary archive used here was verified by MD5 before analysis.

7 Empirical Study

7.1 Artifact and research questions

We use the public artifact accompanying the large-scale APCA study of Wang et al. (Wang et al., 2020). The released PATCH-SIM/E-PATCH-SIM archive contains generator-specific patch directories with a human-readable `result` file. The first line stores the released correctness label; the next two lines contain the two released numeric evidence vectors. The archive analyzed in this study matched the publisher-provided MD5 checksum `e3b68e17dd4be6a2bafadd6c6b30b52e`.

We ask two questions. **RQ1:** Does either released evidence channel, or their combination, contain exact opposite-correctness cells on the observed patch corpus? **RQ2:** If the parent observed corpus is unresolved under an evidence channel, how often can a smaller evaluation surrogate falsely appear resolved?

7.2 RQ1: evidence-channel resolution

Table 1 reports the exact-cell audit. EvoSuite contributes 879 records and Randoop 883, for 1,762 generator-aware records. Vector 1 alone produces nine mixed cells covering 43 records in the combined corpus, corresponding to RD=2.44%. Vector 2 alone and the full pair contain no exact cross-label cells on this observed corpus.

Table 1: Exact repair-evidence resolution on the released ASE 2020 artifact.

Generator	Representation	Records	Cells	Mixed	Mixed members	RD
EvoSuite	Full pair	879	517	0	0	0
EvoSuite	Vector 1	879	451	3	12	0.0137
EvoSuite	Vector 2	879	331	0	0	0
Randoop	Full pair	883	644	0	0	0
Randoop	Vector 1	883	535	6	31	0.0351
Randoop	Vector 2	883	480	0	0	0
Combined	Full pair	1762	1161	0	0	0
Combined	Vector 1	1762	986	9	43	0.0244
Combined	Vector 2	1762	811	0	0	0

Figure 2 visualizes the same result. The important observation is not that “more features improve accuracy.” No classifier is needed for the statement. The additional evidence channel changes the evidence partition itself: concrete mixed cells present under Vector 1 disappear under Vector 2 and the full pair.

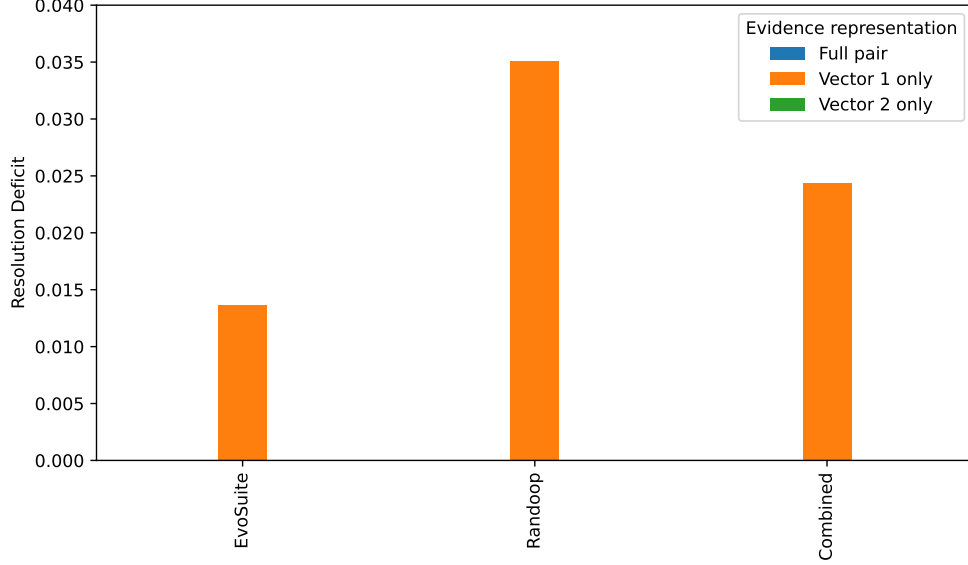


Figure 2: Resolution deficit under the released evidence channels. Vector 1 alone contains mixed correctness cells; Vector 2 and the full pair do not under exact equality on this observed corpus.

The null result for the full pair is equally important. It falsifies the naive expectation that a detailed continuous representation will necessarily contain many exact cross-label duplicates. Exact RER can therefore be highly representation-dependent, and continuous representations may yield singleton or pure cells even when downstream prediction remains imperfect. This motivates future effective-representation and robustness analyses but does not justify redefining near-neighbors as exact CCWs after observing the result.

7.3 RQ2: exact false resolution under sampling

The nine mixed Vector-1 cells have correct/incorrect membership counts (1, 2), (5, 1), (1, 2), (1, 1), (1, 1), (1, 15), (2, 1), (2, 1), and (1, 4). Forty-three records lie in these cells; the remaining 1,719 records are outside them. Algorithm 2 therefore gives an exact hypergeometric/generating-function calculation for fixed-size samples.

Table 2 and Figure 3 show the resulting probabilities. A uniformly drawn sample of 100 records appears completely claim-resolving under Vector 1 with probability 0.9129 even though the 1,762-record parent corpus is unresolved. At $n = 400$ the probability remains 0.3397. It falls below 0.025 only by $n = 800$.

Table 2: Exact probability that a fixed-size evaluation surrogate is claim-resolving under Vector 1 although the full observed parent corpus is not.

Sample size n	Exact false-resolution probability
25	0.993740
50	0.975810
100	0.912890
200	0.725448
400	0.339655
800	0.024354
1200	0.000209
1500	0.000000174

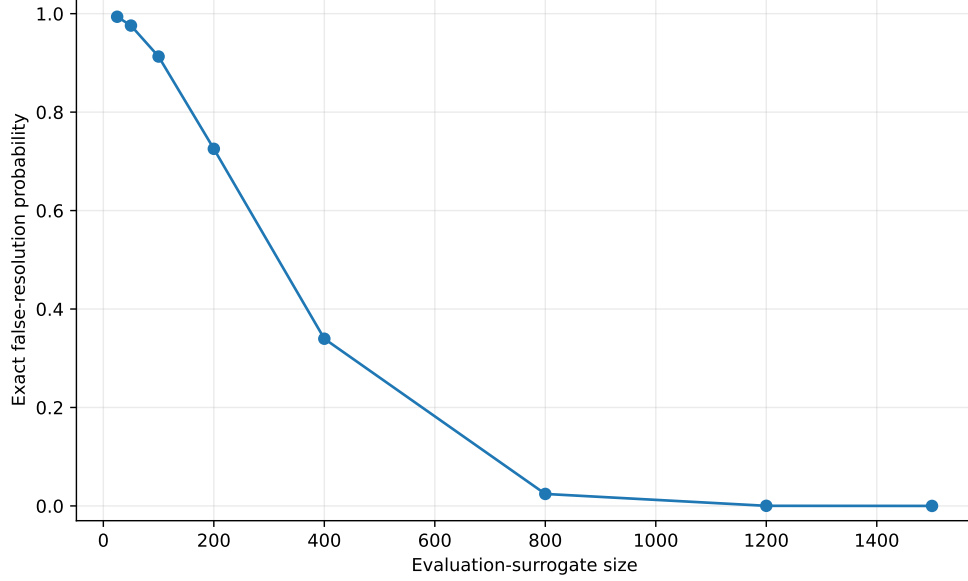


Figure 3: Exact false-resolution probability versus evaluation-surrogate size for Vector 1.

The Vector-2 and full-pair negative controls have no mixed parent cells, so parent-relative false resolution is zero by construction. The result is therefore not a generic sampling artifact: it appears exactly where the parent evidence representation crosses the correctness boundary.

8 Discussion

8.1 What RER adds to APCA evaluation

Prediction metrics and RER metrics answer different questions. Accuracy, F1, AUC, ranking metrics, calibration, and selective risk quantify the behavior of a predictor (Fawcett, 2006; Geifman & El-Yaniv, 2017; Guo et al., 2017; Niculescu-Mizil & Caruana, 2005). RER inspects whether the predictor’s evidence representation merges opposite claims before the downstream model is considered. In the finite deterministic setting, a mixed evidence cell is an explicit certificate that no deterministic function of that representation alone can perfectly classify every member of the cell.

This distinction also clarifies evidence enrichment. APCA work has long observed that static and dynamic techniques can be complementary (Ghanbari & Marcus, 2022; Le-Cong et al., 2023; Wang et al., 2020). RER offers a structural interpretation of complementarity: a new evidence channel can refine an evidence partition and split cells that cross the correctness boundary. Table 1 provides a direct example in the released artifact.

8.2 What RER does not claim

RER does not claim that passing tests can establish semantic correctness, that observational equivalence is new, that feature collisions are new, or that subset representativeness is a new general problem. It also does not claim that the 1,762-record corpus is the complete semantic repair universe. RCR=1 on an observed finite population means only that no exact cross-label collision was found under the declared representation and claim on that population. CBC makes the transfer assumption explicit rather than silently equating a benchmark with the universe it is intended to represent.

8.3 Implications for benchmark design

Benchmark construction in software engineering has repeatedly raised questions of representativeness, labeling quality, hidden duplicates, project imbalance, and transfer (Just, Jalali, & Ernst, 2014; Kochhar et al., 2016; Pearson et al., 2017; Wang et al., 2020). RER adds a claim-specific structural check: does the evaluated patch set expose the claim-boundary crossings that exist within the evidence cells of the population about which the study intends to conclude? The exact false-resolution curve in Figure 3 demonstrates that even random sampling can hide all observed boundary crossings at surprisingly large sample sizes when mixed cells are sparse.

9 Threats to Validity

Construct validity. Exact equality is a strict notion of evidence indistinguishability. For high-dimensional continuous representations, exact duplicates may be rare. We therefore avoid interpreting an absence of exact CCWs as proof of semantic sufficiency. Near-collision or robustness analyses require separately justified tolerances tied to model behavior, precision, or perturbation semantics.

Internal validity. The empirical parser uses only the released human-readable `result` files. Labels are attached after evidence cells are constructed. The primary ZIP checksum was verified before analysis. We observed a provenance ambiguity between stored outputs and bundled classifier bytecode; consequently, this paper treats the stored released results as the artifact under audit and does not claim historical classifier regeneration.

External validity. The current quantitative results concern one published APCA artifact and two test generators. The theory and software are general, but empirical prevalence is not. Independent replication on Shibboleth, Invalidator, xTestCluster, and newer reasoning-based assessors is future work (Fuster-Pena et al., 2026; Ghanbari & Marcus, 2022; Le-Cong et al., 2023; Martinez et al., 2024).

Conclusion validity. The CBC experiment is exact relative to the finite observed parent population and uniform fixed-size sampling; it is not a distribution-free statement about all future repairs. Its purpose is to demonstrate the existence and magnitude of false resolution under a transparent finite model.

10 Conclusion

Patch correctness assessment has largely been evaluated as a prediction problem. This paper introduces a complementary structural question: before asking whether a model predicts correctness well, does the evidence supplied to that model actually separate repairs that demand opposite correctness decisions? Repair Evidence Resolution formalizes this question through repair-claim resolution, claim-conflict witnesses, resolution deficit, claim-resolution profiles, and claim-boundary completeness. The framework deliberately imports rather than renames established ideas where reductions exist, including teaching/certificate-complexity and test-completeness special cases.

A deterministic implementation and a publisher-verified ASE 2020 artifact show why the distinction matters. One released evidence channel leaves nine mixed correctness cells spanning 43 patches, while a second channel and the combined representation remove all observed exact conflicts. More strikingly, a 100-record random evaluation subset has an exact 91.29% probability of appearing resolved even though the 1,762-record parent population is not. These findings support a three-layer view of APCA evaluation: *prediction* asks how often the assessor is right, *resolution* asks whether its evidence can express the required distinction, and *transfer* asks whether that structural conclusion survives beyond the evaluated surrogate. Future work should

test these layers across independent APCA systems and develop principled robustness notions for continuous learned representations.

Data and Code Availability

All software, reproducibility scripts, workflow definitions, and frozen result artifacts are available at <https://github.com/Arithmetic-Power-Geometry/Repair-Evidence-Theory>. The empirical study uses the public ASE 2020 APCA replication artifact described by Wang et al. (2020); the software verifies the publisher checksum before analysis.

Declarations

Funding. No external funding was received for this study.

Competing interests. The author declares no competing interests.

Ethics. This theoretical, methodological, and computational study did not involve human participants, personal data, animals, or clinical interventions.

Author contributions. Mohammad Amir Khusru Akhtar: conceptualization, mathematical foundation, methodology, formal analysis, software, investigation, validation, data curation, visualization, writing—original draft, writing—review and editing.

References

- Alur, R., Bodik, R., Juniwal, G., Martin, M. M. K., Raghothaman, M., Seshia, S. A., Singh, R., Solar-Lezama, A., Torlak, E., & Udupa, A. (2013). Syntax-guided synthesis. *FMCAD*, 1–8.
- Ammann, P., Delamaro, M. E., & Offutt, J. (2014). Establishing theoretical minimal sets of mutants. *IEEE International Conference on Software Testing, Verification and Validation*, 21–30. <https://doi.org/10.1109/ICST.2014.13>
- Angelopoulos, A. N., & Bates, S. (2023). Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4), 494–591. <https://doi.org/10.1561/2200000101>
- Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2, 319–342. <https://doi.org/10.1007/BF00116828>
- Barr, E. T., Harman, M., McMin, P., Shahbaz, M., & Yoo, S. (2015). The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5), 507–525. <https://doi.org/10.1109/TSE.2014.2372785>
- Buhrman, H., & de Wolf, R. (2002). Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1), 21–43. [https://doi.org/10.1016/S0304-3975\(01\)00144-X](https://doi.org/10.1016/S0304-3975(01)00144-X)
- Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*, 39–57. <https://doi.org/10.1109/SP.2017.49>
- Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1), 41–46. <https://doi.org/10.1109/TIT.1970.1054406>
- Collberg, C., & Proebsting, T. A. (2016). Repeatability in computer systems research. *Communications of the ACM*, 59(3), 62–69. <https://doi.org/10.1145/2812803>
- Cousot, P., & Cousot, R. (1977). Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. *4th ACM Symposium on Principles of Programming Languages*, 238–252. <https://doi.org/10.1145/512950.512973>
- Csuvik, V., Horváth, D., Horváth, F., & Vidács, L. (2020). Utilizing source code embeddings to identify correct patches. *International Workshop on Intelligent Bug Fixing*, 18–25. <https://doi.org/10.1109/IBF50092.2020.9034714>

- DeMillo, R. A., Lipton, R. J., & Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4), 34–41. <https://doi.org/10.1109/C-M.1978.218136>
- Dikici, S., & Bilgin, T. T. (2025). Advancements in automated program repair: A comprehensive review. *Knowledge and Information Systems*, 67, 4737–4783. <https://doi.org/10.1007/s10115-025-02383-9>
- Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8), 861–874. <https://doi.org/10.1016/j.patrec.2005.10.010>
- Fraser, G., & Arcuri, A. (2011). Evosuite: Automatic test suite generation for object-oriented software. *19th ACM SIGSOFT Symposium and 13th European Conference on Foundations of Software Engineering*, 416–419. <https://doi.org/10.1145/2025113.2025179>
- Fuster-Pena, M., de-Fitero-Dominguez, D., Garcia Cabot, A., & Garcia Lopez, E. (2026). Repaca: Leveraging reasoning large language models for static automated patch correctness assessment. *Neurocomputing*. <https://doi.org/10.1016/j.neucom.2026.134583>
- Gazzola, L., Micucci, D., & Mariani, L. (2019). Automatic software repair: A survey. *IEEE Transactions on Software Engineering*, 45(1), 34–67. <https://doi.org/10.1109/TSE.2017.2755013>
- Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. *Advances in Neural Information Processing Systems*, 4878–4887.
- Ghanbari, A., & Marcus, A. (2022). Patch correctness assessment in automated program repair based on the impact of patches on production and test code. *31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. <https://doi.org/10.1145/3533767.3534368>
- Ghotra, B., McIntosh, S., & Hassan, A. E. (2015). Revisiting the impact of classification techniques on the performance of defect prediction models. *37th International Conference on Software Engineering*, 1, 789–800. <https://doi.org/10.1109/ICSE.2015.91>
- Goldman, S. A., & Kearns, M. J. (1995). On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1), 20–31. <https://doi.org/10.1006/jcss.1995.1003>
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations*.
- Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. *38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 317–330. <https://doi.org/10.1145/1926385.1926423>
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *34th International Conference on Machine Learning*, 1321–1330.
- Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, 38(6), 1276–1304. <https://doi.org/10.1109/TSE.2011.103>
- Hanneke, S. (2014). Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3), 131–309. <https://doi.org/10.1561/22000000037>
- Herbold, S., Trautsch, A., & Grabowski, J. (2018). A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, 44(9), 811–833. <https://doi.org/10.1109/TSE.2017.2724538>
- Jha, S., Gulwani, S., Seshia, S. A., & Tiwari, A. (2010). Oracle-guided component-based program synthesis. *32nd International Conference on Software Engineering*, 215–224. <https://doi.org/10.1145/1806799.1806833>
- Jia, Y., & Harman, M. (2011). An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5), 649–678. <https://doi.org/10.1109/TSE.2010.62>

- Jiang, J., Xiong, Y., Zhang, H., Gao, Q., & Chen, X. (2018). Shaping program repair space with existing patches and similar code. *27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 298–309. <https://doi.org/10.1145/3213846.3213871>
- Just, R., Jalali, D., & Ernst, M. D. (2014). Defects4j: A database of existing faults to enable controlled testing studies for java programs. *International Symposium on Software Testing and Analysis*, 437–440. <https://doi.org/10.1145/2610384.2628055>
- Just, R., Jalali, D., Inozemtseva, L., Ernst, M. D., Holmes, R., & Fraser, G. (2014). Are mutants a valid substitute for real faults in software testing? *22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 654–665. <https://doi.org/10.1145/2635868.2635929>
- Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 100804. <https://doi.org/10.1016/j.patter.2023.100804>
- Kitchenham, B., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8), 721–734. <https://doi.org/10.1109/TSE.2002.1027796>
- Kochhar, P. S., Xia, X., Lo, D., & Li, S. (2016). Practitioners’ expectations on automated fault localization. *25th International Symposium on Software Testing and Analysis*, 165–176. <https://doi.org/10.1145/2931037.2931051>
- Lau, T., Wolfman, S. A., Domingos, P., & Weld, D. S. (2003). Programming by demonstration using version space algebra. *Machine Learning*, 53, 111–156. <https://doi.org/10.1023/A:1025671410623>
- Le, X.-B. D., Chu, D.-H., Lo, D., Le Goues, C., & Visser, W. (2018). On the efficiency of test suite based program repair: A systematic assessment of 16 automated repair systems for java programs. *Proceedings of the 40th International Conference on Software Engineering*, 615–626. <https://doi.org/10.1145/3180155.3180248>
- Le Goues, C., Nguyen, T., Forrest, S., & Weimer, W. (2012). Genprog: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1), 54–72. <https://doi.org/10.1109/TSE.2011.104>
- Le-Cong, T., Luong, D.-M., Le, X.-B. D., Lo, D., Tran, N.-H., Bui, Q.-H., & Huynh, Q.-T. (2023). Invalidator: Automated patch correctness assessment via semantic and syntactic reasoning. *IEEE Transactions on Software Engineering*, 49(6), 3411–3429. <https://doi.org/10.1109/TSE.2023.3255177>
- Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4), 485–496. <https://doi.org/10.1109/TSE.2008.35>
- Li, Y., Wang, S., & Nguyen, T. N. (2020). Dlfix: Context-based code transformation learning for automated program repair. *42nd International Conference on Software Engineering*, 602–614. <https://doi.org/10.1145/3377811.3380345>
- Liu, K., Koyuncu, A., Kim, D., & Bissyandé, T. F. (2019). Tbar: Revisiting template-based automated program repair. *28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 43–54. <https://doi.org/10.1145/3293882.3330577>
- Long, F., & Rinard, M. (2016a). An analysis of the search spaces for generate and validate patch generation systems. *38th International Conference on Software Engineering*, 702–713. <https://doi.org/10.1145/2884781.2884872>
- Long, F., & Rinard, M. (2016b). Automatic patch generation by learning correct code. *43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 298–312. <https://doi.org/10.1145/2837614.2837617>
- Lutellier, T., Pham, H. V., Pang, L., Li, Y., Wei, M., & Tan, L. (2020). Coconut: Combining context-aware neural translation models using ensemble for program repair. *29th ACM*

- SIGSOFT International Symposium on Software Testing and Analysis*, 101–114. <https://doi.org/10.1145/3395363.3397369>
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*.
- Martinez, M., Durieux, T., Sommerard, R., Xuan, J., & Monperrus, M. (2017). Automatic repair of real bugs in java: A large-scale experiment on the defects4j dataset. *Empirical Software Engineering*, 22, 1936–1964. <https://doi.org/10.1007/s10664-016-9470-4>
- Martinez, M., Kechagia, M., Perera, A., Petke, J., Sarro, F., & Aleti, A. (2024). Test-based patch clustering for automatically-generated patches assessment. *Empirical Software Engineering*, 29, 116. <https://doi.org/10.1007/s10664-024-10503-2>
- Menzies, T., & Shepperd, M. (2019). Bad smells in software analytics papers. *Information and Software Technology*, 112, 35–47. <https://doi.org/10.1016/j.infsof.2019.04.005>
- Milner, R. (1989). *Communication and concurrency*. Prentice Hall.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18(2), 203–226. [https://doi.org/10.1016/0004-3702\(82\)90040-6](https://doi.org/10.1016/0004-3702(82)90040-6)
- Monperrus, M. (2018). *The living review on automated program repair* (tech. rep. No. hal-01956501). HAL Archives Ouvertes.
- Nguyen, H. D. T., Qi, D., Roychoudhury, A., & Chandra, S. (2013). Semfix: Program repair via semantic analysis. *35th International Conference on Software Engineering*, 772–781. <https://doi.org/10.1109/ICSE.2013.6606623>
- Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *22nd International Conference on Machine Learning*, 625–632. <https://doi.org/10.1145/1102351.1102430>
- Northcutt, C. G., Athalye, A., & Mueller, J. (2021). Pervasive label errors in test sets destabilize machine learning benchmarks. *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Offutt, A. J. (1992). Investigations of the software testing coupling effect. *ACM Transactions on Software Engineering and Methodology*, 1(1), 5–20. <https://doi.org/10.1145/125489.125473>
- Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., & Snoek, J. (2019). Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32.
- Pacheco, C., Lahiri, S. K., Ernst, M. D., & Ball, T. (2007). Feedback-directed random test generation. *29th International Conference on Software Engineering*, 75–84. <https://doi.org/10.1109/ICSE.2007.37>
- Pearson, S., Campos, J., Just, R., Fraser, G., Abreu, R., Ernst, M. D., Pang, D., & Keller, B. (2017). Evaluating and improving fault localization. *39th International Conference on Software Engineering*, 609–620. <https://doi.org/10.1109/ICSE.2017.62>
- Peleg, H., Shoham, S., & Yahav, E. (2018). Programming not only by example. *40th International Conference on Software Engineering*, 1114–1124. <https://doi.org/10.1145/3180155.3180189>
- Pineau, J., Vincent-Lamarre, P., Sinha, K., Lariviere, V., Beygelzimer, A., d’Alche-Buc, F., Fox, E., & Larochelle, H. (2021). Improving reproducibility in machine learning research. *Journal of Machine Learning Research*, 22(164), 1–20.
- Qi, Z., Long, F., Achour, S., & Rinard, M. (2015). An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. *International Symposium on Software Testing and Analysis*, 24–36. <https://doi.org/10.1145/2771783.2771791>
- Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). Dataset shift in machine learning. *MIT Press*.

- Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019). Do imagenet classifiers generalize to imagenet? *International Conference on Machine Learning*, 5389–5400.
- Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14, 131–164. <https://doi.org/10.1007/s10664-008-9102-8>
- Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. <https://doi.org/10.1371/journal.pone.0118432>
- Settles, B. (2009). Active learning literature survey. *University of Wisconsin-Madison Computer Sciences Technical Report 1648*.
- Shepperd, M., Bowes, D., & Hall, T. (2014). Researcher bias: The use of machine learning in software defect prediction. *IEEE Transactions on Software Engineering*, 40(6), 603–616. <https://doi.org/10.1109/TSE.2014.2322358>
- Smith, E. K., Barr, E. T., Le Goues, C., & Brun, Y. (2015). Is the cure worse than the disease? overfitting in automated program repair. *10th Joint Meeting on Foundations of Software Engineering*, 532–543. <https://doi.org/10.1145/2786805.2786825>
- Staats, M., Gay, G., & Heimdahl, M. P. E. (2012). Automated oracle creation support, or: How i learned to stop worrying about fault propagation and love mutation testing. *34th International Conference on Software Engineering*, 870–880. <https://doi.org/10.1109/ICSE.2012.6227131>
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.
- Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2017). An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering*, 43(1), 1–18. <https://doi.org/10.1109/TSE.2016.2584050>
- Tantithamthavorn, C., McIntosh, S., Hassan, A. E., & Matsumoto, K. (2019). The impact of automated parameter optimization on defect prediction models. *IEEE Transactions on Software Engineering*, 45(7), 683–711. <https://doi.org/10.1109/TSE.2018.2794977>
- Tian, H., Liu, K., Kaboré, A. K., Koyuncu, A., Li, L., Klein, J., & Bissyandé, T. F. (2020). Evaluating representation learning of code changes for predicting patch correctness in program repair. *35th IEEE/ACM International Conference on Automated Software Engineering*, 981–992. <https://doi.org/10.1145/3324884.3416532>
- Torralba, A., & Efros, A. A. (2011). Unbiased look at dataset bias. *CVPR*, 1521–1528. <https://doi.org/10.1109/CVPR.2011.5995347>
- Vovk, V., Gammernan, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Springer. <https://doi.org/10.1007/b106715>
- Wang, S., Wen, M., Lin, B., Wu, H., Qin, Y., Zou, D., Mao, X., & Jin, H. (2020). Automated patch correctness assessment: How far are we? *35th IEEE/ACM International Conference on Automated Software Engineering*, 968–980. <https://doi.org/10.1145/3324884.3416590>
- Weimer, W., Nguyen, T., Le Goues, C., & Forrest, S. (2009). Automatically finding patches using genetic programming. *31st International Conference on Software Engineering*, 364–374. <https://doi.org/10.1109/ICSE.2009.5070536>
- Wen, M., Chen, J., Wu, R., Hao, D., & Cheung, S.-C. (2018). Context-aware patch generation for better automated program repair. *40th International Conference on Software Engineering*, 1–11. <https://doi.org/10.1145/3180155.3180233>
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in software engineering. *Springer*. <https://doi.org/10.1007/978-3-642-29044-2>
- Xia, C. S., Wei, Y., & Zhang, L. (2023). Automated program repair in the era of large pre-trained language models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA2), 237–265. <https://doi.org/10.1145/3628252>

- Xia, C. S., & Zhang, L. (2022). Less training, more repairing please: Revisiting automated program repair via zero-shot learning. *30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 959–971. <https://doi.org/10.1145/3540250.3549101>
- Xiong, Y., Wang, J., Yan, R., Zhang, J., Han, S., Huang, G., & Zhang, L. (2017). Precise condition synthesis for program repair. *39th International Conference on Software Engineering*, 416–426. <https://doi.org/10.1109/ICSE.2017.45>
- Xuan, J., Martinez, M., DeMarco, F., Clement, M., Lamelas Marcote, S. R., Durieux, T., Le Berre, D., & Monperrus, M. (2017). Nopol: Automatic repair of conditional statement bugs in java programs. *IEEE Transactions on Software Engineering*, 43(1), 34–55. <https://doi.org/10.1109/TSE.2016.2560811>
- Ye, X., Bunescu, R., & Liu, C. (2014). Learning to rank relevant files for bug reports using domain knowledge. *22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 689–699. <https://doi.org/10.1145/2635868.2635874>
- Zhu, Q., Guo, Z., Zhang, Y., Wang, Y., & Wang, J. (2021). A syntax-guided edit decoder for neural program repair. *29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 341–353. <https://doi.org/10.1145/3468264.3468544>
- Zilles, S., Lange, S., Holte, R., & Zinkevich, M. (2011). Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12, 349–384.