VERILOOP STRUCTURAL REPAIR: QUANTIFYING PROGRESS WITHOUT REGRESSION IN CODE REPAIR

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-based program repair can hide damage to previously correct behavior when it reports only aggregate test success. We introduce VSR, which constructs training and evaluation data from protected-obligation rank vectors and the non-regression relation $I > 0, R = 0$. The dataset contains 40 program families, 988 tasks, 2,297 states, and 27,046 executed candidates and passes 228,473 independent checks. Base/SFT comparisons at both 4B and 9B show that supervised repair data improves closed-loop completion. Construction ablations show that coupling-aware sampling exceeds equal-size random sampling at both model scales and on all three test distributions, while the full SFT mixture similarly exceeds COMPLETE-only in every matched comparison. Sequential 9B SFT-to-DPO raises completion by 0.0119, 0.0417, and 0.0476 on ID, OOD-family, and OOD-τ, respectively. In the DPO error-type ablations, removing either aggregate-trap or protected-regression examples causes a comparatively large drop in completion and usually lowers NRIR or increases calls, highlighting their roles in resisting aggregate-score shortcuts and preserving previously correct behavior.

1 INTRODUCTION

Large language models can propose patches from specifications, failing tests, and repository context. Repair differs from code generation because an incumbent program usually satisfies many behaviors already. A safe patch must correct observed failures and preserve those behaviors. Existing benchmarks emphasize real faults, repository context, or final test-suite success (Just et al., 2014; Lin et al., 2017; Jimenez et al., 2024); code-generation benchmarks similarly score candidates by execution against a test collection (Chen et al., 2021; Liu et al., 2023). These protocols measure functional correctness but rarely represent newly repaired behavior and regression of previously correct behavior as distinct variables.

This omission is closely related to test-suite overfitting in automated program repair. Generate-and-validate systems can produce plausible patches that pass available tests without implementing the intended semantics (Qi et al., 2015; Smith et al., 2015). For language-model repair, a weaker form can occur within one feedback round: a candidate fixes the disclosed cases and improves the aggregate pass count while breaking an undisclosed obligation that previously passed. A scalar reward or a final patch alone does not explain why that candidate should be rejected.

VSR puts non-regression into data construction. Each obligation is a protected behavioral coordinate. A candidate must improve at least one failing coordinate while remaining unchanged on every previously passing coordinate. The resulting protected partial order supports aggregate traps whose net pass count improves even though the candidate is unsafe.

We evaluate supervised training, obligation sampling, SFT target mixtures, and sequential SFT-to-DPO at 4B and 9B scales. SFT exceeds the native Base at both scales. Coupling-aware sampling outperforms equal-size random sampling on ID, OOD-family, and OOD-τ at both scales, and the full SFT mixture exceeds COMPLETE-only in all six matched comparisons. Relative to its SFT starting point, Full 9B DPO raises completion by 0.0119, 0.0417, and 0.0476 on ID, OOD-family, and OOD-τ. In the DPO ablations, removing aggregate-trap or protected-regression has the largest negative effect on completion at both scales.

Our contributions are:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- We define an executable non-regression contract that separates improvement I from protected regression R and generates safe actions and structured negatives.
- We introduce a coupling signature $h(u)$ and task statistic τ that provide a detectability condition, controlled obligation sampling, and a measurable out-of-distribution axis.
- We construct hindsight-aligned supervision by fixing target code before selecting diagnostics and visible cases from obligations repaired by that target.
- We construct single-step supervision, preference pairs, state graphs, and training-isolated execution oracles, and independently audit 228,473 executable checks.
- We provide cross-scale construction ablations that quantify obligation sampling, SFT target mixtures, and structured error-type negatives under a common three-seed protocol.

2 Related Work

Program-repair benchmarks. Defects4J provides reproducible real Java faults with versioned buggy and fixed programs (Just et al., 2014). QuixBugs offers small Python and Java algorithmic defects for cross-language repair comparison (Lin et al., 2017). SWE-bench evaluates repository-level agents on GitHub issues and pull requests (Jimenez et al., 2024). These resources prioritize realism, system context, or final resolution. VSR is complementary: it gives up repository scale to make defect sites, obligation coupling, candidate behavior changes, and negative classes controllable and independently recomputable.

Patch correctness and stronger tests. Prior work shows that patches passing the available suite can still violate intended semantics or damage untested behavior (Qi et al., 2015; Smith et al., 2015). EvalPlus expands code-generation tests and demonstrates that limited tests can change model rankings (Liu et al., 2023). VSR moves this concern into training-data construction: each state retains failing and passing obligations, and the latter form a protected set that candidates may not regress.

Execution supervision and preference learning. Code models are commonly evaluated with execution pass rates (Chen et al., 2021). NExT uses execution traces to improve reasoning about program behavior (Ni et al., 2024). DPO learns directly from chosen and rejected responses without fitting an explicit reward model (Rafailov et al., 2023). VSR supplies a repair-specific structure for these paradigms: execution determines I, R, and J; hindsight alignment selects prompt evidence; and protected preference pairs make alternatives tie on visible cases whenever possible so that the remaining signal concerns hidden non-regression.

3 Problem Formulation

Let $O = \{o_j\}_{j=1}^{m}$ be protected obligations. Obligation o_j contains an input u_j, a stable identifier, and expected observation y_j produced by a reference implementation $f^\star$. The observation includes both the return value and the argument after the call:

$$T(f,u) = \mathrm{canonical}\big(\mathrm{return}(f,u), \mathrm{argAfter}(f,u)\big). \tag{1}$$

This makes mutation and aliasing errors first-class evidence. Exceptions, recursion errors, and timeouts are canonical error observations.

Program f has rank vector $r(f) \in \{0,1\}^m$:

$$r_j(f) = \mathbb{I}[T(f,u_j) \neq y_j]. \tag{2}$$

Passing coordinates are retained because they define behavior the patch must preserve. For incumbent f and candidate g, define

$$I(g;f) = \big|\{j : r_j(f) = 1 \wedge r_j(g) = 0\}\big|, \tag{3}$$

$$R(g;f) = \big|\{j : r_j(f) = 0 \wedge r_j(g) = 1\}\big|, \tag{4}$$

$$J(g;f) = \{j : r_j(f) = 1 \wedge r_j(g) = 0\}. \tag{5}$$

A candidate is *admissible* when $I(g;f) > 0$ and $R(g;f) = 0$. It is a *complete repair* when it is admissible and $r(g) = \mathbf{0}$. This coordinate-wise protection is stronger than a positive aggregate gain $I - R > 0$.

4 Dataset Construction Method

All construction mechanisms are collected in this section. Each subsection is one method component and contains no further nested structure.

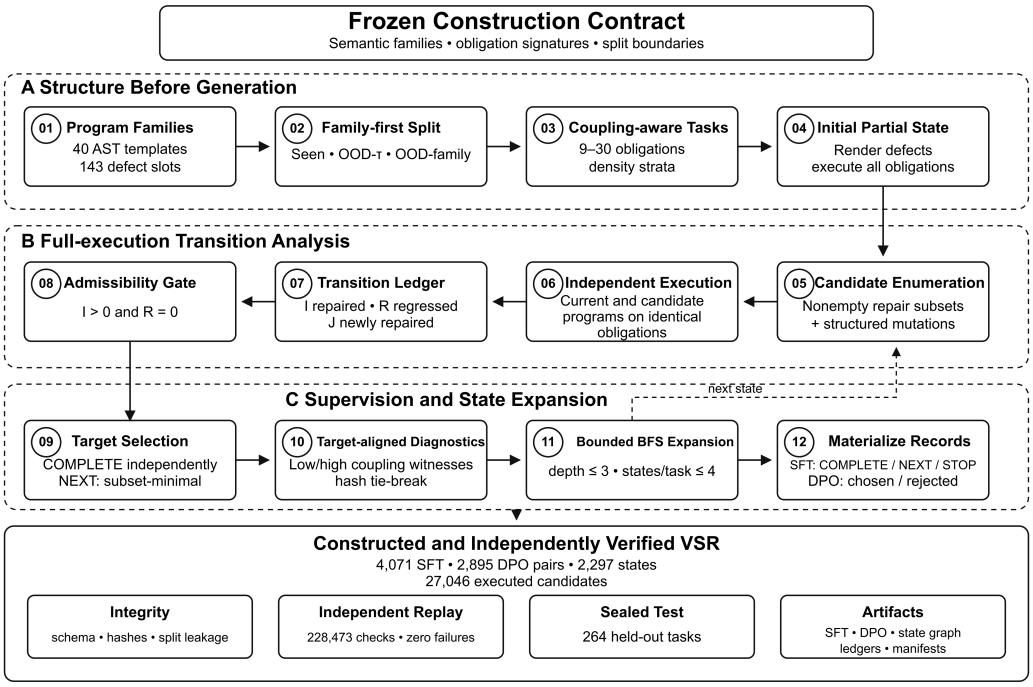

Figure 1: VSR dataset construction pipeline. The process first freezes program families, obligation signatures, split boundaries, and the random seed; it then independently executes every candidate on all obligations, applies the admissibility gate, and materializes hindsight-aligned supervision, the bounded state graph, and release views. The dashed loop sends a state produced by `NEXT_SAFE_STEP` back to candidate enumeration.

4.1 Program Families and Defect Lattice

The current instantiation contains 40 short Python families across text, integer sequence, key-value text, dictionary, interval, and matrix domains. Each family provides a natural-language contract, a `solve` template, and AST defect sites. Grade zero is the correct fragment; positive grades encode incorrect conditions, boundaries, indices, return structures, update rules, or mutation handling. A grade vector uniquely renders an executable program.

An entry state receives one to four wrong sites. The builder rejects an invalid reference, insufficient obligations, an entirely correct or entirely failing incumbent, a repair that only restores constants, or a state without observable single-site improvement. Retained states are scored only by protected-order structure: masked passing coordinates jointly observed by multiple wrong sites, single-site repair traps, and observable single-site improvements.

4.2 Obligations and Coupling Signatures

For correct program $f^\star$, site q, and input u, the builder substitutes each defective grade at q while keeping every other site correct. The coupling signature is

$$h(u)_q = \mathbb{I}\left[\exists a \in G_q \setminus \{0\} : T(f^{\star[q \leftarrow a]}, u) \neq T(f^\star, u)\right]. \tag{6}$$

Thus $h(u)$ identifies the latent sites observable through input u. Its weight $w(u) = \|h(u)\|_1$ counts how many sites can affect that obligation under isolated perturbations. Weight-one obligations ap-

proximate local witnesses. High-weight obligations span sites and are more likely to expose cancellation, repair interference, and hidden regression. Task coupling is

$$\tau(O) = \frac{1}{|O|} \sum_{u \in O} \|h(u)\|_1. \tag{7}$$

Each task first assigns every expressible site a lightest available witness. This is a detectability floor: every site must be observable somewhere in the task. The remaining budget follows the weight quotas in Table 1. If a bucket is empty, its allocation moves to the nearest nonempty bucket rather than reducing the number of obligations. Tasks contain 16–30 obligations subject to a minimum imposed by family size.

Table 1: Obligation-weight quotas.

Regime	$w = 1$	$w = 2$	$w = 3$	$w \geq 4$
τ_{low}	.55	.30	.10	.05
τ_{mid}	.35	.30	.22	.13
τ_{high}	.22	.25	.27	.26

The signature makes difficulty claims testable. Detectability is the coverage condition $\forall q, \exists u : h(u)_q = 1$. Locality is summarized by the share of $w = 1$ obligations. τ supports performance curves under increasing coupling. A *masked coordinate* is currently passing while observed by at least two wrong sites, quantifying fragile correctness due to error cancellation. If performance is insensitive to these controlled changes, then $h(u)$ has not formed the intended difficulty axis.

4.3 Candidate Space and Safe Actions

At a nonterminal state the generator modifies only currently wrong sites. It enumerates every nonempty subset of wrong sites to restore, alternative defective grades at wrong sites, and bounded repair-plus-misfix combinations. A currently correct grade is never directly corrupted. Behavioral regressions therefore arise from coupling rather than an artificial edit to a correct site. The sole exception is terminal supervision: a zero-rank program has no benign candidate, so one-site perturbations provide explicit negatives for continued editing and are isolated as `post_completion_edit`.

All candidates are re-executed and classified as shown in Table 2. The `aggregate_trap` class is central: its aggregate pass count improves, but it violates the protected order.

Table 2: Execution-defined candidate relations.

Relation	Condition	Safe	Interpretation
strict progress	$I > 0, R = 0$	yes	repairs behavior without regression
equal rank	$I = 0, R = 0$	no	no behavioral gain
protected regression	$I = 0, R > 0$	no	only damages protected behavior
aggregate trap	$I > R > 0$	no	net gain hides a regression
Pareto incomparable	$R \geq I > 0$	no	improvement and damage coexist

Complete and incomplete safe repairs first enter the same admissible action set

$$\mathcal{A}(f) = \{g : I(g; f) > 0,\ R(g; f) = 0\}. \tag{8}$$

The subsets $\mathcal{C}(f) = \{g \in \mathcal{A}(f) : r_O(g) = 0\}$ and $\mathcal{N}(f) = \mathcal{A}(f) \setminus \mathcal{C}(f)$ distinguish complete repairs from safe intermediate states. Restoring all wrong sites yields the reference program, so $\mathcal{C}(f)$ and hence $\mathcal{A}(f)$ are nonempty for every generated nonterminal state.

Let $S_g \subseteq W(f)$ be the repaired-site set of an admissible pure repair. The operative forward filter is

$$\mathcal{M}(f) = \{g \in \mathcal{A}_{\text{pure}}(f) : \nexists g' \in \mathcal{A}_{\text{pure}}(f) \text{ with } S_{g'} \subset S_g\}. \tag{9}$$

A safe intermediate step is retained only when no strictly smaller repaired-site subset is admissible. This defines an atomic safe move without rewarding long trajectories.

Every expanded nonterminal state independently emits one `COMPLETE_REPAIR` and selects at most one `NEXT_SAFE_STEP` from the subset-minimal forward-action set $\mathcal{M}(f)$; a zero-rank state emits `TERMINAL_STOP` with target `STOP`. Subset-minimal `NEXT_SAFE_STEP` examples teach the model to make the smallest safe advance, reducing the risk of damaging obligations that already pass. This bias can, however, favor partial repairs and introduce unnecessary multi-step execution on easier tasks. Training therefore also includes `COMPLETE_REPAIR` and controls its sampling ratio relative to `NEXT_SAFE_STEP` to preserve the model's ability to complete a repair in one step. `TERMINAL_STOP` teaches the model not to modify an already correct program and to recognize when repair is complete.

4.4 Hindsight-Aligned Supervision

Sampling visible cases before choosing a target can misalign prompt and label: the prompt may show failure A while the target repairs only B. VSR therefore fixes target g first, computes $J(g; f)$, and only then reads visible examples from the incumbent's pre-repair execution record. Every repair record enforces

$$D \subseteq J(g; f). \tag{10}$$

Generation asserts this inclusion. The independent verifier re-executes f and g and checks the recorded input, pre-repair observation, reference expectation, and target observation for every displayed diagnostic.

When J contains more than two obligations, the deterministic selector chooses one lightest and one heaviest witness. The light witness supplies relatively local evidence; the heavy witness reveals cross-site dependence. Hash salts include task, state, repair mode, and record type. This stratification prevents uniform sampling from systematically underexposing high-coupling coordinates when low-weight obligations are more numerous.

Preference pairs share task, state, repair mode, prompt, and diagnostics. For aggregate-trap and Pareto-incomparable pairs, diagnostics are preferentially drawn from $J(g^+; f) \cap J(g^-; f)$. Both alternatives then solve the disclosed cases, leaving hidden protected regression as the preference signal. The `dpo_protected` view keeps aligned pairs with equal visible scores. Its visible-diagnostic proxy accuracy is 0.500, versus 0.650 on the full preference set. Pair selection also balances character count, changed lines, and edit distance; the strongest audited surface heuristic reaches 0.523.

4.5 State Graph Expansion

The generator performs breadth-first expansion only along incomplete subset-minimal safe actions and deduplicates states by normalized program skeleton and evidence vector. Each task is capped at four expanded states—the initial state plus at most three reached states—and depth three. These are enumeration budgets, not optimization objectives: depth does not enter task scoring or target selection, and all enumerated candidates at every visited state retain their execution-derived labels.

The release is therefore a deterministic bounded prefix of the reachable subset-minimal graph rather than its exhaustive transitive closure. Each stored training example is one transition; no multi-round prompt sequence is precomputed. The graph is acyclic because every traversed edge has $I > 0$ and $R = 0$ and strictly reduces the failure count.

To reduce template-format dependence, a deterministic 25% sample applies `ast.unparse` jointly to incumbent and target. Independent verification re-executes both forms. The release contains 813 style-jittered and 2,461 unchanged repair records.

4.6 Data Views and Split Protection

Table 3 summarizes the release. The final SFT mixture is 54.41% complete repair, 26.01% next safe step, and 19.58% terminal stop. DPO contains 921 aggregate traps, 921 Pareto-incomparable negatives, 395 protected regressions, 395 equal-rank negatives, and 263 post-completion edits.

Table 3: Released views.

View	Records	Purpose
SFT repair	3,274	complete repair and minimal safe step
SFT terminal	797	stopping after verified completion
DPO pairs	2,895	safe choices and structured negatives
DPO protected	1,842	alternatives tied on visible evidence
state graph	9,136	admissible edges and minimality labels
candidate ledger	27,046	complete executed candidate audit
Pareto probe	2,297	per-state selection measurements
sealed evaluation	988	complete obligations and entry programs

The 40 families are partitioned before task generation into 28 seen, four OOD development, and eight OOD test families. Test data contains standard-τ tasks from seen families, high-τ tasks from seen families, and unseen-family tasks. Validation and test entries are generated first; training cannot reuse their exact entry-grade vectors. A second pass removes lower-priority records that reproduce evaluation entry states. Checkpoint selection is restricted to the in-distribution validation subset. OOD-τ and OOD-family subsets are monitoring sets only.

4.7 Independent Verification

The independent verifier reads inputs and expected outputs from the sealed oracle and re-executes reference programs, entry programs, SFT targets, and DPO candidates. It does not trust recorded I, R, relation, or diagnostic fields. The release passes 228,473 hard checks with zero failures. An oracle policy completes all three test regimes with no unsatisfiable round and mean 1.186 steps.

5 Experiments

5.1 Research Questions

We evaluate four questions:

- RQ1: Does VSR supervised repair data improve closed-loop completion over the native model?
- RQ2: How does coupling-aware obligation sampling affect completion relative to an equal-size random sampler?
- RQ3: Does the full SFT mixture outperform COMPLETE_REPAIR-only training?
- RQ4: Does DPO continued from SFT provide further gains, and how does removing each structured error type affect them?

5.2 Models and Experimental Protocol

We use Qwopus3.5-4B-Coder and Qwopus3.5-9B-Coder. All training experiments use three seeds, and we report their means. SFT starts from the native Base, and DPO continues from the same-scale, same-seed Full SFT checkpoint selected only on ID validation. The sealed test is fixed at 264 tasks: 84 ID, 96 OOD-family, and 84 OOD-τ. Evaluation uses deterministic generation and at most six repair rounds. Every training variant uses the same non-regression criterion $I > 0$ and $R = 0$.

5.3 Metrics

The primary endpoint is six-round zero-rank completion. Non-regressive improvement rate is

$$\mathrm{NRIR} = \Pr[I > 0 \wedge R = 0], \tag{11}$$

which measures the frequency of non-regressive candidate improvements. Mean calls is the average number of model invocations per task. Every formal strict unit uses the same $I > 0, R = 0$ acceptance rule, so the results compare training data. Completion is available for every experiment; NRIR and calls are reported where the corresponding trajectory audit is part of the comparison.

5.4 SUPERVISED TRAINING BASELINE

Table 4 reports the 264-task Base/SFT mechanism baseline and tests whether supervised repair data is effective.

Table 4: Six-round completion and OOD-family NRIR for native Base and SFT.

Scale	Method	ID	OOD-family	OOD-τ	Family NRIR
4B	Base	.7143	.3229	.7262	.2651
4B	SFT	.9405	.5625	.8452	.4340
9B	Base	.8690	.6250	.7738	.4655
9B	SFT	.9762	.7292	.8929	.5404

Relative to Base, 4B SFT raises ID, OOD-family, and OOD-τ completion by 0.2262, 0.2396, and 0.1190; OOD-family NRIR rises from 0.2651 to 0.4340. The corresponding 9B SFT completion gains are 0.1072, 0.1042, and 0.1191, while NRIR rises from 0.4655 to 0.5404. Improvements in both completion and NRIR validate the training value of the supervised repair data.

5.5 COUPLING-AWARE VERSUS EQUAL-SIZE RANDOM SAMPLING

Table 5 isolates coupling-aware obligation sampling from its equal-size random control and reports three-seed means.

Table 5: Coupling-aware versus equal-size random obligation sampling.

Scale	Sampler	ID	OOD-family	OOD-τ
4B	Coupling-aware	.9405	.5625	.8452
4B	Equal-size random	.9167	.5104	.8214
9B	Coupling-aware	.9762	.7292	.8929
9B	Equal-size random	.9405	.6875	.8690

Coupling-aware sampling exceeds the equal-size random control on all three distributions at both scales. The 4B completion gains on ID, OOD-family, and OOD-τ are 0.0238, 0.0521, and 0.0238; the corresponding 9B gains are 0.0357, 0.0417, and 0.0238. This validates the training value of coupling-signature-based obligation sampling. A plausible explanation is that the sampler preserves local witnesses while exposing more high-coupling obligations, so the model more often observes site interactions, defect cancellation, and hidden regressions than under uniform random sampling.

5.6 FULL SFT MIXTURE VERSUS COMPLETE-ONLY

Table 6 compares the full SFT target mixture (COMPLETE, NEXT, and STOP) against the COMPLETE-only control and reports three-seed means.

Table 6: Full SFT target mixture versus COMPLETE-only.

Scale	SFT target mixture	ID	OOD-family	OOD-τ
4B	Full SFT mixture	.9405	.5625	.8452
4B	COMPLETE-only	.9048	.5208	.8095
9B	Full SFT mixture	.9762	.7292	.8929
9B	COMPLETE-only	.9286	.7188	.8333

The full SFT mixture exceeds COMPLETE-only in every matched comparison. The 4B completion gains on ID, OOD-family, and OOD-τ are 0.0357, 0.0417, and 0.0357; the corresponding 9B gains are 0.0476, 0.0104, and 0.0595. This validates the joint value of COMPLETE, NEXT, and STOP supervision. NEXT teaches safe intermediate progress in multistep repair, while STOP discourages

editing an already correct program; together with COMPLETE, they cover one-step repair, safe decomposition, and termination.

5.7 SEQUENTIAL SFT-TO-DPO AND ERROR-TYPE ABLATIONS

Table 7 compares 9B Full DPO with its same-seed SFT starting point across three paired seeds.

Table 7: Mean 9B completion across three seeds for SFT and Full DPO.

Stage	ID	OOD-family	OOD-τ
SFT	.9762	.7292	.8929
Full DPO	.9881	.7708	.9405
DPO–SFT	+.0119	+.0417	+.0476

Continuing from the same 9B SFT starting point with Full DPO raises completion by 0.0119, 0.0417, and 0.0476 on ID, OOD-family, and OOD-τ. This validates that structured preference data can further improve closed-loop repair after supervised training, with the largest gains on unseen families and high-coupling tasks.

Table 8 and Table 9 report the 4B and 9B DPO ablations that remove one structured error type at a time. Full is the common baseline, NRIR and calls are OOD-family metrics, and all entries are three-seed means.

Table 8: 4B DPO error-type ablations.

Variant	ID	Family	τ	NRIR	Calls
Full	.9643	.6042	.8929	.4622	3.969
No aggregate trap	.9524	.5625	.8690	.4146	3.969
No Pareto incomparable	.9643	.5938	.8810	.4502	3.938
No protected regression	.9524	.5833	.8571	.4364	4.073
No equal rank	.9524	.5938	.8929	.4578	3.958

Table 9: 9B DPO error-type ablations.

Variant	ID	Family	τ	NRIR	Calls
Full DPO	.9881	.7708	.9405	.5542	3.021
No aggregate trap	.9881	.7292	.9167	.5092	3.208
No Pareto incomparable	.9762	.7708	.9286	.5309	3.042
No protected regression	.9881	.7396	.9048	.5114	3.250
No equal rank	.9762	.7604	.9405	.5136	3.031

In the 4B ablation, Full exceeds the no-aggregate-trap variant by 0.0119, 0.0417, and 0.0238 in ID, OOD-family, and OOD-τ completion and by 0.0476 in NRIR. Relative to no-protected-regression, its completion gains are 0.0119, 0.0208, and 0.0357; NRIR rises by 0.0258 and mean calls fall by 0.104. The 9B results reinforce this pattern. Relative to no-aggregate-trap, Full DPO raises OOD-family and OOD-τ completion by 0.0417 and 0.0238, raises NRIR by 0.0450, and reduces mean calls by 0.187. Relative to no-protected-regression, it raises OOD-family and OOD-τ completion by 0.0313 and 0.0357, raises NRIR by 0.0428, and reduces mean calls by 0.229. Aggregate-trap and protected-regression are therefore the two most important negative types to retain: the former exposes candidates that improve aggregate score while damaging protected obligations and prevents the model from treating a higher total pass count as sufficient for safety; the latter supplies the most direct regression examples and teaches the model to preserve obligations that already pass. They respectively cover hidden positive-net-gain shortcuts and explicit damage to previously correct behavior, making their roles complementary in non-regression preference learning.

6 DISCUSSION

Supervised repair data improves ID, OOD-family, and OOD-τ completion at both scales and also raises OOD-family NRIR, showing that execution-verified COMPLETE, NEXT, and STOP records convert native code capability into closed-loop repair capability.

Coupling-aware sampling exceeds equal-size random sampling on all three distributions at both 4B and 9B, indicating that $h(u)$-guided obligation selection converts defect coverage and high-coupling interactions into training gains. The especially clear OOD-family gain is consistent with coupling witnesses teaching cross-site protection relations. Full SFT exceeds COMPLETE-only in every comparison at both scales, supporting the joint value of safe intermediate actions and stopping supervision.

Sequential 9B SFT-to-DPO improves completion on all three distributions, showing that preference training complements SFT. At both 4B and 9B, removing aggregate-trap or protected-regression lowers OOD-family completion and is accompanied by lower NRIR or more calls. This matches their complementary roles: one corrects aggregate-score shortcuts and the other directly protects previously correct behavior.

7 LIMITATIONS

The study has three main limitations. First, short functions and templated defects do not replace repository-level repair. Second, a finite obligation set is behavioral evidence rather than a proof of semantic equivalence. Third, experiments cover only two model scales and three random seeds, and reporting only means may be insufficient for broader statistical generalization.

8 CONCLUSION

VSR constructs quantifiable non-regressive code-repair data through a protected behavioral partial order, coupling signatures, hindsight alignment, and independent execution verification. Base/SFT comparisons at both scales establish clear training value from the supervised view. Coupling-aware sampling exceeds equal-size random sampling on all three distributions at both 4B and 9B, and Full SFT exceeds COMPLETE-only in every matched comparison. Full 9B DPO further improves completion on all three distributions. Error-type ablations consistently show that aggregate-trap and protected-regression negatives are especially important for OOD-family completion, NRIR, and call efficiency. VSR therefore provides both training data and an experimental framework for quantifying the contribution of individual construction modules.

REPRODUCIBILITY STATEMENT

Generation is deterministic from a fixed master seed and requires neither network access nor an external model. Upon acceptance, we will publicly release the VSR dataset, dataset-generation and independent-verification code, training and evaluation configurations, manifest, file hashes, and reproduction instructions.

ETHICS STATEMENT

The dataset consists of synthetic short programs and automatically generated inputs. It contains no personal data and no content mined from developer repositories. Performance is not a deployment-safety guarantee. Practical use of repair models requires comprehensive testing, human review, and isolated execution.

AI USE STATEMENT

Generative AI tools assisted with drafting prose from author-provided content, Chinese–English translation, and language editing. The experimental design, substantive summaries and scientific interpretation of the results, and all final decisions about the manuscript structure and argument

were made by the authors. The authors independently verified dataset code, construction statistics, execution results, citations, and all scientific claims, and take responsibility for the final content.

REFERENCES

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *ICLR*, 2024.

René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: A database of existing faults to enable controlled testing studies for java programs. In *ISSTA*, 2014.

Derrick Lin, James Koppel, Angela Chen, and Armando Solar-Lezama. Quixbugs: A multi-lingual program repair benchmark set based on the quixey challenge. In *SPLASH Companion*, 2017.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, Charles Sutton, and Pengcheng Yin. NExT: Teaching large language models to reason about code execution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37929–37956, 2024.

Zichao Qi, Fan Long, Sara Achour, and Martin Rinard. An analysis of patch plausibility and correctness for generate-and-validate patch generation systems. In *ISSTA*, 2015.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Edward K. Smith, Earl T. Barr, Claire Le Goues, and Yuriy Brun. Is the cure worse than the disease? overfitting in automated program repair. In *ESEC/FSE*, 2015.

A REPRODUCTION ASSETS

The post-acceptance release will include deterministic generation, independent verification, baseline evaluation, training-route guards, all JSONL views, a manifest with SHA256 hashes, and a one-command CPU rebuild. The training loader rejects paths containing sealed evaluation, candidate ledger, or Pareto probe names. Model selection is restricted to the in-distribution validation regime.