SEAMBENCH: EVALUATING APPEND-ONLY MODEL HANDOFFS IN CODE

Anonymous authors
Paper under double-blind review

ABSTRACT

When an LLM service switches models mid-response, the replacement must finish code it did not write and cannot revise, and the client must decide whether to accept the result. We introduce SeamBench, a benchmark of 2,488 such append-only code handoffs from 428 HumanEval and MBPP tasks, built only from tasks both models solve on their own and labelled by executing the combined program. The acceptance decision turns out to be the hard part, and what a detector is allowed to see is a first-order design variable: in a controlled re-judging of 517 BigCodeBench handoffs, giving the judge the task description raises recall in every condition, while revealing where the switch occurred lowers AUC without the task and raises it with the task. Continuations fail on 10–26% of prefixes; 91% of failures are caught by syntax and entry-point checks, but the remaining semantic failures are caught by no rule and by a content-only judge fewer than half the time, at a false-rejection cost that varies from 2% to 42% across model directions. Switching itself is not the source of the failures: on 261 matched prefixes with fresh same-model restarts, replacing the continuation model never increases failure, and in one direction lowers it from 26% to 10%. All seams, labels and scorers are released under MIT and CC-BY licenses.

1 INTRODUCTION

If a model stops halfway through a streamed code response, can another model finish it without changing what was already emitted? A plausible continuation is not enough: it may redefine the function, duplicate a loop, or close the wrong block. An append-only client must keep the prefix and accept or reject the continuation as supplied. The object to evaluate is the combined program.

Prior work on cross-model continuation studies mathematical reasoning (Lu et al., 2025), inference-cost switching (Song et al., 2026), and agent handoffs that may edit state (Pandey & Singh, 2026; Khraishi et al., 2026; Ganz et al., 2026). None fixes an emitted code prefix and executes the result. We ask two questions: does replacing the continuation model make failure more likely than restarting the original model, and which failures can be caught before the continuation is accepted?

SeamBench answers both on 2,488 handoffs (*seams*) from 428 HumanEval and MBPP tasks, three model pairs and six directions. The second question turns out to be the hard one. Each seam pairs a task both models solve on their own with a cut of one model's answer and a raw continuation by the other, and records structural, task-test and append-contract outcomes separately.

Switching is not the cost. On 261 prefixes with fresh same-model restarts collected under identical settings, replacing the continuation model never increases the failure rate. Sonnet 4.6 restarting its own code fails on 26% of prefixes; GPT-4o-mini continuing the same prefixes fails on 10%. The other two directions change failure by $+1.1$ and -8.0 points, with intervals spanning zero. A prospectively selected BigCodeBench validation on 517 prefixes from 90 tasks shows the same: -2.3 and $+3.1$ points, both spanning zero. Continuation failures are real, but they are not caused by changing the model.

Continuation is the cost, and most of it is visible. Same-model restarts fail on 10–26% of prefixes, and 569 of the 2,488 core seams (23%) fail the combined check. Of these, 520 (91%) already

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.

violate public structural rules: the program does not parse, or the entry point is missing or duplicated. Syntax and interface checks alone therefore recover most of the loss.

The rest is hard to see. The 49 failures that pass structural checks split into 37 task-test failures, 11 append-contract violations and one timeout. Rule-based detectors catch none of the 37. An LLM judge given only the combined code catches 15, and on BigCodeBench 10 of 46, while rejecting 2–42% of correct outputs depending on direction. Which detector to prefer depends on the false-rejection budget: at $\leq 1\%$ the hybrid attains 95% recall against 93% for rules, and adding the judge to rules buys 15 extra detections for 321 extra false rejections on the core but 10 for 4 on BigCodeBench.

Task information beats boundary information. Re-judging all 517 BigCodeBench outputs under a common template, showing the task raises recall by 3–5 points in both boundary conditions; revealing the cut position lowers AUC without the task and raises it with the task. A detector's information access, not just its architecture, determines what it can see.

SeamBench separates the decision to switch from the decision to accept, and releases every seam, label and scorer under the source datasets' MIT and CC-BY-4.0 licenses.

2 Related work

Within-generation handoffs. Reasoning Relay (Lu et al., 2025) transfers partial mathematical reasoning between models. RelayGen (Song et al., 2026), R-Stitch (Chen et al., 2025) and SplitReason (Akhauri et al., 2025) switch models at segment level to trade inference cost against quality, while R2R (Fu et al., 2025), CITER (Zheng et al., 2025) and FusionRoute (Xiong et al., 2026) learn token-level routing between a small and a large model. Speculative decoding also interleaves two models, but the larger model may overwrite draft tokens (Leviathan et al., 2023; Chen et al., 2023a). We evaluate raw code continuations from prefixes that cannot be revised, with matched same-model restart controls.

Routing, cascades and conversation handoffs. Query-level routers and cascades choose one model per request (Chen et al., 2023c; Ding et al., 2024; Ong et al., 2025); they do not address a response already partially delivered. ContinuityBench (Pandey & Singh, 2026) studies conversation-state preservation during failover, Khraishi et al. (2026) compare directional model substitutions against no-switch baselines, and the Handoff Tax (Ganz et al., 2026) compares trajectory-transfer interfaces in coding agents that preserve repository state. Those successors can edit files; ours must retain an emitted text prefix. Because hosted aliases can change behavior over time (Chen et al., 2023b), we record returned model identities.

Infilling and partial programs. Fill-in-the-middle training (Bavarian et al., 2022) and infilling code models (Fried et al., 2023; Allal et al., 2023; Li et al., 2023; Rozière et al., 2023) condition on both a prefix and a suffix; our setting supplies only the prefix and the task. Dinh et al. (2023) study completion from potentially buggy prefixes, including removal and rewriting strategies, and self-repair loops revise a candidate after feedback (Chen et al., 2024; Olausson et al., 2024). Our prefixes have a recorded test-passing completion witness and cannot be changed, so neither rewriting nor repair is available.

Executable and structured benchmarks. HumanEval, MBPP, APPS, DS-1000, LiveCodeBench, SWE-bench and BigCodeBench supply executable checks for generated code (Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Lai et al., 2023; Jain et al., 2025; Jimenez et al., 2024; Zhuo et al., 2025); EvalPlus (Liu et al., 2023) shows that stronger tests expose errors accepted by original suites, which motivates our enhanced-test overlay. Grammar-constrained decoding can guarantee syntactic validity at generation time (Willard & Louf, 2023; Geng et al., 2023; Beurer-Kellner et al., 2024; Dong et al., 2024), sometimes at a cost to content (Tam et al., 2024), but cannot be applied to text already emitted; structured-output benchmarks such as JSONSchemaBench (Geng et al., 2025) measure only schema validity. We therefore separate structural, task-test and append-contract outcomes rather than equating parse success with correctness.

Output judges. MT-Bench (Zheng et al., 2023) evaluates agreement with human preferences; JudgeBench (Tan et al., 2025) uses correctness-based response pairs and filters format-induced grading errors. LLM judges show position and self-preference biases (Wang et al., 2024; Panickssery et al., 2024), and code-specific judges are evaluated against execution (Zhuo, 2024; Tong & Zhang, 2024; Gu et al., 2024). Structural violations remain part of our evaluation target. We report their contribution separately and test detectors after public checks pass, as well as under matched task and boundary information.

3 TASK AND EVALUATION PROTOCOL

For a source task t, model A produces a standalone answer y_A. Normalization N trims outer whitespace and applies a single fence-stripping pass. Cutting at character offset k gives the fixed prefix $p_k = N(y_A)_{:k}$. Model B receives this prefix together with the task prompt and returns continuation c_B. We evaluate $h = p_k \| c_B$: the continuation is appended verbatim, without normalization or repair.

Validation layers. We record three checks separately. Public structural validity $S_t(h)$ requires the program to parse, compile, and define the required top-level entry point exactly once. Task validity $T_t(h)$ runs the stored unit tests without a separate entry-uniqueness rejection. The append checks $H_t(p_k, c_B)$ test prefix preservation, exact suffix–prefix overlap, and a leading Markdown fence in the continuation. The versioned combined validator determines the archived append label, which we preserve alongside the replayed layers. One execution timeout remains unknown and is carried through as a labelled range.

3.1 CONDITIONAL CONTINUATION RISK

Within each cohort, let $D_t^{A,B} = 1$ when both recorded standalone answers pass the task tests after normalization by N. This *dual-success* condition isolates continuation failure from standalone inability. Conditional Seam Risk (CSR) is the append-failure fraction among retained seams; task-only failure is reported as a separate endpoint.

3.2 MATCHED SAME-MODEL CONTROL

Let $\mathcal{A}_{A\to B}$ be the frozen set of retained task/prefix anchors for a direction. At each anchor, we request new continuations from both A and B in the same collection wave, using the same prompt, prefix, temperature, recorded seed, and output-token limit. Dispatch order is randomized. For the task-only or append-contract endpoint E, let $E_{a,M}$ be 1 for failure and 0 for success of model M's continuation at anchor a. All matched outcomes under the original tests are resolved. We estimate

$$\widehat{\Delta}^{E}_{A\to B} = \frac{1}{|\mathcal{A}_{A\to B}|} \sum_{a\in\mathcal{A}_{A\to B}} (E_{a,B} - E_{a,A}). \tag{1}$$

Positive values mean more failures after replacing the continuation model. Each anchor receives equal weight; uncertainty is computed over whole source-task clusters, keeping repeated cuts together. The BigCodeBench validation in Section 6.3 uses one request per arm and anchor; the small MBPP follow-up (Appendix B.6) averages two.

3.3 DETECTOR METRICS

We reject risk scores ≥ 0.5. False-safe rate (FSR) is the fraction of unsafe seams accepted, and false-unsafe rate (FUR) is the fraction of safe seams rejected. ROC AUC treats unsafe as positive and gives half credit to ties; it is undefined in one-class subsets. The structural-pass analysis restricts to $S_t(h) = 1$, independently of detector predictions, and uses task-only labels to measure what remains after public validation.

Using the benchmark. Table 1 separates the comparisons supported by the records. Detector evaluation reuses fixed continuations; comparing a new continuation model requires fresh generations, not relabeling the archived outputs. A new model pair also requires new standalone eligibility

checks. Learned detectors should split by source task across all cuts and directions, since the core was used during development. Hidden tests are withheld from generation and judging.

Table 1: Evaluation uses. Repeated cuts share a source-task cluster.

Question	Keep fixed	Report
Replace the continuation model?	Task, prefix, eligibility rule, prompt and decoding settings	Fresh cross-minus-self failure, by direction
Accept a supplied continuation?	Combined output, labels and detector information access	Missed failures and rejected valid outputs; also structural-pass results
Change the task tests?	Recorded programs; recheck both standalone witnesses	Coverage, changed eligibility and paired labels on the retained population

4 DATA CONSTRUCTION

Tasks come from HumanEval (157 tasks) and MBPP (271 tasks) (Chen et al., 2021; Austin et al., 2021). The core contains 2,488 seams from these 428 tasks across six model directions (Table 2). Each record links the source task and pinned dataset revision to both standalone outputs, the cut, the raw continuation, all validation results, and request/response hashes.

Table 2: Core composition by direction. Unsafe counts archived append-contract failures; structural counts how many of those already fail public parse and entry-point checks.

Direction	Seams	Tasks	Unsafe	Structural
Sonnet 4.6 → GPT-5-mini	688	270	58	31
GPT-5-mini → Sonnet 4.6	1,190	417	383	373
GPT-4o-mini → Sonnet 4.6	349	130	93	91
Sonnet 4.6 → GPT-4o-mini	91	36	10	7
GPT-4o-mini → Qwen3-32B	95	37	18	14
Qwen3-32B → GPT-4o-mini	75	31	7	4
All	2,488	428	569	520

From tasks to seams. After dual-success filtering, we cut the normalized primary output at the line boundary nearest to 25%, 50% and 75% of its character length, requiring at least three nonempty lines on each side. Duplicate offsets are used once; tasks with no eligible cut yield no seams. Cuts are simulated interruption points. We verify distinct example IDs and content identities (task, request, contract, direction, prefix and continuation) for all 2,488 seams. All core tasks were available during development, and the public base benchmarks may appear in model pretraining data (Sainz et al., 2023; Golchin & Surdeanu, 2024).

Added cohorts and matched anchors. The three GPT-4o-mini/Qwen/Sonnet cohorts each start from 80 hash-ranked source tasks and retain up to 50 dual-success tasks in order, a plan fixed before generating these continuations. Table 3 separates selected tasks from those yielding cuts; the two Qwen directions select the same tasks but yield different counts because their primary outputs differ. Together they contribute 261 seams from 46 tasks, and every one of these seams is an anchor in the matched restart control. The control's fresh outputs do not add rows to the core.

Table 3: Construction of the three added cohorts. Each direction starts from 80 hash-ranked source tasks and retains up to 50 dual-success tasks in order; "with cuts" counts tasks yielding at least one seam.

Direction	Selected tasks	With cuts	Seams
Sonnet 4.6 → GPT-4o-mini	50	36	91
GPT-4o-mini → Qwen3-32B	44	37	95
Qwen3-32B → GPT-4o-mini	44	31	75

Validation and scope. We replay task and structural checks separately because the original append validator stops at the first rejection. An enhanced-test overlay using HumanEval+/MBPP+ rechecks both recorded standalones and the combined programs (Section 6.7); archived labels are unchanged. Earlier pilots on JSON Schema generation, prose (ASQA (Stelmakh et al., 2022)) and tool calls (BFCL (Patil et al., 2025)) are excluded from the benchmark: JSON has no semantic oracle beyond schema validity, prose anchors conflate task completion with continuation correctness, and the 59-example tool-call pilot had no failures. Appendix A documents these exclusions.

All 2,488 seams, labels, detector scores, the 261 matched anchors, the 517 BigCodeBench anchors and the fair-judge records are released with offline scorers; HumanEval is MIT and MBPP is CC-BY-4.0. A fresh network-disabled Linux container reproduces every reported statistic from the released records (Appendix B).

5 DETECTORS

Rules. Full rules use public text and contracts, including parsing and entry-point checks, without hidden unit tests or gold labels. They return the first matching rule's score. The `rules-clean` feature ablation removes selected validator checks while retaining other heuristics shared with the oracle.

LLM judges. The two archived judge baselines request a Sonnet 4.6 gateway alias at temperature zero with a 1,024-token output limit. All-at-once receives the original request, prefix, public contract, and numbered continuation segments; only the first 40 segments are shown, each limited to 400 characters. The `round-trip` judge instead makes one call over the full combined program and a checklist of continuation defects. It is told that two models contributed, but is not separately given the task, contract, or cut position. Appendix C records the prompts and input construction.

Matched information access. The new BCB judge audit holds the full combined code and one common template fixed, crossing public task access with disclosure of the zero-based Unicode cut offset. It uses Sonnet 4.6 at temperature zero with a 256-token output cap. Neither hidden tests nor reference answers are shown. This is a new comparison, not a rerun of either archived prompt. The earlier Qwen ablation varies only the boundary sentence. Both audits test a numeric cue, not an inline split marker.

Hybrid. If rule risk is at least 0.95, the hybrid returns it; otherwise it takes the maximum of rule and judge risk. With valid component scores, it rejects whenever either risk score is at least 0.5. The short circuit can still change AUC relative to an unconditional maximum. Audited comparisons reuse the same frozen judge responses.

Response validity. The decoder requires exactly one complete verdict with a finite risk score in $[0, 1]$, allowing surrounding text but rejecting ambiguous verdicts and duplicate keys. Invalid, empty and failed responses remain missing judgments and are reported as such. The additional-task study uses a separately frozen score-validation rule (Appendix B.6).

6 EXPERIMENTS

6.1 SETUP AND UNCERTAINTY

The six directions use Claude Sonnet 4.6, GPT-5 mini, GPT-4o-mini and Qwen3-32B through hosted endpoints at temperature zero; recorded model identities are in Appendix B.1. Detector comparisons use the same example IDs and threshold 0.5. We report pointwise 95% percentile intervals from source-task bootstrap resampling (Efron & Tibshirani, 1993; Miller, 2024): 2,000 replicates for detector tables and 5,000 for matched controls and structural-pass analyses. The three planned control tests form one Holm family (Holm, 1979); all other comparisons are descriptive.

Decoding and determinism. All generations use temperature zero with a recorded seed. Hosted endpoints do not promise deterministic decoding, but repeats are close to it: in the MBPP follow-up 22 of 28 same-condition repeat pairs and in the cursor pilot 90 of 192 returned identical text

Table 4: Matched restart controls on all 261 code anchors. Self and Cross are task-test failure rates (%); Δ is cross minus self in percentage points with a pointwise 95% source-task-clustered bootstrap interval. p is a two-sided exact task sign-flip test; Holm corrects the family of three. Append-contract and task-only outcomes coincide on every code anchor.

Direction	n/t	Self (%)	Cross (%)	Δ [95% CI]	p	Holm p
Sonnet 4.6 → GPT-4o-mini	91/36	26.4	9.9	-16.5 [-29.9, -3.2]	0.038	0.115
GPT-4o-mini → Qwen3-32B	95/37	16.8	17.9	+1.1 [-11.6, 13.3]	1.000	1.000
Qwen3-32B → GPT-4o-mini	75/31	17.3	9.3	-8.0 [-20.0, 4.4]	0.313	0.626

(Appendix B). Each archived core label therefore reflects essentially one realization of the model at that prefix, not a sample from a distribution, and the per-seam failure rates should be read as such. The matched controls are unaffected: both arms are fresh requests issued in the same collection wave with identical settings, the cross-minus-self contrast is taken within each prefix, and all uncertainty is estimated over source tasks, which is the only level at which the design provides replication.

6.2 Matched restart controls

On the 261 code anchors, replacing the continuation model never increases failure (Table 4). Sonnet 4.6 restarting its own prefixes fails on 26.4%; GPT-4o-mini continuing the same prefixes fails on 9.9%, a difference of -16.5 points $[-29.9, -3.2]$ ($p = 0.038$, Holm 0.115). GPT-4o-mini to Qwen3-32B changes failure by $+1.1$ points $[-11.6, 13.3]$ and Qwen3-32B to GPT-4o-mini by -8.0 $[-20.0, 4.4]$. Same-model failure rates of 17–26% show that a successful standalone answer does not guarantee a successful restart; the failure belongs to continuation, not to the switch.

Fresh arms share the protocol of Section 3.2 and a common offline oracle. On the 217 anchors that keep dual success under the stronger HumanEval+/MBPP+ tests, the differences are -15.5, -2.4 and -1.6 points (Section 6.7).

6.3 Prospectively selected BigCodeBench tasks

To test the finding on tasks not used during development, we hash-selected 200 BigCodeBench v0.1.4 tasks (Zhuo et al., 2025) from a compatibility-screened pool before observing any outcome. Native references passed in the pinned environment for 187 tasks; 90 then passed dual-success checks and supplied 517 anchors, each receiving one self and one cross continuation.

Table 5: Prospectively selected BigCodeBench validation: task failures on fresh same-prefix continuations. Each direction covers 89 source tasks (90 in union), with one request per arm and anchor. Differences are cross minus self in percentage points; intervals resample source tasks.

Direction	Anchors	Self fail	Cross fail	Δ [95% CI]
GPT-4o-mini → Qwen	259	43/259	37/259	-2.3 $[-8.3, +3.5]$
Qwen → GPT-4o-mini	258	28/258	36/258	$+3.1$ $[-3.1, +9.6]$

Both cross-minus-self differences span zero (Table 5): -2.3 points for GPT-4o-mini to Qwen and $+3.1$ for the reverse. Same-model continuations fail on 43/259 and 28/258 anchors. All 1,034 outcomes and 517 judge scores are known.

Of 73 task-failing cross outputs, 27 fail public structural checks and 46 pass them. Rules catch none of the 46; round-trip and hybrid each catch ten (Table 6). Across 444 task-passing outputs, rules reject one and round-trip/hybrid reject five. The judge therefore adds ten detections for four false rejections, and misses 36 residual failures. Native tests include library and output-contract requirements; Appendix B.7 details them.

6.4 Controlling judge information

We judge the same 517 outputs again under a 2×2 design: task description shown or hidden, numeric cut position shown or hidden. All conditions use the full combined code, one common

Table 6: BigCodeBench cross-continuation detection at threshold 0.5. Fractions show rejected task failures (all and structural-pass only) and rejected task passes. Round-trip and hybrid make identical decisions on this sample.

Direction	Detector	All failures	Residual failures	Rejected passes
GPT-4o-mini → Qwen	Rules	12/37	0/25	1/222
GPT-4o-mini → Qwen	Round-trip / hybrid	19/37	7/25	3/222
Qwen → GPT-4o-mini	Rules	15/36	0/21	0/222
Qwen → GPT-4o-mini	Round-trip / hybrid	18/36	3/21	2/222

prompt, Sonnet 4.6 at temperature zero and a 256-token cap; hidden tests and references are never supplied.

Table 7: Matched judge-information control on the same 517 BCB cross outputs. All entries are rounded missing-outcome bounds, not confidence intervals or primary estimates; singleton bounds are retained. Recall and FUR use 73 failures and 444 passes.

Task	Boundary	Known	Recall (%)	FUR (%)	AUC
hidden	hidden	516/517	$[49.3, 49.3]$	$[0.90, 1.13]$	$[0.8670, 0.8693]$
hidden	shown	516/517	$[50.7, 50.7]$	$[0.90, 1.13]$	$[0.8579, 0.8602]$
shown	hidden	516/517	$[53.4, 54.8]$	$[0.68, 0.68]$	$[0.8840, 0.8977]$
shown	shown	517/517	$[53.4, 53.4]$	$[0.90, 0.90]$	$[0.9008, 0.9008]$

All 2,068 judgments were attempted; three are unknown, so Table 7 reports ranges over the unknowns rather than point estimates. Showing the task raises recall by 2.7–5.5 points in both boundary conditions and raises AUC. Showing the boundary lowers AUC when the task is hidden ($0.867 \rightarrow 0.858$) and raises it when the task is shown (0.884–$0.898 \rightarrow 0.901$). Information access is a first-order design choice for handoff detectors. Appendix B.8 gives all contrasts.

6.5 Error composition

Public structural violations account for 520 of the 569 unsafe seams (91.4%). The remaining 49 pass parsing and entry-point checks and comprise 37 task-test failures from 30 tasks, 11 append-contract violations whose tests pass, and one unresolved timeout. Two of the 37 fail with undefined names during module execution. Resolved replays agree with the archived labels.

A failure that compiles. In HumanEval/6, a continuation places `elif char == ')'` inside the `char == '('` branch, beneath an inner maximum-depth comparison. The program compiles but never decrements depth for closing parentheses. On `'(()()) ((())) () ((())()())'` it returns `[3,3,1,5]` instead of `[2,3,1,3]`: a task-test failure invisible to structural validation (replay trace in Appendix B).

6.6 Detector performance

Against the append labels, rules have zero false rejections in every direction and FSR from 2.2% to 42.9% (Table 9). The judge trades the two: in the two large directions its FSR is 12.1% and 1.0% against rules' 25.9% and 2.6%, but its FUR is 2.9% and 22.4%; in the GPT-4o-mini to Sonnet direction it rejects 42.2% of correct outputs while matching rules' FSR. Rules AUC ranges 0.786–0.989 and round-trip 0.779–0.993.

After structural checks pass, the comparison sharpens (Table 10). Round-trip detects 8 of 15 and 6 of 10 residual task failures in the two largest directions, but 1 of 12 across the four smaller ones. Rules detect none. Small failure counts make these per-direction intervals wide; the aggregate is 15 of 37.

Which detector is better depends on the false-rejection budget (Figure 1). Enumerating all whole-tie thresholds, the hybrid attains 94.9% append recall at $\leq 1\%$ FUR against 93.5% for rules and 92.1% for round-trip; at $\leq 5\%$ these become 95.8, 93.5 and 94.2. At the fixed 0.5 threshold, adding the

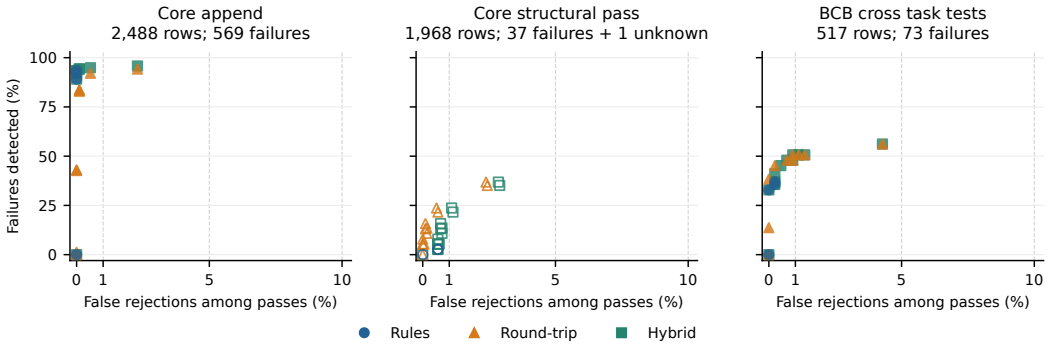

Figure 1: Attainable detection–false-rejection operating points in the 0–10% FUR region for every whole-tie threshold. Open symbols in the middle panel show both labels of one unresolved timeout.

judge to rules catches 15 more failures and rejects 321 more valid outputs on the core, versus 10 and 4 on BigCodeBench. On structurally valid BCB outputs the judge's attainable recall stays at 30.4% under a 10% budget. Full tables are in Appendix B.2.

Strict decoding leaves all round-trip scores unchanged. The all-at-once prompt returns invalid responses on 3/91, 3/95 and 1/75 code seams in the three added cohorts; on the valid rows its AUC is 0.706, 0.696 and 0.865, against 0.882, 0.906 and 0.743 for round-trip on the same rows. The two prompts differ in task access, segmentation and truncation, so this compares implementations rather than isolating boundary visibility.

6.7 Sensitivity to stronger code tests

Using cached HumanEval+/MBPP+ tests (Liu et al., 2023), we re-evaluate 219 of the 428 tasks (1,606 seams) whose specifications, interfaces and enhanced references match. Success now requires both suites; we recheck both standalones to define enhanced dual-success D^+ without changing archived labels. The added tests expose 75 seams that passed the original tests, but 63 lose D^+ because a standalone answer also fails. The remaining 12 failures span eight tasks and pass structural validation; rules detect none and round-trip detects four. In the full structural-pass D^+ subset (32 known failures, 1,150 passes, one unknown), round-trip catches 16 failures but rejects 231 passes; rules catch one and reject five. Appendix B.5 reports selection and uncertainty.

6.8 Exploratory interventions and ablations

On the 24 target tasks of a 32-task BCB development pilot, a cursor-state reminder improved on a generic reminder but not on the original prompt or on original plus syntax-only repair; it failed the predeclared gate and we stopped this method family (Appendix B.9). Withholding the numeric boundary cue changes round-trip AUC in opposite directions across the two Qwen cohorts. Appendix B.4 reports feature-removal ablations and historical fallback-strength comparisons.

7 Discussion and limitations

What the benchmark shows. The switching question and the acceptance question have different answers. Replacing the continuation model did not raise failure in any of five matched directions across two task collections; in one direction it lowered failure by 16 points because the replacement model was simply better at finishing that prefix. The acceptance question is where the difficulty lies: continuation fails on a fifth of core seams, public structural checks recover 91% of that, and the last 9% is where detectors diverge. Rules never reject a correct output but never catch a structurally valid failure; the content-only judge catches two in five of them at the cost of 2–42% false rejections depending on direction. The controlled judge experiment shows that the task description is worth more to the judge than the cut position, and that the cut position helps only once the task is known.

Aggregate AUC hides all of this: the budget-conditional envelope reverses the ranking of rules and judge between the 1% and 5% false-rejection budgets.

Coverage. The core covers three model pairs with fresh restart controls in three directions and reuses tasks that were available during development; the BigCodeBench validation adds 90 prospectively selected tasks but one model pair. Pretraining exposure to HumanEval, MBPP and BigCodeBench is possible and would, if anything, make standalone success easier and continuation failure more surprising. Repeated requests to hosted endpoints do not guarantee independent sampling, and gateway aliases do not guarantee immutable backends; we record returned model identities.

Validity. Unit tests are incomplete specifications (Liu et al., 2023): the stronger HumanEval+/MBPP+ suites expose 75 additional failures, and 63 of them also disqualify a standalone answer, which shrinks the eligible set rather than changing the direction of any result. Enhanced tests cover 219 of 428 tasks. BigCodeBench tests include environment-sensitive behaviour and output requirements not always explicit in the prompt (Appendix B.7); one random-value-sensitive test outside the eligible set prevented exact full replay without affecting the 90 eligible tasks. Results concern Python correctness under executable tests, not general safety. Generated code is untrusted and needs a sandbox stronger than subprocess limits.

Release. All 2,488 seams, 261 matched anchors, 517 BigCodeBench anchors, 2,068 judge records, labels, archived detector scores and offline scorers are released; the source tasks are MIT and CC-BY-4.0. A same-host network-disabled Linux container reproduces every reported number from the released files. An earlier JSON Schema pilot (6,937 seams from JSONSchemaBench) is withheld: its only oracle is schema validity, and the per-schema upstream licenses of its GitHub-sourced tasks could not be established (Appendix A).

8 Conclusion

SeamBench evaluates append-only code handoffs on 2,488 seams from 428 tasks with executable labels. Replacing the continuation model does not increase failure in any of five matched directions; continuation itself fails on 10–26% of restarts, 91% of failures are structural and visible to rules, and the residual semantic failures are caught by a content-only judge less than half the time at a false-rejection cost that varies tenfold across directions. Giving the judge the task helps more than telling it where the switch occurred. The benchmark, labels and scorers are released under MIT and CC-BY-4.0, so these results and new detectors can be checked on the same seams.

AI USE STATEMENT

An AI coding and writing assistant (Codex) was used for part of the code design and implementation, for literature and licensing searches, and for drafting and revising the manuscript. The experimental LLM generation and judging procedures are described separately in Sections 3–6 and Appendix C. The author takes responsibility for the final text, conclusions and artifacts, including AI-assisted work.

ETHICS STATEMENT

The study uses public programming tasks and model-generated code, with no human-participant evaluation. Labels concern correctness under executable tests, not general safety. Source datasets are MIT (HumanEval) and CC-BY-4.0 (MBPP, BigCodeBench Apache-2.0); their notices are retained in the release. Generated code is untrusted and was executed under resource limits; users should apply a stronger sandbox. An earlier JSON Schema pilot is withheld because per-schema upstream licenses could not be established (Appendix A).

REPRODUCIBILITY STATEMENT

Sections 3–6 specify the evaluation units, controls and metrics. Appendices B and C document recorded settings, validation and exact prompt construction. The supplementary package contains all 2,488 seams with labels and archived detector scores, the 261 matched anchors, the 517 BigCodeBench anchors with detector inputs and labels, the 2,068 fair-judge records, and offline scorers that recompute every table; a network-disabled container reproduced them from the package alone. Appendix A documents the release.

REFERENCES

Yash Akhauri, Anthony Fei, Chi-Chih Chang, Ahmed F. AbouElhamayed, Yueying Li, and Mohamed S. Abdelfattah. SplitReason: Learning to offload reasoning. *arXiv preprint arXiv:2504.16379*, 2025. URL https://arxiv.org/abs/2504.16379.

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. SantaCoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023. URL https://arxiv.org/abs/2301.03988.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022. URL https://arxiv.org/abs/2207.14255.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding LLMs the right way: Fast, Non-Invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2403.06988.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a. URL https://arxiv.org/abs/2302.01318.

Lingjiao Chen, Matei Zaharia, and James Zou. How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009*, 2023b. URL https://arxiv.org/abs/2307.09009.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023c. URL https://arxiv.org/abs/2305.05176.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to Self-Debug. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2304.05128.

Zhuokun Chen, Zeren Chen, Jiahao He, Lu Sheng, Mingkui Tan, Jianfei Cai, and Bohan Zhuang. R-Stitch: Dynamic trajectory stitching for efficient reasoning. *arXiv preprint arXiv:2507.17307*, 2025. URL https://arxiv.org/abs/2507.17307.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware query routing. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2404.14618.

Tuan Dinh, Jinman Zhao, Samson Tan, Renato Negrinho, Leonard Lausen, Sheng Zha, and George Karypis. Large language models of code fail at completing code with potential bugs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/819cebb05f993840e8a52d7564c5c282-Abstract-Conference.html.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024. URL https://arxiv.org/abs/2411.15100.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. InCoder: A generative model for code infilling and synthesis. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2204.05999.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. R2R: Efficiently navigating divergent reasoning paths with small-large model token routing. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. URL https://arxiv.org/abs/2505.21600.

Roy Ganz, Mor Shpigel Nacson, Adi Kalyanpur, and Ron Litman. The handoff tax: Continuing non-native trajectories in LLM agents. *arXiv preprint arXiv:2608.24358*, 2026. URL https://arxiv.org/abs/2608.24358v1.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021. URL https://cacm.acm.org/research/datasheets-for-datasets/.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-Constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://arxiv.org/abs/2305.13971.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025. URL https://arxiv.org/abs/2501.10868.

Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2308.08493.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on LLM-as-a-Judge. *arXiv preprint arXiv:2411.15594*, 2024. URL https://arxiv.org/abs/2411.15594.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2105.09938.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2403.07974.

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 *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06770.

Raad Khraishi, Iman Zafar, Katie Myles, and Greig A. Cowan. Evaluating performance drift from model switching in multi-turn LLM systems. In *ICLR 2026 Workshop: Catch, Adapt, and Operate: Monitoring ML Models Under Drift*, 2026. URL https://openreview.net/forum?id=Bhi0hiRxX2.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2211.11501.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2211.17192.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2305.06161.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2305.01210.

Leo Lu, Jonathan Zhang, Sean Chua, Spencer Kim, Kevin Zhu, Sean O'Brien, and Vasu Sharma. Reasoning Relay: Evaluating stability and interchangeability of large language models in mathematical reasoning. *arXiv preprint arXiv:2512.20647*, 2025. URL https://arxiv.org/abs/2512.20647.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024. URL https://arxiv.org/abs/2411.00640.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019. URL https://arxiv.org/abs/1810.03993.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is Self-Repair a silver bullet for code generation? In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2306.09896.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2406.18665.

Vishal Pandey and Gopal Singh. ContinuityBench: A benchmark and systems study of stateful failover in multi-provider LLM routing. *arXiv preprint arXiv:2607.15899*, 2026. URL https://arxiv.org/abs/2607.15899.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2404.13076.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, pp. 48371–48392, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023. URL https://arxiv.org/abs/2308.12950.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023. URL https://arxiv.org/abs/2310.18018.

Jiwon Song, Yoongon Kim, and Jae-Joon Kim. RelayGen: Intra-generation model switching for efficient reasoning. *arXiv preprint arXiv:2602.06454*, 2026. URL https://arxiv.org/abs/2602.06454.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022. URL https://arxiv.org/abs/2204.06092.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on performance of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2024. URL https://arxiv.org/abs/2408.02442.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.12784.

Weixi Tong and Tianyi Zhang. CodeJudge: Evaluating code generation with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://arxiv.org/abs/2410.02184.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2305.17926.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023. URL https://arxiv.org/abs/2307.09702.

Nuoya Xiong, Yuhang Zhou, Hanqing Zeng, Zhaorun Chen, Furong Huang, Shuchao Bi, Lizhu Zhang, and Zhuokai Zhao. Token-Level LLM collaboration via FusionRoute. *arXiv preprint arXiv:2601.05106*, 2026. URL https://arxiv.org/abs/2601.05106.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.05685.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P. Xing, Hongyi Wang, and Huaxiu Yao. CITER: Collaborative inference for efficient large language model decoding with Token-Level routing. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2502.01976.

Terry Yue Zhuo. ICE-Score: Instructing large language models to evaluate code. In *Findings of the Association for Computational Linguistics: EACL 2024*, 2024. URL https://arxiv.org/abs/2304.14317.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a6a90bcc2aa470c3871b2d39a67d26e8-Abstract-Conference.html.

A Dataset documentation and release

This documentation follows datasheets for datasets (Gebru et al., 2021) and model cards (Mitchell et al., 2019).

Purpose and unit. The resource supports evaluation of raw append-only code continuations after a simulated interruption. Each core row identifies a source task, two recorded test-passing standalone generations, a cut, a visible prefix, a continuation by the other model, and an archived append-contract label; task-only and structural labels are separate overlays. There are 2,488 unique core

rows from 428 source tasks; repeated cuts of one task are one cluster for inference. The 261 matched-control anchors are the seams of the three added cohorts and add no rows. The BigCodeBench validation contributes 517 anchors from 90 tasks and the earlier MBPP follow-up 14 anchors from four tasks; neither is added to the core.

Sources and selection. Tasks are HumanEval (revision `7dce6050`) and MBPP (revision `4bb6404f`) as published on Hugging Face; BigCodeBench uses the public v0.1.4 snapshot. Source identifiers, split, revision, contract, adapter configuration and hashes accompany every row. Eligibility requires both models' recorded standalone answers to pass the task tests, which selects a conditional population and does not estimate production outage frequency. The base benchmarks may appear in model pretraining data.

Collection and processing. Generations use configured API endpoints, versioned prompts (Appendix C), temperature zero, a 4,096-token output cap and recorded seeds. A single outer Markdown fence may be removed from a standalone answer before validation; prefixes are cut from that normalized answer, and continuations are appended without repair. API failures and unresolved execution timeouts remain unknown rather than being counted as failures.

Validation and exclusions. The archived label combines task validation with append-contract checks; overlays expose public structural validity and task-only execution separately. Cached HumanEval+/MBPP+ tests form a partial overlay on 219 tasks. The following pilots are excluded from the benchmark: a JSON Schema pilot of 6,937 seams from 965 JSONSchemaBench tasks (no semantic oracle beyond schema validity; 435 of its 435 failures were schema violations; upstream per-schema licenses unresolved for 597 GitHub-sourced tasks), a prose pilot on ASQA, a 59-example tool-call pilot on BFCL with no failures, synthetic defect injections, and a legacy 164-row code pilot. Statistics for the JSON pilot are available on request as text-free numeric records only.

Intended use. Compare detectors on the same fixed rows; study sensitivity to validators, direction and information access; measure switching cost against matched restarts. The core was used during detector development, so learned detectors should hold out by source task. The benchmark is not a safety certification and does not estimate real failure prevalence.

Distribution and rights. The supplementary package contains: (i) all 2,488 core seams as detector inputs (public task, entry-point contract, exact combined output, cut offset) with labels and archived detector scores stored separately; (ii) the 261 matched anchors with self and cross outcomes; (iii) the 517 BigCodeBench anchors with detector inputs, labels and 1,034 self/cross outcomes; (iv) all 2,068 fair-judge records; (v) offline scorers, validators and tests that recompute every table in this paper with the standard library and NumPy. HumanEval is MIT, MBPP is CC-BY-4.0 and BigCodeBench is Apache-2.0; their license texts, dataset cards and an attribution note are included. Project software is Apache-2.0. Model outputs were generated through Amazon Bedrock (`us-east-1`; Claude Sonnet 4.6, Qwen3-32B) and an OpenAI-compatible gateway (GPT-4o-mini, GPT-5 mini); the applicable provider terms assign output interest to the customer and impose no publication pre-approval, and AWS Service Terms §1.8 require that reproducibility information accompany publication and be provided to AWS, which this appendix and the package satisfy.[1] A credential and identity scan of the extracted package found no hits.

Scorers. The core scorer exposes two endpoints: archived append labels on all 2,488 IDs, and task-test labels on the 1,968 structurally valid seams (the filter is applied by the scorer; all 2,488 IDs must be supplied). The BCB scorer requires the 517 cross IDs. Scores are finite numbers in $[0, 1]$ or explicit nulls; omitted, duplicate and foreign IDs are rejected. Unresolved labels are carried as ranges. An optional source-task bootstrap is available but disabled by default; the intervals in this paper come from the frozen analysis scripts, which the package also includes.

Maintenance. Versions are immutable and hashed. Corrections are issued as new versions with a recorded reason. Problems should be reported against `example_id` values.

[1] https://aws.amazon.com/service-terms/, checked 2026-09-16.

B ADDITIONAL RESULTS AND REPRODUCTION

All tables are generated from the released numeric records by the scripts in the supplementary package; construction counts come from the frozen eligibility manifests. Model/request failures, invalid judge responses, unresolved labels and source-task denominators are retained in the records. Paths below are relative to the package root.

B.1 MODEL AND REGION PROVENANCE

Table 8: Recorded model identities and the scope of their supporting evidence.

Model	Requested / returned ID	Observed in
GPT-5 mini	*Requested:* `openai/gpt-5-mini` *Returned:* `gpt-5-mini-2025-08-07`	Older core forward and reverse cohorts
GPT-4o mini	*Requested:* `openai/gpt-4o-mini` *Returned:* `gpt-4o-mini-2024-07-18`	Matched self/cross wave; 614 successful ledger entries
Claude Sonnet 4.6	*Requested:* `bedrock/us.anthropic.claude-sonnet-4-6` *Returned:* `us.anthropic.claude-sonnet-4-6`	Matched self/cross wave; 239 successful ledger entries
Qwen3-32B	*Requested:* `bedrock/qwen.qwen3-32b-v1:0` *Returned:* `qwen.qwen3-32b-v1:0`	Matched self/cross wave; 405 successful ledger entries

Returned IDs are gateway-reported observations for the stated scope; they do not attest an immutable backend revision or establish identities for other cohorts. The older core adapter did not retain response fingerprints. BCB route allowlists use the same three non-GPT-5 strings above; these planned constraints are distinct from observations. Bedrock calls were issued from `us-east-1` (author-confirmed historical configuration); Sonnet used the US geographic inference profile, and per-request execution regions were not recorded. Relative evidence paths and SHA-256 bindings accompany the table in `provenance/model_routes.json`.

Historical GPT-5 mini parameters. In the older core forward and reverse cohorts, the adapter changed requested temperature 0 to effective temperature 1. Preserved GPT-5 mini cache entries record temperature 1, a 4,096-token output cap, and request seed 42; original generation metadata can still show the requested temperature 0. This correction applies to those two historical cohorts, not to the later matched or BCB experiments. The adapter transmitted and cached non-null seeds. It could transmit optional request metadata but did not retain that metadata in the cache or request hash. Actual backend versions and fingerprints cannot be reconstructed from the stored request aliases.

B.2 FULL DETECTOR RESULTS AND OPERATING BUDGETS

Tables 9 and 10 give fixed-threshold estimates with source-task bootstrap intervals for every direction. The tradeoff analysis enumerates every distinct risk score as a threshold, including 0.5 and accept-all, rejecting scores at or above the threshold without splitting ties. The hybrid retains its 0.95 rule short circuit and otherwise takes the larger of rule and round-trip score. The 1%, 5% and 10% false-rejection budgets were fixed before enumeration; at each budget we report the greatest attainable recall among thresholds whose worst-case FUR satisfies it, carrying the one unknown label as a range. These envelopes describe the inspected cohorts.

B.3 MATCHED CONTROLS

Both fresh arms use temperature zero, seed 20260911 and a 4,096-token output cap; hash-randomized dispatch interleaves arms within task/cut blocks, and same-wave duplicates share a cached response. Task-only and append-contract outcomes coincide on every code anchor, so Table 4 reports one endpoint per direction and the Holm family has three tests.

Table 9: Detector performance on all 2,488 code seams against the archived append-contract gold, by direction. FSR and FUR are percentages (lower is better); AUC uses unsafe as positive. Brackets are pointwise 95% source-task-clustered bootstrap intervals. The hybrid uses the recorded 0.95 rule short-circuit and the same round-trip scores.

Detector	FSR (%)	FUR (%)	AUC
Sonnet 4.6 → GPT-5-mini: 688 seams, 270 tasks, 58 unsafe			
Rules	25.9 [14.5, 37.7]	0.0 [0.0, 0.0]	0.871 [0.811, 0.927]
Round-trip	12.1 [3.6, 22.6]	2.9 [1.6, 4.3]	0.953 [0.913, 0.986]
Hybrid	12.1 [3.6, 22.6]	2.9 [1.6, 4.3]	0.957 [0.915, 0.990]
GPT-5-mini → Sonnet 4.6: 1,190 seams, 417 tasks, 383 unsafe			
Rules	2.6 [0.8, 4.9]	0.0 [0.0, 0.0]	0.987 [0.976, 0.996]
Round-trip	1.0 [0.0, 2.4]	22.4 [19.1, 25.7]	0.993 [0.984, 0.999]
Hybrid	1.0 [0.0, 2.4]	22.4 [19.1, 25.7]	0.994 [0.985, 1.000]
GPT-4o-mini → Sonnet 4.6: 349 seams, 130 tasks, 93 unsafe			
Rules	2.2 [0.0, 5.9]	0.0 [0.0, 0.0]	0.989 [0.971, 1.000]
Round-trip	2.2 [0.0, 5.9]	42.2 [35.4, 49.4]	0.984 [0.957, 1.000]
Hybrid	2.2 [0.0, 5.9]	42.2 [35.4, 49.4]	0.984 [0.957, 1.000]
Sonnet 4.6 → GPT-4o-mini: 91 seams, 36 tasks, 10 unsafe			
Rules	30.0 [0.0, 66.7]	0.0 [0.0, 0.0]	0.850 [0.667, 1.000]
Round-trip	30.0 [0.0, 66.7]	4.9 [1.2, 9.8]	0.883 [0.717, 1.000]
Hybrid	30.0 [0.0, 66.7]	4.9 [1.2, 9.8]	0.883 [0.717, 1.000]
GPT-4o-mini → Qwen3-32B: 95 seams, 37 tasks, 18 unsafe			
Rules	22.2 [0.0, 45.5]	0.0 [0.0, 0.0]	0.889 [0.773, 1.000]
Round-trip	16.7 [0.0, 39.2]	10.4 [4.3, 17.1]	0.907 [0.782, 1.000]
Hybrid	16.7 [0.0, 39.2]	10.4 [4.3, 17.1]	0.908 [0.781, 1.000]
Qwen3-32B → GPT-4o-mini: 75 seams, 31 tasks, 7 unsafe			
Rules	42.9 [0.0, 100.0]	0.0 [0.0, 0.0]	0.786 [0.500, 1.000]
Round-trip	42.9 [0.0, 100.0]	2.9 [0.0, 7.6]	0.779 [0.486, 1.000]
Hybrid	42.9 [0.0, 100.0]	2.9 [0.0, 7.6]	0.779 [0.486, 1.000]

B.4 Validation and detector ablations

Requiring both raw (unnormalized) standalone answers to pass excludes seams whose Qwen standalone answer carried an outer Markdown fence; this changes the eligible population, not any label.

Removing validator features weakens rules, and round-trip recovers signal over that restricted baseline. This alone cannot establish independent semantic detection. Withholding the numeric boundary cue changes AUC by $+0.0154$ (95% CI $[0.0021, 0.0306]$) for mini-to-Qwen, but -0.0182 ($[-0.0442, 0.0012]$) in reverse. The first direction's task-cluster FSR test has raw $p = 0.0078$ and Holm-adjusted $p = 0.414$ in the 106-test family; it is not an AUC test. A numeric offset does not ensure that the judge locates the split. Historical exact-prefix comparisons contain 205 and 219 prefixes, but every pair differs in generation seed. These and fallback-strength plots remain exploratory, not a causal scaling law.

B.5 Enhanced code-test details

The overlay pins cached HF HumanEval+/MBPP+ scripts and metadata by hash; it is not an evaluation with the latest official EvalPlus CLI. Of 428 code tasks, 124 are missing from these snapshots and 85 have unresolved specification or interface differences. Reference implementation changes alone do not exclude tasks. Mapping and execution rules were fixed before this replay, after the original experiments. No model calls were made.

Across 2,488 code seams, 1,453 retain definite D^+ eligibility, 129 lose it, and 906 remain unresolved. This selection uses only the two original standalone recordings, never fresh self/cross outcomes. The 75 added failures all pass their base replay, so none is explained by a base-test label reversal. The structural-pass D^+ subset has 1,183 seams from 197 tasks; the 12 additional failures come from eight tasks, not 12 independent replications.

Table 10: Task-only detection on code seams that pass the public structural validators. Headings give seams/tasks and unsafe seams. FSR/FUR are percentages with pointwise 95% task-bootstrap intervals. † gives the finite-cohort range over both labels of one unresolved timeout. Residual failures are few; intervals are correspondingly wide.

Detector	FSR (%)	FUR (%)	AUC
Sonnet 4.6 → GPT-5-mini: 657/269; unsafe 15–16			
Rules	[93.3, 93.8]†	[1.7, 1.7]†	[0.523, 0.525]†
Round-trip	[43.8, 46.7]†	[4.5, 4.7]†	[0.826, 0.838]†
Hybrid	[43.8, 46.7]†	[4.5, 4.7]†	[0.817, 0.829]†
GPT-5-mini → Sonnet 4.6: 817/391; unsafe 10			
Rules	100.0 [100.0, 100.0]	0.0 [0.0, 0.0]	0.500 [0.500, 0.500]
Round-trip	40.0 [0.0, 66.7]	22.4 [19.3, 25.7]	0.764 [0.596, 0.969]
Hybrid	40.0 [0.0, 66.7]	22.4 [19.3, 25.7]	0.764 [0.596, 0.969]
GPT-4o-mini → Sonnet 4.6: 258/120; unsafe 2			
Rules	100.0 [100.0, 100.0]	0.0 [0.0, 0.0]	0.500 [0.500, 0.500]
Round-trip	100.0 [100.0, 100.0]	42.2 [35.1, 49.2]	0.275 [0.240, 0.311]
Hybrid	100.0 [100.0, 100.0]	42.2 [35.1, 49.2]	0.275 [0.240, 0.311]
Sonnet 4.6 → GPT-4o-mini: 84/35; unsafe 3			
Rules	100.0 [100.0, 100.0]	0.0 [0.0, 0.0]	0.500 [0.500, 0.500]
Round-trip	100.0 [100.0, 100.0]	4.9 [1.2, 9.8]	0.609 [0.431, 0.932]
Hybrid	100.0 [100.0, 100.0]	4.9 [1.2, 9.8]	0.609 [0.431, 0.932]
GPT-4o-mini → Qwen3-32B: 81/36; unsafe 4			
Rules	100.0 [100.0, 100.0]	0.0 [0.0, 0.0]	0.500 [0.500, 0.500]
Round-trip	75.0 [0.0, 100.0]	10.4 [4.2, 17.1]	0.589 [0.426, 1.000]
Hybrid	75.0 [0.0, 100.0]	10.4 [4.2, 17.1]	0.584 [0.421, 1.000]
Qwen3-32B → GPT-4o-mini: 71/30; unsafe 3			
Rules	100.0 [100.0, 100.0]	0.0 [0.0, 0.0]	0.500 [0.500, 0.500]
Round-trip	100.0 [100.0, 100.0]	2.9 [0.0, 7.6]	0.485 [0.463, 0.500]
Hybrid	100.0 [100.0, 100.0]	2.9 [0.0, 7.6]	0.485 [0.463, 0.500]

Table 11: Attainable recall (%) under preselected false-rejection budgets, enumerating all whole-tie thresholds on the inspected data. Budgets apply to the worst-case false-rejection rate over unknown labels; ranges are identification bounds.

Population	Detector	FUR $\leq 1\%$	$\leq 5\%$	$\leq 10\%$
Core append	rules	93.5	93.5	93.5
	round-trip	92.1	94.2	94.2
	hybrid	94.9	95.8	95.8
Core structural pass	rules	2.6–2.7	2.6–2.7	2.6–2.7
	round-trip	21.6–23.7	35.1–36.8	35.1–36.8
	hybrid	13.5–15.8	35.1–36.8	35.1–36.8
BCB cross	rules	37.0	37.0	37.0
	round-trip	50.7	56.2	56.2
	hybrid	50.7	56.2	56.2
BCB structural pass	rules	0.0	0.0	0.0
	round-trip	21.7	30.4	30.4
	hybrid	21.7	30.4	30.4

The matched subset retains 217 anchors, excludes 12, and leaves 32 with uncertain eligibility. Task-only and append-contract outcomes coincide on the definite subset. Table 13 distinguishes its original-label contrast from its enhanced-test contrast. Intervals describe this observed subset only. Conservative task-only bounds allowing unknown eligibility include zero in all three code directions; these bounds are not confidence intervals. Full fixed-anchor bounds and both endpoints are retained in the analysis JSON.

Table 12: Paired changes from rules at threshold 0.5. +TP/−TP are additional/lost correct rejections; +FP/−FP are additional/avoided false rejections of passing outputs. Ranges retain one unknown core task label.

Population	Detector	+TP	−TP	+FP	−FP
Core append	round-trip	15	0	321	0
	hybrid	15	0	321	0
Core structural pass	round-trip	14–15	0	321–322	0
	hybrid	14–15	0	321–322	0
BCB cross	round-trip	10	0	4	0
	hybrid	10	0	4	0
BCB structural pass	round-trip	10	0	4	0
	hybrid	10	0	4	0

Table 13: Code-only enhanced-test sensitivity. D^+, D^- and $D^?$ denote eligible, ineligible and unresolved anchors; n/t counts anchors/source tasks. Both Δ columns use the same D^+ subset. Differences are cross-minus-self task-failure risk in pp. The final column admits unresolved eligibility and outcomes: these task-only outer bounds are not confidence intervals. JSON is unchanged.

Direction	D^+ (n/t)	$D^-/D^?$	Original Δ on D^+	Enhanced Δ on D^+ [95% CI]	Full eligible outer bounds
Sonnet → mini	71/29	6/14	-16.9	-15.5 [-30.3, -1.3]	[-26.8, 1.2]
mini → Qwen	83/32	3/9	-3.6	-2.4 [-14.5, 9.4]	[-12.0, 7.6]
Qwen → mini	63/26	3/9	-1.6	-1.6 [-12.5, 9.7]	[-13.9, 11.1]

B.6 Earlier MBPP follow-up

Of 49 source-mapped MBPP candidates, screening for prior local use, duplicates, specification problems and reference-test failures admitted 10 tasks (seven train, three validation) before generation. The task list, prompts, thresholds and analysis were frozen; no outcome-based replacement was allowed. Both models passed structure, base and enhanced tests on five tasks; four supplied 14 anchors. Cuts targeted 0.25/0.5/0.75 with at least three nonempty lines per side; coincident offsets were merged.

Each arm used two separate requests at temperature 0 and seed 20260911. We do not assume independent draws: 22 of 28 anchor/arm groups returned identical text. All 104 calls succeeded: 20 standalones, 56 continuations and 28 judgments; all continuation labels and judge scores were known. Judges saw only combined code under the unchanged boundary-hidden prompt. The frozen score policy accepts parsed, finite, non-boolean numeric risks in $[0, 1]$; invalid scores remain unknown, not imputed. It retains the original JSON extraction; ancillary-field errors are flagged without discarding a valid risk score. The rejection threshold remains 0.5 and the hybrid short circuit 0.95. Candidate tests run in the locked, non-root, network-disabled Docker image.

Paired estimates retain the frozen algorithm: average draws within each anchor, weight anchors equally, and bootstrap global source tasks 5,000 times with seed 20260911. Pointwise intervals use four task clusters overall and only two in the reverse direction; no new p-values are added. Round-trip versus rules recall was 11/15 versus 7/15 (mini→Qwen) and 2/2 versus 0/2 (reverse); hybrid added no gain.

The two failure differences are $+72.2$ and -20.0 percentage points, but only four source tasks contribute (two in reverse). Among structurally valid task failures, round-trip catches 4/8 and 2/2; rules catch none. All detectors reject 0/11 passing outputs, too few to establish a low false-rejection rate.

Table 14: Additional-task matched controls. Counts are failures/requests; failure means not passing both test suites. Δ is cross minus self in percentage points. Two requests per arm are averaged within each cut. Pointwise 95% task-bootstrap intervals use four tasks overall, only two in reverse; repeated requests are not independent tasks.

Direction	Tasks/cuts	Self failures	Cross failures	Δ [95% CI] (pp)
mini $\rightarrow$ Qwen	4/9	2/18	15/18	$+72.2$ $[38.9, 100.0]$
Qwen $\rightarrow$ mini	2/5	4/10	2/10	-20.0 $[-33.3, 0.0]$

B.7 BIGCODEBENCH VALIDATION DETAILS

The public v0.1.4 snapshot was screened for supported static imports, network-dependent libraries, native test classes, adapter-incompatible namespace patching, prior local use, and exact or close task duplicates. From this environment-limited pool, 200 source tasks were hash-selected before reference execution. Thirteen reference failures were retained as attrition, without replacement. The remaining 187 tasks each received one fresh standalone answer from GPT-4o-mini and Qwen3-32B. Ninety passed structure and native tests for both recordings and supplied 517 distinct cuts across the two directions. These are public tasks, not a claim of pretraining-unseen evaluation.

The unchanged code wrappers, normalization, cut rules, temperature zero, seed 20260911 and 4,096-token generation cap were frozen before collection. Each anchor received one fresh request per arm; no identical requests were coalesced in the collector. Cross outputs received 517 content-only Sonnet judgments, using the score policy in the previous subsection. The task oracle is a single native BCB test suite. Compatibility fields named "base" and "enhanced" store callability and native-test checks, not an original-versus-enhanced-test comparison.

A gateway configuration initially injected tools into a request intended to contain none, causing rejection before any successful task response. A recorded transport amendment disabled tool injection; the rejected assignment was explicitly reissued. Its charge reservation and diagnostic probes remain in the ledger. The completed tool-free phase then made 1,925 successful requests: 374 standalones, 1,034 continuations and 517 judgments. It used no automatic retries or model fallbacks.

Across both arms, 144 outputs fail the task oracle: 63 fail syntax checks, 49 have assertion failures and 32 have runtime errors. A separately versioned diagnostic replay of all 144 failures and 144 successful counterpart/same-task controls reproduced all pass/status/structure decisions, with all 56 involved references passing. This verifies the diagnostic instrument in the same environment, not a second-environment replication. Failure traces are not automatically continuation mechanisms. For example, two BCB/649 failures concern the Series name rather than its values, a requirement not explicit in the prose; BCB/692 exposes an API absent from the pinned pandas version. Labels are retained under the native oracle with these caveats, not silently reclassified.

After these outcomes were inspected for a lexical-state intervention, the BCB tasks became development data for that intervention. Their prospective status applies to the unchanged continuation/detector study, not to a subsequently designed method.

B.8 MATCHED JUDGE-INFORMATION CONTROL

The control reuses all 517 archived BCB cross outputs from 90 source tasks, with unchanged native labels: 73 failures and 444 passes. It adds judgments, not tasks, continuations or oracle executions. For every output, four separate requests cross public-task visibility with numeric-boundary visibility. The complete combined code and the common new template remain fixed; task information includes the public description and entry-point contract. The boundary is a zero-based Unicode character offset, not an inline marker. No hidden tests, reference answers, labels, original suffixes, model directions or earlier judge responses enter the requests.

All requests use the recorded Bedrock Sonnet 4.6 route, temperature zero and a 256-token cap; no seed is sent to the model. A hash-based order (seed 20260914) fixes case blocks and the four-condition order within each block before collection. The new prompt and cap differ from the legacy 517-judgment audit, whose results are retained separately. Only the two information slots vary within this control (Appendix C.4).

Of 2,068 attempted assignments, 2,066 returned responses and 2,065 yielded valid risks. The three unknowns are a malformed JSON response for a task failure in t_1b_0, an HTTP 503 for a task pass in t_0b_1, and a timeout for a task pass in t_0b_0. Two recorded transport-stop amendments continued only never-attempted assignments, in their original order; none of these three observations was retried or replaced. Here t denotes task information and b the boundary, with 1 meaning shown. The parser requires a finite, non-boolean numeric risk in $[0, 1]$ and normal completion; malformed, truncated and failed responses remain unknown. Rejection is fixed at risk ≥ 0.5.

The analysis was frozen before these judgments. Any unknown suppresses all primary point estimates and intervals, including conditions with no missing scores. Thus the planned 5,000 global source-task bootstrap draws (seed 20260914) were not run. We retain full denominators and vary missing risks over $[0, 1]$; AUC bounds include endpoint ties. Contrast bounds follow the frozen signed interval arithmetic. Singleton bounds are still fixed-cohort bounds, not confidence intervals. No complete-case inference, additional p-values or significance-selected contrasts are reported.

Table 15: All planned information contrasts, shown minus hidden. Recall and FUR differences are in percentage points; AUC differences are unscaled. These are rounded fixed-cohort missing-outcome bounds, including singleton bounds, not CIs. Interaction is the boundary effect with task information minus that without it.

Contrast	Recall Δ (pp)	FUR Δ (pp)	AUC Δ
Task, boundary hidden	$[4.11, 5.48]$	$[-0.45, -0.23]$	$[0.0147, 0.0307]$
Task, boundary shown	$[2.74, 2.74]$	$[-0.23, 0.00]$	$[0.0407, 0.0429]$
Boundary, task hidden	$[1.37, 1.37]$	$[-0.23, 0.23]$	$[-0.0114, -0.0068]$
Boundary, task shown	$[-1.37, 0.00]$	$[0.23, 0.23]$	$[0.0031, 0.0168]$
Interaction	$[-2.74, -1.37]$	$[0.00, 0.45]$	$[0.0100, 0.0282]$

Task visibility raises recall by $[4.11, 5.48]$ points with the boundary hidden and $[2.74, 2.74]$ with it shown. Boundary visibility changes AUC by $[-0.0114, -0.0068]$ without task information and $[0.0031, 0.0168]$ with it. These ranges describe this fixed set under this new prompt; they do not establish a general advantage for hiding or revealing a boundary, nor identify why the older prompts differ.

B.9 Cursor-state pilot details

Table 16: Exploratory cursor-state pilot on 32 development tasks. Task passes / evaluated outputs; each task has two fresh requests per self/cross arm. The 24 target tasks are split equally between open strings and pending indented suites. Repair uses the original outputs and no added model calls.

Condition	Open string	Pending suite	Ordinary	All
Original	36/48	39/48	30/32	105/128
Generic reminder	25/48	32/48	29/32	86/128
Reminder + state facts	40/48	27/48	31/32	98/128
Original + fixed repair	42/48	40/48	30/32	112/128

On the 24 target tasks, state facts gain 10.4 points over generic (95% task-bootstrap CI $[-6.3, 27.1]$), but lose 8.3 points to original ($[-25.0, 6.3]$) and 15.6 to original plus repair ($[-31.3, -1.0]$). Repair rescues seven outputs across four tasks without harming a pass in this sample. This is an exploratory baseline observation, not an independently validated method. All conditions use two fresh requests per arm and anchor; repair adds no model requests and is selected using compilation only, never native task tests.

The mechanism audit gives partial observational annotations, not an exhaustive or independent human taxonomy. Of 144 failed outputs, 41 have unbalanced multiline strings at open-string cuts (23 tasks; 20 cross, 21 self), and 16 lack the required indented suite at the cursor (14 tasks; five cross, 11 self). These patterns do not attribute failures causally to model substitution. Versioned notes retain a corrected suffix-local name typo initially misdescribed as a prefix-name conflict.

The frozen selector reads prefix states, not continuation outcomes, and selects one anchor per task by hash: 12 open-string, 12 pending-suite, and eight ordinary-boundary tasks, balanced by primary model within each stratum. Generic and state prompts have the same output cap and call count but different input lengths. Temperature is zero and the recorded seed is 20260911. Of 64 same-condition anchor/arm repeat pairs, 30 original, 29 generic and 31 state pairs return identical text; repeats are not assumed independent. All 384 requests, 512 evaluations and 32 reference gates completed without unknown outcomes.

The repair menu adds a matching string delimiter at either suffix end or shifts the first nonempty line or all nonempty lines by $\pm 4, \pm 8, \pm 12$ spaces. It preserves prefix bytes and leaves already compiling programs unchanged. Fixed edit-cost order selects the first compiling variant; no hidden oracle participates in selection.

Before generation, independent confirmation was made conditional on state-fact gains of at least five points against both generic and original prompts, three points against original plus repair, improved task means on at least five target tasks, three points of structural gain over generic, nonnegative gains in both arms, and at most three points of harm on ordinary controls. All outcomes also had to be known and references consistent. The original and repair comparisons failed these gates. We did not change thresholds, revise the prompt, substitute a winning method, or add tasks. Pilot intervals use equal-task means and 5,000 task-cluster resamples (seed 20260913), not hypothesis tests.

B.10 Package map

The supplementary package is organized as follows; a top-level manifest hashes every file and `check.py` verifies it.

- `core/`: 2,488 detector-input rows (public task, entry-point contract, exact combined output, cut offset), a separate label/score sidecar, source index, validators, the two-endpoint scorer and tests.
- `controls/`: 261 matched anchors with self and cross outcomes and the clustered bootstrap and sign-flip code.
- `bcb/`: 517 detector inputs, separate labels, 1,034 self/cross outcomes, source selection and reference attrition records, scorer and a parse-only example detector.
- `fair\_judge/`: 2,068 judgment records, prompts, protocol and the bound reconstruction.
- `stats/`: the script that regenerates every table and Figure 1 in this paper from the files above, with input hashes; the enhanced-test overlay, MBPP follow-up and cursor pilot records; the HumanEval/6 replay trace.

Scorers read only the label sidecars and never execute task programs; `tools/replay\_oracles.py` reruns the packaged programs in a read-only, non-root, network-disabled container for those who want to re-derive labels.

Independent-environment replay. A fresh Linux aarch64 container with Python 3.12.12 installed locked dependencies from public PyPI and then ran without network access or the original checkout. All tests passed. Fresh oracle execution, raw judge decoding, full-rule evaluation, source-task bootstrap intervals and the paired-control inference reproduced the archived results with no differences beyond absolute 10^{-10} / relative 10^{-9}; categorical labels matched exactly. The code-only tables in this paper were then regenerated from the same records on a second machine and matched.

C Prompt templates

Base prompt templates are versioned and stored under `prompts/`. The manuscript source companion includes the exact template files; request-specific rendered prompts remain in the separate experimental archive. The matched boundary-visible variant is derived in the frozen replication runner, which records its rendered text in the request cache. The version tag is folded into the request hash used for caching (§4), so revising a prompt produces a distinct cache lineage; every generation record in the dataset carries the exact version tag it was produced with.

The listings below include core prompts and explicitly identified exploratory templates. Typography and brace escaping are normalized for display; repository files contain the exact templates used for replay. Most placeholder fields wrapped in braces (`{task_prompt}`, `{visible_prefix}`, etc.) are Python `str.format`-style substitution slots. The matched information control below instead serializes its varying fields as JSON.

C.1 Standalone generation prompts

These prompts drive the standalone-generation stage (§4) where each model produces a full answer to the source task without any prefix constraint.

Code (`standalone_code_v1`).

```
You are solving a Python programming task.

Return only a complete Python source file that implements the requested
↪  function.
Do not use Markdown fences. Do not include an explanation before or
↪  after the code.

TASK
{task_prompt}
```

Excluded pilots. The JSON, prose and tool-call pilots used analogous standalone prompts (`prompts/standalone_\{json,prose,toolcall\}_v1.txt`).

C.2 Continuation prompts

These prompts drive the continuation stage (§4) where the fallback model is asked to continue from a visible prefix.

Code (`continue_code_v1`).

```
You are continuing a Python source file after an upstream model was
↪  interrupted.

The prefix below has already been shown to the user and is immutable.
Return only the exact source text that should be appended after the
↪  final character
of the prefix.

Do not repeat the prefix.
Do not restart the file.
Do not use Markdown fences.
Do not explain your answer.

ORIGINAL TASK
{task_prompt}

IMMUTABLE PREFIX
<BEGIN_PREFIX>
{visible_prefix}
<END_PREFIX>
```

C.3 Detector prompts

All-at-once judge (`judge_all_at_once_v1`). The original request and visible prefix are not truncated. The contract is reduced to the entry-point name. Only the first 40 continuation segments are shown, each capped at 400 characters after newline escaping, with truncation markers added. The prompt requests a JSON verdict. See `prompts/judge_all_at_once_v1.txt`.

Round-trip judge (`judge_round_trip_v1`). The judge sees the full combined output blob, without client-side input truncation, plus a modality tag (always `code` here). It is told that two models contributed, but not where the seam is. The prompt includes a checklist of structural, semantic, and restart-like tells — displayed below. This is a single content-based judgment, not a second generation that reconstructs the output.

```
You are auditing a single blob of model output that was assembled from
↪ two
different LLMs -- a primary model produced the beginning, then a fallback
model wrote the ending. You do NOT know where the cut is. Your job is to
decide whether the join is visible in the content itself.

Read the OUTPUT below and decide: is there a seam a careful reader would
↪ notice?

Concrete "seam smell" checklist (a positive on any one is enough to call
↪ it unsafe):

  [structural -- code / json / tool_call]
  - Indentation inconsistent across an obvious block boundary
    (e.g. an `if:` line followed by a body at the wrong indent).
  - The same identifier defined twice at top level
    (two `def foo`, two `class Bar`, two entries with the same JSON key).
  - Bracket / brace / quote counts don't balance across a boundary.
  - A Markdown code fence ``` appearing in the middle of
  ↪ otherwise-fence-free
    output -- a strong tell that one model thought it was starting fresh.
  - A statement is half-written and then abandoned (`if x ==` on one
  ↪ line,
    then `def something_else` immediately after).

  [semantic]
  - The output introduces a variable / function / concept that the
  ↪ earlier
    text never defined or referenced.
  - The output states a fact or picks a value that contradicts something
    earlier in the same output.
  - Style / voice / register changes mid-way (formal prose to casual
  ↪ prose,
    docstring-style comments to inline comments).

  [restart-like]
  - Any phrase like "The answer is ...", "Let me answer ...",
    "There are several interpretations ...", "To answer this ..."
    appearing inside the body (not at position 0).
  - A JSON document that has already opened a root object suddenly opens
    another {...} or [...].
  - Code that already imports something re-imports it, or redefines the
    entry-point function it already defined.

  [prose-only]
  - The answer drops or forgets an anchor / entity that it explicitly
    committed to earlier (e.g., prefix says "there are three
    ↪ interpretations:
    A, B, C" and the ending only discusses A).
  - The ending repeats verbatim a full sentence that appeared earlier.

Note: subtle style shift alone, without any of the above, is not enough
↪ to
call it unsafe -- many long outputs naturally shift. Look for at least
↪ one
concrete structural or semantic tell.

MODALITY: {modality}
```

```
OUTPUT
<BEGIN_OUTPUT>
{combined_output}
<END_OUTPUT>

Allowed defect types:
NONE, OVERLAP_DUPLICATION, RESTART, PREFIX_CONTRADICTION,
SYNTAX_STATE_CONFLICT, SCHEMA_CONFLICT, SEMANTIC_INCOMPATIBILITY,
TASK_DRIFT, OTHER.

Return exactly this JSON shape (no fences, no commentary):
{
  "seam_visible": boolean,
  "safe_to_append": boolean,
  "defect_type": string,
  "risk_score": number between 0 and 1 (higher = less safe),
  "confidence": number between 0 and 1,
  "seam_location_hint": string,
  "rationale": string
}
```

Round-trip judge, no-checklist ablation (`judge_round_trip_no_checklist_v1`). Identical to the round-trip prompt above except the entire "seam smell" checklist section is removed. This is an exploratory ablation; strict decoding failures are not imputed as scientific scores.

Matched boundary-cue ablation. `judge_round_trip_boundary_visible_v1` adds the numeric Unicode character offset of the cut to the same content and checklist. This is a derived prompt in `scripts/run_frozen_replication.py`, not a separate template file under `prompts/`. It does not insert an inline boundary marker. The matched pair holds judge model, temperature, token budget, prefix and raw concatenation fixed; this tests the effect of supplying that cue, not guaranteed recognition of the exact split by the judge.

Boundary-state extractor (`bsv_state_v1`). See `prompts/bsv_state_v1.txt` in the repository. Used by the `bsv-llm-state` and `bsv-hybrid` variants of the boundary-state verifier baseline.

C.4 Matched judge-information prompt

The new version is `fair-judge-public-task-x-numeric-boundary-v1`, not the legacy round-trip prompt above. The byte-identical renderer is included as `prompts/fair_judge_v1.py`; readable system and instruction copies are `prompts/fair_judge_system_v1.txt` and `prompts/fair_judge_instruction_v1.txt`. The system display copy adds a final newline; the renderer defines the actual request bytes.

The system tells the judge to treat task/code text as data, avoid code execution and tools, and return JSON. The shared instruction asks for append-risk judgments using structural errors, inconsistent identifiers, contradictions, duplicated work and any supplied task requirements; style change alone is insufficient. It requests only a numeric `risk_score` and one-sentence `rationale`.

After the literal `DATA` delimiter, the renderer appends compact JSON with sorted keys and `ensure_ascii=False`. Its three fields are:

- `combined_output`: the same complete code string in all four conditions, without truncation or inserted markers.
- `public_task`: either `null`, or an object containing `task_prompt` and `public_contract`; the latter has only `kind` and `entry_point`.
- `boundary_char_offset`: either `null`, or the zero-based Unicode character offset of the first continuation character.

The two visibility flags alter only the latter two values. The instruction explicitly forbids inferring withheld information from its absence. Model, temperature and output cap are identical across conditions; strict risk decoding and unknown handling are specified in Appendix B.8.