ByteDance | Seed

# Harness-IF: Evaluating Instruction Following Across Instruction Surfaces in Coding Agents

**Zining Huang**[1,2,*], **Haoran Que**[1,3,*], **Hong Zeng**[1,*], **Ge Zhang**[1], **Zuo Wang**[1], **Jin Chen**[1], **Haodong Wang**[1], **Zhongfei Hou**[1,*], **Changxin Pu**[1], **Shen Yan**[1,†], **Wenhao Huang**[1]

[1]ByteDance Seed, [2]Tsinghua University, [3]Peking University

[*]Work done at ByteDance Seed, [†]Corresponding author



## Abstract

When a coding agent obeys a rule, it may simply have been going to do that anyway. Existing instruction-following benchmarks cannot tell the difference: they concentrate rules in the user turn, while coding-agent benchmarks emphasize final task success. We introduce **Harness-IF**, which scores operational rules one at a time from execution evidence: 60 realistic multi-turn coding items drawn from a 642-rule library, 256 rules receiving verdicts, placed on the five configurable surfaces a deployed agent reads. To separate compliance from coincidence we introduce Against-Prior Accuracy (**AP-Acc**), which scores only rules labeled as opposing unprompted defaults, observed by re-running tasks with the rule withheld across nine probe builds and curated otherwise. Across 12 frontier models, accuracy spans 72.1–85.9% and AP-Acc 66.1–78.6%; every model is worse on against-prior rules, by 3.6 to 7.4 points (mean 5.81), and the direction survives a common-support analysis with item-clustered intervals. Aggregate scores therefore overstate compliance by a model-specific margin: prior control leaves the top build unchanged and exchanges three adjacent rank pairs. A counterbalanced conflict pilot on nine separate builds adds a second result: pooled precedence does not follow prompt depth, with system prompts, project files, and user instructions ahead of tool and skill descriptions.

**Date:** July 15, 2026
**Correspondence:** Shen Yan at sheny@bytedance.com

## 1 Introduction

Coding agents operate under a stack of instructions while inspecting repositories, editing files, running commands, calling tools, and responding across multiple turns [1, 8, 30]. Compliance must therefore persist beyond a single response and across system prompts, tool and skill descriptions, project files such as `CLAUDE.md`, and user instructions. Existing instruction-following benchmarks concentrate rules in user prompts [13, 26, 37, 48], whereas coding-agent benchmarks emphasize final task success [14, 15]. Neither directly reveals which operational rule an agent followed during a long harnessed workflow.

Instruction surface and instruction hierarchy are distinct. Hierarchy concerns which privileged source should prevail under conflict [35]; Harness-IF turns operational delivery surface into an evaluation dimension. Its controlled-relocation design holds rule meaning fixed while moving delivery across surfaces, enabling matched

comparisons. We exercise this design in a controlled conflict pilot (E0) and complement it with a larger coding panel that assigns rules to operationally admissible surfaces, combining controlled identification with broad, realistic coverage.

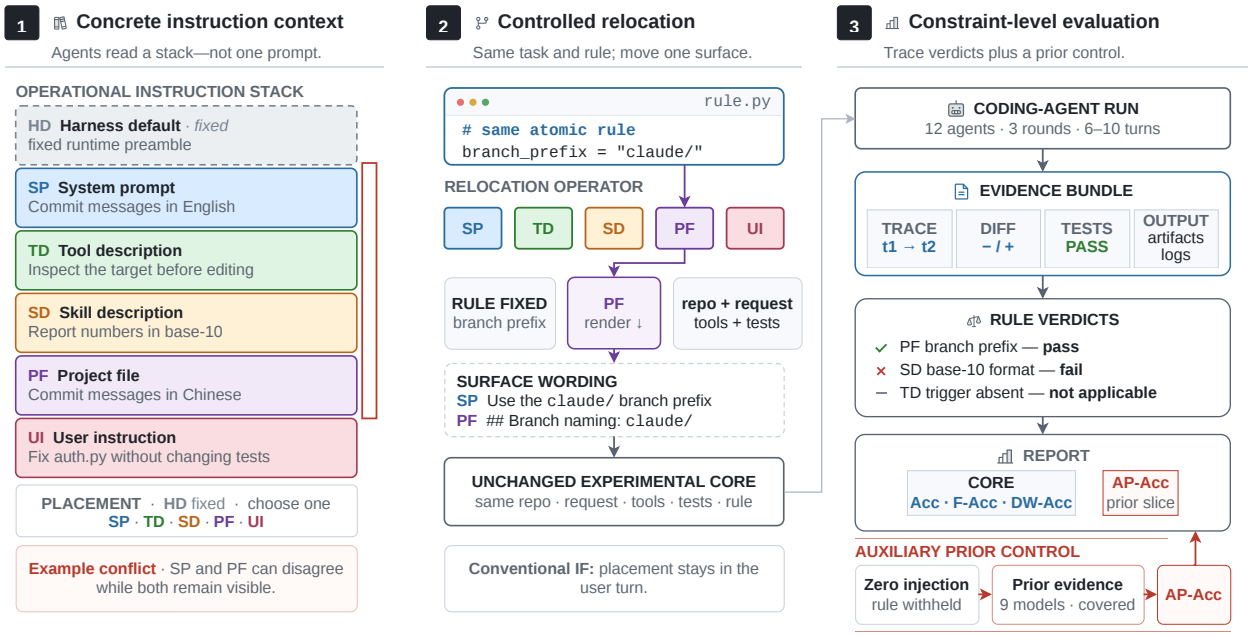


**Figure 1** Overview of Harness-IF. The left column shows an operational instruction stack. The middle column shows how the benchmark can relocate an atomic rule across configurable surfaces while holding its semantics fixed; the released main coding panel instead assigns suitable rules to surfaces. The right column records execution evidence, produces one verdict per applicable rule, and reports standard and prior-stratified metrics.

We introduce **Harness-IF**, a benchmark that turns operational instruction following into a rule-level measurement problem: its 642-rule library is instantiated as 60 realistic multi-turn coding items scoring 256 distinct rules, and every run yields a verdict per applicable rule rather than one task-level outcome. We then ask what those verdicts are worth. Against-Prior Accuracy (**AP-Acc**) scores only rules labeled as opposing unprompted defaults, and across 12 frontier models it is lower than accuracy for every one. The inflation is not a constant that cancels when builds are compared: it varies twofold across the cohort. The resulting verdict traces also expose sharply different failure signatures across rule families and modalities.

The evaluation has three complementary components. The main coding panel measures rule-level compliance and prior alignment across 12 models. E0 isolates surface precedence under four counterbalanced conflicts and nine model builds. The non-coding extension tests breadth on 40 cases using a domain-appropriate case-macro metric.

Our contributions are:

- **Benchmark that scores rules, not tasks:** a 642-rule library of which 302 rules are placed on the five configurable instruction surfaces of a deployed coding agent across 60 realistic multi-turn items, and 256 receive execution-grounded verdicts, one per applicable rule per run.
- **Metric that controls for unprompted defaults:** AP-Acc scores only rules labeled as opposing the unprompted default—observed in a zero-injection probe where that evidence is recoverable, and curated otherwise—separating instruction following from coincidence.
- **Evidence:** all 12 models perform worse on against-prior rules under like-for-like and common-support analyses, while rule families and modalities exhibit distinct difficulty and failure signatures; E0 further reveals a robust pooled surface ordering that is inconsistent with simple prompt-depth accounts.

## 2 Related Work

*Instruction-Following Evaluation.* Instruction-following (IF) evaluation has moved from coarse preference judgment to explicit constraint checking. IFEval [48] made verifiable constraints a standard protocol; FollowBench [13], InfoBench [26], and ComplexBench [37] expanded the space to graded difficulty, information constraints, and multi-constraint composition. The most recent benchmarks test harder prompts rather than only more prompts: Multi-IF [11] adds multilingual multi-turn dialogue, CFBench [45], LIFBench [38], and EIFBench [50] stress complex or long-context constraint sets, IFBench [24] broadens verifiable rule checking, and AgentIF [25] introduces agentic IF scenarios. These benchmarks establish constraint-level measurement, but their experimental variable is still mainly the instruction content or scenario. Harness-IF asks a different question: when the same rule is delivered through different harness surfaces, does the agent still follow it?

**Table 1** Comparison with representative recent benchmarks. Surfaces is the number of distinct instruction-delivery surfaces a benchmark represents; for Harness-IF this is six (HD, SP, TD, SD, PF, UI), of which the harness default (HD) is fixed and the remaining five are configurable placement surfaces: placement is varied as an experimental variable in the E0 conflict pilot and assigned by admissibility in the main panel. Prior control means explicit control for unprompted default behavior. ✓ marks full support, • partial support, and – absence.

| Benchmark | Surfaces | Multi-Turn | Tool Use | Prior Control | Rule-Level Scoring |
|---|---|---|---|---|---|
| IFEval [48] | 1 | – | – | – | ✓ |
| ComplexBench [37] | 1 | – | – | – | ✓ |
| IFBench [24] | 1 | • | • | – | ✓ |
| AgentIF [25] | 2 | • | • | – | ✓ |
| CodeIF-Bench [36] | 1 | ✓ | • | – | ✓ |
| BFCL [23] | 1 | ✓ | ✓ | – | – |
| AppWorld [34] | 1 | ✓ | ✓ | – | • |
| $\tau^2$-bench [4] | 2 | ✓ | ✓ | – | • |
| SWE-Bench Pro [6] | 1 | ✓ | ✓ | • | – |
| Terminal-Bench [16] | 1 | ✓ | ✓ | • | – |
| **Harness-IF** | **6** | ✓ | ✓ | ✓ | ✓ |

*Agentic and Coding Evaluation.* Agent benchmarks now cover realistic environments in which models must plan, call tools, inspect state, and recover from intermediate errors. AgentBench [15] and GAIA [17] evaluate general-purpose assistants; WebArena [49], OSWorld [39], BFCL [23], AppWorld [34], $\tau$-bench [42], and $\tau^2$-bench [4] evaluate web, desktop, tool-calling, application, and user-interaction agents. A parallel line evaluates technical work—repository repair, command-line work, and ML engineering—through the SWE-bench family [6, 14, 21, 40, 41], Terminal-Bench [16], and the MLAgentBench/MLE-bench/PaperBench line [5, 12, 31]. These benchmarks are high fidelity, but most headline scores collapse a trajectory into task success. Harness-IF is complementary: it uses realistic coding-agent execution, but treats instruction following itself as the object of measurement.

*Harness and Trajectory Evaluation.* For deployed agents, the model is only one part of the system: behavior is shaped by system prompts, tool schemas, skills, project files, memory, retries, and the trace visible to the evaluator. Toolformer [28] and Gorilla [22] show why tool context matters; instruction-hierarchy work [35] and IHEval's synthetic conflicts over system, user, history, and tool-output messages [46] show why authority context does. IHEval tests compliance with a prescribed message hierarchy; Harness-IF instead measures distributed constraints in executable workspaces, and E0 estimates precedence over a broader surface set without assuming a universal order. New agent-evaluation papers make the same point from different angles: MCP tool descriptions affect agent efficiency [9], open skill manifests require auditing [43], process defects can survive final-task evaluation [10], and long-horizon coding agents can exploit benchmark specifications [47]. Harness-IF turns that conclusion into a surface-aware benchmark design: it records rule-level verdicts and reports AP-Acc to separate prompted compliance from prior-aligned default behavior.

# 3 Harness-IF

In a complex agentic workflow, the instruction is not a single standalone user request. As shown in Figure 1, the agent reads multiple prompts in sequence; we refer to these prompt levels as instruction surfaces. Requirements on different surfaces can conflict, yet how agents follow the full instruction stack remains underexplored [25, 37]. We therefore build Harness-IF around where a rule is delivered; Table 1 summarizes how it differs from prior benchmarks.

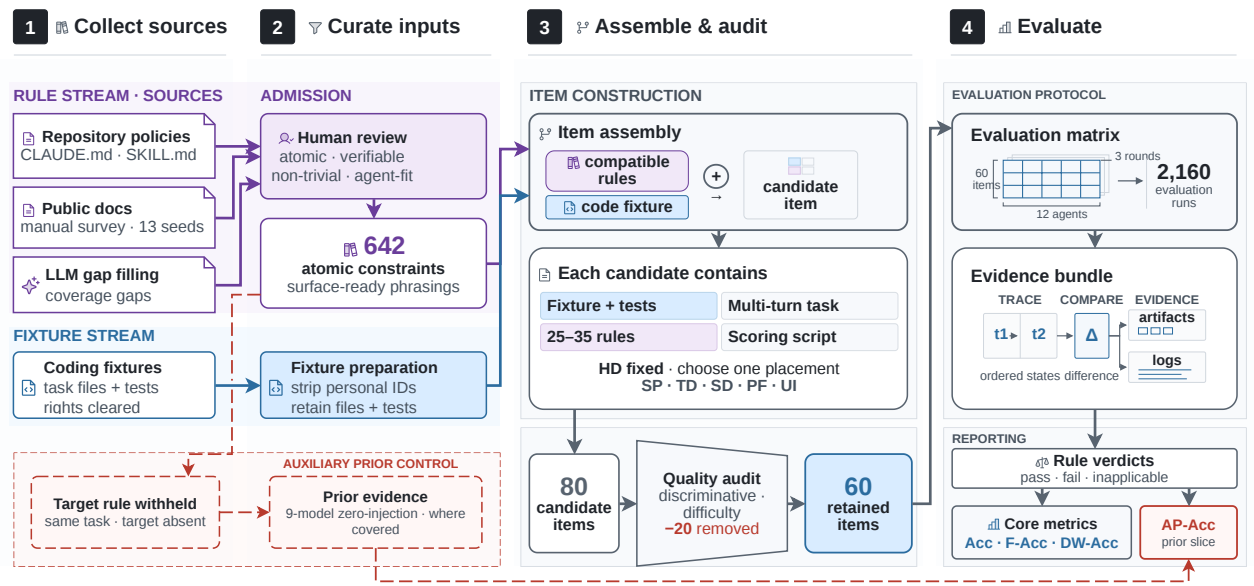


**Figure 2** The Harness-IF construction and evaluation pipeline. Candidate constraints and coding fixtures are curated and assembled into 60 quality-audited multi-turn items. The resulting $60 \times 12 \times 3 = 2{,}160$ agent–item–round runs produce traces, state differences, artifacts, and logs for rule-level verdicts and core/AP-Acc reporting. The auxiliary zero-injection branch supplies prior evidence without entering the core metrics.

## 3.1 Instruction Surfaces

Following the common configuration used by modern coding agents [1, 3], we distinguish six instruction surfaces:

- **Harness Default (HD).** The default instructions inserted by the agent platform at the start of each run; users usually cannot edit them in deployment.
- **System Prompt (SP).** Instructions written by the system developer to set the agent's role and general behavior, such as "never expose secrets" or "answer with high confidence."
- **Tool Description (TD).** Instructions that describe what a tool does and how the agent should use it.
- **Skill Description (SD).** Instructions that describe when a reusable skill should be used and what rules the agent should follow when using it.
- **Project File (PF).** Project-level instructions stored in files such as `CLAUDE.md`, `CONTRIBUTING.md`, or `AGENTS.md`.
- **User Instruction (UI).** The user's current request to the agent.

These surfaces are written by different parties, such as platform developers, tool authors, project maintainers, and users, and appear in different parts of the prompt. Formally, let $x$ be a single constraint, $y$ be the agent output, and $s$ be the surface on which the constraint is placed. We write $F(x, y, s) \in \{\textbf{pass}, \textbf{fail}, \textbf{n/a}\}$ for the rule-level evaluator that judges whether output $y$ satisfies constraint $x$ when $x$ is delivered on surface $s$;

§3.4 turns its pass/fail outcomes into the indicator $z$ used by the metrics. Most benchmarks [6, 37] keep $s$ fixed at the user instruction. Harness-IF evaluates how well agents follow specific constraints placed on five configurable surfaces, $s \in \{\mathrm{SP}, \mathrm{TD}, \mathrm{SD}, \mathrm{PF}, \mathrm{UI}\}$.

## 3.2 Data Structure

*Constraints.* A constraint is one specific rule that an agent is asked to follow. Each rule is narrow enough to be judged pass, fail, or not applicable from the execution evidence of a run. The seven families are professional writing, output control, code style, workflow, quantitative limits, conditional logic, and tool use.

*Surfaces.* A constraint is admissible on a surface when that surface's authoring role could plausibly carry the rule in deployment: a branch-naming rule can sit in a project file or a system prompt, but not in a tool schema. When several surfaces are admissible we sample one uniformly, using surface-appropriate phrasings that preserve rule semantics. This design supports descriptive surface stratification in the main panel; E0 provides the controlled conflict comparison.

*Scenarios.* Each scenario provides working files for a realistic coding task—fixing a backend API bug, updating a frontend component, editing a data-processing script, writing a test, or changing project documentation—so items differ in language, file layout, and the kind of edit required. The evaluated panel draws on eight scenarios from a 13-scenario library, spanning backend, frontend, systems, data/ML, automation, security testing, tool orchestration, and technical documentation.

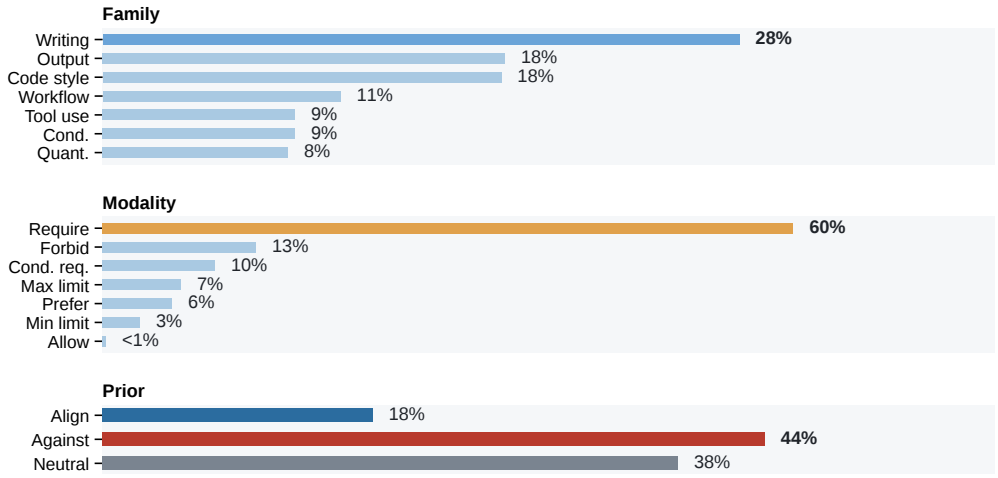


**Figure 3** Composition of the 642-rule library across three independent annotation axes: rule family, logical modality, and behavioral prior. Prior labels indicate whether a rule is aligned with, against, or neutral to observed or curated unprompted behavior; Cond. req. denotes conditional requirement. The figure describes the library, not the evaluated subset: the coding panel instantiates 302 of these rules and scores 256 of them, and professional-writing rules are exercised only in the non-coding extension.

Each data item contains a code scenario, multi-turn user instructions, selected constraints, and their assigned surfaces. Figure 3 shows the library's composition. Requirements are the dominant modality (60%), and professional writing is the largest single family (28%) even though it is exercised outside the coding panel. Prior labels are deliberately mixed—18% align-prior, 44% against-prior, and 38% neutral—so the panel cannot be passed by defaults alone. Scenario is a separate annotation axis and is not counted in this figure.

## 3.3 Data Curation

As shown in Figure 2, the data curation process has four steps: we collect constraints from raw sources, filter them, format them into benchmark items, and apply quality checks.

*Raw Sources.* The constraint library began with a manual survey of publicly accessible GitHub software projects, focusing on project-instruction and contributor documents such as `CLAUDE.md`, `AGENTS.md`, `CONTRIBUTING.md`, skill descriptions, and tool schemas across multiple programming languages and software-engineering workflows. We atomized recurring requirements into benchmark-ready rules, normalized them across surface-specific phrasings, and used reviewed LLM-assisted proposals plus author review to fill under-covered taxonomy cells. The resulting tasks, specifications, and scoring artifacts are project-authored rather than direct copies of source repositories. Detailed provenance is maintained for internal audit and rights review, but the public artifact exposes only coarse source categories so individual constraints cannot be joined directly to named repositories. Coding workspaces were assembled to match realistic technology stacks and are included in the release following the 2026-07-27 owner attestation. Appendix G provides the detailed release boundary.

*Filtering and Item Formatting.* Candidate constraints must be atomic, verifiable, non-trivial, and suitable for coding-agent tasks; proposals are deduplicated and human-reviewed before admission, with targeted expansion for under-covered categories. For each scenario and set of working files we then select constraints and place each one on a single admissible surface.

*Quality Filtering.* From 80 candidates, quality review retained 60 items. Because selection used observed difficulty and discriminativeness, the reported panel may favor items that separated the pilot models (selection optimism); Appendix G documents the full disposition and validation sequence.

In total, the Harness-IF library contains 642 atomic constraints and the evaluated panel comprises 60 multi-turn items. Each item injects a pack of 25–35 rules, of which 10–27 are scorable given the opportunities the item creates; across the panel, 302 distinct library rules are instantiated and 256 receive at least one verdict.

### 3.4 AP-Acc

We use rule-level accuracy metrics to measure whether agents follow the selected constraints. For each agent output, the evaluator judges each constraint as passed, failed, or not applicable. Let $z_{a,i,r} = 1$ if agent $a$ satisfies constraint $r$ in item $i$, and $z_{a,i,r} = 0$ otherwise. Let $\mathcal{E}_a$ be the set of pass/fail constraint instances for agent $a$; not-applicable constraints are excluded from the denominator because the item does not create a valid opportunity to judge whether the agent followed that constraint.

For a like-for-like binary recomputation, the basic accuracy treats all eligible constraint instances equally:

$$\mathrm{Acc}(a) = \frac{\sum_{(i,r)\in\mathcal{E}_a} z_{a,i,r}}{|\mathcal{E}_a|}. \tag{1}$$

We also report two cohort-adaptive diagnostics. First, filtered accuracy (**F-Acc**) removes item–rule pairs that do not distinguish agents in the evaluated cohort. Let $\mathcal{D}$ be the set of item–rule pairs that produce different pass/fail outcomes across agents:

$$\text{F-Acc}(a) = \frac{\sum_{(i,r)\in\mathcal{E}_a\cap\mathcal{D}} z_{a,i,r}}{|\mathcal{E}_a \cap \mathcal{D}|}. \tag{2}$$

Second, discrimination-weighted accuracy (**DW-Acc**) gives more weight to rules that better separate the evaluated agents. Let $d_r \geq 0$ be the non-negative Pearson correlation between agents' pass rates on rule $r$ and their overall accuracy. Rules with non-positive or undefined discrimination receive $d_r = 0$:

$$\text{DW-Acc}(a) = \frac{\sum_{(i,r)\in\mathcal{E}_a} d_r z_{a,i,r}}{\sum_{(i,r)\in\mathcal{E}_a} d_r}. \tag{3}$$

Accuracy and AP-Acc can differ when a rule matches default behavior. We therefore assign behavioral prior labels using zero-injection evidence and curated prior annotations. AP-Acc is a behavioral stratification rather than a training-provenance claim; Appendix B reports label lineage and sensitivity analyses (Figure 3).

We introduce against-prior accuracy (**AP-Acc**), which only evaluates constraints in the against-prior set $\mathcal{P}$. This metric reports performance on constraints labeled as opposing the corresponding default behavior:

$$\text{AP-Acc}(a) = \frac{\sum_{(i,r)\in\mathcal{E}_a,\ r\in\mathcal{P}} z_{a,i,r}}{|\{(i,r) \in \mathcal{E}_a : r \in \mathcal{P}\}|}. \tag{4}$$

All four displayed columns are computed from the released 2,160-record panel under the equations above, over the same eligible verdicts, so $\Delta$ is a like-for-like contrast rather than a difference between aggregation conventions. F-Acc and DW-Acc are cohort-relative diagnostics and are interpreted descriptively: both change if the evaluated cohort changes. Exact denominators, the common-support analysis, and the per-rule discrimination weights appear in Appendix B, and the release includes the script that regenerates every displayed number from the shipped verdict records.

# 4 Results

## 4.1 Main Benchmark Results

**Table 2** Rule-level accuracy (%) across 12 model builds, recomputed from the released 2,160-record panel. All four columns are binary rates over the same eligible verdicts, so $\Delta$ = Acc − AP-Acc is a like-for-like contrast. Rows are ordered by unrounded Acc, which separates MiniMax-M2.7 from Seed-2.0-Pro by 0.03 points; bold marks each column's best value.

| Models | Acc | F-Acc | DW-Acc | AP-Acc | Δ |
|---|---|---|---|---|---|
| Claude-Opus-4.7 [2] | **85.9** | **79.3** | **88.5** | **78.6** | +7.3 |
| GPT-5.5 [20] | 83.1 | 75.5 | 81.2 | 77.0 | +6.1 |
| Claude-Sonnet-4.6 [2] | 82.5 | 73.9 | 82.2 | 78.5 | +4.0 |
| Claude-Haiku-4.5 [2] | 79.0 | 69.4 | 75.9 | 71.9 | +7.2 |
| GLM-5.1 [44] | 78.8 | 67.9 | 75.6 | 75.2 | +3.6 |
| Qwen-3.6-Max [27] | 76.7 | 65.9 | 70.5 | 71.7 | +5.1 |
| Hy3 [33] | 76.2 | 65.2 | 69.7 | 70.8 | +5.4 |
| Kimi-K2.6 [19] | 76.1 | 65.0 | 70.3 | 70.0 | +6.1 |
| Gemini-3.1-Pro [8] | 75.4 | 64.0 | 70.0 | 69.8 | +5.6 |
| MiniMax-M2.7 [18] | 73.6 | 61.2 | 65.1 | 67.7 | +5.9 |
| Seed-2.0-Pro [29] | 73.6 | 61.1 | 63.2 | 66.1 | +7.4 |
| StepFun-3.5 [32] | 72.1 | 59.1 | 62.5 | 66.2 | +5.9 |

*Evaluation scale.* The main coding panel evaluates 12 frontier model builds on 60 items over three rounds: 2,160 agent–item–round runs and 40,104 rule-level verdict rows, one per applicable constraint per run. Deterministic checks (regex, AST, cross-file, command-output) cover 13.3% of eligible verdicts; a GPT-5.2 judge [30] scores rubric constraints by three-vote majority and adjudicates hybrid checks, so 86.8% of rows involve the judge. The like-for-like binary analysis contains 37,616 eligible verdicts, including 19,449 against-prior verdicts. Appendix D tables the exact build identifier behind each row and documents the serving configuration, run status, and exclusion rules.

Table 2 supports two conclusions. First, every model is less successful on the against-prior subset, so aggregate scores overstate compliance where a rule departs from model defaults. Second, the overstatement is model-specific: it ranges from 3.6 to 7.4 points, a twofold spread. Claude-Opus-4.7 leads all four columns, so prior control does not change the top-ranked build, but it exchanges three adjacent rank pairs (2–3, 4–5, and 11–12). Accuracy spans 13.7 points across the cohort, with a standard deviation of 4.2 points.

Models largely agree on which rules are hard: correlating each model's per-rule pass-rate vector against the cohort mean, over the 242 rules every model attempted, gives 0.57–0.89 (mean 0.80). The panel therefore measures a shared difficulty ordering, which is what makes the adjacent-rank exchanges informative rather than noise.

*Against-prior rules expose a consistent compliance gap.* Every evaluated model scores lower on the against-prior subset. Under the like-for-like binary definition, the mean Acc–AP-Acc gap is 5.81 points across 37,616 eligible verdicts and remains positive for all 12 models. A stricter common-support analysis retains 2,430 of 3,342 (item, round, rule) observation keys (72.7%) on which every model produced a clean pass/fail outcome; item-clustered 95% intervals for the paired gap remain positive model by model. On this fixed panel the direction is stable: for all 12 builds, aggregate instruction-following scores exceed performance on rules that oppose observed defaults.

*The benchmark resolves broad patterns more clearly than adjacent ranks.* On common support, item-clustered intervals leave every adjacent model comparison unresolved, so we treat the displayed order as a point ranking while resting the paper's claims on the prior-alignment and failure-pattern results.

## 4.2 Failures Concentrate on Rules That Demand Action

We group failures by what the violated rule demands, which is recoverable from the released records without free-text classification: a rule that requires an action or sets a minimum can only fail by the agent falling short, and a rule that forbids an action or caps a quantity can only fail by the agent overstepping. Shortfall rules absorb 77.1% of the 8,440 failures, against 20.8% for overstep rules and 2.1% for soft preferences (Figure 4). The asymmetry reflects exposure rather than propensity: the panel contains 27,306 shortfall instances against 8,443 overstep instances, and the two classes fail at similar rates (23.8% versus 20.8%). Most compliance failures in realistic coding work are therefore omissions of demanded behavior, so verifiers tuned to detect excess output address only about a fifth of the failure mass. Appendix C gives the full decomposition.

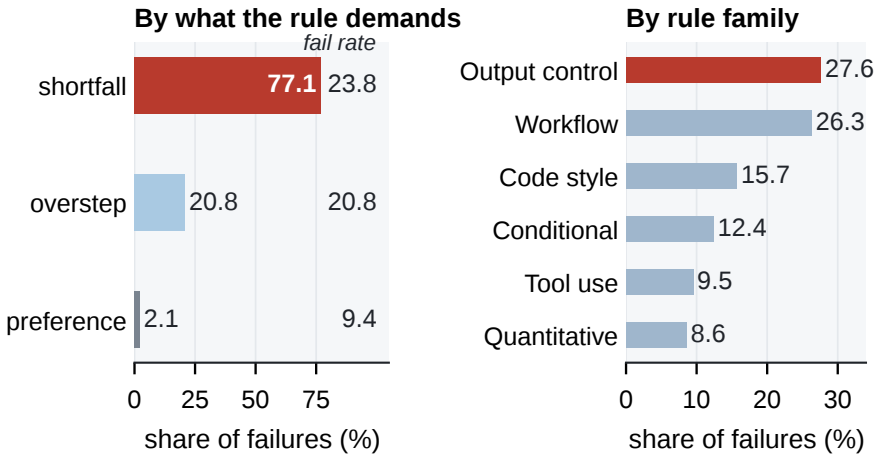


**Figure 4** Failure decomposition over the released panel. Left: share of the 8,440 failures by what the violated rule demands, beside each class's own failure rate; shortfall rules carry most of the mass at a comparable rate, so the gap is one of exposure. Right: share of failures by rule family.

Failure mass is unevenly distributed across families. Output control and workflow together account for 53.9% of all failures (27.6% and 26.3%), followed by code style (15.7%), conditional logic (12.4%), tool use (9.5%), and quantitative limits (8.6%). Output control combines the largest failure mass with the lowest pass rate, which makes it the most productive remediation target; workflow's mass instead follows from its size in the panel, since its pass rate is mid-range.

## 4.3 Commands and Output-Control Rules Are Hardest

Difficulty varies substantially across both logical modality and rule family (Figure 5); Appendix A defines these axes. For difficulty reporting we pool the seven governed modality operators into four coarse classes: Commanding (require/forbid), Conditional (conditional-require), Quantitative (limit-max/limit-min), and Preference (prefer/allow); the coarse "Quantitative" class is the pooled modality and is distinct from the

like-named rule family. The released coding scorecard provides family aggregates for six of the seven library families; professional writing is not included in this comparison.

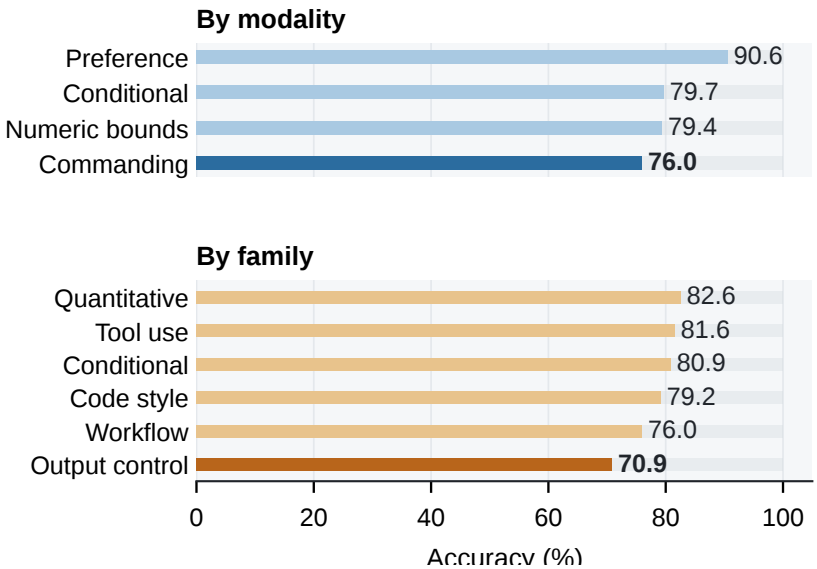


**Figure 5** Pooled binary accuracy over the 12 builds by modality (top) and by the six families that receive verdicts (bottom); the lowest bar in each panel is emphasized. Commanding constraints are hardest by modality, and output control is the lowest-scoring family overall and for 11 of the 12 builds.

Commanding rules are hardest by modality (76.0%), against 79.4% for numeric bounds, 79.7% for conditional rules, and 90.6% for preferences. The preference result is consistent with default alignment: preferences can match existing behavior, whereas commands often require departure from it. Family differences are narrower: output control scores 70.9% against 82.6% for quantitative limits, an 11.7-point spread; it is the lowest-scoring family for 11 of the 12 builds, and no build clears 79% on it.

## 4.4 Surface Precedence Does Not Follow Prompt Depth

E0 provides a controlled, scoped test of surface precedence. System prompts, project files, and user instructions tie exactly for the best mean rank: each has a model-level rank sum of 20, hence $20/9 = 2.22\ldots$. They precede tool descriptions at 3.78 and skill descriptions at 4.56 (Figure 6). By prompt depth we mean position in the assembled context, with the user turn last; a depth account predicts that later-placed instructions win. That the user instruction only ties for first is inconsistent with such an account.

E0 contains 916 runs from nine older model builds on four synthetic conflict pairs, uses deterministic-only scoring, and is not pooled with the main coding panel. Counterbalancing assigns 458 runs to each direction; 889 produce decisive outcomes. A pooled Bradley–Terry analysis places SP, PF, and UI above TD, with SD last. The ordering survives separate direction fits, equal-cell weighting, all leave-one-pair/model-out fits, and four conservative assignments of the 27 errors. A crossed model-and-pair bootstrap preserves the complete ordering in 9,652/10,000 resamples; only 6/9 individual-build fits reproduce it exactly, so we interpret it as a pooled cross-build tendency, not a universal hierarchy (Appendix D.4). The main panel instead assigns suitable rules to surfaces and supports descriptive stratification rather than a paired surface effect.

## 4.5 Reliability: Cross-Model Patterns Are Stable, Rank Margins Are Not

The prior-alignment gap stays positive for all 12 models in the fully deterministic subset, averaging 13.09 points over 5,013 eligible verdicts; it is larger there because that subset over-represents the pattern- and command-checked families, against 5.81 points panel-wide. Test–retest ICC is 0.725 across models and 0.599 across agent–item cells, and an earlier human-reference audit, run under a five-vote rather than the released three-vote configuration, showed 69.0% agreement ($\kappa = 0.515$ over 919 rows). Verdicts are far less stable

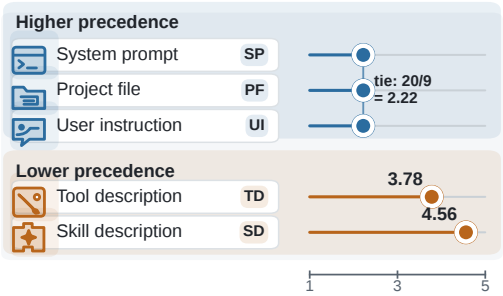


**Figure 6** E0 mean conflict ranks over nine older model builds (lower = higher precedence). SP, PF, and UI are an exact three-way tie: each model-level rank sum is 20, so each mean is $20/9 = 2.22\ldots$; TD = 3.78 and SD = 4.56. This deterministic-only pilot is separate from the main 12-model coding panel; ranks summarize four synthetic conflict pairs and are not pass rates.

under a judge swap (62.1% agreement, $\kappa = 0.163$ on 116 paired clean verdicts). Since 86.8% of verdict rows involve the judge, this is the dominant uncertainty in the measurement, and we therefore report cross-model patterns and treat the displayed order as a point ranking. Appendix E provides the complete calibration protocols. On $n = 65$ common non-error samples, Claude–GPT inter-LLM agreement is $\kappa = 0.4717$, which is not human validation.

*The instrument transfers beyond code, the ranking does not.* A separate exploratory panel scores 40 non-coding cases across five domains over 1,428 valid trajectories from the same 12 builds. Case-macro pass rates span 65.8–84.8%, led by GPT-5.5 rather than the coding leader; that panel's metric and population differ and are never pooled with the coding results (Appendix G).

## 5 Limitations

*Scope.* Harness-IF targets multi-turn coding agents. The 60 evaluated items were selected from an 80-item working set through quality and discriminativeness review, and they instantiate 302 of the library's 642 rules and score 256 of them; professional-writing rules are exercised only in the non-coding extension. Our claims therefore hold for this panel of items and models; the 40-case non-coding extension provides complementary breadth under a separate case-macro metric.

*Measurement.* AP-Acc is a behavioral stratification over observed or curated defaults, not a claim about model training provenance. We report label lineage in the appendix, including the overlap between the zero-injection probe cohort and the evaluated panel, and use one like-for-like binary definition throughout. Because 86.8% of verdicts involve an LLM judge whose labels shift under a judge swap, absolute levels are instrument-specific and the cross-model comparisons, which share the instrument, carry the claims. F-Acc and DW-Acc are cohort-adaptive diagnostics, while common-support and item-clustered analyses provide the primary denominator and uncertainty checks. The failure decomposition groups violations by what the rule demands and therefore describes rule structure rather than latent causal mechanisms.

*Resolution and surfaces.* Calibration and test–retest analyses support the benchmark's cross-model patterns but not fine distinctions among neighboring models. The main panel places rules on operationally admissible surfaces for realistic coverage; E0 separately supplies the controlled comparison. Its pooled ordering is robust across crossed-bootstrap, direction, deletion, weighting, and error analyses and is interpreted as an E0 cross-build tendency rather than a universal hierarchy.

## 6 Conclusion

Harness-IF makes operational instruction following measurable at the level of individual rules and delivery surfaces. Across 60 multi-turn coding items, 256 scored rules, and 12 frontier models, every model performs worse on against-prior rules under both full-panel and common-support analyses, revealing compliance differences that aggregate scores obscure. The benchmark further localizes failure signatures by family and modality, while its counterbalanced E0 study identifies a robust pooled ordering in which system prompts, project files, and user instructions lead tool and skill descriptions. Harness-IF therefore provides a unified measurement framework for determining which operational rules an agent follows, where those rules are delivered, and how reliably they survive execution in realistic workflows. Coding-agent evaluation should treat instruction compliance as an execution-level object: preserve rule provenance, score individual opportunities, separate against-prior behavior from aggregate success, and counterbalance delivery surfaces when making precedence claims.

## Data and Code Availability

The benchmark is prepared for public release as a self-contained package: the 642-rule constraint library in YAML, the 60 assembled coding items with their scenario fixtures and ground-truth scoring scripts, the 2,160-record verdict panel behind every number reported here, and the evaluation and analysis code. A single script recomputes each displayed figure and table from the shipped verdict records, so the results in this paper can be reproduced offline, without model API access and without re-running any agent. Evaluating a new model additionally requires a coding-agent harness and API access to both the model under test and a judge model.

Project-authored software is released under Apache-2.0, and the project-authored benchmark text, data, and sanitized derived results under CC-BY-4.0. Raw provider outputs and agent run workspaces are not redistributed. The release location will be given in a later version of this preprint.

## References


[1] Anthropic. Claude code, 2024. https://claude.com/claude-code.

[2] Anthropic. Models overview. https://platform.claude.com/docs/claude/docs/models-overview, 2026.

[3] Anthropic, Erik Schluntz, and Barry Zhang. Building effective agents. Anthropic Research Blog, 2024.

[4] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. $\tau^2$-Bench: Evaluating conversational agents in a dual-control environment. arXiv preprint arXiv:2506.07982, 2025.

[5] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In Proc. ICLR, 2025.

[6] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? arXiv preprint arXiv:2509.16941, 2025.

[7] 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.

[8] Google DeepMind. Gemini 3.1 Pro: Model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026.

[9] Mohammed Mehedi Hasan, Hao Li, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E. Hassan. Model context protocol (MCP) tool descriptions are smelly! towards improving AI agent efficiency with augmented MCP tool descriptions. arXiv preprint arXiv:2602.14878, 2026.

[10] Jiawei He, Jie Jia, Chenbo Liu, Chaoyi Xue, Yapeng Song, Xikai Yang, and Dong Sun. ProcCtrlBench: Evaluating process-level defects and control preservation in LLM coding agents. arXiv preprint arXiv:2605.20251, 2026.

[11] Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, Shruti Bhosale, Chenguang Zhu, Karthik Abinav Sankararaman, Eryk Helenowski, Melanie Kambadur, Aditya Tayade, Hao Ma, Han Fang, and Sinong Wang. Multi-IF: Benchmarking LLMs on multi-turn and multilingual instructions following. arXiv preprint arXiv:2410.15553, 2024.

[12] Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In Proc. ICML, 2024.

[13] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Proc. ACL, 2024.

[14] 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 Proc. ICLR, 2024.

[15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In Proc. ICLR, 2024.

[16] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael

Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. arXiv preprint arXiv:2601.11868, 2026.

[17] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In Proc. ICLR, 2024.

[18] MiniMax. MiniMax M2.7: Model self-improvement. https://www.minimax.io/models/text/m27, 2026.

[19] Moonshot AI. Kimi K2.6 model card. https://huggingface.co/moonshotai/Kimi-K2.6, 2026.

[20] OpenAI. GPT-5.5 model. https://developers.openai.com/api/docs/models/gpt-5.5/, 2026.

[21] OpenAI Preparedness Team. Introducing SWE-bench verified. https://openai.com/index/introducing-swe-bench-verified/, 2024.

[22] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In Proc. NeurIPS, 2024.

[23] Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Ji, Vivek Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of LLMs. In Proc. ICML, 2025.

[24] Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In Proc. NeurIPS Datasets and Benchmarks Track, 2025.

[25] Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. AgentIF: Benchmarking instruction following of large language models in agentic scenarios. In Proc. NeurIPS Datasets and Benchmarks Track, 2025.

[26] Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. InfoBench: Evaluating instruction following ability in large language models. arXiv preprint arXiv:2401.03601, 2024.

[27] Qwen Team. Qwen3.6-Max-Preview released. https://qwen.ai/blog?id=qwen3.6-max-preview, 2026.

[28] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Proc. NeurIPS, 2023.

[29] Seed Team. Seed2.0 model card. https://yfz.ai/Seed2.0_Model_Card.pdf, 2026.

[30] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 System Card. arXiv preprint arXiv:2601.03267, 2025.

[31] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. arXiv preprint arXiv:2504.01848, 2025.

[32] StepFun. Step 3.5 Flash: Open frontier-level intelligence with 11b active parameters. arXiv preprint arXiv:2602.10604, 2026.

[33] Tencent. Tencent unveils Hy3 preview. https://www.tencent.com/en-us/articles/2202320.html, 2026.

[34] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Proc. ACL, 2024.

[35] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training LLMs to prioritize privileged instructions. arXiv preprint arXiv:2404.13208, 2024.

[36] Peiding Wang, Li Zhang, Fang Liu, Lin Shi, Minxiao Li, Bo Shen, and An Fu. CodeIF-Bench: Evaluating instruction-following capabilities of large language models in interactive code generation. arXiv preprint arXiv:2503.22688, 2025.

[37] Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition. In Proc. NeurIPS Datasets and Benchmarks Track, 2024.

[38] Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Xiangju Lu, Junmin Zhu, and Wei Zhang. LIFBench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. In Proc. ACL, 2025.

[39] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. arXiv preprint arXiv:2404.07972, 2024.

[40] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. arXiv preprint arXiv:2405.15793, 2024.

[41] John Yang, Carlos E. Jimenez, Alex L. Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R. Narasimhan, Diyi Yang, Sida I. Wang, and Ofir Press. SWE-bench multimodal: Do AI systems generalize to visual software domains? In Proc. ICLR, 2025.

[42] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. arXiv preprint arXiv:2406.12045, 2024.

[43] Jiahao Ying, Boxian Ai, Wei Tang, Siyuan Liu, and Yixin Cao. OpenSkillEval: Automatically auditing the open skill ecosystem for LLM agents. arXiv preprint arXiv:2605.23657, 2026.

[44] Z.ai. GLM-5.1 release notes. https://docs.z.ai/release-notes/new-released, 2026.

[45] Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. CFBench: A comprehensive constraints-following benchmark for LLMs. In Proc. ACL, 2025.

[46] Zhihan Zhang, Shiyang Li, Zixuan Zhang, Xin Liu, Haoming Jiang, Xianfeng Tang, Yifan Gao, Zheng Li, Haodong Wang, Zhaoxuan Tan, Yichuan Li, Qingyu Yin, Bing Yin, and Meng Jiang. IHEval: Evaluating language models on following the instruction hierarchy. In Proc. NAACL, 2025.

[47] Bingchen Zhao, Dhruv Srikanth, Yuxiang Wu, and Zhengyao Jiang. SpecBench: Measuring reward hacking in long-horizon coding agents. arXiv preprint arXiv:2605.21384, 2026.

[48] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023.

[49] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In Proc. ICLR, 2024.

[50] Tao Zou, Xinghua Zhang, Haiyang Yu, Minzheng Wang, Fei Huang, and Yongbin Li. EIFBench: Extremely complex instruction following benchmark for large language models. In Proc. EMNLP, 2025.

# Appendix

## A Design axis definitions

Each of the 642 atomic constraints in the HARNESS-IF library is tagged along eight separable axes.

**Family (7).** The rule-family annotation: professional-writing, output-control, code-style, workflow, quantitative, conditional-logic, and tool-use. Family is independent of scenario applicability. Synthetic examples include requiring a verification step after implementation (workflow) and requiring structured rather than free-form tool parameters (tool-use).

**Modality (7).** The logical operator: require, forbid, conditional-require, limit-max, limit-min, prefer, allow. Modality captures difficulty structure that family alone misses: a forbid and a require on the same content are different instructions. Synthetic examples include requiring a short completion summary, forbidding generated cache files, requiring validation when a configuration changes, and preferring a compact report format.

**Prior (3).** The model's observed or curated default behavior in the absence of the instruction: align-prior (the instruction matches the default tendency), against-prior (the instruction pushes against it), neutral. A zero-injection ablation runs each task with the target rule withheld across nine probe builds; a rule receives a consensus label when at least five of those nine agree, which is recoverable for 287 rules. Other final labels use recoverable pre-existing curation or have unknown lineage, as detailed in the full reliability analysis. This probe cohort is the same set of nine builds used in the E0 conflict pilot, run there on different tasks for a different purpose; the two experiments are separate and their results are never pooled. Five of the nine share an identifier with a model in the evaluated panel, and the full reliability analysis quantifies that overlap. The current governed totals are 115 align-prior, 282 against-prior, and 245 neutral. These labels are behavioral strata, not causal claims about training provenance.

**Observability (4).** Where compliance is visible: surface (tokens in the final turn), structural (file layout, AST), behavioral (test pass/fail, side effects), deep (semantic intent). Synthetic examples include a required summary heading (surface), a required module layout (structural), successful local validation (behavioral), and preservation of an argument's causal order (deep).

**Verifiability (3).** How compliance can be checked: deterministic (regex, AST match, cross-file equality), rubric (LLM judge over a written rubric), subjective (human-only, currently unused in scoring). Synthetic examples include regex over a generated heading (deterministic), a pass-if/fail-if rubric for concise explanatory prose (rubric), and aesthetic fluency (subjective, not scored).

**Universality (4).** Applicability: universal, cross-coding, cross-non-coding, specific. This axis governs which scenarios a constraint can be composed into: a completion summary can be universal, local validation can be cross-coding, an audience-specific call to action can be cross-non-coding, and a framework toggle can be scenario-specific.

**Surface fit.** Each constraint carries a per-surface suitability score {none, low, medium, high} indicating whether it can be placed in each surface, preventing semantically impossible placements. For example, a synthetic tool-parameter rule fits naturally in TD or PF, may be stated globally in SP, and is less natural as an ad hoc UI request.

**Surface variants.** Each applicable surface has a pre-authored rendering that preserves semantics while matching that surface's role. For a synthetic compact-summary rule, SP may state "Keep generated summaries compact," PF may state "Output style: compact summaries," and UI may request "Return a compact completion summary."

The (family $\times$ modality $\times$ prior $\times$ observability $\times$ verifiability $\times$ universality) combinatorial surface is $7 \times 7 \times 3 \times 4 \times 3 \times 4 = 7056$ cells. Our 642 constraints populate roughly 420 of these cells after deduplication on rhetorical content; the remaining cells are either semantically empty (e.g., forbid $\times$ subjective) or underpopulated in the current library.

### Authoring waves

A hash-pinned historical artifact verifies that public project-instruction and contributor documents informed the initial authoring wave, and a later commit preserves 13 human-promoted seed records whose mapping to current rules is not established. Subsequent LLM-assisted proposals and human review filled low-coverage taxonomy cells, including conditional requirements. Detailed source labels remain in the private provenance ledger for audit and rights review; the public release exposes coarse provenance classes and synthetic examples, preventing a direct join from a released rule to a named source repository. The library should therefore be read as empirically grounded benchmark content, not a repository census.

## B Scoring methods and cascade handling

Each rule in an item is scored by one of six methods, selected at authoring time based on what the constraint can be checked against:

**regex.** Surface pattern match on the final-turn message or specific files.

**ast.** AST structural match on generated code (e.g., function presence, decorator use, import ordering).

**cross-file.** Inter-file consistency check across the generated workspace.

**command-output.** Execute a provided compile-or-test script and grade the exit code / stdout.

**hybrid.** Deterministic pre-check (narrows candidates) followed by LLM refinement (resolves ambiguity).

**LLM-judge.** Rubric-based grading by an LLM judge reading the full run.

For LLM-judge and hybrid methods we use majority voting over three independent evaluations at temperature 0.3. The frozen coding evaluation uses GPT-5.2 as judge; robustness to this choice is summarized in the main paper's reliability subsection and examined in full in Section E below.

*Severity labels.* Every rule instance carries a severity of MUST, SHOULD, or MAY, and the released records retain both the unweighted status and the severity-weighted earned/possible totals (weights 3, 2, and 1 respectively). All metrics reported in this paper are unweighted binary rates, so severity affects none of the displayed numbers; it is retained for downstream users who want a weighted view and for the per-item MUST gate used during item authoring.

*Denominators and exclusions.* Acc and AP-Acc use pass/fail opportunities only. The frozen panel contains 40,104 method rows: 29,176 pass, 8,440 fail, 2,320 no-opportunity, and 168 partial. The like-for-like binary recomputation retains the 37,616 pass/fail verdicts; AP-Acc uses the 19,449 eligible verdicts whose final rule label is against-prior. Denominators vary by agent and are retained in the evidence snapshot. These counts apply only to the 12-model, 60-item, three-round coding panel; non-coding, E0, and online panels use separate populations and metrics.

*All-model common support.* To remove model-specific denominator differences, we additionally restrict to identical $(\text{item}, \text{round}, \text{rule})$ observations with a clean pass/fail outcome for all 12 models. This sensitivity retains 2,430 of 3,342 unique observations (72.7%), including 1,214 against-prior observations across 58 items. We report both views: released-set scores preserve the benchmark panel, while common support enables paired model and Acc–AP-Acc comparisons on an identical denominator.

*Cascade-dedup.* A single missing artifact can cause many dependent rules to fail simultaneously. The _-dedup-cascade pass retains one highest-severity fail and converts the remaining dependent outcomes to no-opportunity, preventing one missing artifact from multiplying failures.

*Cascade-fairness audit.* After a full batch, audit-cascade-fairness promotes a rule to untestable-design-gap when $\geq 50\%$ of agents miss the artifact needed to test it. Such rules are excluded from denominators. The audit requires at least 5 tested agents.

# C Failure decomposition

The main paper's failure-analysis subsection reports the decomposition by what the violated rule demands. This section gives the full table, the family breakdown, and the reason this grouping replaces a free-text taxonomy.

*Method.* Every rule in the library carries a logical modality, so each failure can be assigned to a class from the rule definition alone, with no inspection of the judge's reason string. Rules that require an action (REQUIRE, CONDITIONAL_REQUIRE) or set a floor (LIMIT_MIN) can only be failed by falling short of the demand; rules that forbid an action (FORBID) or set a ceiling (LIMIT_MAX) can only be failed by overstepping it; PREFER and ALLOW rules express soft preferences. The decomposition is therefore fully recomputable from the released verdict records and the released rule definitions, and it involves no tuned keywords, no per-model adjustment, and no catch-all bucket.

| Class | Failures | Share | Fail rate |
|---|---|---|---|
| Shortfall | 6,507 | 77.1% | 23.8% |
| Overstep | 1,758 | 20.8% | 20.8% |
| Preference | 175 | 2.1% | 9.4% |
| Total | 8,440 | 100.0% | 22.4% |

*Mass versus propensity.* The share column and the failure-rate column tell different stories, and only the second is a statement about agent behavior. Shortfall rules carry 3.7 times the failure mass of overstep rules, but they also account for 27,306 of the eligible verdicts against 8,443 for overstep rules, and the two classes fail at nearly the same rate (23.8% versus 20.8%). The asymmetry in observed failures is therefore a property of what operational rule sets ask for—mostly actions—rather than evidence that agents are markedly more prone to omission than to excess. Soft preferences are the one class that is genuinely easier, failing at 9.4%.

*Family-conditional distribution.* Failure mass by family is output control 27.6%, workflow 26.3%, code style 15.7%, conditional logic 12.4%, tool use 9.5%, and quantitative limits 8.6%. Output control combines the largest share of failures with the lowest pass rate; workflow's share instead reflects its size in the panel, since its pass rate is mid-range. A pooled accuracy number obscures this family-specific structure.

*Relation to the retired keyword taxonomy.* Earlier drafts reported a twelve-bucket taxonomy assigned by a keyword classifier over judge reason strings, including a 19.2% unclassified catch-all. We retired it because its bucket shares could not be regenerated from the released artifacts and because the reason strings are multilingual free text in which negation cues appear in nearly every failure, making keyword assignment unreliable. The decomposition above is reported instead: it is coarser, but every number in it can be recomputed by a reader from the shipped records.

# D Additional Experimental Detail

This appendix expands the analyses summarized in the main-text Experiments section. Unless a subsection is explicitly labeled E0, numbers are from the 12-agent × 60-item × 3-round coding evaluation (2,160 scoring records). Coding, non-coding, E0, and online panels are never pooled.

## D.1 Evaluated Model Builds

Table 3 gives the model identifier behind each row of the main leaderboard, as logged by the runner at collection time rather than reconstructed afterwards. Where a provider published a dated snapshot, the date is part of the identifier. The three Anthropic builds were served with their extended 1M-context variant. Two entries were preview releases when the panel was collected and have since been promoted, so the current catalogue names differ from what was run—`qwen/qwen3.6-max-preview` is now `qwen/qwen3.6-max`, and `tencent/hy3-preview:free` is now `tencent/hy3`; we record what was run. Seed-2.0-Pro was reached

through a provider-hosted deployment rather than a shared catalogue route, so the table gives its model identifier rather than a routing slug. The judge is `openai/gpt-5.2` throughout, at temperature 0.3 with three-vote majority.

**Table 3** Model identifier behind each leaderboard row, as logged at collection time.

| Displayed name | Model identifier |
|---|---|
| Claude-Opus-4.7 | `anthropic/claude-4.7-opus-20260416` |
| Claude-Sonnet-4.6 | `anthropic/claude-4.6-sonnet-20260217` |
| Claude-Haiku-4.5 | `claude-haiku-4-5-20251001` |
| GPT-5.5 | `openai/gpt-5.5` |
| Gemini-3.1-Pro | `google/gemini-3.1-pro-preview` |
| Qwen-3.6-Max | `qwen/qwen3.6-max-preview` |
| Hy3 | `tencent/hy3-preview:free` |
| Kimi-K2.6 | `moonshotai/kimi-k2.6` |
| MiniMax-M2.7 | `minimax/minimax-m2.7` |
| GLM-5.1 | `z-ai/glm-5.1` |
| StepFun-3.5 | `stepfun/step-3.5-flash` |
| Seed-2.0-Pro | `seed-2.0-pro` |

## D.2 Per-Surface and Per-Family Accuracy

Table 4 reports pooled accuracy for each configurable surface, and Table 5 reports pooled accuracy for each constraint family. Both are recomputed from the released panel over the same 37,616 eligible verdicts as the main table. The user-instruction (UI) surface carries only against-prior placements in this panel (1,476 of 1,476 eligible verdicts), so its rate is not comparable with the other surfaces, whose placements are mixed; it is reported for completeness rather than as a surface-difficulty estimate.

**Table 4** Pooled accuracy by instruction surface, with the number of eligible verdicts. PF is the project file (e.g. `CLAUDE.md`); SD is the skill description; TD the tool description; SP the system prompt; UI the user instruction.

| Surface | Pooled Acc | $N$ |
|---|---|---|
| TD (tool description) | 83.1% | 3,934 |
| PF (project file) | 79.1% | 11,756 |
| SD (skill description) | 78.6% | 11,072 |
| SP (system prompt) | 73.6% | 6,589 |
| UI (user instruction)[†] | 54.5% | 1,476 |

[†]All UI placements in this panel are against-prior, unlike every other surface.

**Table 5** Pooled accuracy by constraint family, with the number of eligible verdicts and the per-build range. Output control is lowest overall and for 11 of the 12 builds; quantitative limits are highest.

| Family | Mean Acc | $N$ | Per-build range |
|---|---|---|---|
| QT (quantitative) | 82.6% | 4,168 | 74.9–92.4 |
| TU (tool use) | 81.6% | 4,352 | 77.7–90.6 |
| CL (conditional) | 80.9% | 5,489 | 78.1–88.2 |
| CS (code style) | 79.2% | 6,342 | 68.8–93.5 |
| WF (workflow) | 76.0% | 9,265 | 71.0–84.1 |
| OC (output control) | 70.9% | 8,000 | 63.9–78.9 |

## D.3 Cross-Agent Agreement and Clusters

Agents largely agree on which constraints are hard, though the strength of that agreement depends on the comparison used. Correlating each build's per-rule pass-rate vector with the cohort mean over the 242 rules

every build attempted gives 0.57–0.89 (mean 0.80); the stricter build-to-build pairwise correlation gives 0.32–0.83 (mean 0.62). The leaderboard therefore reflects a largely shared difficulty structure with real per-build idiosyncrasy on individual rules.

Under the two-cluster grouping recorded in the release (5 builds versus 7), mean accuracy differs by +5.9 points (81.2% versus 75.3%) and mean AP-Acc by +5.5 points (75.1% versus 69.7%). Prior control therefore does not explain the cluster difference: the gap is almost unchanged when the comparison is restricted to against-prior rules. Pairwise cosine similarity between per-build modality×surface profile vectors stays in $[0.987, 0.999]$ over the 16 cells with at least 20 eligible verdicts, so the clusters differ in level rather than in the shape of their compliance profile, and we treat them as a weak behavioral gradient rather than discrete agent types.

### D.4 Separate E0 Surface-Precedence Pilot

The main surface-rank figure comes from E0, a distinct pilot collected on 2026-04-25 using nine older model builds, 916 recorded runs, four synthetic conflict pairs, and deterministic-only scoring. The nine builds are Claude Opus 4.6, Claude Sonnet 4.6, GPT 5.4, Gemini 3.1 Pro, DeepSeek V3.2, Kimi K2.5, MiniMax M2.7, Qwen 3.6 Plus, and Seed 2 Pro; four of them do not appear in the main coding panel, which is one reason E0 is not pooled with it. E0 predates the current surface naming; its two legacy channel labels map onto the canonical set as CM → project file (PF) and SK → skill description (SD), and we report E0 in canonical terms throughout. Ordinal ranks are recomputed from each model's head-to-head win counts and averaged equally across models despite unequal focused rerun counts: SP/PF/UI = 2.22, TD = 3.78, and SD = 4.56. These ranks describe precedence under the synthetic conflicts, not coding accuracy, and are not pooled with the main 12-model panel. The two views therefore order the skill description differently and are not in conflict: E0 asks which surface wins when two surfaces demand opposite things, and places SD last; the main panel asks how often a rule is followed when it is the only instruction, where SD reaches 78.6% above SP at 73.6%. A surface can be easy to comply with in isolation and still lose a direct conflict, so the E0 ranking is a statement about precedence and the panel rates are statements about difficulty.

*Identification and uncertainty.* E0 uses counterbalanced assignment rather than paired outcomes: each AB and BA run is a separate fresh session, with the mutually exclusive members of one rule pair assigned to opposite surfaces for the same fixed task. The design contains 458 assigned runs per direction; after 27 errors, 445 AB and 444 BA runs are decisive. On these 889 outcomes, a pooled Bradley–Terry fit gives sum-to-zero log-strengths SP = +0.57, PF = +0.65, UI = +0.60, TD = −0.53, and SD = −1.29. Because model and conflict pair are crossed factors, the primary uncertainty analysis independently resamples the nine models and four pairs (10,000 fixed-seed replicates), rather than treating 36 model–pair cells as independent clusters. The complete SP/PF/UI > TD > SD ordering appears in 9,652/10,000 crossed-bootstrap resamples (top-group-above-TD: 98.81%; TD-above-SD: 97.71%); these are Monte Carlo diagnostics conditional on the resampling scheme, not $p$-values. The three leading surfaces are not statistically distinguished at the available precision.

The pooled ordering survives both direction-specific fits (445/444 decisive rows), all four leave-one-pair-out fits, all nine leave-one-model-out fits, and equal weighting of the 694 model×pair×direction×surface-matchup cells (repeat range 1–5). It also survives four deterministic assignments of every error—channel A wins, channel B wins, the higher expected group wins, or the lower expected group wins. Errors are concentrated in DeepSeek V3.2 (14) and Seed 2 Pro (10), so this explicit bound is preferable to assuming outcome-independent deletion. Heterogeneity remains material: all four pair-only fits, but only six of nine model-only fits, reproduce the exact ordering. We therefore interpret E0 as a robust pooled tendency within four synthetic style conflicts and nine older builds, not a build-invariant hierarchy or evidence for the main panel's descriptive surface strata. AB was executed before BA in the retained collection; the counterbalanced assignment addresses rule identity but the fixed temporal order cannot be corrected retrospectively.

The pooled ordering survives both direction-specific fits (445/444 decisive rows), all four leave-one-pair-out fits, all nine leave-one-model-out fits, and equal weighting of the 694 model×pair×direction×surface-matchup cells (repeat range 1–5). It also survives four deterministic assignments of every error—channel A wins,

channel B wins, higher expected tier wins, or lower expected tier wins. Errors are concentrated in DeepSeek V3.2 (14) and Seed 2 Pro (10), so this explicit bound is preferable to assuming outcome-independent deletion. Heterogeneity remains material: all four pair-only fits, but only six of nine model-only fits, reproduce the exact tier ordering. We therefore interpret E0 as a robust pooled tendency within four synthetic style conflicts and nine older builds, not a build-invariant hierarchy or evidence for the main panel's descriptive surface strata. AB was executed before BA in the retained collection; the counterbalanced assignment addresses rule identity but the fixed temporal order cannot be corrected retrospectively.

### D.5 Surface × Family Interaction

The main coding panel does not provide a paired surface intervention: constraints are assigned to admissible surfaces rather than counterbalanced across all placements, and surface wording and family composition can co-vary. We therefore treat its surface-by-family cells as descriptive strata rather than a variance decomposition or controlled surface effect. A historical held-out analysis reports 648 additional conflict runs, Kendall's $\tau = 0.91$, and an 89% stronger-group win rate, but its row-level artifact is not retained. We treat those figures as descriptive historical evidence only, not as a robustness result.

## E Full reliability analysis

The main paper's reliability subsection summarizes the reliability of the reported results. This appendix provides the full stratified numbers, ablations, and method details. Figure 7 shows the reported per-model Accuracy and AP-Acc and the descriptive gap between them.

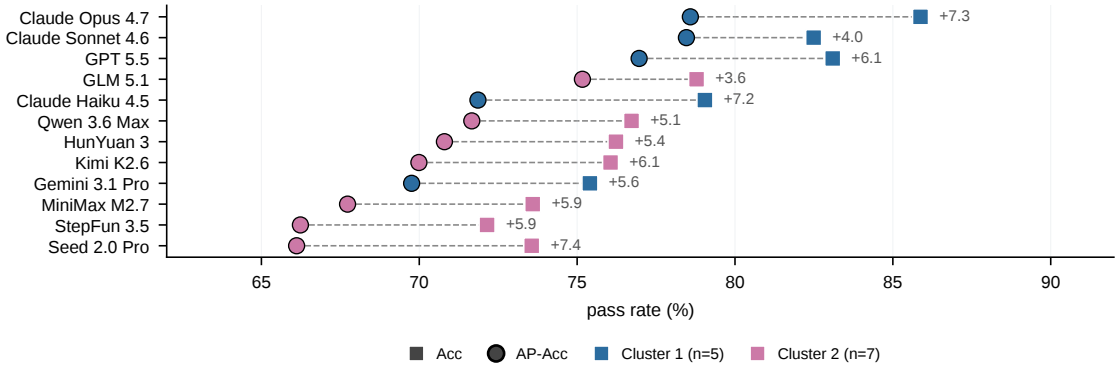


**Figure 7** Per-agent Accuracy (square) and AP-Acc (circle), sorted by AP-Acc, both recomputed from the released panel under the same binary definition. The dashed span is the Acc–AP-Acc prior-alignment gap, positive for all 12 agents. Marker fill encodes the two behavioral clusters described under experimental detail.

### E.1 Judge calibration and judge-swap sensitivity

The frozen coding panel uses a 3-vote protocol for rubric and hybrid methods, with the majority outcome as the consensus verdict. Two retained calibration studies answer different questions. First, on 65 common non-error samples, Claude–GPT four-class inter-LLM agreement is $\kappa = 0.4717$, whereas auto-vs-Claude is 0.0332 and auto-vs-GPT is $-0.0491$; these are inter-LLM comparisons, not human-reference results, and expose sensitivity around no-opportunity boundaries. Second, a historical human-reference audit of a related five-vote auto-scoring configuration records a three-class confusion matrix over $n = 919$ rows, with 69.0% observed agreement and $\kappa = 0.515$. This provides moderate human-reference calibration for the judge family, not direct validation of the exact frozen three-vote panel. The matrix arithmetic and per-case disagreement reasons are retained, but the raw joined labels, the sampling frame and probabilities, rater assignments, and the independent adjudication record are not, so it is not a fully auditable human calibration study and we report it as qualified evidence only.

*Judge-swap ablation (Claude Opus 4.7 in place of GPT 5.2).* We re-judge a stratified 200-verdict subset (balanced across family × modality × prior cells) with Claude Opus 4.7 as the judge, using the same 3-vote protocol. Both judges produce a clean pass/fail/partial label for 116 rows; 84 rows fall outside the paired comparison because at least one side is no-opportunity, an error, or lacks evidence. On the paired subset, raw agreement is 62.1% and Cohen's $\kappa = 0.163$; per-agent pass-rate deltas range from $-40$ pp (Seed) to $+33$ pp (Kimi, MiniMax). This sensitivity motivates our emphasis on cross-model patterns and common-support uncertainty rather than narrow leaderboard margins. The retained release includes the aggregate judge-swap summary, but not row-level swap verdicts, so an alternate-judge prior-alignment gap is unavailable.

## E.2 Test-retest (ICC) detail

**Table 6** Test-retest stability across the three rounds, by stratum.

| Stratum | ICC(1,1) |
|---|---|
| Agent-level (12 agents, pooled cells) | 0.725 |
| Cell-level (agent × item) | 0.599 |
| Mean per-cell Acc range (across rounds) | 15.6 pp |

Round-to-round dispersion is substantial at the cell level: the mean per-cell Acc range across the three rounds is 15.6 points, which exceeds the 13.7-point spread of the whole leaderboard. Single-round comparisons between adjacent models are therefore uninformative, and all reported numbers pool three rounds.

## E.3 Scoring-method composition and ranking robustness

Method shares are computed over the 37,616 eligible pass/fail verdicts of the released panel. Kendall $\tau$ compares each method-specific per-agent ranking with the overall ranking; $N$/agent is the mean number of eligible verdicts per agent. The judge and the deterministic–judge hybrid together cover 86.7% of eligible verdicts, so the judge-swap sensitivity reported above applies to most of the measurement.

**Table 7** Per-method agreement with the overall ranking (Kendall's $\tau$), mean eligible verdicts per agent, each method's share of the 37,616 eligible verdicts, and its pass rate. The judge and the deterministic–judge hybrid together cover 86.7% of eligible verdicts.

| Method | $\tau$ vs. overall | $N$/agent | Share | Pass rate |
|---|---|---|---|---|
| LLM-judge | +0.91 | 2,147 | 68.5% | 72.6% |
| hybrid | +0.72 | 570 | 18.2% | 90.8% |
| regex | +0.79 | 275 | 8.8% | 86.1% |
| command-output | +0.39 | 84 | 2.7% | 84.1% |
| ast | +0.23 | 31 | 1.0% | 89.3% |
| cross-file | +0.00 | 28 | 0.9% | 76.3% |

## E.4 Prior-label lineage and sensitivity

The current governed distribution is 115 align-prior, 282 against-prior, and 245 neutral. The recoverable 5/9 zero-injection consensus covers 287 rules (106/170/11); a separate historical report describes a 280-rule binary determined subset (114/166), which is not interchangeable with either the consensus count or the final 282 against-prior labels. Lineage matches the zero-injection consensus for 275 rules, matches a recoverable pre-existing value for 331, and remains unknown for 36; 12 final labels differ from recoverable consensus without a preserved override reason.

*Probe cohort overlap with the evaluated panel.* Because the against-prior set is defined partly from observed model behavior, we state how much that behavior overlaps the models being scored. The zero-injection votes come from a nine-build probe cohort recorded in the frozen lineage snapshot, which ships with the release. Seven of the twelve evaluated models—Claude Haiku 4.5, GLM-5.1, GPT-5.5, Hy3, Kimi K2.6, Qwen 3.6 Max,

and StepFun-3.5—are absent from that cohort and therefore contributed no prior labels. The against-prior gap is positive for all seven, averaging +5.63 points (range +3.62 to +7.19), against +6.06 points for the five that share an identifier with a probe build. The 0.43-point difference between the two groups is small relative to the effect itself and, on twelve models, is not resolved in either direction; we report it as a bound rather than as evidence of independence. Two further limits apply. Because only four probe builds sit outside the evaluated panel's identifier set, a 5/9 consensus necessarily includes at least one overlapping build, so no zero-injection label is independent of the scored cohort. And absence from the probe cohort is not vendor independence: GPT-5.5, Kimi K2.6, and Qwen 3.6 Max are successor builds of probe models, leaving GLM-5.1, Hy3, and StepFun-3.5 as the only evaluated models with no same-vendor probe build; their mean gap is +4.99 points. What does bound the exposure is the label provenance itself: of the 19,449 against-prior eligible verdicts, 8,574 (44.1%) carry a label sourced from the zero-injection consensus, against 9,051 (46.5%) from prior curation and 1,824 (9.4%) of unknown lineage. Fewer than half of the AP-Acc denominator can therefore be affected by the overlap at all.

Under the paper-like binary equations, the frozen full panel contains 37,616 eligible pass/fail verdicts, of which 19,449 are against-prior; no-opportunity and other statuses are excluded. The resulting agent-macro mean Acc–AP-Acc gap is +5.8076 points and is positive for all 12 agents. The deterministic-only subset contains 5,013 eligible verdicts (977 against-prior) and gives +13.0933 points, also positive for all agents; larger historical estimates are not retained.

The available threshold sweep relabels the recoverable zero-injection consensus subset. Under the same binary definition, the mean gap is +11.8161 points at 4/9 and 5/9, +10.6022 at 6/9, and +10.2438 at 7/9; the 4/9–6/9 change is 1.2139 points. The direction is therefore stable across the tested thresholds.

### E.5 Common-support uncertainty

Common support requires a clean pass/fail outcome from every model for the same (item, round, rule) key. It retains 2,430/3,342 unique observations (72.7%), including 1,214 against-prior observations over 58 items. We use a deterministic 2,000-resample item-clustered percentile bootstrap (seed 20260723), retaining all common-support rounds and rule rows within each sampled item. The paired Acc–AP-Acc interval is positive for every model; lower bounds range from +1.06 to +4.83 points. By contrast, all adjacent common-support Acc intervals include zero, yielding one conservative adjacent tie group containing all 12 models. Thus the prior-alignment contrast is resolved model by model, whereas neighboring point ranks are not.

## F Failure gallery

Five failures from the frozen scoring panel illustrate the benchmark's main findings. To preserve source abstraction, the vignettes use paper-local labels and behavioral paraphrases rather than stable library identifiers or verbatim constraint text.

**Example A: skill-surface placement miss.** Two agents created the requested skill artifact but placed it outside the required project convention. The same rule was followed more often when delivered through a project file (100% of 144 eligible verdicts) than through the skill description (67.0% of 103). These are distinct rule instances in distinct items rather than a counterbalanced pair, so the contrast is descriptive.

**Example B: tool-surface workflow miss.** An agent produced a free-form change summary rather than the required structured form when the workflow rule appeared in a tool description. The rule is placed only on the tool description in this panel, where it is followed in 91.9% of its 172 eligible verdicts, so this is an instance failure on an otherwise well-followed rule rather than evidence about the surface.

**Example C: against-prior output failure.** Two agents generated an additional documentation artifact despite an explicit prohibition. The task naturally invited explanatory output, so compliance required overriding a common default rather than merely completing the requested implementation.

**Example D: output-control mirroring.** An agent copied language and formatting cues from the surrounding fixture into its final response, violating a response-language constraint. This reflects the surface-mirroring pattern common in the output-control family.

**Example E: budget overshoot.** Two agents produced artifacts substantially beyond a stated size cap. The result illustrates failure on a quantitative constraint when unconstrained generation favors longer outputs.

## G Exploratory Non-Coding Extension

*Scope and protocol.* We evaluated 40 non-coding cases, eight in each of five domains: customer support, legal/compliance, marketing content, financial analysis, and research/academic writing. The panel comprises 12 models, each assigned three generator runs per case (1,440 attempted trajectories). These tasks primarily produce natural-language artifacts rather than repository patches; their language, tools, interaction horizons, and rule ontology therefore differ from the coding panel. The model builds are those recorded by the frozen Stage-5 evaluation, not necessarily the identically named builds in later panels; two rows use the Stage-5 display names Hunyuan 3 and StepFun Flash, which correspond to the coding leaderboard's Hy3 and StepFun-3.5.

*Metric.* For accepted run $r$ of case $c$ and model $m$, let $E_{cmr}$ be the eligible rule verdicts after the evaluation's exclusions and let $p_{cmr} = |E_{cmr}|^{-1} \sum_{j \in E_{cmr}} \mathbf{1}[j = \text{PASS}]$. We first average $p_{cmr}$ over the valid runs for each case–model pair, then average the 40 case means equally:

$$\text{NC-Macro}_m = \frac{1}{40} \sum_{c=1}^{40} \frac{1}{|R_{cm}|} \sum_{r \in R_{cm}} p_{cmr}. \tag{5}$$

This case-macro construction prevents cases with more rubric checks from dominating. Empty-shell trajectories are excluded; a non-empty trajectory with at least 25% judge errors uses its clean eligible-rule rate, following the frozen evaluation protocol. Confidence intervals in Table 8 are percentile intervals from 20,000 fixed-seed (20260714), domain-stratified bootstrap replicates that resample cases within each domain and retain each sampled case's observed run aggregate. Cases, rather than runs or individual rule verdicts, are the resampling units.

**Table 8** Exploratory non-coding results. NC-Macro and its 95% case-bootstrap interval are percentages. $\sigma_r$ is the mean within-case run standard deviation; $\sigma_c$ is the cross-case standard deviation. Valid is out of 120 attempted trajectories per model.

| Model | NC-Macro [95% CI] | $\sigma_r$ | $\sigma_c$ | Valid |
|---|---|---|---|---|
| GPT 5.5 | 84.8 [83.0, 86.6] | 3.0 | 6.6 | 120 |
| GLM 5.1 | 78.7 [76.6, 80.7] | 4.5 | 8.2 | 120 |
| Claude Opus 4.7 | 78.4 [75.4, 81.2] | 4.2 | 9.8 | 116 |
| Gemini 3.1 Pro | 76.9 [74.5, 79.1] | 6.1 | 8.9 | 120 |
| Claude Sonnet 4.6 | 76.1 [73.5, 78.6] | 5.3 | 8.9 | 119 |
| Hunyuan 3 | 73.3 [70.4, 76.4] | 6.5 | 10.4 | 120 |
| Qwen 3.6 Max | 72.4 [69.7, 74.9] | 6.3 | 9.0 | 119 |
| Claude Haiku 4.5 | 71.3 [69.0, 73.5] | 6.0 | 8.5 | 120 |
| Seed 2 Pro | 69.6 [66.6, 72.7] | 4.5 | 11.7 | 120 |
| Kimi K2.6 | 68.6 [65.3, 71.9] | 7.6 | 11.4 | 114 |
| MiniMax M2.7 | 68.5 [65.6, 71.3] | 6.2 | 10.3 | 120 |
| StepFun Flash | 65.8 [63.0, 68.6] | 6.6 | 10.8 | 120 |

*Coverage and variability.* Of 1,440 attempted trajectories, 1,428 (99.2%) were valid. The 12 invalid trajectories were empty long-context network shells and were excluded rather than scored. Eleven case–model cells consequently have incomplete replication: ten retain two valid runs and one retains a single valid run. Their means use the available accepted runs; $\sigma_r$ uses the observed sample size and is undefined for the single-run cell. Table 8 reports both run-wise and cross-case variation because the latter is generally larger and the 40 cases, not the 1,428 trajectories, define the external-validity sample. Per-domain case-macro pass rates appear in Table 9.

*Non-comparability and provenance.* NC-Macro is not AP-Acc: it uses a different task population, ontology, scoring/exclusion protocol, and model-build snapshot. We therefore neither pool the 40 cases with the 60

**Table 9** Case-macro pass rate (%) by non-coding domain; each cell averages eight case means. Model order follows overall NC-Macro, not the coding leaderboard.

| Model | Customer support | Legal/ compliance | Marketing | Financial analysis | Research/ academic |
|---|---|---|---|---|---|
| GPT 5.5 | 80.5 | 84.8 | 84.9 | 90.2 | 83.7 |
| GLM 5.1 | 73.7 | 82.2 | 77.7 | 85.8 | 73.8 |
| Claude Opus 4.7 | 76.8 | 80.2 | 75.6 | 81.9 | 77.3 |
| Gemini 3.1 Pro | 72.4 | 77.7 | 77.8 | 85.1 | 71.3 |
| Claude Sonnet 4.6 | 72.5 | 78.1 | 73.2 | 81.4 | 75.3 |
| Hunyuan 3 | 72.2 | 70.7 | 71.7 | 80.9 | 71.1 |
| Qwen 3.6 Max | 71.2 | 72.3 | 74.5 | 75.9 | 67.9 |
| Claude Haiku 4.5 | 67.0 | 68.1 | 72.4 | 78.8 | 70.1 |
| Seed 2 Pro | 69.4 | 68.5 | 73.6 | 77.5 | 59.3 |
| Kimi K2.6 | 71.8 | 71.7 | 66.2 | 71.4 | 62.1 |
| MiniMax M2.7 | 65.7 | 66.9 | 67.5 | 77.5 | 64.8 |
| StepFun Flash | 64.3 | 65.7 | 64.6 | 75.8 | 58.4 |

coding items nor use this table to revise the paper's coding leaderboard. This historical analysis is also distinct from the July 2026 online archive, which contains new-build, single-trial operational runs and is not an additional round of either panel. The values above come from a frozen 12-model Stage-5 aggregate snapshot preserved in the release provenance. Complete raw trajectories no longer survive in the current repository, so the snapshot supports the stated aggregate-level claims while trajectory-level regeneration remains outside scope. This provenance boundary, together with protocol and build differences, makes the analysis exploratory rather than confirmatory.

## Datasheet

Following Gebru et al. [7].

*Motivation.* The benchmark was created to evaluate instruction following in coding agents across multiple instruction-placement surfaces (system prompt, tool description, skill description, project file, user message, plus a harness default)—a gap left by existing IF benchmarks that only test user-message prompts.

*Composition.* The 80-candidate disposition retains 60 items, deprecates 15, and excludes 5 after quality review. Several per-item promotion and review artifacts are not recoverable. The diagnostic sampler is separate from and does not define the final assembled items. The release contains three explicitly separate evaluation populations. The frozen coding core comprises: (i) a library of 642 atomic constraints authored in YAML and tagged along 8 axes (family, modality, prior, observability, verifiability, universality, surface fit, and surface variants); (ii) 60 assembled coding items (quality-audited from an 80-item working set), each combining a scenario fixture, an injected pack of 25–35 rules of which 10–27 are scorable given the opportunities the item creates, a surface assignment, a multi-turn task specification, and a ground-truth scoring script; across the panel 302 distinct library rules are instantiated and 256 receive at least one verdict, and professional-writing rules are exercised only in the non-coding panel; and (iii) 2,160 scoring records (12 agents $\times$60 coding items $\times$3 rounds) with per-rule pass/fail/no-opportunity status and judge reasons. A frozen exploratory non-coding panel separately contains 40 cases across five domains (eight cases each), with 1,440 attempted trajectories (12 models $\times$40 cases $\times$3 generator runs) and 1,428 valid trajectories. Its case-macro rule pass rate averages valid runs within each case and then weights the 40 case means equally; it is not pooled with coding AP-Acc. A July 2026 online operational audit is a third, new-build single-trial panel covering 60 coding items and 40 general cases; outages and API-quality flags are retained, and its metrics are not comparable to either frozen panel. Constraints include natural-language text; items include source code fixtures and task specifications.

*Collection process.* A hash-verified manual survey of publicly accessible GitHub project-instruction and contributor documents seeded the constraint library. Surveyed material included `CLAUDE.md`, `AGENTS.md`, `CONTRIBUTING.md`, skill descriptions, and tool schemas across multiple software stacks. Requirements were atomized and normalized into project-authored benchmark constraints rather than copied as repository files. LLM-assisted proposals and author review subsequently filled taxonomy gaps. A later commit preserves 13 human-promoted historical seed records in Git history, although their mapping to current rules is not recoverable, so the retained labels do not support a rule-level transformation ledger. Detailed source labels are retained privately for audit and rights review; the public artifact uses coarse provenance classes and synthetic paper examples rather than repository-shaped labels or source-searchable quotations. Coding workspaces were assembled for realistic technology stacks and are project-authored, owner-attested on 2026-07-27 after a mechanical provenance audit of all 785 fixture files found no copied third-party material; earlier drafts held them back pending that provenance and redistribution rights are cleared. No crowdworkers were used.

*Preprocessing / cleaning / labelling.* The retained evidence does not support a comprehensive construction-time deduplication or PII-removal claim. Current prior labels comprise 115 `align_prior`, 282 `against_prior`, and 245 `neutral` rules. A reproducible zero-injection 5/9 consensus covers a distinct 287-rule subset; a historical report's 280-rule determined subset is reported-only and lacks a retained row-level manifest. Where label lineage or override rationale is not recoverable, the frozen lineage snapshot records it as unknown rather than inferring it.

*Uses.* Intended for benchmarking instruction-following performance of coding agents and for validating surface-aware and prior-aware evaluation methodologies. Not intended as a certification instrument, agent capability certification, or hiring/procurement decision tool.

*Data availability.* A planned public release will contain the 642 project-authored constraints, active item specifications, surface assignments, ground-truth scoring scripts, sanitized derived verdicts for the frozen 2,160-record coding panel, analysis code and snapshots, Croissant metadata, and provenance documentation. The executable coding workspaces are included following the 2026-07-27 owner attestation; raw provider traces remain excluded until their privacy review is complete, which affects end-to-end trajectory replay but not inspection of the constraint library, item design, scoring logic, or published aggregate analyses.

*Distribution.* The release does not grant rights to or publicly redistribute raw third-party repository snapshots, provider outputs, request traces, or unreviewed third-party material. Project-authored software is intended for distribution under Apache-2.0 and project-authored benchmark text/data plus sanitized derived results under CC-BY-4.0; the file-level rights ledger and exporter allowlists govern actual inclusion.

*Maintenance.* Maintained by the authors; after public release, issues and contributions will be handled through the public repository. The benchmark may be updated (new constraints, items, or agent evaluations); versioned via manifest.yaml.

*Known biases.* The constraint library reflects an English-language, software-engineering-centered manual survey plus targeted expansion; free-form source labels do not establish a repository sampling frame. The 12-agent evaluation sample consists of generally available commercial and public API models at time of writing; it does not include specialized coding-only models or fine-tuned variants. Coding-item selection used observed pilot/full-panel discriminativeness and difficulty diagnostics, which may favor separation among the observed models (selection optimism). Any training-provenance correspondence in model behavior should be interpreted descriptively rather than causally.

*Legal and ethical considerations.* Human reviewers assessed model outputs and automated verdicts during quality control; the retained historical comparison is reported as qualified calibration evidence, and no rater identities or personal data are released. The evaluation did not collect behavioral or demographic data from

human subjects. Public exports must pass fail-closed privacy and rights checks and exclude credentials, private endpoints, request identifiers, raw provider responses, hidden reasoning, unreviewed personal data, and material without established redistribution rights. Public availability of a source does not by itself establish redistribution permission.