HOW GOOD IS YOUR HARNESS? STATISTICAL EVALUATION OF CODING HARNESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agent benchmarks conflate the underlying language model with the harness that wraps the model in tools, prompts, and control flow. We develop a simple statistical method to disentangle the effects of the LLM and its harness on the agent's score; *i.e.*, to attribute the variation in agents' scores to their LLMs and harnesses. We use the method to evaluate harnesses and LLMs on Terminal-Bench 2.0, and our results show that

1. the harness can matter as much as an LLM upgrade: the range of harness effects is comparable to the gain from Claude Opus 4.1 to Claude Opus 4.6.
2. the harness effects can be heterogeneous; *i.e.*, some harnesses work better with some LLMs than with others.
3. the effects of individual harness features can also vary across LLMs.

Our results confirm and, more importantly, quantify practitioners' intuition on the importance of the harness and validate efforts to tailor harnesses to specific applications.

1 INTRODUCTION

Terminology AI agents consist of an LLM and a *harness*: a supporting framework of prompts, tools, and control logic that enables the agent to interact with its *in silico* environment. The terms agent and harness are sometimes used interchangeably, but we shall distinguish between them in the rest of this paper; *i.e.*, we always use the term agent to refer to a combination of an LLM with a harness.

When an agent fails/succeeds at a task, the outcome does not directly inform us whether the failure/success is due to the harness or LLM (or some interaction between the two). There is a growing body of work that examines the agent's traces/trajectories (in addition to the outcome) to obtain more detailed diagnoses of the agent's failures/successes (see Cemri et al. (2025); Zhang et al. (2025); Barke et al. (2026); Ma et al. (2024) and follow-up works). Such methods typically rely on LLMs to annotate/label the agent's traces with failure analyses before clustering/taxonomizing the labels.

In this paper, we develop an alternative purely statistical approach that relies on the variation in agents' scores across different harness–LLM pairs to attribute the variation to the harness and the LLM. As we shall see, the method is simple and general, and can be applied to any benchmark with a sufficiently crossed design of agents; *i.e.*, the scores of enough harness–LLM pairs are observed. Using the method, we study the agents on the Terminal-Bench 2.0 leaderboard (Merrill et al., 2026) and find that

1. the harness can matter as much as an LLM upgrade: the range of harness effects is comparable to the gain from Claude Opus 4.1 to Claude Opus 4.6.
2. the harness effects can be heterogeneous: OpenHands' advantage over Mini-SWE-Agent is larger for the evaluated OpenAI models than for their Anthropic and Google counterparts.

The first finding confirms practitioners' intuition that the harness is not a minor implementation detail, but rather a critical component of the agent's performance. That said, the sizes of the harness effects are surprising: even though LLMs still differ more overall, a better harness can pay off as much as upgrading to a newer flagship LLM!

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.

The second finding suggests that the harness and LLM are not perfectly exchangeable components: some harnesses may be better suited for some LLMs than others. The sizes of the interaction effects are also surprising: they're comparable in size to the harness and LLM (main) effects!

The leaderboard findings motivate a practical question: which harness features are useful for a given LLM and task set? To answer this question, we design and conduct ablation experiments with the little-coder harness (Inbar, 2026a), varying individual harness features, *i.e.*, modular components that can be enabled or disabled. We find that

3. the effects of individual harness features vary across LLMs: quality monitoring helps agents solve Terminal-Bench 2.0 tasks, but the size of this benefit depends on the LLM.

The third finding shows that the choice of LLM matters even when selecting individual harness components. By quantifying feature contributions on the target tasks, our approach provides an empirical basis for deciding which features to build into a harness.

2 Additive harness–LLM decomposition

Model To decompose the variation in agent scores into harness effects and LLM effects, we consider the leaderboard as a table with rows and columns corresponding to harnesses and LLMs respectively: each cell contains the score of an agent consisting of the harness and the LLM corresponding to the cell's row and column respectively. We fit an additive model to the scores, which posits that the score of each agent can be expressed as the sum of a baseline effect, a harness effect, and an LLM effect. More concretely, if $Y_{i,j}$ is the score of the agent formed by harness i and LLM j, we model its mean as

$$g\big(\mathbf{E}\big[Y_{i,j}\big]\big) = \mu + \alpha_i + \beta_j, \tag{2.1}$$

where g is an invertible link function, μ is the baseline score on the link scale, α_i is the additive effect of harness i, and β_j is the additive effect of LLM j.[1] The choice of the link function g depends on the nature of the scores. For example, if the scores are pass rates (proportions), a common choice is the logit link, which models the log-odds of passing as a linear function of the harness and LLM effects. In the rest of this paper, we use the logit link for the pass rates on Terminal-Bench 2.0 (but the method can be adapted to leaderboards with different score types by changing the link function) (McCullagh & Nelder, 1989).

Methods For the logit link, we interpret μ as the baseline log-odds of success for the reference agent, α_i as the average (over LLMs) log-odds change from changing the baseline harness to harness i, and β_j as the average (over harnesses) log-odds change from changing the baseline LLM to LLM j. Equivalently, $\exp(\alpha_i)$ and $\exp(\beta_j)$ are multiplicative odds ratios relative to the corresponding baseline harness/LLM. We fit Equation 2.1 to data with off-the-shelf logistic regression software.

Data Terminal-Bench 2.0 evaluates coding agents on 89 terminal-based tasks, with 5 trials per task. Each leaderboard entry reports an agent's accuracy $Y_{i,j} \in [0, 1]$, averaged over tasks and trials. We use the first five recorded trials of each task; some entries record fewer, including three with only one trial per task, and for these we use all recorded trials. The leaderboard snapshot we collected contains 118 entries after excluding entries listing "Multiple" as the model. After normalizing duplicate LLM names, this corresponds to 39 LLM levels, but the design is sparse and unbalanced: many harnesses are evaluated on only one or two LLMs, and many LLMs are evaluated under only one or two harnesses. We keep the LLMs evaluated with more than one harness, leaving 105 harness–LLM pairs across 28 harnesses and 26 LLMs. This filter removes two entries with inseparable harness and LLM effects, as well as 11 singleton-LLM entries whose removal leaves the retained coefficients unchanged. Although the resulting design remains sparse, its design matrix is full rank, so the additive effects in Equation 2.1 are identifiable under the reference constraints.

[1] Writing $f(i, j) = g(\mathbf{E}[Y_{i,j}])$ and viewing it as a function of the two attributes harness identity and LLM identity, Equation 2.1 is the first-order (main-effects-only) functional decomposition $f(i, j) = f_\emptyset + f_{\mathrm{H}}(i) + f_{\mathrm{L}}(j)$, with $f_\emptyset = \mu$, $f_{\mathrm{H}}(i) = \alpha_i$, $f_{\mathrm{L}}(j) = \beta_j$. In our two-attribute reference game these first-order components are exactly the Shapley contributions of harness and LLM identity on the link scale. See Appendix D.1 for the derivation and Bordt & von Luxburg (2023) for the correspondence between generalized additive models and Shapley explanations.

Table 1: Largest fitted harness effects (left) and LLM effects (right). Coefficients are log-odds shifts relative to the Mini-SWE-Agent reference harness and the Claude Sonnet 4.5 reference LLM. Standard errors are model-based. Bold coefficients have 95% confidence intervals excluding zero.

Harness	Coef ± SE	Entries
Ante	$\mathbf{0.926} \pm 0.129$	1
ForgeCode	$\mathbf{0.890} \pm 0.114$	2
Capy	$\mathbf{0.656} \pm 0.127$	1
TongAgents	$\mathbf{0.653} \pm 0.110$	2
Terminus-KIRA	$\mathbf{0.588} \pm 0.109$	2

LLM	Coef ± SE	Entries
GPT-5.3-Codex	$\mathbf{1.072} \pm 0.076$	7
Gemini 3.1 Pro	$\mathbf{0.929} \pm 0.111$	3
Claude Opus 4.6	$\mathbf{0.836} \pm 0.063$	10
GPT-5.2	$\mathbf{0.568} \pm 0.075$	4
Claude Opus 4.5	$\mathbf{0.507} \pm 0.059$	8

Overall results We fit the additive model in Equation 2.1 to the Terminal-Bench 2.0 leaderboard. Table 1 reports the five largest fitted harness and LLM coefficients (Appendix Figure 1 shows all of them), and Table 6 reports the five highest-scoring LLMs when evaluated with the Terminus 2 harness. The fitted effects are adjusted comparisons from an observational leaderboard, not causal estimates.

We choose Mini-SWE-Agent (Yang et al., 2024) as a minimal harness baseline because it uses only bash actions and a linear message history, with no specialized tool interfaces or elaborate control flow. We choose Claude Sonnet 4.5 as the reference LLM because it has the highest reported accuracy among the 13 LLMs evaluated with Mini-SWE-Agent in our collected leaderboard (42.5%). The references are only a coding convention: changing them shifts every harness (or LLM) coefficient by the same constant, leaving the rankings and the differences between coefficients unchanged. Throughout the paper, the leaderboard harness and LLM effects should be read relative to these references; the subsequent feature analyses use their separately specified baselines.

Our central empirical result comes from comparing the harness effects with familiar LLM upgrades. The fitted contrast between ForgeCode and Mini-SWE-Agent is 0.89 logits, comparable to the contrast between Claude Opus 4.6 and Claude Sonnet 4.5 (0.84). More broadly, the harness effects span 1.12 logits, from Gemini CLI (-0.20) to Ante (0.93), comparable to recent progress within a flagship line: 1.01 logits from Claude Opus 4.1 to Claude Opus 4.6 and 1.15 from GPT-5 to GPT-5.3-Codex. The LLM effects, however, span a wider range: 4.10 logits overall, from GPT-OSS-20B (-3.02) to GPT-5.3-Codex (1.07), and 3.07, 2.12, and 1.47 logits within the proprietary GPT, Gemini, and Claude families.

Ante has the largest estimated harness effect, but it is observed with only one LLM, so its coefficient may also reflect a pairing-specific interaction; Section 3 finds significant interactions for some harness–LLM pairs. ForgeCode has the largest estimated effect among harnesses with more than one entry.

Per-category results The Terminal-Bench 2.0 tasks are grouped into 16 categories, seven of which contain at least five tasks (data science, debugging, file operations, scientific computing, security, software engineering, and system administration). For each task category, imagine a leaderboard in the same shape as the overall leaderboard but with the entries computed as the average accuracy across tasks in that category. We can fit the same additive model in Equation 2.1 to each category-specific leaderboard to obtain category-specific harness and model effects.

We fit within each of these seven categories by computing each pair's average accuracy across tasks in that category. Table 2 shows the top three harness coefficients by category, with standard errors shown after ±. As in Table 1, standard errors are model-based; sandwich standard errors give the same significance calls (Appendix C.1). For file operations, GPT-OSS-20B has no successes in 50 trials, causing separation in the full design. We remove this model level and its 50 outcomes before fitting; the remaining design is full rank and passes the separation check.

ForgeCode ranks first in three of the six non-debugging categories. The exceptions are data science and scientific computing, where Ante has the largest estimated effect, and security, where SageAgent leads. This suggests that many strong harnesses are broadly useful, but harness design can also confer category-specific strengths.

Table 2: Top fitted harness effects by task category. The debugging category is omitted because it exhibits complete or quasi-complete separation and admits no finite regular MLE. Standard errors are model-based; bold coefficients have 95% confidence intervals excluding zero.

Category	Rank 1	Rank 2	Rank 3
Data science (8)	Ante $\mathbf{1.395} \pm 0.425$	ForgeCode $\mathbf{1.330} \pm 0.367$	Simple Codex $\mathbf{1.113} \pm 0.420$
File operations (5)	ForgeCode $\mathbf{2.121} \pm 0.494$	TongAgents $\mathbf{1.679} \pm 0.467$	Capy $\mathbf{1.140} \pm 0.502$
Scientific computing (8)	Ante $\mathbf{1.310} \pm 0.425$	Simple Codex 0.848 ± 0.448	OpenCode 0.828 ± 0.757
Security (8)	SageAgent $\mathbf{1.161} \pm 0.383$	Terminus-KIRA $\mathbf{1.083} \pm 0.401$	Codex CLI $\mathbf{1.062} \pm 0.213$
Software engineering (26)	ForgeCode $\mathbf{1.102} \pm 0.226$	Ante $\mathbf{1.065} \pm 0.238$	Terminus-KIRA $\mathbf{0.793} \pm 0.215$
System administration (9)	ForgeCode $\mathbf{0.902} \pm 0.374$	CAMEL-AI $\mathbf{0.771} \pm 0.385$	Capy 0.722 ± 0.422

3 SATURATED HARNESS–LLM DECOMPOSITION

Implicitly, Equation 2.1 assumes that the harness and LLM effects are separable on the log-odds scale; *i.e.*, there is no interaction between the harness and LLM effects. We now relax this assumption by adding the harness–LLM interaction term.

Model To probe whether certain harness–LLM pairs exhibit synergy beyond their additive main effects, we extend Equation 2.1 with an interaction term:

$$g\big(\mathbf{E}\big[Y_{i,j}\big]\big) = \mu + \alpha_i + \beta_j + \gamma_{ij}, \tag{3.1}$$

where γ_{ij} is the interaction effect of harness i and LLM j. As in Section 2, we take g to be the logit link. Under our reference coding, the interaction is a difference-in-differences in log-odds: $\gamma_{ij} = \eta_{ij} - \eta_{i,j_0} - \eta_{i_0,j} + \eta_{i_0,j_0}$, where $\eta_{ij} = g(\mathbf{E}[Y_{i,j}])$, i_0 denotes Mini-SWE-Agent, and j_0 denotes Claude Sonnet 4.5. It measures how the harness contrast changes relative to the reference LLM, rather than a residual from the separately fitted additive model. Including all harness–LLM interactions gives a saturated model on a fully observed block.[2] Unlike Equation 2.1, which we could fit to the sparse leaderboard in Section 2, each γ_{ij} is identified only if the agents (i, j), (i, j_0), (i_0, j), and (i_0, j_0) are all observed; a fully observed block guarantees this for every pair.

Data For the interaction model, we use the largest fully observed submatrix of the leaderboard of Terminal-Bench 2.0: the harnesses Mini-SWE-Agent, OpenHands, and Terminus 2 crossed with nine LLMs evaluated under all three harnesses, yielding 27 agents. We use up to five recorded trials per task on the same 89-task benchmark. We compute an overall accuracy for each agent and repeat index from the available task outcomes, yielding 135 aggregate rows. Weighting each row by its number of observed tasks gives a binomial GLM equivalent to fitting the 11,941 recorded binary outcomes directly. As in the additive analysis, Mini-SWE-Agent and Claude Sonnet 4.5 serve as references; interaction terms involving either reference level are zero by construction.

Overall results We fit the interaction model in Equation 3.1 to the largest fully observed subset of the Terminal-Bench 2.0 leaderboard. Table 3 reports the fitted interaction coefficients γ_{ij} for OpenHands and Terminus 2 across nine LLMs, with standard errors shown after $\pm$. Mini-SWE-Agent is omitted because it is the reference harness, and Claude Sonnet 4.5 has zero interaction by construction as the reference model. A likelihood-ratio test rejects additivity in this block ($G^2 = 91.2$ on 16 degrees of freedom, $p < 10^{-11}$); this test does not depend on which harness and LLM serve

[2] In the decomposition of the previous footnote, γ_{ij} is the second-order component $f_{\mathrm{HL}}(i, j)$ that Equation 2.1 sets to zero, so Equation 3.1 is the complete decomposition $f(i, j) = f_\emptyset + f_{\mathrm{H}}(i) + f_{\mathrm{L}}(j) + f_{\mathrm{HL}}(i, j)$; since harness and LLM identity are the only two attributes, no components of order higher than two exist. Ordinary Shapley values distribute this second-order component equally between the two attributes, giving contributions $\alpha_i + \gamma_{ij}/2$ and $\beta_j + \gamma_{ij}/2$ (Bordt & von Luxburg, 2023). See Appendix D.1 for the derivation.

Table 3: Interaction coefficients in the complete 3 by 9 block. Values are reference-based interaction contrasts on the log-odds scale. Standard errors are model-based. Bold coefficients have 95% confidence intervals excluding zero.

Model	OpenHands	Terminus 2
Claude Haiku 4.5	$\mathbf{-0.951} \pm 0.219$	-0.087 ± 0.201
Claude Opus 4.1	0.081 ± 0.195	0.103 ± 0.195
Claude Sonnet 4.5	0.000 ± 0.000	0.000 ± 0.000
GPT-5	$\mathbf{0.425} \pm 0.194$	0.048 ± 0.196
GPT-5-Mini	0.373 ± 0.206	0.096 ± 0.209
GPT-5-Nano	0.404 ± 0.280	0.127 ± 0.290
Gemini 2.5 Flash	-0.042 ± 0.225	-0.026 ± 0.225
Gemini 2.5 Pro	$\mathbf{-0.582} \pm 0.216$	0.296 ± 0.201
Grok 4	0.103 ± 0.204	-0.122 ± 0.208

as references. The departures come from OpenHands ($G^2 = 63.7$, 8 df, $p < 10^{-10}$) rather than Terminus 2 ($G^2 = 5.5$, 8 df, $p = 0.71$). Averaged over GPT-5, GPT-5-Mini, and GPT-5-Nano, OpenHands' interaction exceeds its average over the five Claude and Gemini LLMs by 0.70 ± 0.13 logits.

The largest harness–LLM interaction contrasts are comparable in magnitude to the fitted harness effects in Table 1, showing that a harness's contribution can vary substantially with the LLM it supports. OpenHands (Wang et al., 2025) exhibits the largest interaction contrasts. Relative to Claude Sonnet 4.5, its interactions are positive for GPT-5, GPT-5-Mini, and GPT-5-Nano (0.37 to 0.43) and negative for Claude Haiku 4.5 and Gemini 2.5 Pro (-0.95 and -0.58). The provider contrast above confirms this pattern: OpenHands' advantage over Mini-SWE-Agent is larger for the evaluated OpenAI LLMs. One possible explanation is that differences in how LLMs handle tool calls interact with OpenHands' action interface, though the success-rate analysis does not identify this mechanism.

On the other hand, Terminus 2, the default Terminal-Bench 2.0 harness, shows smaller interaction contrasts overall, and all of its non-reference intervals include zero. Terminus 2 parses actions from plain-text responses and sends keystrokes to a tmux session rather than using native tool calling. This design may reduce its sensitivity to model-specific tool-calling conventions, though this explanation remains speculative.

4 From harnesses to harness features

The preceding analyses show that harness choice matters and that its effects can vary across LLMs and task categories, suggesting the value of tailoring a harness to both the underlying LLM and the target tasks. To guide this design process, we propose an ablation-based experimental design to identify which features contribute to performance for a given LLM and task set.

Experiment Design We treat each optional harness feature as an on/off switch and randomly sample combinations of these switches to construct harness configurations, keeping the underlying harness structure fixed. For each target LLM, we run the resulting harness configurations on the same benchmark set and record their success rates under a common evaluation protocol.

Model We consider a harness with M configurable features evaluated with J LLMs. We model each agent's mean score on the link scale as the sum of an LLM-specific baseline and the contributions of its enabled features. Let $Y_{i,j}$ denote the score of the agent formed by harness configuration i and LLM j, where $u_i \in \{0,1\}^M$ indicates which features are enabled, with $u_{ik} = 1$ exactly when feature k is enabled. We model its mean as

$$g\big(\mathbf{E}[Y_{i,j}]\big) = \beta_j + \sum_{k=1}^{M} u_{ik} v_{jk} = \beta_j + u_i^\top v_j, \tag{4.1}$$

where g is the link function, β_j is the LLM-specific baseline on the link scale with all M features disabled, and $v_j = (v_{j1}, \ldots, v_{jM})^\top$ is an LLM-specific vector of feature coefficients. Any fixed

infrastructure is shared across configurations and excluded from u_i. As in Sections 2 and 3, we take g to be the logit link, so β_j is the baseline log-odds and v_{jk} is the log-odds change for LLM j from enabling feature k with the other features held fixed. Equivalently, $\exp(v_{jk})$ is a multiplicative odds ratio for that feature.

Equation 4.1 replaces a separate effect for each harness configuration with a sum of individual feature contributions. The LLM-specific coefficient vector v_j allows these contributions to vary across LLMs. To separate average feature contributions across the evaluated LLMs from LLM-specific departures, define

$$\bar{v} = \frac{1}{J}\sum_{j=1}^{J} v_j, \qquad \delta_j = v_j - \bar{v}, \qquad \sum_{j=1}^{J} \delta_j = 0. \tag{4.2}$$

It follows that $u_i^\top v_j = u_i^\top \bar{v} + u_i^\top \delta_j$. At the configuration level,

$$\alpha_i^{\mathrm{FL}} = u_i^\top \bar{v}, \qquad \gamma_{ij}^{\mathrm{FL}} = u_i^\top \delta_j \tag{4.3}$$

play roles analogous to the harness main effect and harness–LLM interaction in Sections 2 and 3. The superscript FL indicates that these effects are constructed from additive feature attribution. Accordingly, $\bar{v}$ is the vector of harness feature main effects, averaged equally across LLMs, and δ_j is the vector of feature–LLM interaction effects for LLM j, expressed as departures from those averages.

Data We study little-coder, originally developed and evaluated on Aider Polyglot (Inbar, 2026b;a), as a case study in adapting a harness to other LLMs and a different benchmark. Its publicly available feature implementations and configurations allow us to evaluate different feature combinations. We sample 100 harness configurations for each of three LLMs: Gemma-4-26B-A4B (Gemma Team, 2026), K2-Horizon-MoVA-36B-A4B (IFM Team, 2026), and Qwen3.6-35B-A3B (Qwen Team, 2026). Each configuration toggles ten feature groups while keeping infrastructure (INFRA) enabled, and is evaluated on the same 89 Terminal-Bench 2.0 tasks. Feature descriptions appear in Appendix Table 7.

We restrict the analysis to configurations with shell session (SS) enabled, since it provides the shell tool required for task execution,[3] leaving nine optional feature groups ($M = 9$): browser (BR), checkpointing (CP), extra tools (ET), evidence management (EV), knowledge and skill injection (KS), output parsing (OP), quality monitoring (Q), thinking budget (TB), and write protection (W). Including baseline and full configurations for Shapley-value analysis in Section 4.1, the final dataset contains 171 profiles: 58 for Gemma, 50 for K2, and 63 for Qwen. For each LLM–configuration pair, we record a binary success outcome for each of the 89 tasks.

With only INFRA and SS enabled, Gemma, K2, and Qwen solve $13/89$ (14.6%), $37/89$ (41.6%), and $26/89$ (29.2%) tasks respectively; with all nine optional features enabled they solve $17/89$ (19.1%), $41/89$ (46.1%), and $21/89$ (23.6%), spanning a wide range of baseline capability. Across all retained configurations, accuracy ranges from 7.9% to 20.2% for Gemma, 31.5% to 46.1% for K2, and 21.3% to 34.8% for Qwen.

Overall results We fit the feature-linear model in Equation 4.1 to the 171 profiles. Table 4 reports the average feature components $\bar{v}_k$ and the LLM-specific departures δ_{jk} for all nine optional features.

Quality monitoring is the only feature whose average coefficient has a 95% interval excluding zero: $\bar{v}_{\mathrm{Q}} = 0.186$, about a fifth of the ForgeCode–Mini-SWE-Agent contrast in Section 2. The fitted LLM-specific effects correspond to predicted accuracy increases of approximately 1–5 percentage points at the observed baseline accuracies. The quality-monitor extension detects execution failures such as empty responses and repeated loops, and prompts the agent to retry (Inbar, 2026a). One possible explanation for its benefit is that these interventions prevent a recoverable failure from ending a multi-step terminal task or exhausting its execution budget. Quality monitoring may therefore help agents complete tasks that they could otherwise solve but fail to finish because execution breaks down.

[3] In our ablation experiments, configurations with SS disabled recorded 0 successes in 3,827 task outcomes for Gemma, 2 in 4,628 for K2, and 2 in 3,471 for Qwen.

Table 4: Average and LLM-specific feature components from the SS-conditioned binomial GLM. Entries are on the log-odds scale, with model-based standard errors shown after $\pm$. Bold coefficients have 95% confidence intervals excluding zero.

Feature	$\bar{v}_k$	$\delta_{\mathrm{Gemma},k}$	$\delta_{\mathrm{K2},k}$	$\delta_{\mathrm{Qwen},k}$
BR	$+0.016 \pm 0.041$	-0.024 ± 0.062	$+0.033 \pm 0.056$	-0.010 ± 0.057
CP	-0.006 ± 0.044	$+0.008 \pm 0.069$	-0.025 ± 0.058	$+0.018 \pm 0.059$
ET	-0.007 ± 0.041	$+0.002 \pm 0.063$	$+0.023 \pm 0.057$	-0.025 ± 0.056
EV	-0.020 ± 0.042	$+0.028 \pm 0.064$	-0.054 ± 0.060	$+0.026 \pm 0.055$
KS	-0.027 ± 0.042	$+0.039 \pm 0.064$	-0.084 ± 0.059	$+0.045 \pm 0.056$
OP	-0.029 ± 0.040	$+0.006 \pm 0.062$	$+0.008 \pm 0.056$	-0.014 ± 0.054
Q	$\mathbf{+0.186} \pm 0.042$	$+0.095 \pm 0.063$	$+0.028 \pm 0.058$	$\mathbf{-0.123} \pm 0.056$
TB	$+0.017 \pm 0.042$	$+0.063 \pm 0.064$	$+0.044 \pm 0.057$	-0.108 ± 0.056
W	-0.000 ± 0.040	-0.021 ± 0.061	-0.005 ± 0.056	$+0.026 \pm 0.054$

Quality monitoring contributes significantly less for Qwen than the three-LLM average ($\delta_{\mathrm{Qwen,Q}} = -0.123$); its total fitted effect remains positive but is not significant. The other LLM-specific departures have intervals that include zero. The variation in quality monitoring may reflect differences in failure frequency, self-recovery, or responsiveness to corrective feedback, but this remains a hypothesis rather than a conclusion of the success-rate analysis.

Apart from quality monitoring, every average coefficient is within 0.03 logits of zero. The benchmark interface helps interpret these small estimates. In the release we evaluated (`v0.1.27`; Appendix B), the adapter exposes only shell-session tools, limiting the scope of five feature groups: BR and ET provide tools outside the allowed set, CP and W target standalone file tools, and EV's context hooks depend on evidence or browser outputs that are not normally produced in this setting. These features remain in the analysis as part of the evaluated configuration space and provide a useful sanity check. Their small estimates are consistent with their limited role under this interface: none of their 35 components ($\bar{v}_k$, v_{jk}, and δ_{jk}) lies more than 1.02 standard errors from zero. The remaining three features operate through the system prompt (KS) or turn-level hooks (OP, TB), but show no detectable average effect at the precision of this experiment.

4.1 Shapley attribution of harness features

We reanalyze the same 171 profiles, each the set $S_i = \{k : u_{ik} = 1\}$ of features it enables. The feature-linear model in Equation 4.1 assumes that, for each LLM, feature effects add on the log-odds scale, so it excludes interactions among harness features; its coefficients are the Shapley values of this additive game (Appendix D.1). To estimate Shapley attributions without imposing additivity, we apply KernelSHAP (Lundberg & Lee, 2017) directly to the observed coalition log-odds.

Cooperative game We cast harness configuration as a cooperative game (Shapley, 1953) whose *players* are the harness feature groups. The player set $N = \{1, \ldots, M\}$ consists of the nine optional feature groups ($M = 9$); INFRA and SS remain enabled throughout and are therefore not players. A *coalition* $S \subseteq N$ is the harness configuration that enables exactly the features in S, and its *value* $V_j(S)$ is the performance of LLM j under that configuration, defined below. The *baseline* configuration is the empty coalition ($S = \varnothing$), with all nine optional features disabled; the *full* configuration is the grand coalition ($S = N$), with all nine enabled.

For LLM j, define the coalition value on the log-odds scale as

$$V_j(S) = \mathrm{logit}(p_j(S)), \qquad p_j(S) = \mathbf{E}\big[Y_j(S)\big], \tag{4.4}$$

where $Y_j(S)$ is the mean success score on the fixed set of 89 benchmark tasks and the expectation is over run-level randomness. A feature's marginal contribution can depend on the other enabled features; its Shapley value averages these coalition-dependent contributions.

For each evaluated coalition, we observe $X_j(S)$ successes among $n_j(S) = 89$ tasks. The empirical game uses the observed success proportions:

$$\widehat{p}_j(S) = \frac{X_j(S)}{n_j(S)}, \qquad \widehat{V}_j(S) = \mathrm{logit}\{\widehat{p}_j(S)\} = \log \frac{X_j(S)}{n_j(S) - X_j(S)}. \tag{4.5}$$

Table 5: KernelSHAP attributions on the log-odds scale (INFRA and SS enabled), with $\bar{v}_k$ from Table 4 for comparison. Standard errors after $\pm$ account for binomial noise in the coalition values and for random coalition selection (Appendix D.3); bold entries have 95% intervals excluding zero.

Feature	$\phi_k^{(\text{Gemma})}$	$\phi_k^{(\text{K2})}$	$\phi_k^{(\text{Qwen})}$	$\bar{\phi}_k$	$\bar{v}_k$
BR	$+0.000 \pm 0.134$	$+0.088 \pm 0.093$	$+0.022 \pm 0.107$	$+0.037 \pm 0.065$	$+0.016 \pm 0.041$
CP	-0.106 ± 0.163	$+0.018 \pm 0.192$	-0.049 ± 0.204	-0.045 ± 0.108	-0.006 ± 0.044
ET	-0.088 ± 0.113	-0.107 ± 0.117	-0.099 ± 0.114	-0.098 ± 0.066	-0.007 ± 0.041
EV	$+0.143 \pm 0.228$	$+0.104 \pm 0.208$	$+0.034 \pm 0.169$	$+0.094 \pm 0.117$	-0.020 ± 0.042
KS	$+0.092 \pm 0.252$	-0.036 ± 0.112	$+0.047 \pm 0.111$	$+0.034 \pm 0.099$	-0.027 ± 0.042
OP	-0.009 ± 0.112	-0.038 ± 0.094	-0.093 ± 0.117	-0.047 ± 0.062	-0.029 ± 0.040
Q	$\mathbf{+0.357} \pm 0.107$	$+0.134 \pm 0.107$	$+0.030 \pm 0.082$	$\mathbf{+0.174} \pm 0.058$	$\mathbf{+0.186} \pm 0.042$
TB	$+0.030 \pm 0.117$	-0.100 ± 0.132	-0.154 ± 0.216	-0.075 ± 0.093	$+0.017 \pm 0.042$
W	-0.095 ± 0.109	$+0.120 \pm 0.208$	-0.029 ± 0.095	-0.001 ± 0.084	-0.000 ± 0.040
Sum	$+0.322$	$+0.183$	-0.290	$+0.072$	—

All observed SS-enabled configurations have success rates strictly between zero and one, so their empirical logits are finite.

Methods Given the empty and grand coalitions, the sampled-coalition KernelSHAP approximation solves the constrained weighted regression

$$\min_{\phi_0^{(j)},\,\phi^{(j)}} \sum_{\substack{S\in\mathcal{D}_j \\ 0<|S|<M}} \pi(S)\left[\widehat{V}_j(S) - \phi_0^{(j)} - \sum_{k\in S}\phi_k^{(j)}\right]^2, \qquad \pi(S) = \frac{M-1}{\binom{M}{|S|}|S|(M-|S|)}, \tag{4.6}$$

subject to

$$\phi_0^{(j)} = \widehat{V}_j(\varnothing), \qquad \sum_{k=1}^{M}\phi_k^{(j)} = \widehat{V}_j(N) - \widehat{V}_j(\varnothing). \tag{4.7}$$

The first constraint fixes the baseline and the second enforces Shapley efficiency: the nine feature attributions sum to the empirical log-odds difference between the full and baseline configurations. With all 2^M coalitions, this regression recovers the Shapley values of the empirical game (Appendix D.2); with the observed subset of the $2^9 = 512$ coalitions, it is an approximation whose accuracy depends on the evaluated coalitions (Olsen & Jullum, 2026).

Overall results Table 5 reports the attributions $\phi_k^{(j)}$ fitted separately for each LLM and their equally weighted average $\bar{\phi}_k$. Quality monitoring has the largest estimated average contribution under both analyses and is the only feature whose average interval excludes zero in both. Its KernelSHAP estimates are positive for all three LLMs, although only Gemma's individual interval excludes zero. The LLM-specific patterns also broadly agree: knowledge and skill injection has its lowest attribution for K2, while quality monitoring and thinking budget have their lowest attributions for Qwen. EV, KS, and TB change sign on average between methods, but their intervals include zero under both, so these reversals remain inconclusive.

The feature GLM pools information across configurations under an additive specification, whereas KernelSHAP allows feature interactions at the cost of wider intervals. Most of that width is noise in the coalition values, amplified by the endpoint constraint and the concentrated kernel weights; the estimated coalition-selection component is small (Appendix D.3). The additive GLM gives the more precise summary, and both analyses support a positive average contribution from quality monitoring.

5 Summary and Discussion

The additive model in Equation 2.1 provides a practical way to quantify the value-add of a harness to LLM performance on agentic tasks. On Terminal-Bench 2.0, replacing a basic harness with the best one improves the odds of success about as much as upgrading from Claude Sonnet 4.5 to

Table 6: Top LLMs evaluated with the Terminus 2 harness on Terminal-Bench 2.0. The right-most column is the rank change relative to the additive LLM-effect ranking; positive values mean the LLM ranks higher under Terminus 2 than by additive LLM effect.

Rank	LLM	Score	Δ
1	GPT-5.3-Codex	64.7% ± 2.7%	0
2	Claude Opus 4.6	62.9% ± 2.7%	+1
3	Claude Opus 4.5	57.8% ± 2.5%	+2
4	Gemini 3 Pro	56.9% ± 2.5%	+2
5	GPT-5.2	54.0% ± 2.9%	-1

Claude Opus 4.6. Our findings have consequences for both benchmark interpretation and harness development/engineering. For example, it is common in model release reports to compare the new LLM's scores on agentic benchmarks with those of competitor LLMs (*e.g.*, Kimi Team (2025)). Such comparisons are misleading because they fail to adjust for harness effects, and they risk overstating/understating a model's ability depending on the harness used for evaluation. On the other hand, our findings confirm reports that suggest it is possible to gain substantial accuracy by improving the harness while keeping the underlying LLM fixed (Lee et al., 2026; Pryzant et al., 2023; Romera-Paredes et al., 2024; Novikov et al., 2025; Lee et al., 2025; Agrawal et al., 2026).

That said, the interaction analysis shows that the simple additive model can be a useful first-order approximation, but harnesses and models are not perfectly exchangeable components. The benefits of more full-featured harnesses may (unintentionally) depend on the underlying LLM due to seemingly unimportant design choices (*e.g.*, how tool calls are issued and parsed), but the benefits of minimal harnesses that parse actions from plain text, such as Terminus 2, seem to be more model agnostic. This justifies emerging harness engineering efforts that develop harnesses to make up for deficiencies in the tool calling capabilities of LLMs. For example, ForgeCode's model-specific fixes—schema field ordering, flatter schemas, explicit truncation notices, and enforced reviewer-mode verification—brought both GPT-5.4 and Opus 4.6 to 81.8% on Terminal-Bench 2.0 at the time of that report (ForgeCode, 2026).

Our ablation study shows that individual harness features contribute differently across LLMs, suggesting that their benefits should be reevaluated when models or tasks change. In little-coder, quality monitoring has the largest average contribution, but its contribution is smallest for Qwen among the three LLMs; Qwen also has the lowest thinking-budget contribution. For example, Rajasekaran (2026) reports removing sprint-based task decomposition when adapting an application-development harness from Opus 4.5 to Opus 4.6, while retaining the planner and evaluator because they continued to add value; other studies report similar reuse across model versions and task settings (Xu et al., 2026; Zhang et al., 2026; Lee et al., 2026). Little-coder illustrates the same point: developed against Aider Polyglot and later evaluated on Terminal-Bench (Inbar, 2026a), it gains from quality monitoring on the new benchmark, while five of its features have limited scope under the benchmark's shell-only interface, indicating which components to retain and which to reconsider.

Last (but not least), this paper focuses on harness evaluation, so we spent most of the paper interpreting the harness effects, but the LLM effects are also revealing. Terminal-Bench 2.0 evaluates LLMs together with the Terminus 2 harness (see Table 6) but Terminus 2 is a very basic harness, so the scores may not reflect their abilities to take advantage of the features of more advanced harnesses. On the other hand, the LLM effects in Table 1 do depend on the LLMs' scores on more advanced harnesses, and we see some changes in LLM rankings between Tables 1 and 6. This suggests that the scores with the Terminus 2 harness in Table 6 may not reflect the more advanced capabilities of LLMs.

AI USE STATEMENT

In this work, we used generative AI tools for retrieval and discovery, namely searching for related work, and for research execution, namely writing the code that implements our experiments and analyses. We have not used generative AI tools for research ideation, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to aid and polish writing. We have reviewed all AI-assisted work: AI-suggested references were checked against their original sources, AI-assisted code was verified and tested for correctness, and AI-polished text was compared with the author-written original, with each of these checks carried out by multiple authors over multiple passes. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI agent failures from execution trajectories, 2026. URL https://arxiv.org/abs/2602.02475.

Sebastian Bordt and Ulrike von Luxburg. From Shapley values to generalized additive models and back. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 709–745, 2023. URL https://proceedings.mlr.press/v206/bordt23a.html.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.1093/biomet/39.3-4.324.

A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller. Robust inference with multiway clustering. *Journal of Business & Economic Statistics*, 29(2):238–249, 2011. doi: 10.1198/jbes.2010.07136.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia, Joseph Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4082.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465, 2021. URL https://proceedings.mlr.press/v130/covert21a.html.

ForgeCode. Benchmarks don't matter – until they do (part 2). https://forgecode.dev/blog/gpt-5-4-agent-improvements/, March 2026. Accessed 2026-05-02.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Peter J. Huber. The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 221–233, 1967.

IFM Team. Introducing K2 Horizon: Frontier performance, radically open. https://ifm.ai/blog/k2/, September 2026. Accessed 2026-09-25.

Itay Inbar. little-coder. https://github.com/itayinbarr/little-coder, 2026a. Accessed 2026-09-16.

Itay Inbar. Honey, I shrunk the coding agent. https://itayinbarr.substack.com/p/honey-i-shrunk-the-coding-agent, April 2026b. Accessed 2026-09-16.

Kimi Team. Kimi K2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback Descent: Open-ended text optimization via pairwise comparison, 2025. URL https://arxiv.org/abs/2511.07919.

Yoonho Lee, Roshen Sanjay Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=tmbOUyFx3R.

Hongwei Li, Zhun Wang, Qinrun Dai, Yuzhou Nie, Jinjun Peng, Ruitong Liu, Jingyang Zhang, Kaijie Zhu, Jingxuan He, Lun Wang, Yangruibo Ding, Yueqi Chen, Wenbo Guo, and Dawn Song. OpenSage: Self-programming agent generation engine, 2026. URL https://arxiv.org/abs/2602.16891.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362, 2024. doi: 10.52202/079017-2365.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks, 2024. URL https://arxiv.org/abs/2406.10229.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: Evaluating LLMs with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326, 2024. URL https://proceedings.mlr.press/v235/maia-polo24a.html.

Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, second edition, 1989.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, 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 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, Christopher Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, pp. 40903–40986, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/444a3737adaee10d86ad2ef5f74468e6-Abstract-Conference.html.

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

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? A call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024. doi: 10.1162/tacl_a_00681.

Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Lars Henry Berge Olsen and Martin Jullum. Improving the weighting strategy in KernelSHAP. In *Explainable Artificial Intelligence*, volume 2577 of *Communications in Computer and Information Science*, pp. 194–218. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-032-08324-1_9.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all. https://qwen.ai/blog?id=qwen3.6-35b-a3b, April 2026. Accessed 2026-09-25.

Prithvi Rajasekaran. Harness design for long-running application development. https://www.anthropic.com/engineering/harness-design-long-running-apps, March 2026. Accessed 2026-09-25.

Pedro Rodriguez, Joe Barrow, Alexander Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4486–4503, 2021. doi: 10.18653/v1/2021.acl-long.346.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, pp. 25055–25083, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6c0e99d736da621403018ca7b32b1a4d-Abstract-Conference.html.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games, Volume II*, number 28 in Annals of Mathematics Studies, pp. 307–317. Princeton University Press, 1953. doi: 10.1515/9781400881970-018.

Mukund Sundararajan and Amir Najmi. The many Shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9269–9278, 2020. URL https://proceedings.mlr.press/v119/sundararajan20b.html.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, pp. 65882–65919, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a4b6ad6b48850c0c331d1259fc66a69c-Abstract-Conference.html.

Halbert White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–25, 1982. doi: 10.2307/1912526.

Tianshi Xu, Huifeng Wen, and Meng Li. Adapting the interface, not the model: Runtime harness adaptation for deterministic LLM agents, 2026. URL https://arxiv.org/abs/2605.22166.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601.

Scott L. Zeger, Kung-Yee Liang, and Paul S. Albert. Models for longitudinal data: A generalized estimating equation approach. *Biometrics*, 44(4):1049–1060, 1988. doi: 10.2307/2531734.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, Yizhou Jin, and Yan Xu. HarnessCompass: Guiding automatic harness evolution toward generalizable and effective agent harnesses, 2026. URL https://arxiv.org/abs/2608.01918.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

A Related Work

Agent harnesses. A growing body of work studies how the prompts, tools, and control logic surrounding an LLM shape agent performance. Approaches to improving these components include prompt optimization (Pryzant et al., 2023; Agrawal et al., 2026; Lee et al., 2025), program and agent search (Romera-Paredes et al., 2024; Novikov et al., 2025; Li et al., 2026), and harness optimization or adaptation (Lee et al., 2026; Zhang et al., 2026; Xu et al., 2026). Engineering reports document model-specific design choices (Rajasekaran, 2026; ForgeCode, 2026; Inbar, 2026b), and open-source harnesses make these choices accessible to evaluation (Yang et al., 2024; Wang et al., 2025; Inbar, 2026a). These efforts motivate the evaluation question we study: how much does a harness contribute to agent performance, and how does that contribution depend on the LLM? We address this question through a decomposition of leaderboard scores, then use little-coder ablations to examine the contributions of individual harness features.

Statistical evaluation. Prior work studies system rankings (Bradley & Terry, 1952; Chiang et al., 2024), item difficulty (Rodriguez et al., 2021; Maia Polo et al., 2024), score uncertainty (Miller, 2024; Madaan et al., 2024), and prompt sensitivity (Mizrahi et al., 2024; Sclar et al., 2024). We use standard generalized linear models (McCullagh & Nelder, 1989) to decompose variation in agent scores into harness and LLM effects. This requires a sufficiently crossed harness–LLM design and provides a basis for evaluating each component's contribution. Appendix C describes the variance estimators and robustness checks.

Shapley attribution. Shapley values allocate a cooperative game's value among its players (Shapley, 1953) and are widely used for model explanations (Lundberg & Lee, 2017; Sundararajan & Najmi, 2020). Our harness–LLM models admit a Baseline Shapley interpretation (Sundararajan & Najmi, 2020), with harness and LLM identity as players and a reference pair defining the baseline. The correspondence between Shapley explanations and functional decompositions (Bordt & von Luxburg, 2023) relates the main effects and their interaction to first- and second-order components (Appendix D.1). For individual harness features, we apply KernelSHAP (Lundberg & Lee, 2017) to estimate coalition-averaged marginal contributions, allowing interactions among features. These estimates use only the observed configurations; their accuracy as approximations to full-game Shapley values depends on coalition sampling and weighting (Covert & Lee, 2021; Olsen & Jullum, 2026).

B Little-Coder

B.1 Background and Architecture

Little-coder is a coding harness designed for small, locally served LLMs (Inbar, 2026b;a). It provides explicit mechanisms for editing, context selection, reasoning limits, and recovery to support models that may struggle with long-horizon planning or reliable tool use. Its modular, publicly available implementation makes it a useful case study for our feature analysis: we can vary these mechanisms while holding the LLM fixed, then compare their contributions across LLMs on the same tasks.

Reported performance. The developer reports a mean Aider Polyglot accuracy of 45.56% with Qwen3.5-9B across two 225-exercise runs, compared with 19.11% for vanilla Aider using the same model (Inbar, 2026b). With Qwen3.6-35B-A3B, the reported accuracy is 78.67% (177/225) (Inbar, 2026a). These results use the earlier Python implementation. From `v0.1.0`, little-coder uses Pi for its agent loop and model interface. For this implementation, the developer reports Terminal-Bench 2.0 accuracies of 24.6% with Qwen3.6-35B-A3B (`v0.1.13`) and 9.2% with Qwen3.5-9B (`v0.1.24`), each over 89 tasks with five trials per task (Inbar, 2026a). These results motivate studying the harness, but do not isolate individual feature contributions; the two Terminal-Bench entries also differ in harness version.

Execution and extensions. In the Pi-based implementation, the runtime sends the task instructions and conversation context to the LLM, executes its tool calls, and returns tool results to the conversation for subsequent generation. Little-coder extensions intervene at specific points in this loop. Knowledge and skill injection (KS) augments the system prompt at agent start. The thinking-budget extension (TB) monitors streamed reasoning and, when its estimated token budget is exceeded, aborts

generation and requests continuation with thinking disabled. Quality monitoring (Q) and output parsing (OP) inspect completed turns and can queue corrective follow-ups; on turns with tool calls, these checks occur after tool execution. Other extensions provide tools, preserve file contents, or manage evidence and browser results. Table 7 summarizes the feature groups used in our ablations. These descriptions and the table are based on the source of release `v0.1.27`, which the experiments used (Appendix B.2).

Benchmark interface. In `v0.1.27`, the Harbor adapter passes task instructions to Pi and permits only the shell-session tools (SS), which execute commands in the task environment, inspect the working directory, or reset the session. This restriction affects how the other extensions can operate. Write protection (W) and checkpointing (CP) apply to standalone file tools and do not intercept shell commands that modify files. Browser (BR), extra-tool (ET), and evidence (EV) tools are also outside the allowed set. EV's context hooks remain registered, but require stored evidence or browser outputs to have an effect. The available mechanisms must therefore be distinguished from those exercised under this benchmark interface.

Table 7: Feature mechanisms in little-coder `v0.1.27`. EV groups `evidence`, `evidence-compact`, and `browser-extract-retention`; KS groups `knowledge-inject` and `skill-inject`. Other codes each toggle one extension.

Code	Mechanism and trigger
BR	Registers browser navigation, interaction, and extraction tools.
CP	Saves a file's original contents on its first qualifying standalone write or edit call in a session.
ET	Registers additional file search and web retrieval tools.
EV	Stores evidence, supplies a compaction reminder, and prunes older browser outputs from context.
KS	Selects algorithm references and tool guidance at agent start, using lexical matches and error, recency, and intent signals.
OP	Detects tool calls embedded in completed text responses and requests native tool calls in a follow-up.
Q	Inspects completed turns for empty responses and problematic response or tool-call patterns, then requests correction.
SS	Provides shell execution, working-directory inspection, and reset tools through the benchmark adapter.
TB	Monitors streamed reasoning usage and, on reaching its budget, aborts and requests continuation with thinking disabled.
W	Replaces the standalone write tool with one that refuses to overwrite an existing file.

B.2 Experimental Setup

We used little-coder `v0.1.27` with Gemma-4-26B-A4B, K2-Horizon-MoVA-36B-A4B, and Qwen3.6-35B-A3B.

Configurations and evaluation. For each LLM, we randomly sampled 100 configurations by toggling ten feature groups while keeping infrastructure (INFRA) enabled. Each configuration was evaluated on the same 89 Terminal-Bench 2.0 tasks through the Harbor adapter, with one recorded attempt per task and a binary success outcome. The feature groups and their mechanisms are listed in Table 7.

The analyses condition on shell session (SS) being enabled, leaving nine optional feature groups. Including the baseline configuration with only INFRA and SS enabled and the full configuration with all optional features enabled, the retained dataset contains 58 configurations for Gemma, 50 for K2, and 63 for Qwen: 171 configurations and 15,219 task outcomes in total. The feature GLM and KernelSHAP analyses use this same dataset; the baseline and full configurations supply the endpoint constraints for KernelSHAP.

C Robustness Checks

The main text reports model-based standard errors for the binomial GLMs, with Wald intervals $a^\top\widehat{\theta} \pm 1.96\,\widehat{\mathrm{se}}(a^\top\widehat{\theta})$. They assume that the working GLM is correctly specified. Appendix C.1 compares them with trial-row HC0 sandwich standard errors as a sensitivity check. Appendix C.2 checks the additive point estimates against a model with task fixed effects. All reported 95% intervals are approximate and pointwise, without adjustment for multiple comparisons or selection of the largest effects.

C.1 Sandwich Standard Errors

Each GLM is fitted to trial rows: one binary outcome Y_i per harness–LLM–task–repeat trial for the leaderboard fits (restricted to a category's tasks for Table 2), and per configuration–task outcome for the feature GLM. Row i has design vector x_i, including the intercept, and fitted probability $\widehat{p}_i = \mathrm{logit}^{-1}(x_i^\top\widehat{\theta})$. With N trials, the model-based and sandwich covariance estimators are

$$\widehat{V}_{\mathrm{model}} = \widehat{H}^{-1}, \qquad \widehat{V}_{\mathrm{sand}} = \widehat{H}^{-1}\Big(\sum_{i=1}^{N}(Y_i-\widehat{p}_i)^2 x_i x_i^\top\Big)\widehat{H}^{-1}, \qquad \widehat{H} = \sum_{i=1}^{N}\widehat{p}_i(1-\widehat{p}_i)x_i x_i^\top. \tag{C.1}$$

We use the trial-row HC0 sandwich (Huber, 1967; White, 1982) as a sensitivity check on the working-model standard errors. With tasks and configurations fixed, squared residuals can include squared mean-model errors as well as execution variance. Independence of trials alone therefore does not make HC0 consistent for re-execution variance under mean misspecification. Both covariance estimates use normal-quantile intervals.

All trials of a leaderboard entry or ablation configuration e share its design row x_e and fitted probability $\widehat{q}_e$. With N_e trials and observed success rate $\widehat{p}_e$, the sandwich meat is $\sum_e N_e\{\widehat{p}_e(1-\widehat{p}_e) + (\widehat{p}_e-\widehat{q}_e)^2\}x_e x_e^\top$, whereas $\widehat{H} = \sum_e N_e\widehat{q}_e(1-\widehat{q}_e)x_e x_e^\top$. The two estimators therefore differ only through the entry-level lack of fit $\widehat{p}_e - \widehat{q}_e$, and they coincide exactly for the saturated interaction block. Table 8 compares them.

Table 8: Model-based versus sandwich standard errors on trial rows. The ratio is the sandwich SE divided by the model-based SE, as a median (and range) over coefficients; the counts are coefficients whose 95% intervals exclude zero. Category counts pool the six regular category fits; the feature row includes the baselines β_j and the contrasts v_{jk}, $\bar{v}_k$, and δ_{jk}.

Coefficients	Trials	SE ratio	Model-based	Sandwich
Overall additive (Table 1)	45,331	1.000 (0.986–1.015)	43/53	43/53
Categories (Table 2)	32,531	1.001 (0.946–1.056)	141/317	141/317
Interaction block (Table 3)	11,941	1.000 (exact)	12/27	12/27
Feature GLM (Table 4)	15,219	0.999 (0.993–1.016)	7/66	7/66

No coefficient or contrast changes its significance call, so every bold entry in Tables 1–4 is unchanged. The HC1 correction $N/(N-p)$ would enlarge these standard errors by at most 1.1%. Like the model-based errors, the sandwich treats trials as independent. Outcomes on the same task may share a difficulty that the GLMs omit; Appendix C.2 adds task effects to the additive model and finds its conclusions unchanged.

C.2 Task Fixed Effects

The additive model in Equation 2.1 has no task term, so shared task difficulty enters its residuals. As an ablation, we refit it to the trial-level outcomes with a fixed effect for each task,

$$\mathrm{logit}\,\mathbf{P}(Y_{ijtr}=1) = \mu + \alpha_i + \beta_j + \tau_t, \tag{C.2}$$

where t indexes tasks, r indexes trials, and $\tau_t = 0$ for a reference task. In the retained 105-pair leaderboard dataset, `make-doom-for-mips` has zero successes across 513 recorded trials. It is the only task with all-fail or all-pass outcomes. Its fixed effect tends to $-\infty$, and its contributions to

the harness and LLM score equations vanish in that limit, so we drop it from the task-fixed-effects fit, leaving 88 tasks. The main additive fit retains this task and uses all 89 tasks. Both fits use model-based standard errors.

Logit coefficients are non-collapsible, so conditioning on task difficulty changes their interpretation and need not preserve their magnitudes (Zeger et al., 1988). We therefore compare the rankings and relative sizes of the harness and LLM contrasts (Table 9). Although the coefficients increase after including task fixed effects, the principal qualitative conclusions remain unchanged.

Table 9: Additive fit (Table 1) versus the fit with task fixed effects. Contrasts and ranges are on the log-odds scale; ratios summarize their relative magnitudes within each fit. Both fits use model-based 95% confidence intervals.

	Additive	Task fixed effects
ForgeCode − Mini-SWE-Agent	0.890	1.594
Claude Opus 4.6 − Claude Sonnet 4.5	0.836	1.477
Ratio of the two contrasts	1.06	1.08
Harness range	1.12	1.98
LLM range	4.10	6.11
Ratio of the two ranges	0.27	0.32
Harnesses with 95% intervals excluding zero	20/27	20/27
LLMs with 95% intervals excluding zero	22/25	22/25

The rankings are essentially unchanged. Spearman correlations between the two fits are 0.999 for harnesses and 1.000 for LLMs, and the five largest harness and LLM effects are those of Table 1, in the same order. Every non-reference harness and LLM coefficient keeps its sign and its significance call. The harness contrast remains comparable to the LLM upgrade, and the harness range remains well below the LLM range. Accounting for task difficulty therefore preserves the qualitative conclusions of Section 2, so the main text reports the simpler additive model.

D Shapley Interpretation of Decompositions

D.1 Shapley Attributions of Logistic Regression

We provide a game-theoretic interpretation of the contributions of individual components of LLM agents, including LLMs and harnesses. For the additive decomposition in Equation 2.1 and the saturated decomposition in Equation 3.1, we construct a two-player game for each target harness–LLM pair (i, j) that explains the change in population log-odds relative to a reference pair (i_0, j_0). The two players are not the specific harnesses and LLMs themselves, but rather the two attributes $N = \{\mathrm{H}, \mathrm{L}\}$, where H denotes harness identity and L denotes LLM identity. A coalition specifies which attributes take their target values and which remain at their reference values. This construction provides two related interpretations of our statistical models. First, ordinary Shapley values allocate the total change in log-odds between harness and LLM identity. Second, the same game induces a functional decomposition into first-order harness and LLM components and a second-order harness–LLM interaction component. The latter connects our models directly to the Shapley–GAM correspondence of Bordt & von Luxburg (2023). Thus, the MLEs provide natural plug-in estimators of these model-based population quantities.

Baseline Shapley game. Let p_{ij} denote the population success probability of harness i paired with LLM j, with log-odds $\eta_{ij} = \mathrm{logit}(p_{ij}) = \log\{p_{ij}/(1 - p_{ij})\}$. For each target pair (i, j), define a cooperative game $v_{ij} : 2^N \to \mathbb{R}$ as follows. For a coalition $S \subseteq N$, let $i(S) = i$ if $\mathrm{H} \in S$ and $i(S) = i_0$ otherwise, let $j(S) = j$ if $\mathrm{L} \in S$ and $j(S) = j_0$ otherwise, and set

$$v_{ij}(S) = \eta_{i(S),j(S)}. \tag{D.1}$$

This construction is a direct instance of *Baseline Shapley* (BShap) (Sundararajan & Najmi, 2020). Given an explicand x, a baseline x', and a function f, BShap defines a cooperative game by replacing the coordinates outside a coalition S with their baseline values, $v(S) = f(x_S; x'_{N\setminus S})$. In our setting,

the explicand is the target harness–LLM pair (i, j), the baseline is the reference pair (i_0, j_0), and the function of interest is the population log-odds $(i, j) \mapsto \eta_{ij}$.

For a general cooperative game, the Shapley value of player $k \in N$ is (Shapley, 1953)

$$\phi_k = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left\{ v(S \cup \{k\}) - v(S) \right\}. \tag{D.2}$$

Because our game has only two players, the corresponding attributions simplify to

$$\begin{aligned} \phi_{\mathrm{H}}(i, j) &= \tfrac{1}{2}\left[\eta_{i,j_0} - \eta_{i_0,j_0}\right] + \tfrac{1}{2}\left[\eta_{ij} - \eta_{i_0,j}\right], \\ \phi_{\mathrm{L}}(i, j) &= \tfrac{1}{2}\left[\eta_{i_0,j} - \eta_{i_0,j_0}\right] + \tfrac{1}{2}\left[\eta_{ij} - \eta_{i,j_0}\right]. \end{aligned} \tag{D.3}$$

Thus, each attribution averages the marginal effect of changing one attribute over the two possible values of the other attribute. By Shapley efficiency, $\phi_{\mathrm{H}}(i, j) + \phi_{\mathrm{L}}(i, j) = \eta_{ij} - \eta_{i_0,j_0}$.

Functional decomposition. The same baseline game also provides a functional-decomposition view of our models. Bordt & von Luxburg (2023) show that a subset-compliant value function induces a particular functional decomposition $f(x) = \sum_{S \subseteq N} f_S(x_S)$, which they call the *Shapley-GAM*. The component functions are obtained from the value function by its Möbius transform, with $f_\emptyset = v(\emptyset)$. Conversely, summing the components indexed by subsets of a coalition recovers the associated value function. Thus, for a fixed value function, the cooperative-game and functional-decomposition views describe the same attribution structure.

Our baseline value function in Equation D.1 is subset-compliant: once the baseline (i_0, j_0) is fixed, $v_{ij}(S)$ depends on the target pair only through the attributes contained in S, and $v_{ij}(N) = \eta_{ij}$. Since there are only two attributes, its Shapley-GAM contains exactly four possible components:

$$\begin{aligned} f_\emptyset &= \eta_{i_0,j_0}, & f_{\mathrm{H}}(i) &= \eta_{i,j_0} - \eta_{i_0,j_0}, \\ f_{\mathrm{L}}(j) &= \eta_{i_0,j} - \eta_{i_0,j_0}, & f_{\mathrm{HL}}(i, j) &= \eta_{ij} - \eta_{i,j_0} - \eta_{i_0,j} + \eta_{i_0,j_0}. \end{aligned} \tag{D.4}$$

Consequently, $\eta_{ij} = f_\emptyset + f_{\mathrm{H}}(i) + f_{\mathrm{L}}(j) + f_{\mathrm{HL}}(i, j)$. The first two nonconstant terms are first-order effects of harness and LLM identity. The final term is the second-order component and measures the part of the target pair's log-odds that cannot be explained by changing the harness and LLM separately from the reference pair. Equivalently, $f_{\mathrm{HL}}(i, j)$ is the interaction contrast Δ_{ij}, the difference-in-differences of log-odds relative to the reference pair. This makes the additive and interaction models in Sections 2 and 3 two successive orders of the same functional decomposition.

Additive model as a first-order decomposition. Under the additive model in Equation 2.1, $\eta_{ij} = \mu + \alpha_i + \beta_j$, with reference constraints $\alpha_{i_0} = 0$ and $\beta_{j_0} = 0$, the components in Equation D.4 become $f_\emptyset = \mu$, $f_{\mathrm{H}}(i) = \alpha_i$, $f_{\mathrm{L}}(j) = \beta_j$, and $f_{\mathrm{HL}}(i, j) = 0$. Thus, Equation 2.1 is exactly the first-order, main-effect-only decomposition $f(i, j) = f_\emptyset + f_{\mathrm{H}}(i) + f_{\mathrm{L}}(j)$. In the terminology of Bordt & von Luxburg (2023), this is a GAM of order one: there are separate component functions for harness and LLM identity but no component involving both attributes. Substituting the additive model into Equation D.3 gives $\phi_{\mathrm{H}}(i, j) = \alpha_i - \alpha_{i_0}$ and $\phi_{\mathrm{L}}(i, j) = \beta_j - \beta_{j_0}$. Under the reference constraints, this reduces to $\phi_{\mathrm{H}}(i, j) = \alpha_i$ and $\phi_{\mathrm{L}}(i, j) = \beta_j$. Hence, in the additive model, the harness and LLM coefficients are simultaneously the first-order functional components and the ordinary BShap attributions relative to the reference pair.

Interaction model as a second-order decomposition. Under the interaction model in Equation 3.1, $\eta_{ij} = \mu + \alpha_i + \beta_j + \gamma_{ij}$, with reference constraints $\alpha_{i_0} = 0$, $\beta_{j_0} = 0$, $\gamma_{i,j_0} = 0$, and $\gamma_{i_0,j} = 0$, the functional components become $f_\emptyset = \mu$, $f_{\mathrm{H}}(i) = \alpha_i$, $f_{\mathrm{L}}(j) = \beta_j$, and $f_{\mathrm{HL}}(i, j) = \gamma_{ij}$. In particular, $\Delta_{ij} = \gamma_{ij}$. Thus, Equation 3.1 retains the second-order component omitted by the additive model, $f(i, j) = f_\emptyset + f_{\mathrm{H}}(i) + f_{\mathrm{L}}(j) + f_{\mathrm{HL}}(i, j)$. Because harness and LLM identity are the only two attributes, there are no possible components of order greater than two. The interaction model is therefore the full second-order functional decomposition of the population log-odds with respect to these attributes. The coefficient γ_{ij} is not merely a residual from the additive fit: under the reference game, it is exactly the second-order harness–LLM component.

Shapley axioms. The Shapley value is characterized by four standard properties. First, *efficiency* requires that the component attributions sum to the total change from the reference pair, $\phi_{\mathrm{H}}(i,j) + \phi_{\mathrm{L}}(i,j) = \eta_{ij} - \eta_{i_0,j_0}$. Second, *symmetry* requires two players with identical marginal contributions in every coalition to receive the same attribution. Third, the *null-player* property assigns zero attribution to a player whose marginal contribution is zero for every coalition. Finally, *linearity* requires $\phi_k(av + bw) = a\phi_k(v) + b\phi_k(w)$ for any two games v and w and scalars a and b. The Shapley value is the unique allocation rule satisfying these properties (Shapley, 1953).

Plug-in estimation. The quantities above are defined in terms of the population log-odds η_{ij} and are therefore population-level Shapley contributions. In practice, the population log-odds are unknown. We estimate them by fitting the corresponding logistic regression model by maximum likelihood and then substitute the fitted parameters into the Shapley formulas above. For example, under the additive model with reference coding, $\widehat{\phi}_{\mathrm{H}}(i,j) = \widehat{\alpha}_i$ and $\widehat{\phi}_{\mathrm{L}}(i,j) = \widehat{\beta}_j$. Thus, the fitted harness and LLM coefficients are naturally interpreted as plug-in estimators of the corresponding population Shapley contributions. Under the interaction model, the analogous plug-in estimates are $\widehat{\phi}_{\mathrm{H}}(i,j) = \widehat{\alpha}_i + \frac{1}{2}\widehat{\gamma}_{ij}$ and $\widehat{\phi}_{\mathrm{L}}(i,j) = \widehat{\beta}_j + \frac{1}{2}\widehat{\gamma}_{ij}$.

Extension to harness features. The same baseline-game construction applies within a harness. Fix LLM j and take the players to be its M optional harness features, $N = \{1, \ldots, M\}$. The target enables all features in N, and the baseline disables them, with the underlying harness infrastructure held fixed. For a coalition $S \subseteq N$, define $u_k(S) = \mathbf{1}\{k \in S\}$ and let $p_j(S)$ denote the population success probability with exactly the features in S enabled. The resulting game is $V_j : 2^N \to \mathbb{R}$ with $V_j(S) = \mathrm{logit}\{p_j(S)\}$, as in Equation 4.4. Each coalition sets its members to their target values and the remaining features to their baseline values, with LLM identity held fixed.

Under the additive feature model in Equation 4.1, $V_j(S) = \beta_j + \sum_{k \in S} v_{jk}$ and $V_j(\varnothing) = \beta_j$. Thus, for every $S \subseteq N \setminus \{k\}$, $V_j(S \cup \{k\}) - V_j(S) = v_{jk}$. Substituting these coalition-independent contributions into Equation D.2 gives $\phi_k^{(j)} = v_{jk}$ and $\sum_{k \in N} \phi_k^{(j)} = V_j(N) - V_j(\varnothing)$. As in the harness–LLM game above, the additive coefficients are both first-order functional components and baseline Shapley attributions. Their estimates $\widehat{v}_{jk}$ are plug-in estimates of these population quantities under the additive model, and are exactly the Shapley values of the fitted additive game. Without additivity, a Shapley value instead combines a feature's first-order component with its share of each higher-order component it enters (Bordt & von Luxburg, 2023), so the KernelSHAP attributions of Section 4.1 are interaction-aware and need not agree with $\widehat{v}_{jk}$ when feature interactions are present.

D.2 KernelSHAP: exactness and approximation

By Lundberg & Lee (2017, Theorems 1–2), unregularized regression with the Shapley kernel recovers the exact Shapley vector of the specified game when it is supplied with all 2^M coalition values: the $2^M - 2$ coalitions of finite Shapley-kernel weight enter the objective, and the two endpoints enter through the baseline and efficiency constraints. The game need not be additive. Applied to the population game V_j in Equation 4.4, this gives population Shapley values; applied to the empirical game $\widehat{V}_j$ in Equation 4.4, it gives empirical-game Shapley values.

Our analysis uses only the observed empirical coalitions, so it approximates the full-coalition regression and provides a plug-in estimate of population attribution. Two sources of error remain: approximation from incomplete coalition coverage and sampling uncertainty in the empirical coalition values. Observing all coalitions would remove the former but not the latter; the endpoint constraints alone remove neither.

D.3 Joint Value and Coalition Uncertainty for KernelSHAP

Our target is the full-game Shapley vector of $V_j(S)$ in Equation 4.4, where success probabilities average over run-level randomness on the fixed set of 89 benchmark tasks. The estimates have two sources of uncertainty: noise in the evaluated coalition values and random selection of the coalitions to evaluate. Conditional on SS being enabled and on the interior sample sizes $n_j = 56$, 48, and 61 for Gemma, K2, and Qwen, uniform distinct-subset sampling gives a simple random sample without replacement from the $2^9 - 2 = 510$ interior coalitions. The empty and full coalitions are

always retained. We assume independent coalition samples across LLMs, independent of execution outcomes. The sample objective estimates the full-game Shapley-kernel objective; see Covert & Lee (2021) for the regression view of Shapley estimation. The following joint-variance construction is our approximation for noisy coalition values under this sampling design. It requires finite game logits and identified constrained fits, and does not correct finite-sample bias in the attributions.

Value component. For a fixed coalition design, the constrained regression in Equations 4.6–4.7 is linear in the empirical logits: $\widehat{\phi} = L\widehat{v}$. Here $\widehat{v}$ stacks all observed coalition logits, including both endpoints for each LLM, and L is the block-diagonal map obtained by solving the three constrained regressions. With $G = 89$ and configuration success rates $\widehat{p}_c$, the working independent-binomial model and delta method give

$$\widehat{\Sigma}_v = \mathrm{diag}\left\{ \frac{1}{G\,\widehat{p}_c(1-\widehat{p}_c)} \right\}, \qquad \widehat{V}_{\mathrm{val}} = L\widehat{\Sigma}_v L^\top. \tag{D.5}$$

This is model-based value uncertainty, not task-clustered uncertainty. It propagates noise in both endpoints as well as the interior values. All observed success counts lie strictly between 0 and 89, so their empirical logits are finite.

Coalition component and overlap. A delete-one coalition jackknife reflects both coalition selection and noise in the estimated values. Adding it directly to $\widehat{V}_{\mathrm{val}}$ therefore counts some value noise twice; we estimate and subtract this overlap. For LLM j, delete each interior coalition i once and refit, retaining both endpoints. Let $L_{j,-i}$ map $\widehat{v}_j$ to that refit's attributions, with a zero column for the deleted coalition. Define $D_{ji} = L_{j,-i} - n_j^{-1}\sum_l L_{j,-l}$ and $\kappa_j = (1 - n_j/510)(n_j - 1)/n_j$. The finite-population jackknife and its estimated value-noise contribution are

$$\begin{aligned} \widehat{V}_{C,j} &= \kappa_j \sum_{i=1}^{n_j} (D_{ji}\widehat{v}_j)(D_{ji}\widehat{v}_j)^\top, \\ \widehat{B}_j &= \kappa_j \sum_{i=1}^{n_j} D_{ji}\widehat{\Sigma}_{v,jj} D_{ji}^\top, \end{aligned} \tag{D.6}$$

where $\widehat{\Sigma}_{v,jj}$ is the within-LLM block of $\widehat{\Sigma}_v$ in Equation D.5. Every delete-one fit must be identified. Because the jackknife uses estimated logits, $\widehat{V}_{C,j}$ also contains the noise in $\widehat{v}$: for fixed D_{ji}, mean-zero additive logit noise contributes $D_{ji}\Sigma_{v,jj}D_{ji}^\top$ to the expected outer product, which $\widehat{B}_j$ estimates. This correction is a delta approximation for empirical logits, rather than the coalition-batch variance estimator of Covert & Lee (2021).

Joint intervals. Subtracting the overlap gives the candidate covariance $\widehat{V}_{\mathrm{val}} + \mathrm{blockdiag}_j(\widehat{V}_{C,j} - \widehat{B}_j)$. Its coalition term estimates a variance component and need not be positive semidefinite, so the candidate can give negative variances for some contrasts. As for non-positive-semidefinite multiway cluster covariances (Cameron et al., 2011), we set these eigenvalues to zero:

$$\widehat{V}_{\mathrm{joint}} = \widehat{V}_{\mathrm{val}} + \mathrm{blockdiag}_j\left[\widehat{V}_{C,j} - \widehat{B}_j\right]_+, \qquad a^\top\widehat{\phi} \pm t_{\nu_a,0.975}\sqrt{a^\top\widehat{V}_{\mathrm{joint}}a}, \tag{D.7}$$

where $[\cdot]_+$ zeroes negative eigenvalues on the subspace orthogonal to the endpoint sum. The degrees of freedom ν_a are specific to the contrast. Write a_j for the entries of a belonging to LLM j, $s_{a0} = a^\top\widehat{V}_{\mathrm{val}}a$, and $s_{aj} = a_j^\top[\widehat{V}_{C,j} - \widehat{B}_j]_+a_j$. We use the approximate Welch–Satterthwaite calibration

$$\nu_a = \frac{\left(s_{a0} + \sum_j s_{aj}\right)^2}{\sum_j s_{aj}^2/(n_j - 1)}, \tag{D.8}$$

which treats the plug-in binomial value component as known and the coalition components as independent of it, with $n_j - 1$ degrees of freedom each. These are calibration approximations: the estimated components share the observed outcomes, and clipping further affects their sampling distributions. Coalition selection preserves each LLM's endpoint sum, so its variance comes entirely from the value component; contrasts with no coalition variance use the normal quantile. Across-LLM means and departures use the full joint covariance.

Clipping increases contrast variances relative to the unmodified candidate, but guarantees neither an upper bound on the true variance nor nominal finite-sample coverage. The corrected and clipped coalition components have small estimated traces (0.002, 0.023, and 0.016 for Gemma, K2, and Qwen); these are variance estimates, not a test of additivity. The intervals in Table 5 are approximate and pointwise, and do not provide distribution-free guarantees for arbitrary unobserved coalition values.

Sensitivity. Kernel weights concentrate on few coalitions, so $n_j - 1$ may overstate the information available for estimating a coalition component. Replacing it in Equation D.8 by $\mathrm{ESS}_j - 1$, where ESS_j is the Kish effective sample size of the interior kernel weights, leaves the significance calls unchanged. As a second check, we retain the overlap and use $\widehat{V}_{\mathrm{val}} + \mathrm{blockdiag}_j(\widehat{V}_{C,j})$, without subtraction or eigenvalue clipping, with the same degrees-of-freedom rule applied to the raw coalition components. This construction double counts some value noise. Quality monitoring's average attribution still excludes zero: 0.174 with a 95% interval of $[0.011,\ 0.336]$. Gemma's individual quality-monitoring attribution also remains significant, and the bold entries in Table 5 are unchanged. Thus, the positive average contribution does not depend on the overlap correction or clipping.

E Full Additive Effects

Figure 1 shows every harness and LLM coefficient of the additive model in Section 2, extending the five largest of each reported in Table 1.

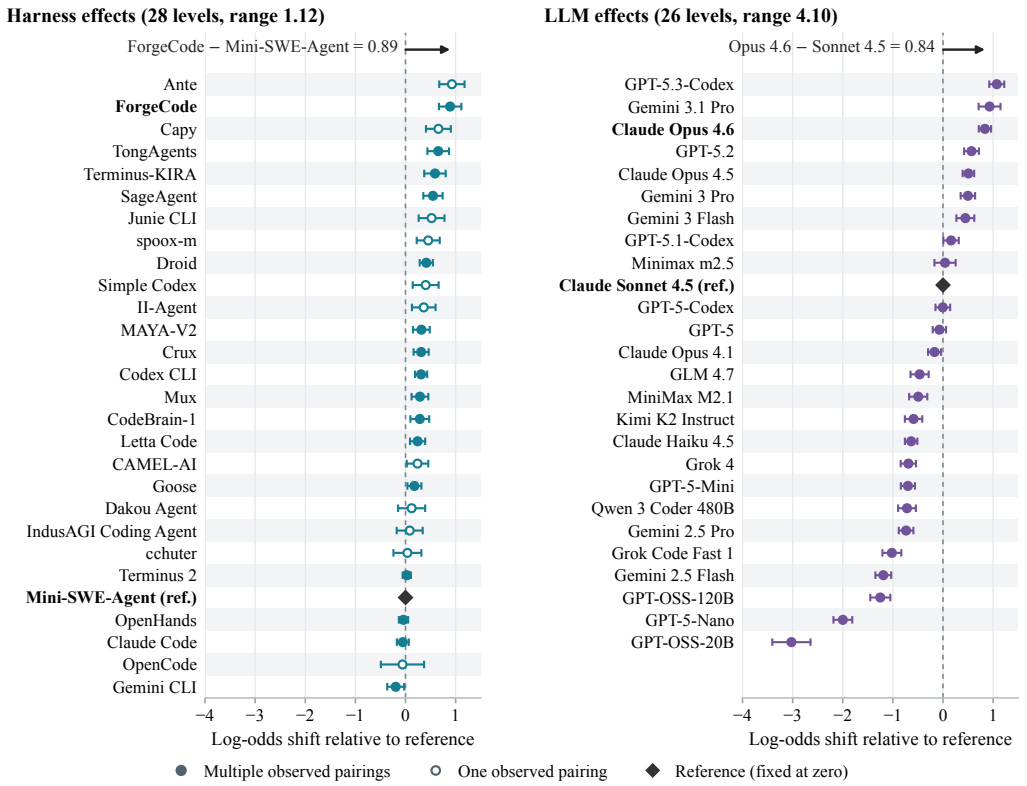

Figure 1: Additive harness effects (left) and LLM effects (right) for every level in the Section 2 fit, as log-odds shifts relative to Mini-SWE-Agent and Claude Sonnet 4.5 (diamonds). Bars are pointwise model-based 95% confidence intervals; hollow markers denote harnesses observed with a single LLM. Both panels share one scale, and the arrows mark the two contrasts compared in Section 2: ForgeCode versus Mini-SWE-Agent (0.89) and Claude Opus 4.6 versus Claude Sonnet 4.5 (0.84). LLM effects nonetheless span a wider range overall (4.10 versus 1.12 logits).