STOCHASTIC YET STRUCTURED: REASONING BRANCH SELECTION DYNAMICS UNDER RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning dynamics of LLM post-training provides insight into how model behaviors emerge and evolve throughout optimization. In this paper, we study the learning dynamics of reinforcement learning with verifiable rewards (RLVR) by tracking the evolution of probability distributions over behavioral branches along training trajectories and controlled continuations from fixed training states. By projecting RLVR training trajectories into an interpretable behavioral state space, we characterize RLVR's reallocation of probability mass among behavioral branches across models and tasks. Controlled continuations from the same complete training state show that future branch allocations are not fully reproducible, even though the dominant branch often remains stable. Across continued training, these differences retain only short-range directional memory, while the current branch state provides a near-first-order predictive summary of short-horizon dynamics. The remaining variation is nevertheless structured: residual dependence shapes long-horizon divergence, while the current branch state predicts future instability across independent lineages and over substantially longer training horizons. Together, these results characterize RLVR branch selection as a stochastic but state-dependent dynamical process.

1 INTRODUCTION

Large Language Models (LLMs) improve their reasoning capabilities through post-training methods such as reinforcement learning with verifiable rewards (RLVR), which reinforces sampled solutions according to verifiable outcomes. Studying the learning dynamics of this process asks not only what behavior a trained model exhibits, but how that behavior emerges and changes over successive optimization updates (Ren & Sutherland, 2025). Such dynamical views have helped characterize when model behaviors emerge and how successive updates shape generalization and training efficiency (Pruthi et al., 2020; Ren et al., 2022; Park et al., 2024).

Prior work has shown that RLVR can improve reasoning by reallocating probability toward successful behaviors already accessible to the model (Yue et al., 2025; Zhao et al., 2025), while subsequent work connects the contraction of behavioral diversity to the on-policy nature of RLVR or the underdetermination of outcome-based objectives (Yuan et al., 2026; Lochab et al., 2026). More recently, Zhou & Li (2026) localize much of this contraction to early reasoning choices, showing that RLVR can narrow which behavioral branches are entered. Relatedly, Wang et al. (2025) identify high-entropy tokens as critical forks that steer models toward different reasoning pathways, providing a token-level view complementary to our branch-level state representation. These findings motivate a dynamical question: once a branch distribution has formed at an intermediate training state, how does it evolve under continued RLVR? We ask whether the resulting dynamics are reproducible and how much of their future evolution is already encoded in the current behavioral state.

To answer these questions, we represent an RLVR checkpoint by its distribution over behavioral branches and study the induced dynamics in this behavioral state space. Along ordinary checkpoint trajectories, this view reveals how probability is reallocated among branches over training. We then restart multiple continuations from the same complete resumable training state, holding the entire prior training history fixed and varying only subsequent training realizations. This state-conditional design distinguishes variation generated after a specified checkpoint from differences inherited across independently trained runs. We find that future branch allocations are not fully

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.

reproducible from a fixed state, yet their stochastic evolution remains strongly state-dependent. Directional differences decay quickly, while the current behavioral state provides a near-first-order predictive summary and identifies which problems are most susceptible to future instability.

These dynamics can arise naturally from RLVR's on-policy feedback, where the current policy shapes which trajectories are sampled and, in turn, which behaviors receive learning signal in subsequent updates (Huang et al., 2026; Tang et al., 2026; Wei et al., 2026). When multiple viable branches compete, finite-sample updates can perturb their relative probabilities, while later sampling feeds those differences back into training. This motivates the branch-level dynamical picture developed from Sections 3 to 5.

Contributions. Our analysis establishes three properties of RLVR branch dynamics. First, RLVR selectively reallocates probability among behavioral branches beyond scalar sharpening of an existing distribution. Second, controlled full-state continuations show that future branch allocations are not fully reproducible from a fixed training state. Finally, this variation is structured: directional memory is short-ranged, the current branch state provides a near-first-order predictive summary, residual dependence shapes longer-horizon divergence, and the pre-fork branch margin predicts future instability in both the dense Track-D continuation and an independently trained multi-root Track-A lineage, including horizons up to 100 updates. A matched split-half control indicates that finite-rollout uncertainty is not the primary driver. Detailed connections to prior learning-dynamics, RLVR-diversity, and stochastic-training work are provided in Appendix A.

2 Behavioral State and Branch Dynamics

We focus on one aspect of RLVR learning dynamics: the reallocation of probability across behavioral branches. For a problem q, let

$$\mathcal{B}_q = \{b_1, \ldots, b_{K_q}\}$$

denote a finite set of possible behavioral branches, and define $g_q(y)$ as a mapping from a generated trajectory y to the branch it enters. Given policy π_t at training update t, we define

$$p_t(b \mid q) = \Pr_{y \sim \pi_t(\cdot|q)}[g_q(y) = b],$$

and summarize the model's behavior by

$$X_t(q) = \big(p_t(b_1 \mid q), \ldots, p_t(b_{K_q} \mid q)\big).$$

We denote $X_t(q)$ as the *behavioral state* of problem q. In our experiments, branches are task-specific, reproducibly identified behavioral entrances. Their concrete construction is given in Section 4. If S_t denotes the complete resumable training state, including the policy parameters, optimizer, scheduler, and other state required to continue training, then

$$X_t(q) = \Phi_q(S_t)$$

is an interpretable behavioral projection of the full training state. Our basic dynamical object is therefore

$$X_t(q) \longrightarrow X_{t+\Delta}(q).$$

RLVR makes this transition stochastic because each update depends on newly sampled trajectories. We write

$$S_{t+1} = \mathcal{T}(S_t, \xi_t),$$

where ξ_t collects stochasticity realized during the update, including on-policy trajectory generation and implementation-level nondeterminism. Thus a fixed S_t can give rise to different future behavioral states. We characterize these possible outcomes through the state-conditional distribution

$$\mathcal{P}_h(\cdot \mid S_t, q) = \mathrm{Law}(X_{t+h}(q) \mid S_t)\,.$$

Controlled continuations from the same S_t provide samples from this distribution and allow us to study the reproducibility of future branch allocation from a fixed training state.

Since $X_t(q)$ lies on a probability simplex, we represent its relative motion using centered log-ratio (CLR) coordinates (Aitchison, 1986),

$$z_t(q, b) = \log p_t(b \mid q) - \frac{1}{K_q} \sum_{b' \in \mathcal{B}_q} \log p_t(b' \mid q).$$

At observation interval Δ, we write a generic behavioral transition as

$$z_{t+\Delta} = F(\mathcal{H}_t) + r_t,$$

where $\mathcal{H}_t$ denotes the observed state or history supplied to the transition model and r_t is the remaining residual. This representation provides a common language for studying branch dynamics: how much the current state predicts future motion, what additional information is carried by recent history, and how the remaining stochastic variation propagates across training.

3 On-Policy Branch Selection Dynamics

The behavioral state above describes how probability is allocated across reasoning branches. We next consider how on-policy RLVR can change this allocation and generate stochastic branch dynamics.

3.1 On-Policy Dynamics of Reasoning-Branch Selection

Consider a sampled trajectory $y \sim \pi_t(\cdot \mid q)$ with normalized training advantage $\widehat{A}_t(y)$. For branch b, define its expected advantage

$$a_t(q, b) = \mathbb{E}\Big[\widehat{A}_t(y) \mid g_q(y) = b,\ q\Big],$$

and let

$$\bar{a}_t(q) = \sum_{b'} p_t(b' \mid q) a_t(q, b')$$

be the policy-average advantage.

Motivated by the standard policy-gradient form (Williams, 1992; Sutton et al., 2000), an idealized branch-level update has the qualitative form

$$\mathbb{E}[\Delta p_t(b \mid q) \mid X_t] \propto p_t(b \mid q)\big(a_t(q, b) - \bar{a}_t(q)\big).$$

Thus branch motion depends jointly on its relative advantage and its current probability. A branch with positive relative advantage tends to gain mass, but a branch that is already more probable is also sampled more often and therefore receives more opportunities for reinforcement. This produces a replicator-like feedback between current probability and subsequent updates.

When several viable branches have similar probability and expected advantage, finite on-policy sampling can produce different realized updates from the same training state. The resulting probability differences then alter the branch distribution encountered by later updates, creating a feedback between stochastic variation and subsequent sampling.

3.2 Stochastic Branch Selection under On-Policy Feedback

Figure 1 summarizes this interaction. Starting from the same behavioral state, stochastic updates can favor different near-tied branches; later sampling can amplify these differences while fresh stochasticity continues to redirect branch motion.

This picture motivates the empirical analysis that follows. We first test whether RLVR changes branch allocation beyond simple sharpening of existing preferences. We then use controlled continuations to measure how reproducible future branch allocation is from a fixed training state. Having isolated post-fork variation, we study how long its direction persists and how much predictive information recent history contributes beyond the current behavioral state. Finally, we examine whether the remaining variation contains dependence that shapes longer-horizon divergence and future instability.

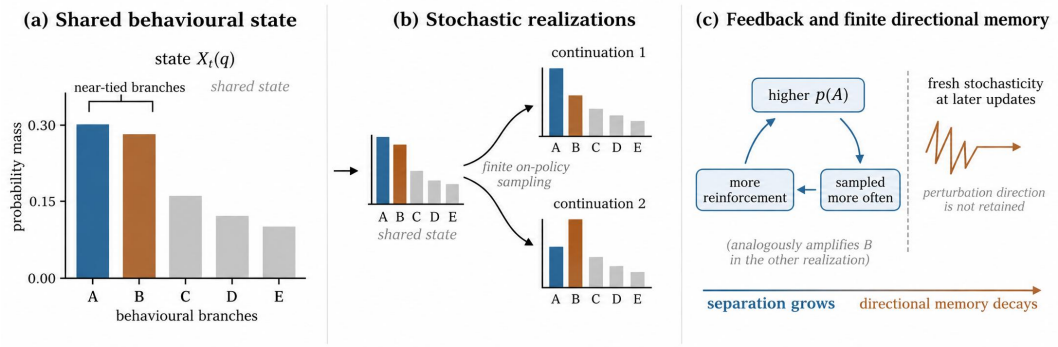


Figure 1: **Stochastic branch selection under on-policy feedback.** (a) Multiple viable branches can have similar probability at the same behavioral state. (b) Finite on-policy sampling can produce different realized updates from the same complete training state, favoring different branches across continuations. (c) The resulting probability differences feed back into later sampling and reinforcement, while fresh stochasticity at subsequent updates prevents a single perturbation direction from persisting indefinitely.

4 Experimental Setup

Experimental settings. We study behavioral branch dynamics primarily on Countdown, an arithmetic reasoning task whose finite action structure allows alternative reasoning entrances to be enumerated mechanically (Katz et al., 2025). Our controlled experiments use Qwen2.5-3B-Instruct (Yang et al., 2025) trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Countdown provides our primary controlled setting, where branch feasibility can be verified exactly and training can be resumed from complete intermediate states. For cross-setting breadth, we additionally analyze Qwen2.5-7B Countdown runs, a public Qwen2.5-3B Countdown continuation, and Qwen- and OLMo-based RLVR Code trajectories on HumanEval (Chen et al., 2021). Our state-conditional analyses rely on the controlled Countdown continuations, while the remaining trajectories provide trajectory-level breadth across model scale, training source, and task. We refer to the self-trained 3B full-state forks as Track A and the public dense 3B continuation as Track D. Training provenance and evaluation details are reported in Appendix C.

Behavioral branch states. On Countdown, each generated trajectory is mapped to its first arithmetic entrance,

$$g_q(y) = (\text{first operand}, \text{operator}),$$

and an exact solver retains entrances from which at least one correct continuation remains possible. The resulting branch distribution therefore measures probability allocation among viable reasoning starts rather than only among ultimately correct solutions. On HumanEval, we analogously define deterministic implementation entrances from the first substantive AST decision of the target function. Because these representations are task-specific, our cross-task comparisons concern probability allocation over reproducibly identified behavioral entrances rather than a shared semantic taxonomy of reasoning strategies. Detailed branch construction, support estimation, and validation are given in Appendix C.3.

Observed trajectories and controlled continuations. We use two complementary experimental views. First, checkpoint trajectories measure how behavioral branch distributions change during ordinary RLVR. Second, controlled full-state continuations restart training multiple times from the same resumable state, holding fixed the model, optimizer, scheduler, data state, and prior training history. Multi-root forks evaluate the reproducibility of future branch allocation from the same training state, while a densely evaluated five-continuation fork resolves how post-fork differences propagate over short training horizons. Because each behavioral state is estimated from finite rollouts, all continuation-divergence analyses use rollout-matched measurement controls to separate

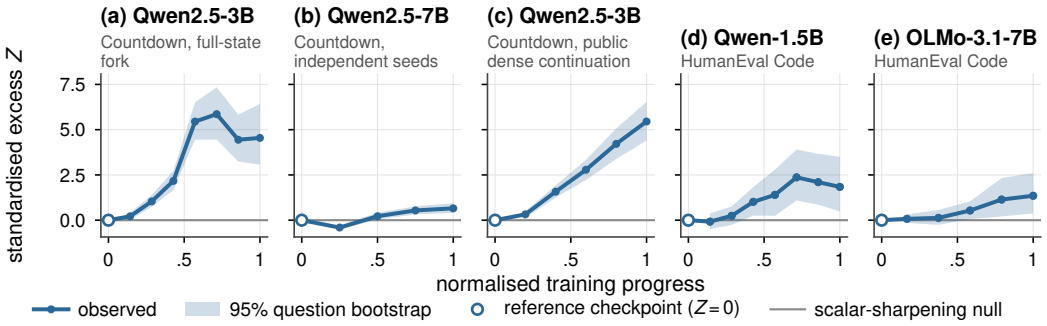


Figure 2: **RLVR departs from scalar sharpening across several settings.** Each panel uses the earliest observed checkpoint of that setting as its reference. The standardized excess is zero for exact scalar sharpening; shaded regions show 95% question-bootstrap intervals. The onset and magnitude vary across settings, with clear later departures in the 3B Countdown and Code trajectories.

training-induced variation from sampling uncertainty. Exact fork locations, rollout budgets, and statistical procedures are reported in Appendix C.

5 Empirical Results

5.1 RLVR Reorganizes Behavioral Branch Distributions

To distinguish branch-specific reweighting from simple concentration of an existing preference, we compare the observed dynamics with a scalar-sharpening baseline. Under this baseline, branches with higher reference probability receive systematically greater mass as the distribution sharpens, with all changes governed by a single scalar exponent. We model this baseline as

$$p_t^{(\gamma)}(b \mid q) \propto p_{t_0}(b \mid q)^{\gamma_t},$$

where t_0 is the earliest observed checkpoint of a trajectory and γ_t controls the concentration of its branch distribution. We fit γ_t at each checkpoint and standardize the discrepancy from the fitted baseline by its rollout-matched bootstrap variation.

Figure 2 shows departures from scalar sharpening across several observed settings. The departure appears in both 3B Countdown trajectories and in the Qwen and OLMo Code trajectories. On the main 3B trajectory, for example, step 50 remains compatible with scalar sharpening while the departure becomes clear from step 75 onward. The shorter 7B trajectory serves as a scale-side scope check rather than evidence for a matched effect size. Thus later RLVR updates can selectively reallocate probability among viable behavioral branches beyond scalar sharpening.

Consistent with this selective reallocation, behavioral states also progressively move away from their earlier checkpoints across the observed settings. We report the corresponding JS-displacement curves and descriptive entropy and top-1 statistics in Appendix E. A single observed trajectory, however, cannot determine whether the direction of this movement is uniquely specified by the current training state.

5.2 Future Branch Allocations Are Not Fully Reproducible from Fixed States

A sequence of checkpoints records only one realized training history. To test whether the resulting movement is reproducible from a fixed state, we use the controlled full-state continuations described in Section 4 and compare their future branch distributions. Because variation accumulated before the fork is held fixed, differences after the fork isolate variation generated during subsequent training.

The dominant branch is often stable across stochastic realizations: modal agreement is 0.92 in the multi-root full-state forks and 0.89 in the dense step-100 continuation, with 0.95 in the independent-run 7B reference. Here modal agreement is the fraction of continuation pairs that share the same

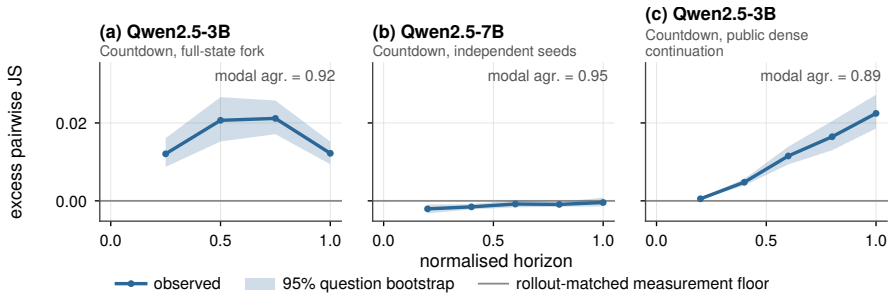


Figure 3: **Future branch allocations are not fully reproducible from fixed training states.** Excess pairwise JS subtracts a matched finite-rollout measurement floor. Panel (a) summarizes different full-state fork roots ending at a common endpoint; panel (b) gives an independent-run 7B reference; panel (c) follows dense continuations from one shared step-100 state.

modal branch; at the problem level, 16–26 of 106 problems per Track-A root have at least one of the four continuations end on a different dominant branch (Table 22). Stability of the modal branch, however, does not imply reproducibility of the full branch distribution. Across the four controlled fork roots, at 256 rollouts per problem and arm, 68–91% of the held-out Countdown problems exhibit detectably different future branch distributions across continuations. Figure 3 shows the corresponding divergence relative to a rollout-matched measurement floor. The multi-root full-state forks remain above the floor, and the dense step-100 continuation shows divergence that grows with post-fork horizon. In panel (a), the points correspond to different fork roots that terminate at a common endpoint, so they should not be read as a single root observed repeatedly over time.

These behavioral differences are not simply ordered by task-performance differences. At the prompt level, continuation pairs that disagree on the dominant branch have a mean absolute accuracy difference of 0.039, compared with 0.045 among pairs whose dominant branch agrees. Across the primary run-level pairs, branch divergence likewise does not track the ordering of pass@1 or training-reward differences. These comparisons show that branch-space divergence is not simply a surrogate for accuracy divergence. A same-seed replay at root 75 that also restores the rollout seed still diverges above its evaluation floor (39/106 problems pass the blocked test; Table 15), showing that implementation-level nondeterminism can remain behaviorally visible. Full paired results, the balanced four-arm tests, and replay controls are reported in Appendix F.

Controlled continuations therefore show that future branch allocation is not fully reproducible under stochastic continuation from a fixed complete training state. Stochastic variation generated after the fork changes the branch distribution even when the complete prior training history is held fixed.

5.3 The Current Branch State Is a Near-First-Order Predictive Summary

We next ask whether branch-space motion preserves a persistent direction. Let

$$v_t(q) = z_{t+\Delta}(q) - z_t(q)$$

be the CLR increment over one observation interval, and measure directional memory by the cosine between increments separated by a given number of observation intervals. Figure 4 compares this quantity with a shuffled-time reference across all five experimental settings.

Directional memory is concentrated at short lags and does not persist over long horizons. The public dense 3B trajectory shows the clearest positive short-range alignment, while the other trajectories return to the shuffled reference more quickly. The controlled dense continuation gives the same qualitative picture when we isolate pairwise perturbations between continuations: the corrected cosine falls from 0.889 at lag 5 to 0.574 at lag 10 and 0.393 at lag 15, with the lag-20 confidence interval including zero. Thus branch differences can grow in magnitude without preserving one early perturbation direction indefinitely.

The rapid decay of directional memory raises a predictive question: how much does recent history add once the current branch state is already known? On the dense five-update trajectory, let

$$v_t = z_t - z_{t-5},$$

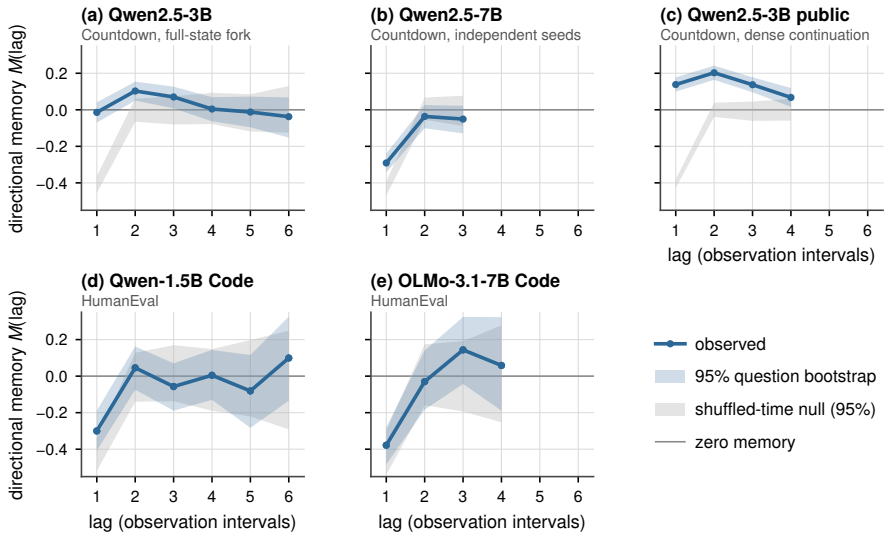


Figure 4: **Directional memory is short-ranged across training trajectories.** Curves show cosine alignment between CLR increments at increasing temporal lags; gray bands give shuffled-time 95% references. The lag-1 shuffled reference is negative because adjacent increments share an intermediate state with opposite sign, so the relevant comparison is to the shuffled band rather than to zero alone.

Table 1: **Recent motion adds little predictive information beyond the current branch state.** Held-out negative log-likelihood per count of a conditional multinomial predictor. The two rows use different transition scopes because M_3 requires one further lag, so absolute NLL values are comparable within a row but not across rows.

Comparison	Held-out NLL	ΔNLL	95% CI
$M_1 \to M_2$	$0.60834 \to 0.60718$	$+0.001154$	$[+0.000747, +0.001594]$
$M_2 \to M_3$	$0.57437 \to 0.57441$	-0.000037	$[-0.000055, -0.000019]$

and compare three nested feature sets,

$$M_1 : (z_t, c), \qquad M_2 : (z_t, c, v_t), \qquad M_3 : (z_t, c, v_t, v_{t-5}),$$

where $c(q, b)$ contains the fixed per-branch controls.

Recent history contributes little additional predictive information once the current branch state is known. Adding v_t improves held-out NLL by $0.001154\,[0.000747, 0.001594]$, corresponding to only 0.19% of the state-only NLL, although this small improvement exceeds a first-order simulation control at $p = .001$. Adding another lag does not improve prediction and slightly worsens held-out NLL. Thus, at this temporal resolution, the current branch state captures nearly all of the short-horizon predictive structure available from recent branch-space history.

This near-first-order picture is also supported after removing the state-predicted drift. Residual temporal alignment remains close to a shuffled-time reference at the tested lags, indicating that most of the observed temporal structure lies in predictable state-dependent drift rather than in a persistent residual direction. Appendix G gives the residual-memory analysis, the exact predictive estimator, and leave-one-out robustness checks.

5.4 Residual Dependence Shapes Long-Horizon Divergence

The near-first-order result leaves a substantial one-step residual. We separate its energy into finite-rollout measurement variation, a component shared across all five continuations for the same problem and transition, and a continuation-specific component. With the state-only transition model,

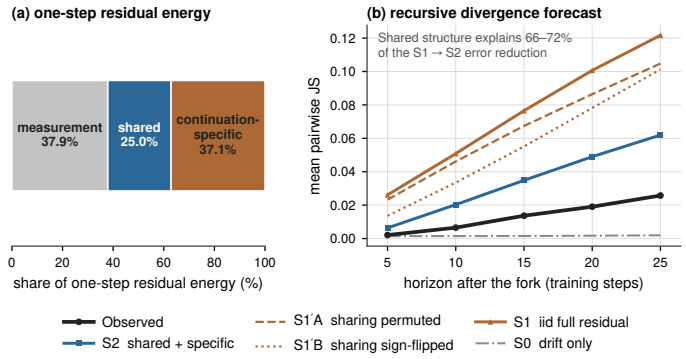


Figure 5: **Residual dependence matters for long-horizon divergence.** (a) One-step residual energy contains measurement, shared, and continuation-specific components. (b) Treating the full residual as fresh iid noise overpredicts divergence; destroying cross-continuation sharing moves the forecast toward the iid baseline, while preserving shared structure substantially improves the forecast.

these account for 37.9%, 25.0%, and 37.1% of raw residual energy, respectively. We use this decomposition to establish structure rather than a universal variance law, since the exact shares depend on the transition model and scope.

This residual dependence matters when transitions are composed recursively. Starting from the real shared step-100 state, a drift-only model S_0 under-disperses, while S_1, which treats the full observed one-step residual as fresh iid process noise, over-disperses relative to the observed continuations. The structured model S_2 instead preserves both a residual component shared across continuations and a continuation-specific component. Controls that destroy cross-continuation sharing move the forecast back toward the iid baseline. Across the evaluated horizons, preserving the shared residual structure accounts for about 66–72% of the forecast-error reduction from the iid-residual model to the structured model. The structured model still over-disperses at the endpoint (0.062 versus 0.026 observed), so we use it to compare residual structures rather than as a calibrated forecast. Figure 5 shows this effect under recursive simulation. Full simulator definitions, horizon-wise values, alternative residual controls, and leave-one-continuation-out forecasts are reported in Appendix H.

5.5 The Current State Predicts Future Instability

The pre-fork behavioral state is already informative about which problems will become unstable under continued training. On the dense Track-D continuation, branch-state features predict future modal disagreement substantially better than ordinary controls (Figure 6).

Separately, in a fixed-horizon analysis over all observed post-fork states, the top-1–top-2 margin provides a particularly simple indicator of susceptibility. Problems with closely competing branches are much more likely to switch their modal branch over the next few updates: ranking states by negative margin yields AUROC 0.935 at the primary five-update horizon. This suggests that future instability is concentrated in behavioral states where no branch has yet established a clear lead.

The same state-dependent susceptibility appears in the independently trained Track-A lineage. Across four full-state fork roots, the pre-fork branch margin predicts endpoint modal disagreement at every tested horizon through 100 updates (Appendix I, Table 22, and Figure 8). The same ranking also predicts continuous endpoint branch divergence, with Spearman correlations from 0.429 to 0.700. A pooled predictor over the four roots likewise separates behavioral state from ordinary controls, reaching AUROC 0.844 with state features compared with 0.542 for controls alone. Thus susceptibility to future branch instability is already visible in the behavioral state well before the continuations separate. A matched split-half control indicates that this Track-A signal is not driven primarily by uncertainty in the estimated endpoint winner: at the same rollout budget, pairwise dis-

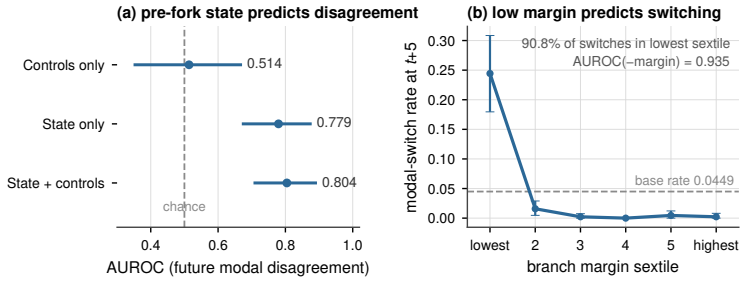


Figure 6: **Pre-fork branch state predicts future instability.** (a) Branch-state features predict future modal disagreement better than ordinary controls: AUROC is 0.779 for state features, 0.514 for controls, and 0.804 when combined. (b) Modal switching is concentrated among low-margin states: ranking by negative top-1–top-2 margin yields AUROC 0.935 at the primary five-update horizon, with 108/119 switches occurring in the lowest margin sextile. The dashed line marks the overall switch base rate.

agreement across continuations is 2.3–$5.3\times$ the within-continuation split-half rate, and excluding measurement-unstable problems changes raw-margin AUROC by at most 0.05.

Appendix I reports complete predictive metrics, continuous future-divergence results, multi-root and multi-horizon analyses, calibration, matched finite-rollout controls, and leave-one-continuation-out robustness.

Implications for RLVR evaluation. Our results suggest that an RLVR run should be interpreted as one realization of a state-conditional stochastic process, not only through its endpoint accuracy. Branch-space differences can emerge from the same complete training state without tracking task-performance differences, while the current branch state identifies where future dominant-branch changes are most likely. This perspective complements broader concerns about run-to-run variability and evaluation uncertainty (Henderson et al., 2018; Agarwal et al., 2021; Hochlehnert et al., 2025) and motivates behavioral-state diagnostics or controlled continuations when reproducibility itself is under study.

Limitations. Our state-conditional evidence is concentrated in Qwen2.5-3B Countdown full-state forks, and our behavioral branches are task-specific entrances rather than a shared semantic taxonomy of reasoning strategies. The 7B and Code trajectories therefore provide trajectory-level breadth rather than larger-scale fixed-state replication. Using the branch-margin signal to guide exploration or diversity-preserving interventions is left to future work.

6 Conclusion

We study RLVR as a stochastic dynamical process over behavioral branch distributions. Across training trajectories, RLVR selectively reallocates probability among viable branches beyond scalar sharpening, while controlled continuations show that future branch allocation is not fully reproducible under stochastic continuation from a fixed complete training state. Yet this stochastic variation remains structured. Directional memory is short-ranged, and the current branch state captures nearly all of the short-horizon predictive structure available from recent history. Dependence in the remaining residual continues to shape longer-horizon divergence, while the pre-fork behavioral state predicts which problems are most susceptible to future instability. Together, these results show that the branch distribution provides a useful behavioral state for characterizing the learning dynamics of RLVR.

Reproducibility Statement

We provide detailed training and evaluation configurations, branch construction procedures, controlled-continuation protocols, statistical estimators, robustness analyses, and artifact provenance in the appendix. Each headline empirical result is linked to a frozen source artifact and executed analysis path. Code, analysis scripts, and frozen artifacts for reproducing the reported figures and tables are available in the anonymous repository: `https://anonymous.4open.science/r/ICLR2027_AnonymousGitHub-AB0F/`.

AI Use Statement

Generative AI tools were used to assist with research methodology and experimental design, implementation and debugging of analysis code, result interpretation, literature search, and drafting and editing parts of the manuscript. All AI-assisted analyses, code, citations, and scientific claims were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

John Aitchison. *The Statistical Analysis of Compositional Data*. Chapman and Hall, London, 1986.

Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. In *International Conference on Learning Representations*, 2024.

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

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11694.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Conference on Language Modeling*, 2025.

Yuzheng Hu, Fan Wu, Haotian Ye, David A. Forsyth, James Y. Zou, Nan Jiang, Jiaqi Ma, and Han Zhao. A snapshot of influence: A local data attribution framework for online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Guanhua Huang, Tingqiang Xu, Mingze Wang, Qi Yi, Xue Gong, Siheng Li, Ruibin Xiong, Kejiao Li, Yuhao Jiang, and Bo Zhou. Low-probability tokens sustain exploration in reinforcement learning with verifiable reward. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24158–24188, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1209. URL `https://aclanthology.org/2026.findings-acl.1209/`.

Renren Jin, Pengzhi Gao, Yuqi Ren, Zhuowen Han, Tongxuan Zhang, Wuwei Huang, Wei Liu, Jian Luan, and Deyi Xiong. Revisiting entropy in reinforcement learning for large reasoning models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25300–25322. Association for Computational Linguistics, 2026.

Michael Katz, Harsha Kokel, and Sarath Sreedharan. Seemingly simple planning problems are computationally challenging: The countdown game. *arXiv preprint arXiv:2508.02900*, 2025.

Anamika Lochab, Bolian Li, and Ruqi Zhang. Uniform-correct policy optimization: Breaking rlvr's indifference to diversity. *arXiv preprint arXiv:2605.00365*, 2026.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 217–227. Association for Computational Linguistics, 2020.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Core Francisco Park, Maya Okawa, Andrew Lee, Hidenori Tanaka, and Ekdeep Singh Lubana. Emergence of hidden capabilities: Exploring learning dynamics in concept space. *arXiv preprint arXiv:2406.19370*, 2024.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, 2020.

Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yi Ren, Shangmin Guo, and Danica J. Sutherland. Better supervisory signals by observing learning paths. In *International Conference on Learning Representations*, 2022.

Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Raluca Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick. The multiberts: Bert reproductions for robustness analysis. In *International Conference on Learning Representations*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 9275–9293, 2020.

Xinyu Tang, Yuliang Zhan, Zhixun Li, Xin Zhao, Zhenduo Zhang, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Rethinking sample polarity in reinforcement learning with verifiable rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2928–2954, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.134. URL https://aclanthology.org/2026.acl-long.134/.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3850.

Shaohang Wei, Zikun Su, Feifan Song, Wen Luo, Wei Li, Guangyue Peng, and Houfeng Wang. Verifier-induced support reshaping in on-policy optimization, 2026. URL `https://arxiv.org/abs/2608.00220`.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Suqin Yuan, Jinkun Chen, Jiyang Zheng, Muyang Li, Lei Feng, Dadong Wang, Tao Xiang, Tongliang Liu, and Bo An. Understanding diversity collapse in rlvr via the lens of overtraining. *arXiv preprint arXiv:2606.15455*, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *Advances in Neural Information Processing Systems*, 2025.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

Qiancheng Zhou and Ruizhe Li. Locked at the entrance, open inside: Where rlvr narrows the solution space. *arXiv preprint arXiv:2608.29188*, 2026.

A Related Work

Learning dynamics of neural networks and language models. A broad line of work studies neural networks through their evolution during training rather than only through the properties of a converged model. Early work used example-level training trajectories to identify forgetting events, characterize ambiguous or difficult examples, and estimate the influence of training data across optimization (Toneva et al., 2019; Swayamdipta et al., 2020; Pruthi et al., 2020). A complementary line follows the emergence of model-internal structure and capabilities over training. For example, induction heads appear alongside the development of in-context learning (Olsson et al., 2022); mechanistic progress measures reveal distinct phases underlying grokking (Nanda et al., 2023); and syntactic attention structure emerges during a narrow training window associated with later grammatical capabilities (Chen et al., 2024). More recently, Ren & Sutherland (2025) study the learning dynamics of LLM finetuning by decomposing how the influence of successive updates accumulates among possible responses, connecting this view to instruction and preference optimization.

Our work follows the same general principle that intermediate training states contain information obscured by endpoint evaluation. We represent an RLVR checkpoint by its distribution over reproducibly identified behavioral branches. This representation makes it possible to study how behavior changes along a realized training trajectory and how reproducible future branch allocation is when training is continued repeatedly from the same state.

Reasoning behavior and diversity under RLVR. Recent work has increasingly examined how reinforcement learning with verifiable rewards changes the distribution of reasoning behaviors, beyond its effect on aggregate task accuracy. RLVR can shift probability toward successful behaviors already accessible to the base model (Yue et al., 2025; Zhao et al., 2025), while several studies report contraction in the broader solution distribution during training (Yuan et al., 2026; Lochab et al., 2026). Related analyses study entropy collapse and its relationship to exploration and reasoning diversity (Jin et al., 2026). At a finer level, Zhou & Li (2026) localize much of this contraction to early reasoning choices, showing that RLVR can narrow which solution families are entered even when multiple continuations remain available within a chosen family.

Several recent methods further connect this contraction to the on-policy training process. Tang et al. (2026) distinguish the roles of positive and negative self-generated samples, finding that they affect sharpening and exploration differently. Huang et al. (2026) study the progressive loss of low-probability exploratory tokens, while Wei et al. (2026) show that on-policy optimization can reshape which successful behaviors remain reachable within a finite rollout budget. Other work modifies the RLVR objective to preserve broader support over correct solutions (Lochab et al., 2026). These studies establish that probability allocation and exploration evolve substantially during RLVR. Our analysis studies the dynamics of the resulting branch distribution across successive updates. We characterize its reproducibility from a fixed training state and the extent to which the current behavioral state and remaining stochastic variation organize subsequent evolution.

Stochastic training, underspecification, and multiple learned solutions. A separate literature shows that nominally identical training procedures need not produce behaviorally identical models. Different random seeds can yield models with similar aggregate performance but different generalization behavior, representations, or downstream predictions (McCoy et al., 2020; Sellam et al., 2022; D'Amour et al., 2022). This observation motivates treating stochastic variation as part of the object of study rather than only as experimental noise. Related work in online reinforcement learning also studies how individual data or trajectories affect subsequent learning while the data distribution itself changes with the policy (Hu et al., 2025). Broader reinforcement-learning work documents run-to-run variability and the statistical uncertainty of few-run comparisons (Henderson et al., 2018; Agarwal et al., 2021), while recent language-model reasoning evaluations show sensitivity to decoding, random seeds, prompt formatting, and systems configuration (Hochlehnert et al., 2025). These works address evaluation practice across independently trained runs; our state-conditional design isolates the component of this variability that arises after a fixed checkpoint.

Because stochastic differences in independently trained runs may have accumulated throughout their entire histories, our controlled-continuation design conditions on the same complete resumable training state—including model parameters, optimizer state, scheduler state, and prior training history—and varies only subsequent training realizations. This separates variation generated after a specified

training state from variation inherited before it, allowing us to characterize a state-conditional distribution over future branch allocations.

Relation to our behavioral-dynamics view. These lines of work provide complementary views of training dynamics. Example-influence methods connect training data to later predictions; developmental and mechanistic analyses characterize when capabilities or structures emerge; RLVR diversity studies characterize changes in the support of reasoning behaviors; and multi-seed studies measure variability across trained models. Our analysis combines the temporal perspective of learning dynamics with an explicit behavioral state space and state-conditional continuations. The resulting object is a stochastic transition process over branch distributions, allowing selective reweighting and fixed-state non-reproducibility to be studied within the same representation. It also exposes how the current behavioral state and residual dependence organize subsequent branch evolution.

B Derivation and Scope of the Branch-Level Dynamics

The main text uses a branch-level view to motivate the empirical analysis. This section makes explicit the approximation behind that view and separates it from our later predictive claims.

B.1 A branch-level softmax surrogate

For a fixed problem q, write the branch probabilities as a softmax over surrogate branch logits u_b,

$$p(b \mid q) = \frac{\exp u_b}{\sum_{b'} \exp u_{b'}}.$$

Suppose that, after aggregating trajectory-level policy-gradient updates within a branch, the expected small update to these logits is

$$\mathbb{E}[\Delta u_b \mid X_t] = \eta\, a_t(q, b) + O(\eta^2),$$

where $a_t(q, b)$ is the expected normalized training advantage of trajectories entering branch b. The softmax Jacobian gives

$$\frac{\partial p_b}{\partial u_j} = p_b\big(\mathbf{1}[b = j] - p_j\big),$$

and therefore, to first order in η,

$$\begin{aligned}
\mathbb{E}[\Delta p_t(b \mid q) \mid X_t] &\approx \sum_j \frac{\partial p_b}{\partial u_j} \mathbb{E}[\Delta u_j \mid X_t] \\
&= \eta\, p_t(b \mid q) \left(a_t(q, b) - \sum_j p_t(j \mid q) a_t(q, j) \right) \\
&= \eta\, p_t(b \mid q)\big(a_t(q, b) - \bar{a}_t(q)\big).
\end{aligned}$$

This is the replicator-like relation used in Section 3. It is an *idealized branch-level surrogate*; the full GRPO update is not represented as a closed-form process in branch space. The real update couples branches through shared parameters, group-normalized advantages, optimizer state, and the trajectories sampled at that step. We use the relation only for the qualitative prediction that current branch probability and relative advantage jointly shape expected reweighting.

B.2 State-conditional futures are defined from the complete training state

Let S_t denote the complete resumable training state and $X_t(q) = \Phi_q(S_t)$ its behavioral projection. The state-conditional future law

$$\mathcal{P}_h(\cdot \mid S_t, q) = \mathrm{Law}(X_{t+h}(q) \mid S_t)$$

is defined with respect to repeated continuations from the same S_t. This is different from comparing independently trained runs, which may already differ before the observation window. Our strongest claims about fixed-state non-reproducibility therefore rely on the full-state Countdown forks; the 7B and Code trajectories are used only as trajectory-level breadth checks.

Table 2: **Experimental inventory.** Only Track A provides the primary full-state forks. Track B is a same-seed replay control of Track A. Tracks C–F provide trajectory-level breadth.

Track	Model / task	Trajectory design	Evaluation	Role
A	Qwen2.5-3B / Countdown	self-trained trunk + four-root full-state forks	106 prompts, 256 rollouts	primary controlled setting
B	Qwen2.5-3B / Countdown	same-seed replays of Track A	106 prompts, 256 rollouts	implementation-nondeterminism control
C	Qwen2.5-7B / Countdown	two independent RLVR seeds	steps 5–25, 106×256	scale-side trajectory check
D	public Qwen2.5-3B / Countdown	five continuations from public step 100	steps 105–125, 106×1024	dense short-horizon dynamics
E	Qwen-1.5B Code / HumanEval	public released trajectory, 8 states	24 prompts, 128 rollouts/state	cross-task trajectory check
F	OLMo-3.1-7B Code / HumanEval	public released trajectory, 6 states	24 prompts, 128 rollouts/state	cross-family trajectory check

B.3 Predictive near-first-order dynamics

An exact first-order Markov statement at observation interval Δ would require

$$p(X_{t+\Delta} \mid X_t, H_t^-) = p(X_{t+\Delta} \mid X_t),$$

where H_t^- denotes earlier observed history. Our test is predictive rather than nonparametric: within the fixed conditional multinomial model family of Section 5.3, we measure how much held-out next-state log loss improves when recent velocity and then one additional lag are added to the current branch state. The observed gain from recent velocity is positive and amounts to 0.19% of the state-only NLL, while one older velocity does not improve prediction. We therefore use *near-first-order predictive summary* rather than *Markov state* or *sufficient state*. The claim applies to this behavioral projection and this five-update observation scale, not to the complete parameter-space training state.

C Experimental Details and Reproducibility

This section records the training, evaluation, branch-construction, and continuation details omitted from the main text. Countdown is the primary controlled setting. The remaining trajectories broaden the trajectory-level analysis across model scale, training source, and task. We report upstream training metadata only when they can be verified from frozen artifacts rather than reconstructing them from edited launchers or unverified defaults.

C.1 Experimental inventory and provenance

The primary base model is `Qwen/Qwen2.5-3B-Instruct`; its full revision hash is recorded in the artifact manifest (short hash `aa8e72537993`). Track C uses the local `Qwen2.5-7B-Instruct-RL` lineage. Track D begins from the public `philschmid/qwen-2.5-3b-r1-countdown` checkpoint-100. Its download-time revision was not recorded, so we do not assign one retroactively. Tracks E and F use `polaris-73/DeepSeek-R1-Distill-Qwen-1.5B-RLVR-Code` and `allenai/Olmo-3.1-7B-RL-Zero-Code`, respectively. Exact per-checkpoint commit hashes for both Code lineages are present in the frozen provenance artifacts. Their upstream RLVR hyperparameters are not available in a form we can verify; all claims using these tracks are therefore within-lineage comparisons based on the verified evaluation outputs.

C.2 Primary Countdown training configuration

Held-out Countdown evaluation uses temperature 0.7, top-p 0.9, top-k disabled, maximum 1024 generated tokens, and evaluation seed 909. The main multi-root analyses use 256 rollouts per problem. The dense Track D analysis combines disjoint evaluation batches to obtain 1024 rollouts per problem at each post-fork checkpoint.

Table 3: **Effective configuration for the controlled 3B Countdown training.** Values were verified against the runtime manifest and executed trainer configuration.

Field	Value
Base model	Qwen2.5-3B-Instruct
RL objective	GRPO, $\beta_{\mathrm{KL}} = 0.001$
Learning rate	5×10^{-7}
Schedule	cosine, warmup ratio 0.03, 450-step horizon
Prompt batch / update	24
Completions / prompt	8
Completion batch / update	192
Training ranks	3
Training temperature	0.9
Training top-p / top-k	1.0 / disabled
Maximum completion length	1024
DeepSpeed	ZeRO stage 3
Model seed / data seed	42 / 42

Table 4: **Realized Countdown branch coverage.** Coverage is the fraction of all sampled rollouts that are both parseable and solver-feasible.

Track	Feasible coverage	Unparseable	Parseable infeasible
A: controlled 3B	0.7923	0.0024	0.2054
B: same-seed replay	0.8381	0.0001	0.1617
C: 7B	0.6668	0.0714	0.2617
D: public dense 3B	0.7890	0.0092	0.2018

C.3 Countdown branch construction and conditioning

The Countdown entrance parser is executed when each rollout is generated. It searches only the reasoning prefix preceding the first `</think>`, `<feasible>`, or `<answer>` marker. The executed pattern is

```
(?<![\w.])(?P<first>-?\d+(?:\.\d+)?)\s*(?P<op>[+\-*/×÷])
```

Unicode $\times$ and $\div$ are normalized to $\star$ and `/`. Negative integer operands are supported. A decimal first operand is treated as unparseable. The branch key is

$$g_q(y) = (\text{first integer operand}, \text{first operator}),$$

so concrete moves such as `8*3` and `8*7` both map to the family `8*`.

The feasible support $\mathcal{B}_q$ is frozen once per problem using an exact integer-arithmetic solver over ordered pairs. Division is legal only when it is exact. A family is viable if applying at least one concrete first move in that family leaves a residual multiset from which the target remains reachable. This is a property of the problem rather than of the sampled completion: a rollout can enter a viable branch and later produce an incorrect final answer. Unparseable and solver-infeasible entrances are counted separately and excluded from the conditional branch distribution. For feasible counts $n_t(q, b)$,

$$\widehat{p}_t(b \mid q) = \frac{n_t(q, b) + 0.5}{\sum_{b' \in \mathcal{B}_q} n_t(q, b') + 0.5|\mathcal{B}_q|}.$$

Track C has the lowest realized coverage and the largest unparseable fraction. Accordingly, we use it as a scale-side scope check and do not infer unchanged effect sizes at 7B.

C.4 HumanEval implementation entrances and validation

For HumanEval, branch labels are deterministic functions of the AST of the extracted target function. The target function is selected using the dataset `entry_point`; among module-level defini-

Table 5: **Validation of the deterministic HumanEval branch representation.**

Criterion	Value	Threshold / result
AST + target-function mapping coverage	0.9995	≥ 0.90 / pass
Prompts with at least two viable branches	23/24	≥ 12 / pass
Median effective evaluation coverage	0.9948	≥ 0.70 / pass
Cells where an Other label is modal	2/48	$< 1/2$ / pass
Whitespace invariance	600/600	pass
Comment/docstring-removal invariance	600/600	pass
Fenced/unfenced extraction invariance	600/600	pass

Table 6: **Representative Countdown entrances.** Both problems have five solver-feasible entrance families. The examples illustrate that viability is a property of the entrance, not a guarantee that the sampled completion ends correctly.

Problem	Numbers $\rightarrow$ target	Example entrance	Submitted answer	Outcome
cd/490267	$[75, 38, 31] \rightarrow 68$	`75-`	$75 - 38 + 31$	feasible, correct
cd/490267	$[75, 38, 31] \rightarrow 68$	`75+`	$75 - 38 + 31$	feasible, correct; first attempted move differs
cd/490267	$[75, 38, 31] \rightarrow 68$	`75-`	$(75 - 31 + 38) - 17$	feasible, wrong
cd/490334	$[14, 46, 38] \rightarrow 70$	`46+`	$46 + 38 - 14$	feasible, correct
cd/490334	$[14, 46, 38] \rightarrow 70$	`46-`	$38 + 14 - 46$	feasible, wrong

tions with that name, the last one is used, and nested definitions are considered only if no module-level match exists. We skip one leading docstring and any leading `pass` or bare `...`, then label the first remaining substantive statement. Nothing else, including apparently trivial initialization, is skipped.

Statement families. `Return`, `Assign`, `AnnAssign`, `AugAssign`, `If`, `For`, `While`, `Try`, `With`, `Expr`, and `OtherStmt`. Asynchronous `for` and `with` fold into `For` and `With`.

Expression families. `Call`, `Compare`, `Arithmetic`, `Boolean`, `ListComp`, `SetComp`, `DictComp`, and `Generator`; `SliceSubscript`, `ContainerLiteral`, `Constant`, `Name`, and `OtherExpr`. A `Try` statement is forced to `OtherExpr`; a bare `return` maps to `Constant`; and unary `Not` maps to `Boolean` while other unary operators map to `Arithmetic`.

Call families. When the governing expression is a call, we append one of `sorted`, `set`, `len`, `range`, `sum`, `max`, `min`, `any`, `all`, `enumerate`, `zip`, `recursive_self_call`, or `other_call`. Attribute calls such as `s.split()` map to `other_call`. The resulting label is therefore `statement:expression`, with a third call field when needed.

The viable support is frozen from discovery rollout IDs 0–63, pooled across both Code lineages and all observed states. A branch is viable for prompt q if at least one discovery completion with that label passes all HumanEval tests. Rollout IDs 64–127 form the evaluation split and are never consulted when the codebook is constructed. An evaluation completion can therefore pass all tests while remaining off-support if its branch was never established as viable in discovery.

All final Code results use the deterministic AST representation above.

C.5 Worked branch examples

The examples below are selected by deterministic median-based rules rather than by effect size. For Countdown, one problem is the lower-median support size and a second is the lower-median Track-A excess JS. For HumanEval, examples are selected around the median number of viable branches among the 23 prompts with at least two viable branches.

For cd/490267, the modal family covers 240/256 completions, while three sampled completions enter a feasible branch but end with an incorrect final answer. For cd/490334, the modal family

Table 7: **Representative HumanEval implementation entrances.** Viability is frozen from discovery; an evaluation completion may therefore pass all unit tests while remaining off-support.

Prompt	Entrance label	Evaluation behavior	Discovery viability
HumanEval/2, `truncate_number`	`Assign:Call:other_call`	passes 3/3 tests	viable
HumanEval/2, `truncate_number`	`Assign:SliceSubscript`	passes 3/3 tests	viable
HumanEval/2, `truncate_number`	`If:Boolean`	passes 3/3 tests	off-support
HumanEval/41, `car_race_collision`	`Assign:Arithmetic`	fails evaluation tests	viable
HumanEval/41, `car_race_collision`	`Assign:Call:other_call`	fails evaluation tests	viable

Table 8: **Controlled-continuation designs.**

Design	Shared root(s)	Arms	Evaluation
Multi-root Track A	75, 100, 125, 150	4 per root	common endpoint 175
Dense Track D	100	5	every 5 updates through 125

covers 128/256 completions and one feasible entrance ends incorrectly. These examples also show that many distinct full reasoning traces collapse to the same entrance family.

For HumanEval/2, two passing implementations begin by assigning a value through different syntactic forms, while a third passing evaluation completion begins with `If:Boolean` but is off-support because no discovery completion had established that branch as viable. Hence *off-support* means "not established by the discovery split," not "incorrect."

C.6 Controlled-continuation protocol

The multi-root experiment restores complete resumable states at $t \in \{75, 100, 125, 150\}$ and trains four arms from each root to the common endpoint 175. The balanced arms are the trunk continuation plus rollout seeds 1, 2, and 3. The dense Track D experiment instead follows five continuations from one shared step-100 state at steps 105, 110, 115, 120, and 125. The former evaluates fixed-state reproducibility at multiple training states; the latter resolves short-horizon propagation from one state.

C.7 What is fixed and what varies after a fork

The full-state fork changes the rollout stream rather than the training configuration. Table 9 summarizes the components that are restored from the checkpoint and the quantities allowed to vary after the fork.

The trunk and same-seed replays use rollout seed 0; the three stochastic fork arms use seeds 1, 2, and 3. Dataloader-worker and any separate sampler state are not hashed individually, but the observable ordered prompt-row stream they would affect is verified step by step.

C.8 Same-seed replay control plane

Same-seed replays additionally hold the rollout seed fixed at 0. All four replays eventually diverge at the completion-token level, despite matching the prompt rows and the restored control plane. Importantly, the fraction of bitwise-identical completions is *not* monotone in replay length, so we do not interpret it as a decay law.

At root 75 the replay remains bitwise identical for two full post-root updates before the first differing completion at step 78. At the other three roots the first post-root step already differs. Learned states are different by step 175 in every replay. State hashes are not available at every optimizer step, so the exact optimizer update at which the learned weights first differ is not reported.

Table 9: **Fork semantics.** The rollout seed is the single intended between-arm difference. Subsequent completions, gradients, and model states are consequences of that difference.

Status	Components	Verification
Fixed	bf16 parameters; FP32 master weights; Adam moments; optimizer; scheduler; global step	checkpoint + state hashes
Fixed	trainer state; data cursor; prompt order; batch construction	trainer state + per-step prompt-row hashes
Fixed	Python, NumPy, Torch CPU, and CUDA RNG state	per-rank RNG checkpoint files
Fixed	DeepSpeed ZeRO-3 state; verifier; generation configuration	checkpoint shards + source/config hashes
Varies by design	rollout / generation seed	`rollout_seed` $\times 100000 +$ `global_step`
May vary after fork	completion tokens; realized gradients; later model states	consequences of the realized rollout stream
Not forced deterministic	CUDA / kernel nondeterminism	measured separately by same-seed replay

Table 10: **First completion-level divergence in same-seed replays.** The row ID still matches at the first differing slot, localizing the initial mismatch to generation rather than to data ordering.

Root	Replay length	First divergence	Steps after root	First slot	Identical slots / total	Fraction
75	100	78	3	rank 0 / slot 0	629 / 19200	0.0328
100	75	101	1	rank 0 / slot 0	340 / 14400	0.0236
125	50	126	1	rank 0 / slot 0	414 / 9600	0.0431
150	25	151	1	rank 0 / slot 0	494 / 4800	0.1029

C.9 Software, hardware, and compute

The primary Track A environment uses Python 3.12.14, PyTorch 2.8.0+cu128, Transformers 4.57.1, TRL 0.23.0, DeepSpeed 0.19.7, vLLM 0.11.0, CUDA 12.8, and NVIDIA driver 580.65.06. Training uses three ranks plus one rollout process on four NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs. Summing recorded in-step wall times gives 104.93 productive GPU-hours over 1201 optimizer steps, or 116.5 GPU-hours when failed/crashed attempts with recorded steps are included. These figures exclude process launch, model loading, checkpoint writes, and held-out evaluation. Countdown evaluation used the frozen vLLM 0.10.1.1 stack on A6000 GPUs. We do not report total evaluation GPU-hours because the older evaluation artifacts do not contain complete timing records.

D Statistical Procedures

Unless stated otherwise, uncertainty is clustered by problem: all observations from a sampled question are resampled together. This preserves dependence between checkpoints and continuations of the same problem. All stochastic procedures below use the frozen seeds and executed analysis paths recorded in our artifact manifest.

D.1 Scalar-sharpening calibration

For each question and checkpoint pair, the sharpening prediction is

$$p_t^{(\gamma)}(b \mid q) = \mathrm{softmax}_b(\gamma \log p_{t_0}(b \mid q)),$$

and the discrepancy is

$$D_t(q) = \frac{1}{|\mathcal{B}_q|} \sum_{b \in \mathcal{B}_q} \left| \log p_t(b \mid q) - \log p_t^{(\gamma)}(b \mid q) \right|.$$

We search 77 values $\gamma \in \{0.20, 0.25, \ldots, 4.00\}$ and minimize the multinomial NLL of the target counts. The primary sensitivity allows a separate γ for each question. We draw 2,000 parametric

bootstrap replicates from the fitted scalar-sharpening model. On every draw, counts are resampled multinomially at *both* t_0 and t using their realized budgets, γ is refit, and $D_t(q)$ is recomputed. The resulting bootstrap distribution therefore reflects both finite-rollout variation and sharpening-parameter estimation error. Confidence intervals use 10,000 question-bootstrap draws; the cross-setting meta figure uses 2,000 re-aggregation draws. The one-sided p-value uses the usual $+1$ correction.

D.2 Same-state omnibus tests

The primary test behind "detectably different" compares all K arms at one fork root. For each problem, the test statistic is the mean pairwise JS divergence between the Dirichlet-0.5-smoothed branch distributions of all arm pairs. All arms use the same evaluation seed, and rollout slot i therefore uses the same per-slot sampler seed in every arm. The permutation test shuffles arm labels within each rollout slot, preserving this common-random-number pairing. We use 20,000 blocked permutations per problem. A problem counts as detectably different when its permutation p-value survives Benjamini–Hochberg correction at $q \leq 0.05$, with BH applied only across computable problems within the same $(\text{root}, \text{test}, K)$ cell and never pooled across roots. A problem is non-computable only if no arm contains a feasible slot; fully identical arm labels remain computable and receive $p = 1$.

A secondary marginal omnibus uses ratio-estimator influence functions, a Moore–Penrose pseudo-inverse, and 20,000 paired Rademacher wild-bootstrap draws. Rare branches with pooled count below 5 are folded into `OTHER`. It is used as a sensitivity analysis rather than as the headline statistic.

D.3 Budget-matched finite-rollout floors

A finite-rollout floor is constructed by splitting the realized branch counts from one continuation into two halves using a multivariate hypergeometric draw without replacement. The halves are therefore an exact partition of the observed rollout set rather than two bootstrap resamples. We average 25 independent split draws. Crucially, the continuation comparison is recomputed at the *same half budget* as the floor. Subtracting a half-budget floor from a full-budget JS inflates the noise correction and biases excess JS downward; in the dense trajectory the shortest-horizon excess changes from -0.00078 under the unmatched construction to $+0.00053$ under the matched construction. All reported excess-JS curves use the matched estimator. Question-cluster confidence intervals use 2,000 bootstrap draws.

D.4 Directional memory

For consecutive observed states, define

$$v_i(q) = z_{t_{i+1}}(q) - z_{t_i}(q).$$

At lag L we report the mean cosine between increments separated by L checkpoint intervals. Units with norm below 10^{-12} are dropped. The shuffled-time reference uses 200 permutations, and uncertainty uses 2,000 question-bootstrap draws. Adjacent increments share one intermediate state with opposite sign, which induces a negative lag-1 shuffled reference; the relevant comparison is therefore to the shuffled interval rather than to zero. The longest-lag estimates are based on few increment pairs and are treated as weak rather than as precise point estimates.

For the dense continuation, perturbation magnitude is corrected in aggregate by subtracting matched measurement energy before taking the square root. We do not apply row-wise clipping, which would bias small signals upward.

D.5 Predictive-history models

The predictive analysis uses conditional multinomial logits over the frozen solver-feasible support. The response is the next checkpoint's branch-count vector. Each unit is a $(\text{question}, \text{continuation}, \text{transition})$ cell, and its contribution to NLL is weighted by its total next-state count. The three feature sets are

$$M_1 = (z_t, c), \qquad M_2 = (z_t, c, v_t), \qquad M_3 = (z_t, c, v_t, v_{t-5}),$$

where c contains log solver multiplicity, four operator indicators, and log first operand. Features are standardized using training folds only; the model has no intercept and uses an L_2 penalty $\lambda = 10^{-3}$. Optimization uses L-BFGS-B with a maximum of 500 iterations. We use question-grouped five-fold cross-validation with fold seed 20260916. Confidence intervals for NLL gains use 10,000 question-cluster bootstrap draws.

The first-order reference fits the state-only model, simulates complete trajectories at the observed rollout budgets, and reruns the same model comparison. The $M_1 \to M_2$ gain remains positive when any one continuation or any one transition is excluded. Because M_3 requires an additional lag, its absolute NLL is evaluated on a shorter transition scope; only within-row NLL differences are interpreted.

D.6 Residual decomposition and recursive simulation

Using the out-of-fold state-only predictor, define the one-step CLR residual

$$r_{q,t,s} = z^{\text{obs}}_{q,t+\Delta,s} - z^{\text{pred}}_{q,t+\Delta,s}.$$

For a (q,t) cell with $S \geq 3$ continuations, let $\bar{r} = S^{-1}\sum_s r_s$. We estimate

$$E_\eta = \frac{S}{S-1}\frac{1}{S}\sum_s \|r_s - \bar{r}\|^2, \qquad E_\mu = \|\bar{r}\|^2 - \frac{E_\eta}{S}.$$

Finite-rollout measurement energy is estimated from 32 independent multinomial redraws of the observed next-state distribution and is subtracted only from the continuation-specific component. It is not subtracted from the shared component because target-side measurement noise is independent across continuations. Component intervals use 10,000 question-bootstrap draws.

Every recursive simulator starts from the real shared step-100 state and uses the same frozen out-of-fold state-only drift predictor. S_0 adds no process noise. S_1 injects independent Gaussian noise matched to the entire corrected one-step residual energy. S_2 preserves an empirical residual vector $\mu_{q,t}$ shared across simulated continuations and independently samples a continuation-specific residual from the corresponding $(\text{transition}, K)$ pool. S_{2g} replaces the empirical continuation-specific draw with a Gaussian of matched corrected energy. $S_1'A$ permutes shared residual vectors across questions independently for each continuation within each $(\text{transition}, K)$ stratum; $S_1'B$ preserves the question-specific vector but independently flips its sign across continuations. We use 200 simulation replicates per simulator. Observation noise is introduced only when a latent state is converted back to sampled branch counts at the real rollout budget.

D.7 Instability and winner-switching metrics

The pre-fork analysis uses features measured only at step 100 and outcomes at step 125. Its binary target is whether the five continuations disagree about the post-fork modal branch; the continuous target is their future pairwise JS. For the Track-A multi-root analysis (Appendix I), features are computed at each fork root, the binary target is four-arm modal disagreement at the common endpoint 175 estimated from 256 rollouts per arm, and the split-half measurement control follows Appendix D.3. The fixed-horizon analysis instead defines persistence as equality of the modal branch at t and $t + \Delta$, with switching as its complement. For the raw margin result, the switch score is the negative top-1–top-2 probability gap: smaller current margin predicts a future switch.

Logistic models use `C=1.0`, training-fold standardization, and a maximum of 5000 iterations under the frozen question-level five-fold split. AUROC and AUPRC use standard sklearn definitions. Confidence intervals cluster all cells of one question and use 2,000 bootstrap draws with seed 20260929. The near-tie sensitivity excludes cells with margin below 0.01. Leave-one-continuation-out results recompute the raw-margin AUROC after dropping each of the five continuations in turn.

E Additional Evidence for Selective Reweighting

E.1 Behavioral displacement

Departure from scalar sharpening is our primary evidence for branch-specific reallocation. As a complementary descriptive measure, we also track displacement from the earliest observed check-

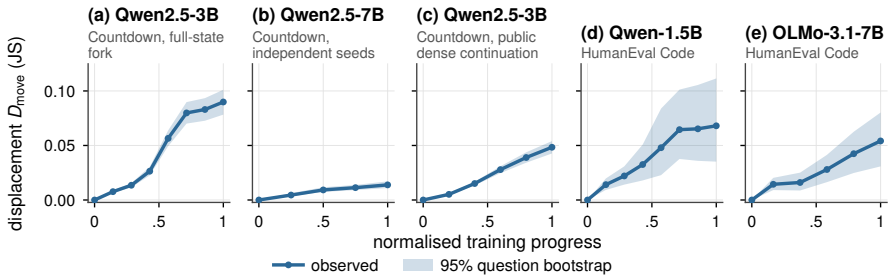


Figure 7: **Behavioral states move away from their earlier checkpoints.** Solid curves show mean JS displacement from the earliest observed state; dashed curves subtract a rollout-matched measurement floor. Shaded regions show 95% question-bootstrap intervals.

Table 11: **Descriptive concentration over each observed trajectory.**

Setting	Normalized entropy	Top-1 mass
Qwen2.5-3B Countdown	$0.6354 \rightarrow 0.3019$	$0.6277 \rightarrow 0.8417$
Qwen2.5-7B Countdown	$0.4784 \rightarrow 0.5109$	$0.7268 \rightarrow 0.7027$
Public Qwen2.5-3B Countdown	$0.4983 \rightarrow 0.3350$	$0.7178 \rightarrow 0.7932$
Qwen Code	$0.4670 \rightarrow 0.4289$	$0.7194 \rightarrow 0.7509$
OLMo Code	$0.4551 \rightarrow 0.4632$	$0.6749 \rightarrow 0.7114$

point,

$$D_{\text{move}}(t, q) = \text{JS}(X_t(q), X_{t_0}(q)) \,.$$

Across the observed windows, branch states generally move away from their earlier values, although the magnitude and checkpoint spacing differ across settings.

E.2 Descriptive concentration statistics

Entropy and top-1 mass describe concentration but do not distinguish branch-specific reweighting from scalar sharpening. We therefore report them only as descriptive quantities.

E.3 Scalar-sharpening sensitivity

On the self-trained 3B trunk, the earliest interval is compatible with scalar sharpening, while later intervals depart from it. The same conclusion survives the more permissive per-question-γ fit that absorbs heterogeneous concentration rates.

Across the broader Countdown analyses, the departure is positive and significant in 30/34 self-trained 3B pair-models and 44/50 public dense- continuation pair-models, compared with 5/16 comparisons on the short 7B lineage. The latter is therefore treated as a scope check rather than as a matched replication.

F Additional Evidence for Fixed-State Non-Reproducibility

F.1 Balanced four-arm omnibus tests

The multi-root headline uses the primary blocked four-arm test defined in Appendix D. Table 13 gives the counts behind the 68–91% range in the main text.

The generalized implementation was also rerun at three arms against the frozen earlier analysis: three roots reproduce exactly and one differs by one problem, within the documented Monte-Carlo tolerance.

Table 12: **Departure from scalar sharpening on the self-trained 3B trunk.** The reference checkpoint is step 25.

Target	γ	Excess discrepancy [95% CI]	p
50	1.00	$+0.013$ $[-0.021, 0.045]$	0.220
75	1.00	$+0.148$ $[+0.105, +0.192]$	5×10^{-4}
100	1.05	$+0.290$ $[+0.238, +0.347]$	5×10^{-4}
125	1.05	$+0.659$ $[+0.585, +0.734]$	5×10^{-4}
150	1.30	$+0.848$ $[+0.758, +0.937]$	5×10^{-4}

Table 13: **Future branch distributions differ across continuations from the same complete training state.** "Different" means blocked-permutation p surviving within-root BH correction at $q \leq 0.05$.

Fork root	Significant / computable	Fraction	Mean statistic
75	86 / 106	0.811	0.01814
100	96 / 106	0.906	0.02611
125	83 / 105	0.790	0.02156
150	72 / 106	0.679	0.01460

F.2 Behavioral divergence versus task performance

The forks measure state-conditional variation in branch allocation; matching task performance is not required by the experimental design. Four of twelve primary continuation pairs have a paired pass@1 confidence interval excluding zero. At the same time, the ordering and magnitude of branch divergence do not track these performance differences.

At the prompt level, continuation pairs that choose different dominant branches have mean absolute accuracy difference 0.0390 over 140 prompt pairs, compared with 0.0450 over 1132 pairs whose dominant branch agrees. This comparison supports the use of branch divergence as a behavioral quantity distinct from accuracy divergence.

F.3 Same-seed replay

At root 75, a same-seed replay that restores the full state, rollout seed, and prompt order still produces endpoint branch divergence above its independent evaluation floor (Table 15).

Replays at roots 100, 125, and 150 also pass the control-plane checks in Appendix C.7. Their completion-level divergence is reported in Table 10.

G Additional Evidence for Short-Memory Dynamics

G.1 Dense continuation dynamics

In the dense public 3B trajectory, pairwise branch divergence increases with post-fork horizon after subtracting a matched finite-rollout floor. The corresponding perturbation norm also grows even though directional alignment weakens. At lags 5, 10, 15, and 20 updates, the measurement-corrected cosine is 0.889, 0.574, 0.393, and 0.121, respectively; the lag-20 confidence interval reaches zero and is based on the sparsest comparison. The result therefore shows clear short-range memory, while the longest-lag confidence interval includes zero; it does not identify a smooth decay law with a precise endpoint.

G.2 Residual temporal memory

After subtracting the out-of-fold state-only drift predictor, temporal alignment is much smaller.

Table 14: **Branch-space divergence is not ordered by pass@1 divergence.** Intervals are paired question bootstraps.

Root	Pair	Branch JS	Δ pass@1	95% CI	Excludes 0
75	trunk vs seed1	0.01659	+0.0413	$[+0.0285, +0.0548]$	yes
75	trunk vs seed2	0.02471	+0.0204	$[+0.0052, +0.0359]$	yes
75	seed1 vs seed2	0.02611	-0.0210	$[-0.0386, -0.0042]$	yes
100	trunk vs seed1	0.02399	-0.0005	$[-0.0123, +0.0113]$	no
100	trunk vs seed2	0.03641	+0.0156	$[+0.0003, +0.0311]$	yes
100	seed1 vs seed2	0.03393	+0.0161	$[-0.0021, +0.0341]$	no
125	trunk vs seed1	0.01219	-0.0135	$[-0.0279, +0.0012]$	no
125	trunk vs seed2	0.00958	-0.0067	$[-0.0153, +0.0018]$	no
125	seed1 vs seed2	0.00945	+0.0068	$[-0.0043, +0.0179]$	no
150	trunk vs seed1	0.00686	-0.0021	$[-0.0070, +0.0030]$	no
150	trunk vs seed2	0.02704	+0.0106	$[-0.0051, +0.0265]$	no
150	seed1 vs seed2	0.01847	+0.0128	$[-0.0027, +0.0284]$	no

Table 15: **Behavioral divergence persists in the frozen root-75 same-seed replay.**

Condition	Endpoint JS	Measurement floor	Prompts passing blocked test
Same-seed replay	$0.00740\,[0.00597, 0.00905]$	$0.00320\,[0.00205, 0.00459]$	39/106

The largest absolute residual cosine is 0.056, compared with approximately 0.203 for raw increment memory on the same trajectory. Under the full shuffled-time intervals, no residual lag provides robust evidence for a persistent residual direction.

G.3 Predictive-model robustness

The $M_1 \to M_2$ held-out NLL gain is $+0.001154\,[+0.000747, +0.001594]$. A first-order simulation control has a slightly negative expected gain, and the observed improvement lies beyond that reference at $p = .001$. Leaving out any one continuation gives a gain of $+0.001055$; leaving out any one transition keeps the gain positive, ranging from $+0.000533$ to $+0.001964$. The older-history comparison $M_2 \to M_3$ is $-0.000037\,[-0.000055, -0.000019]$ and therefore does not support persistent additional memory.

H Additional Evidence for Residual Structure

H.1 One-step decomposition

For the state-only transition predictor, the raw out-of-fold residual energy is 1.816. After measurement correction, the decomposition is reported in Table 17.

The component shares depend on the predictor and transition scope. On the four-transition scope where both state-only and state-plus-velocity predictors are defined, their shared and continuation-specific shares change substantially. We therefore use the decomposition to establish residual dependence rather than as a universal variance law for RLVR.

H.2 Recursive divergence forecasts

At horizon 25, the absolute JS forecast errors are 0.09592 for S_1, 0.07901 for $S_1'A$, 0.07550 for $S_1'B$, and 0.03619 for S_2. Relative to the total $S_1 \to S_2$ error reduction, the two interventions attribute 71.7% and 65.8% of the improvement to preserving cross-continuation sharing. The ordering $S_2 < S_1' < S_1$ holds at every evaluated horizon. S_2 predicts 0.06194 at the endpoint versus 0.02574 observed. We therefore use it to compare residual structures rather than as a calibrated generative model.

H.3 Leave-one-continuation-out robustness

All five reductions are positive, showing that the benefit of preserving shared residual dependence is not driven by any single continuation.

Table 16: **Residual temporal alignment stays near the shuffled reference after removing predicted state-dependent drift.**

Lag	Residual cosine [95% CI]	Shuffled mean
1	$-0.0520\ [-0.0874, -0.0168]$	-0.0078
2	$+0.0501\ [+0.0155, +0.0863]$	-0.0055
3	$+0.0235\ [-0.0227, +0.0665]$	-0.0073
4	$-0.0562\ [-0.1143, -0.0005]$	-0.0048

Table 17: **One-step residual-energy decomposition for the dense continuation experiment.**

Component	Energy	Share of raw residual
Finite-rollout measurement	0.688	0.379
Shared across continuations	0.454	0.250
Continuation-specific, corrected	0.674	0.371

I Additional Evidence for Instability Prediction

I.1 Prediction from the shared pre-fork state

The pre-fork features use only the shared step-100 state, while targets are computed from the five continuations at step 125. At the 1024-rollout budget used for this analysis, 24 of 106 problems end with disagreement among the continuations about the modal branch. The original 256-rollout robustness analysis gives 23/106, with 22–23/106 under OTHER-inclusive variants (Table 23).

The state features are branch entropy, top-1 mass, and top-1–top-2 margin; the controls are branch count, pass@1, and effective sample size. For the continuous future-JS target, the state-only Spearman correlation is $+0.527\,[+0.360, +0.662]$, while the combined model gives $+0.519\,[+0.341, +0.654]$. The strongest predictive signal is therefore for whether the modal winner becomes unstable, while the continuous future-JS target also benefits from the ordinary controls.

I.2 Fixed-horizon winner switching

At $\Delta = 5$, excluding cells with margin below 0.01 leaves AUROC $0.9364\,[0.9000, 0.9630]$. Leaving out one continuation at a time yields AUROCs 0.9346, 0.9307, 0.9245, 0.9420, and 0.9409. A supplementary logistic fit gives an odds ratio of $15.59\,[8.53, 38.06]$ per standard deviation of margin and a Brier score of 0.0310. The raw-margin ranking remains the primary result because it directly tests the proposed state variable without requiring a learned predictor.

The lowest margin sextile contains 108 of 119 observed switches at $\Delta = 5$, showing that future switching is strongly concentrated among currently close branch competitions.

I.3 Multi-root instability prediction

The Track-A full-state forks provide an independent-lineage test of whether future instability is already encoded in the pre-fork behavioral state. For each fork root $r \in \{75, 100, 125, 150\}$, features are computed from the shared state at the root, while the target is whether the four balanced continuations disagree on their modal branch at the common endpoint 175. This gives horizons of 100, 75, 50, and 25 updates. Table 22 reports the per-root results, and Figure 8 summarizes the same estimates by horizon.

All four AUROC confidence intervals exclude chance, and the corresponding correlations with continuous endpoint divergence are positive with confidence intervals excluding zero. The signal therefore remains detectable at every tested horizon through 100 updates. The horizon dependence is non-monotone, so we do not infer a temporal decay law from these four fork roots.

A pooled predictor over the four roots gives AUROC 0.844 using branch-state features, compared with 0.542 using ordinary controls alone and 0.872 when the two are combined. The pooled predictor uses the same frozen question-level folds across roots, so all observations from a given problem

Table 18: **Recursive mean pairwise JS across horizons.** Preserving the shared residual component consistently moves the forecast away from the fresh- iid baseline and toward the observed continuation ensemble.

Model	5	10	15	20	25
Observed	0.00209	0.00652	0.01366	0.01905	0.02574
S_0 drift only	0.00147	0.00146	0.00151	0.00169	0.00193
S_1 iid full residual	0.02611	0.05086	0.07652	0.10066	0.12166
$S_1' A$ sharing permuted	0.02328	0.04616	0.06740	0.08641	0.10476
$S_1' B$ sharing sign-flipped	0.01356	0.03347	0.05512	0.07813	0.10124
S_2 shared + empirical specific	0.00636	0.02030	0.03483	0.04899	0.06194
S_{2g} shared + Gaussian specific	0.00432	0.01726	0.03135	0.04487	0.05642

Table 19: **Leave-one-continuation-out endpoint forecast errors.**

Dropped	S_1 error	S_2 error	Reduction
seed 1	0.09894	0.03174	67.9%
seed 2	0.10113	0.04227	58.2%
seed 3	0.10228	0.03188	68.8%
seed 4	0.09692	0.03087	68.2%
seed 5	0.10003	0.03131	68.7%

remain in the same train or test fold. Leave-one-arm-out AUROCs stay within 0.07 of their four-arm values, and excluding pre-fork margins below 0.01 changes AUROC by at most 0.012.

Finite-rollout modal-uncertainty control. Because endpoint modal disagreement is itself estimated from finite rollouts, we apply the multivariate-hypergeometric split-half construction from Appendix D.3 to the Track-A endpoint counts. At the matched half budget, pairwise modal disagreement across distinct continuations is 7.3–11.3%, compared with 1.7–4.0% between two halves of the same continuation, a 2.3–5.3× separation across the four roots. The four-way cross-arm disagreement rate is also nearly unchanged when the budget is halved (0.139–0.203 versus 0.131–0.212 at full budget). As a second control, we mark a problem as measurement-unstable when, for any arm, the two split-half modal winners disagree in a majority of 100 repeated splits. Excluding these problems changes the raw-margin AUROC by at most 0.05. The split-half analysis is computable for 99–102 problems per root, and the baseline AUROC is recomputed on the same subset. These controls suggest that the Track-A margin signal is not driven primarily by finite-rollout uncertainty in the estimated endpoint modal branch.

J Scope and Additional Analyses

J.1 Entropy and process innovation

The one-step residual analysis permits a direct test of whether a single scalar such as current branch entropy determines the magnitude of future process innovation. Using measurement-corrected continuation-specific residual energy as the outcome, a pooled regression with branch count as a control gives an entropy coefficient of

$$-0.1675\ [-0.2498, -0.1063]$$

per standard deviation. Taken alone, the pooled coefficient is negative. However, transition-specific coefficients have signs $+ - - - -$, and the marginal entropy bins are non-monotone: corrected innovation is 0.811, 0.909, 0.755, 0.542, and 0.353 across increasing entropy bins centered near 0.16, 0.34, 0.44, 0.54, and 0.71. These results indicate that entropy alone does not summarize the state dependence of process-innovation magnitude.

J.2 Scale-side 7B reference

Track C contains two independently trained 7B seeds, not continuations from a common complete state. It therefore cannot test fixed-state reproducibility by design. Five of 16 sharpen-

Table 20: **Prediction from the shared pre-fork state.** The continuous column predicts future pairwise JS.

Features	AUROC [95% CI]	AUPRC	CV R^2
State only	0.779 $[0.670, 0.878]$	0.453	0.161
Controls only	0.514 $[0.349, 0.671]$	0.324	0.142
State + controls	0.804 $[0.705, 0.894]$	0.509	0.262

Table 21: **Current branch margin predicts a future modal switch.**

Horizon	AUROC [95% CI]	Switch AUPRC	Switch base rate
$\Delta = 5$	0.9346 $[0.8993, 0.9611]$	0.4106 $[0.3210, 0.5044]$	0.0449
$\Delta = 10$	0.9091 $[0.8708, 0.9400]$	0.4137 $[0.2974, 0.4970]$	0.0712

ing comparisons are positive, compared with 30/34 on the self-trained 3B trajectory and 44/50 on the public dense 3B trajectory. Under no injected effect, the frozen power analysis gives $P(\text{any rejection}) = 0.032$ and finds that 0.0090 ($d = 0.10$ in the perturbation construction) reaches at least 0.80 power across affected fractions. A smaller $d = 0.05$ effect reaches 0.725 power even when every prompt is affected. Within the observed 7B window, this analysis provides a scale-side reference for the trajectory-level effects while leaving smaller departures below this power range unresolved.

K Global Robustness Summary

Across these checks, the headline conclusions remain stable to support definition, continuation choice, transition choice, tie handling, and code formatting. The fixed-state reproducibility result remains stable to changes in support definition and continuation choice. The temporal, residual-structure, and instability analyses are likewise stable under their corresponding sensitivity checks.

L Frozen Source Tables and Artifact Provenance

For reproducibility, every headline analysis is associated with one frozen source table and one executed analysis path. Table 24 records the authoritative artifact for each main claim; superseded intermediate outputs are not used when writing the paper.

The frozen analysis package is tied to git commit `68a2908ebf83ca096d28c2cc520d99dee0eaa66b`. The canonical branch-state table, Code codebook, trajectory registry, fork plan, and each appendix evidence file are separately SHA256-hashed in the artifact manifest.

Table 22: **Pre-fork branch margin predicts future instability across Track-A fork roots.** AUROC uses the raw negative top-1–top-2 margin at the shared pre-fork state. Intervals are question-level bootstrap 95% confidence intervals.

Root	Horizon	Disagree	AUROC	95% CI	Spearman JS [95% CI]
75	100	19/106	0.789	[0.688, 0.877]	0.517 [0.34, 0.66]
100	75	26/106	0.841	[0.751, 0.916]	0.556 [0.40, 0.68]
125	50	21/106	0.768	[0.628, 0.889]	0.429 [0.23, 0.59]
150	25	16/106	0.924	[0.825, 0.989]	0.700 [0.56, 0.81]

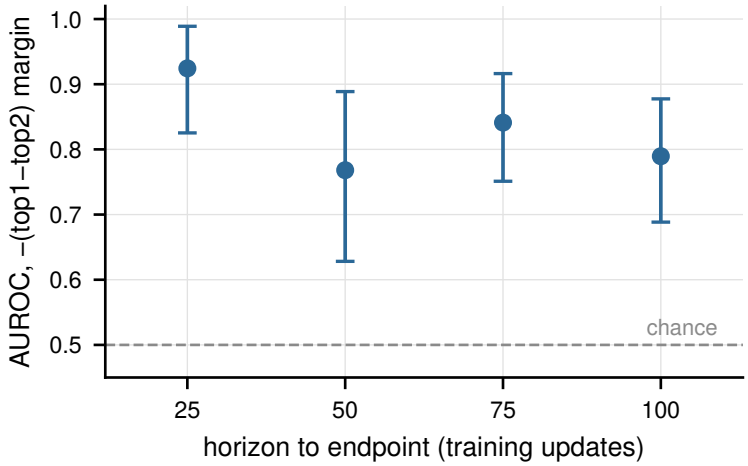


Figure 8: **Pre-fork branch margin predicts future instability across Track-A fork horizons.** Error bars show 95% question-bootstrap intervals. Horizons 25, 50, 75, and 100 correspond to fork roots 150, 125, 100, and 75, respectively. The points are left unconnected because the horizon dependence is non-monotone.

Table 23: **Summary of robustness checks for the main empirical claims.**

Main claim	Robustness check	Result
Fixed-state non-reproducibility	Include OTHER / off-support mass	at 256 rollouts, winner-disagreement count changes from 23/106 to 22–23/106
Fixed-state non-reproducibility	Leave one continuation out	unstable-count range 20–23/106
Directional memory	Leave one continuation out	lag-5 cosine 0.834–0.940; longest-lag estimate remains inconclusive
Near-first-order prediction	Leave one continuation / transition out	ΔNLL remains positive; transition range $+0.000533$ to $+0.001964$
Residual forecast	Leave one continuation out	$S_1 \rightarrow S_2$ endpoint error reduction 58.2–68.8%
Margin predicts switching	Exclude near-ties / leave one continuation out	AUROC 0.9364 after tie exclusion; LOCO 0.9245–0.9420
Track-A multi-root instability	Leave one arm out / exclude margin < 0.01	leave-one-arm AUROCs stay within 0.07 of four-arm values; near-tie exclusion changes AUROC by at most 0.012
Track-A modal measurement control	Matched split-half modal uncertainty	pairwise cross-arm disagreement is 2.3–5.3$\times$ the within-arm rate; excluding measurement-unstable problems changes AUROC by at most 0.05
Code representation	Formatting / comments / code fences	600/600 labels unchanged under each perturbation

Table 24: **Authoritative frozen artifacts for the main empirical claims.**

Claim	Frozen source	Executed analysis
Selective reweighting	`sharpening_null.csv`	`p1_sharpening.py`
Fixed-state non-reproducibility	`balanced_omnibus.json`	`bk4_omnibus.py`
Directional memory	`meta3_temporal_memory.csv`	`meta_build.py`
Near-first-order prediction	`predictive_history.csv`	`p3_predictive.py`
Residual structure	`recursive_forecast_1024.csv`	`sr_p5_residual.py`, `sr_p6_recursive.py`
Instability prediction	`instability_prediction_prefork.csv` + raw-margin JSON	`sr_p789_susceptibility.py`, `tc_item4_raw.py`
Track-A multi-root instability	`trackA_prefork_instability_paper_table.csv` + `trackA_prefork_instability_robustness_final.csv`	`ta_prefork_v2.py`, `ta_v2_package.py`
Track-A modal measurement control	`trackA_matched_budget_comparison.csv` + `trackA_measurement_control_summary.csv`	`ta_p0_measctl.py`