GAUGE: GAUGE-INVARIANT POLICY OPTIMIZATION FOR LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy optimization for language agents typically constrains changes in token likelihood, although environment transitions and rewards depend on executable actions obtained by parsing the generated text. This mismatch can over-penalize different textual renderings of the same action while under-resolving small but behaviorally critical changes in action arguments. We formalize this distinction by decomposing raw-text policy drift into drift over executable actions and within-action *gauge* drift. Based on this decomposition, we introduce GAUGE, a parser-aware trust-control method that aggregates likelihood across candidate renderings and measures drift at both the action and semantic-slot levels. GAUGE preserves sequence-level gradient estimation and is compatible with group-relative policy optimization. We evaluate it in the ARLArena pipeline across ALFWorld, WebShop, Sokoban, and TIR Math. With SFT-initialized Qwen3-4B, GAUGE achieves 75.09% success on WebShop and 85.67% on Sokoban, improving over the corresponding GSPO and DAPO baselines by 2.61 and 3.27 percentage points. These results support executable-action space as a useful representation for controlling language-agent policy updates.

1 INTRODUCTION

Language agents generate text, but environments respond to executable actions. A model output is first interpreted by a parser or executor, which extracts a search query, web click, tool call, code block, or embodied action that determines the next transition and reward. This interface is shared by reasoning-and-acting systems, web agents, tool-using models, and code-generating agents (Yao et al., 2022b;a; Schick et al., 2023; Qin et al., 2024; Wang et al.). Consequently, policy optimization must distinguish between the text produced by the model and the action ultimately observed by the environment.

Existing policy-optimization methods define trust primarily in textual likelihood space. PPO-style algorithms constrain token-level likelihood ratios or KL divergence (Schulman et al., 2015; 2017), and the same machinery is widely used in RLHF and group-relative optimization for language models (Stiennon et al., 2020; Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025). Recent methods improve clipping rules, sequence-level ratios, detached importance weights, smooth gates, and loss aggregation (Yu et al., 2025; Zheng et al., 2025; Chen et al., 2025; Gao et al., 2025; Wang et al., 2026c). These advances refine how textual likelihood updates are weighted, but they do not address a more basic question: *should trust be measured over generated strings or over the actions executed by the environment?*

The distinction matters because the mapping from text to executable behavior is many-to-one. For example, `Search[waterproof backpack]` and `search query: waterproof backpack` may be parsed into the same search action. Their likelihoods can change substantially without changing the environment transition. Conversely, replacing `click[item_3]` with `click[item_8]` changes only a few tokens but may produce an entirely different transition. Under a length-normalized sequence ratio, this local change can be diluted by a long reasoning trace or output template. Textual trust control can therefore be conservative toward behaviorally irrelevant rendering changes while remaining insensitive to small but critical changes in action arguments.

We formalize this mismatch using the policy induced by the parser. The parser partitions textual outputs into equivalence classes whose members execute the same action. We refer to variation

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.

within each class as *gauge variation*, since it changes the textual representation without changing observable behavior. Pushing the textual policy through the parser produces a quotient policy over executable actions. We show that raw-text likelihood drift decomposes exactly into behavioral drift between quotient policies and gauge drift within parser-equivalence classes. The corresponding KL divergence admits the same decomposition. This analysis establishes that raw-text KL upper-bounds executable-action KL but can be arbitrarily loose, that gauge-only policy changes preserve executable trajectory distributions and expected return, and that length-normalized sequence ratios can under-resolve concentrated changes in behavior-critical slots.

Motivated by this analysis, we introduce GAUGE, a parser-aware trust-control layer for language-agent policy optimization. As illustrated in Figure 3, GAUGE aggregates sequence likelihood across a finite set of parser-equivalent renderings to estimate executable-action probabilities. It additionally evaluates counterfactual values of semantic slots, such as target objects, locations, search queries, and tool arguments, while holding the remainder of the action fixed. Action-level and slot-level drift are combined into a smooth semantic trust weight that modulates a quotient-policy score objective. Gradients continue to flow through ordinary sequence log-likelihoods, allowing GAUGE to reuse GRPO-style or other turn-level advantages. Its invariance is scoped to the candidate actions and renderings enumerated by the estimator; the raw-text reference KL remains a separate regularizer.

We evaluate GAUGE in the ARLArena pipeline (Wang et al., 2026c) on ALFWorld, WebShop, Sokoban, and TIR Math (Shridhar et al., 2020; Yao et al., 2022a; Schrader, 2018; Xue et al., 2025). With SFT-initialized Qwen3-4B, GAUGE obtains success rates of 80.54% on ALFWorld, 75.09% on WebShop, and 85.67% on Sokoban. These results exceed the corresponding GSPO baselines by 1.93 and 2.61 percentage points on ALFWorld and WebShop, and the strongest reported DAPO baseline by 3.27 points on Sokoban. Fixed-history diagnostics further identify cases in which textual drift is dominated by parser-equivalent rendering changes or fails to expose concentrated slot drift. We treat the reported results as endpoint comparisons rather than evidence of improved training stability, which would require matched multi-seed learning curves.

Contributions. Our contributions are threefold. First, we introduce a parser-induced view of language-agent policies and derive an exact decomposition of textual drift into behavioral and gauge components. Second, we develop GAUGE, which combines finite-support quotient-policy estimation with semantic-slot-aware trust control while remaining compatible with existing advantage estimators. Third, we provide targeted drift diagnostics and evaluate the method across embodied interaction, web navigation, spatial planning, and tool-integrated mathematical reasoning.

2 Behavioral Trust Misalignment

PPO- and GRPO-style methods control policy updates through textual likelihoods, typically using KL regularization or likelihood-ratio clipping (Schulman et al., 2017; Shao et al., 2024; Guo et al., 2025). For a language agent, however, the generated text is first mapped by a parser or executor to an action that determines the environment transition and reward (Yao et al., 2022b; Shridhar et al., 2020; Yao et al., 2022a; Wang et al.). As illustrated in Figure 1, this interface creates two distinct mismatches: textual drift may be large without changing the executed action, while a localized change in a critical action field may be obscured by sequence-level averaging.

2.1 Parser-Induced Policies

Let h denote an interaction history, $\tilde{a} \sim \pi_\theta(\cdot \mid h)$ a textual action, and $a = \mathrm{Parse}(\tilde{a})$ its executable action. The parser defines an equivalence class and a pushforward policy

$$\mathcal{E}(a) = \{\tilde{a} : \mathrm{Parse}(\tilde{a}) = a\}, \qquad \mu_\theta(a \mid h) = \sum_{\tilde{a} \in \mathcal{E}(a)} \pi_\theta(\tilde{a} \mid h). \tag{1}$$

Invalid outputs may be assigned to a distinguished action so that the parser is total. For $\mu_\theta(a \mid h) > 0$, define the conditional rendering distribution

$$\nu_\theta(\tilde{a} \mid a, h) = \frac{\pi_\theta(\tilde{a} \mid h)}{\mu_\theta(a \mid h)}, \qquad \tilde{a} \in \mathcal{E}(a). \tag{2}$$

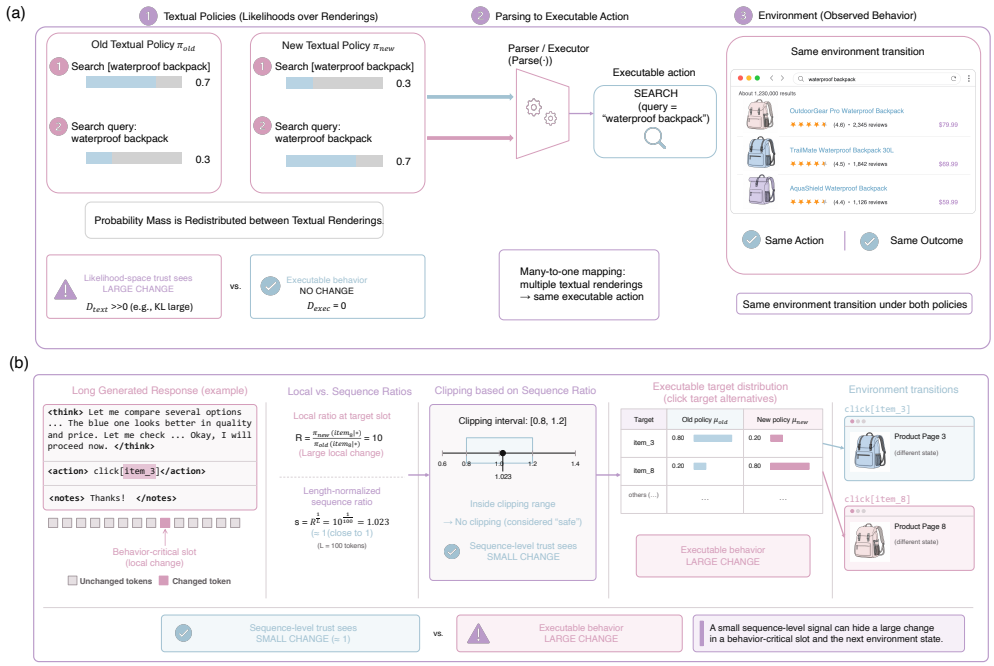


Figure 1: Two failures of likelihood-space trust. Parser-equivalent renderings can exhibit large textual drift without changing the executed action, whereas sequence averaging can dilute a large change in a behavior-critical slot. Numerical values are illustrative.

The textual policy can therefore be written as $\pi_\theta = \mu_\theta \nu_\theta$: μ_θ selects an executable action, while ν_θ selects how that action is expressed in text. When transition and reward depend on $\tilde{a}$ only through a, μ_θ is the behaviorally relevant marginal. This does not make textual quality irrelevant; formatting, validity, and reference-model proximity may still require separate regularization.

2.2 Behavioral–Gauge Decomposition

Proposition 1 (Behavioral–gauge decomposition). *Assume* $\pi_\theta(\cdot \mid h) \ll \pi_{\text{old}}(\cdot \mid h)$. *For* $a = \text{Parse}(\tilde{a})$,

$$\log \frac{\pi_\theta(\tilde{a} \mid h)}{\pi_{\text{old}}(\tilde{a} \mid h)} = \underbrace{\log \frac{\mu_\theta(a \mid h)}{\mu_{\text{old}}(a \mid h)}}_{\textit{behavioral drift}} + \underbrace{\log \frac{\nu_\theta(\tilde{a} \mid a, h)}{\nu_{\text{old}}(\tilde{a} \mid a, h)}}_{\textit{gauge drift}}. \tag{3}$$

Moreover,

$$\begin{aligned} &D_{\text{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\text{old}}(\cdot \mid h)\right) \\ &= D_{\text{KL}}\left(\mu_\theta(\cdot \mid h) \| \mu_{\text{old}}(\cdot \mid h)\right) \\ &\quad + \mathbb{E}_{a \sim \mu_\theta(\cdot|h)} D_{\text{KL}}\left(\nu_\theta(\cdot \mid a, h) \| \nu_{\text{old}}(\cdot \mid a, h)\right). \end{aligned} \tag{4}$$

The first term moves probability between executable actions and can therefore change environment behavior. The second redistributes probability among textual renderings of the same action. Substituting $\pi = \mu\nu$ gives the pointwise identity, while grouping strings by their parsed actions gives the KL chain rule. The complete proof is in Appendix C.

Corollary 2 (Raw-text KL is conservative). *For every fixed history* h,

$$D_{\text{KL}}\left(\mu_\theta(\cdot \mid h) \| \mu_{\text{old}}(\cdot \mid h)\right) \le D_{\text{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\text{old}}(\cdot \mid h)\right), \tag{5}$$

with a gap equal to the expected rendering KL in Equation 4.

Exact raw-text KL therefore controls executable-action KL, but only conservatively. A small textual KL guarantees a small executable KL, whereas an update can have large textual KL while producing little or no behavioral change. This conclusion concerns distributional KL; length-normalized sequence ratios require a separate analysis.

2.3 Two Representation-Specific Failures

Over-constraining rendering changes. Suppose an executable action admits at least two textual renderings. Probability can then be redistributed between those renderings while their total executable-action probability remains unchanged.

Proposition 3 (Unbounded gauge drift under identical behavior)**.** *If $|\mathcal{E}(a_0)| \geq 2$ for some action a_0, then for every $C > 0$ there exist textual policies satisfying*

$$\mu_\theta(\cdot \mid h) = \mu_{\text{old}}(\cdot \mid h), \qquad D_{\text{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\text{old}}(\cdot \mid h)\right) > C. \tag{6}$$

The construction preserves the total probability of every parser class while shifting mass between two renderings. As one old-policy rendering probability approaches zero, raw-text KL becomes arbitrarily large although the executable-action policy remains fixed. Appendix D gives the construction.

Under-resolving a critical slot. The opposite problem occurs when a local textual change selects a different executable action.

Proposition 4 (Sequence ratios can dilute slot drift)**.** *Consider an action containing L generated tokens with token-level likelihood ratios $r_1, \ldots, r_L$. If one behavior-critical token has ratio $R > 1$ and all remaining ratios equal one, then the length-normalized sequence ratio is*

$$s = \exp\left(\frac{1}{L}\sum_{\ell=1}^{L} \log r_\ell\right) = R^{1/L}. \tag{7}$$

For an upper clipping threshold $1 + \epsilon_{\text{clip}}$,

$$L \geq \frac{\log R}{\log(1 + \epsilon_{\text{clip}})} \quad \Longrightarrow \quad s \leq 1 + \epsilon_{\text{clip}}. \tag{8}$$

A large change to a target, object identifier, tool name, or numerical argument can therefore remain inside a sequence-level clipping range when the surrounding completion is long. This result diagnoses length-normalized sequence control and does not imply analogous dilution for every token-level clipping rule. The proof is in Appendix E.

Gauge directions and return. The distinction between behavioral and gauge drift extends from individual actions to complete trajectories.

Proposition 5 (Gauge directions are behaviorally null)**.** *Let z denote an executable history in which previous textual actions are represented by their parsed actions. Assume transition and reward depend on generated text only through $a = \text{Parse}(\tilde{a})$. If two textual policies satisfy*

$$\mu_\theta(\cdot \mid z) = \mu_{\theta'}(\cdot \mid z) \qquad \textit{for every executable history } z, \tag{9}$$

then they induce the same distribution over executable trajectories and satisfy

$$J(\pi_\theta) = J(\pi_{\theta'}). \tag{10}$$

The result follows by induction over executable histories: equal pushforward policies produce equal action distributions, and the shared environment kernel then produces equal next-history distributions. Appendix F provides the proof.

These conclusions diagnose different trust mechanisms. Corollary 2 and Proposition 3 show that exact raw-text KL may be overly conservative. Proposition 4 shows that length-normalized sequence ratios may under-resolve a localized action change. Proposition 5 establishes why within-class drift can be ignored by behavioral trust control. Together, they motivate trust over executable actions, additional resolution at critical slots, and separate textual regularization for fluency and formatting.

2.4 Diagnostic Evidence

We test the two predicted signatures on 3,840 fixed WebShop histories from 256 held-out trajectories. The policies are a Qwen3-4B SFT model and checkpoints 80, 90, and 190 from one GRPO run.

Results are averaged with trajectory normalization so that long episodes do not receive disproportionate weight. Comparisons involving step 200 are retained in Figure 2 only for provenance: the corresponding merged checkpoint used an incorrect RoPE configuration and is excluded from all conclusions. Because the audit uses matched histories, it measures policy drift under fixed contexts rather than changes in on-policy state visitation.

Parser-equivalent aliasing. For each history and policy, we sample 16 responses, deduplicate their union, score every response under both policies, and normalize on the resulting finite support. We then push these distributions through the WebShop parser to obtain projected raw-text drift D_{text}, executable-action drift D_{exec}, and within-action rendering drift D_{gauge}.

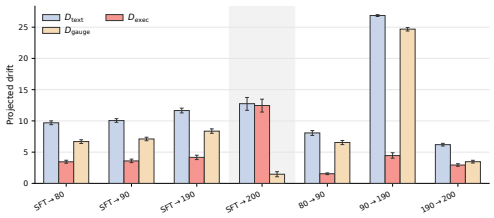


Figure 2: Projected raw-text, executable-action, and gauge drift on finite sampled supports. Step-200 comparisons use the corrupted merge and are excluded from our conclusions.

Figure 2 shows that rendering drift can dominate executable-action drift. For 90→190,

$$(D_{\text{text}}, D_{\text{gauge}}, D_{\text{exec}}) = (26.87, 24.69, 4.45),$$

while the corresponding values for 80→90 are $(8.08, 6.55, 1.55)$. Thus, substantial raw-text drift can primarily reflect redistribution among parser-equivalent strings rather than movement between actions observed by the environment.

These statistics are estimated after finite-support projection, filtering, and trajectory-level aggregation. Their relative magnitudes provide diagnostic evidence, but the three separately aggregated values should not be treated as a numerical verification of the exact identity in Equation 4. Unparseable outputs are mapped to a distinguished `BOT` action and excluded from the primary summaries.

Semantic-slot sensitivity. For each history, we enumerate all legal WebShop `click[target]` actions, averaging 12.302 candidates per history, and score a matched canonical completion for every target. Only the target string changes. The resulting KL between the old and new target distributions is D_{slot}, while R_{full} is the new-policy-weighted absolute mean-token log-ratio of the complete response.

A history is high-slot/low-sequence (HSLS) when $D_{\text{slot}} \geq 1$ and $R_{\text{full}} \leq 0.2$. The clip-based variant requires that at most 10% of legal candidates fall outside the proxy sequence-ratio interval $[0.8, 1.2]$.

Table 1: Fixed-history semantic-slot diagnostic on WebShop. "Seq. clip" uses the proxy interval $[0.8, 1.2]$; corrupted step-200 comparisons are omitted.

Pair	D_{slot}	R_{full}	Seq. clip	HSLS	HSLS clip	Top changed
SFT→80	0.227	0.169	38.30%	4.40%	2.71%	15.10%
SFT→90	0.247	0.179	40.44%	4.45%	2.42%	16.74%
SFT→190	0.563	0.185	59.83%	14.61%	1.82%	26.20%
80→90	0.031	0.022	0.007%	0.052%	0.052%	4.95%
90→190	0.394	0.048	3.28%	11.64%	11.12%	22.27%

For 90→190, the full-sequence signal is small ($R_{\text{full}} = 0.048$), and only 3.28% of legal candidates cross the proxy clipping range. Nevertheless, 11.64% of histories are HSLS, 11.12% satisfy the clip-based criterion, and the most probable target changes on 22.27% of histories. In comparison, 80→90 has a similarly small sequence signal but negligible slot drift and only 4.95% top-target changes. A small sequence signal is therefore not itself problematic; the mismatch occurs when it conceals redistribution over behavior-critical targets.

The interval $[0.8, 1.2]$ is a checkpoint-comparison proxy, not the native GSPO clipping decision. Establishing that an update was actually missed would require current-versus-rollout-old ratios, the native clipping range, and the advantage-dependent clipping decision. The corresponding native-range analysis produces few qualifying cases and does not show elevated failure, so we interpret this audit as evidence of a scale mismatch rather than a causal explanation of performance.

Appendix I defines the audit, Appendix H reports the extended results, and provides the complete sampling, canonicalization, filtering, and thresholding protocol. Overall, the diagnostics match the two theoretical predictions, but do not establish that either drift statistic caused the endpoint gains reported later.

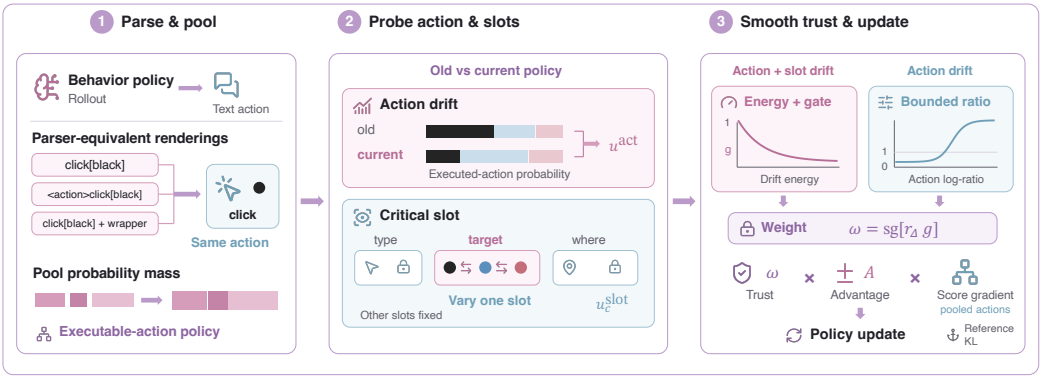


Figure 3: Overview of GAUGE. (1) Probability mass is pooled across parser-equivalent textual renderings to estimate a policy over executable actions. (2) Drift between the old and current policies is measured for the executed action and for critical semantic slots, varying one slot while holding the others fixed. (3) Action and slot drift jointly determine a smooth trust gate, while action drift alone determines a bounded importance ratio. Their stop-gradient product weights the advantage-scaled quotient-policy score gradient, with auxiliary reference-policy KL regularization. Gradients propagate through token likelihoods, while trust and update weighting reflect executable behavior.

3 GAUGE: Parser-Aware Trust Control

The previous section shows that raw textual likelihood drift contains two different components: behavioral drift over executable actions and gauge drift within parser-induced equivalence classes. This suggests that language-agent policy optimization should not use raw token ratios as the primitive object for trust control. We propose GAUGE, a parser-aware policy optimization framework that defines trust in the quotient space of executable actions and then refines it with semantic-slot sensitivity. The key idea is simple: the policy gradient is still implemented through token-level likelihoods, but the *trust signal* is computed from parsed actions and their semantic slots. Thus, GAUGE preserves differentiable language-model training while avoiding trust decisions that are sensitive only to harmless textual renderings.

3.1 Finite Quotient-Policy Estimator

At history h, let $\mathcal{A}_{\mathrm{cand}}(h)$ be a finite executable-action set and $\widehat{\mathcal{E}}(a) \subseteq \mathcal{E}(a)$ a finite set of parser-valid renderings. GAUGE pools rendering likelihoods and normalizes over candidates:

$$S_\theta(a,h) = \log \sum_{\tilde{a}\in\widehat{\mathcal{E}}(a)} \pi_\theta(\tilde{a} \mid h), \tag{11}$$

$$\widehat{\mu}_\theta(a \mid h) = \frac{\exp S_\theta(a,h)}{\sum_{a'\in\mathcal{A}_{\mathrm{cand}}(h)} \exp S_\theta(a',h)}, \qquad u^{\mathrm{act}} = \log \frac{\widehat{\mu}_\theta(a \mid h)+\epsilon_\mu}{\widehat{\mu}_{\mathrm{old}}(a \mid h)+\epsilon_\mu}. \tag{12}$$

This is a conditional estimator on enumerated support, not generally the full deployed pushforward μ_θ: omitted aliases or actions can change its normalization, and a single canonical rendering remains rendering-sensitive. Accordingly, all invariance claims below are restricted to fixed, covered candidate and rendering sets.

Support conditioning. Let $\mathcal{U}(h) = \bigcup_{a\in\mathcal{A}_{\mathrm{cand}}(h)} \widehat{\mathcal{E}}(a)$ and $Z_\theta(h) = \sum_{\tilde{a}\in\mathcal{U}(h)} \pi_\theta(\tilde{a} \mid h)$. The score of $\widehat{\mu}_\theta$ contains the normalization term $-\nabla_\theta \log Z_\theta(h)$, so it is the score of the deployed textual policy conditioned on retained support. It matches the true executable-policy score only when the candidate/rendering support is complete or when omitted mass is controlled in a way that makes this term irrelevant. In admissible-action environments we enumerate legal actions; in open-ended environments we include the sampled action, canonical renderings, parser-equivalent aliases, and contrastive perturbations. This construction trades exact marginalization for a tractable, auditable support and makes coverage a first-class experimental quantity.

3.2 SEMANTIC-SLOT SENSITIVITY

Write an executable action as $a = (\tau, v_1, \ldots, v_J)$. For slot c, GAUGE holds a^{-c} fixed and scores counterfactual values $v \in \mathcal{V}_c(h, a)$:

$$\widehat{q}_{\theta,c}(v \mid h, a^{-c}) = \frac{\exp S_\theta(a[c \leftarrow v], h)}{\sum_{v' \in \mathcal{V}_c(h,a)} \exp S_\theta(a[c \leftarrow v'], h)}, \tag{13}$$

$$u_c^{\text{slot}} = \log \frac{\widehat{q}_{\theta,c}(v_c \mid h, a^{-c}) + \epsilon_q}{\widehat{q}_{\text{old},c}(v_c \mid h, a^{-c}) + \epsilon_q}. \tag{14}$$

The gate uses the sampled value's log-ratio, whereas Section 2.4 reports full slot-distribution KL; the latter motivates localization but is not a direct validation of this sampled-value statistic. Candidate coverage therefore matters. A token-span fallback, when candidates are unavailable, is described in the appendix and is not claimed to be gauge-invariant. Format penalties remain complementary: they reduce malformed outputs but do not control probability movement among valid executable actions.

Granularity of attenuation. A sampled-value log-ratio can be zero even when probability redistributes among alternative values, whereas full slot KL detects that redistribution at additional scoring cost. GAUGE uses the sampled value because it aligns the trust weight with the sampled policy-gradient term. It then attenuates the complete executable-action update, treating the parsed action as the atomic environment transition. A field-local alternative could preserve learning in unaffected slots, but would require a token-to-slot gradient decomposition and a policy for shared wrapper tokens. We therefore present action-wide attenuation as an implementation choice whose benefit must be separated from quotient scoring and ratio bounding by matched ablations.

3.3 SMOOTH ENERGY TRUST

GAUGE converts signed action and slot log-ratios into unsigned energy with

$$\phi_\delta(u) = \delta^2 \log\cosh(u/\delta), \tag{15}$$

$$E^{\text{GAUGE}} = \underbrace{\phi_{\delta_{\text{act}}}(u^{\text{act}})}_{E^{\text{act}}} + \lambda_{\text{slot}} \underbrace{\sum_c w_c \phi_{\delta_c}(u_c^{\text{slot}})}_{E^{\text{slot}}}, \tag{16}$$

$$g^{\text{GAUGE}} = g_{\min} + (1 - g_{\min}) \exp(-E^{\text{GAUGE}}/T). \tag{17}$$

Lemma 6 (Robust drift potential). *ϕ_δ is nonnegative, even, smooth, and strictly convex, with $|\phi'_\delta(u)| \le \delta$ and*

$$\phi_\delta(u) = u^2/2 + \mathcal{O}(u^4/\delta^2) \qquad (u \to 0), \tag{18}$$

$$\phi_\delta(u) = \delta|u| - \delta^2 \log 2 + \mathcal{O}(\delta^2 e^{-2|u|/\delta}) \qquad (|u| \to \infty). \tag{19}$$

Thus ordinary changes receive a local quadratic penalty while extreme ratios have bounded marginal influence; see Appendix L.

Proposition 7 (Energy-to-trust map). *For $E \ge 0, T > 0$, $\exp(-E/T)$ uniquely minimizes $\eta E + T(\eta \log \eta - \eta + 1)$ over $\eta > 0$. Hence $g(0, T) = 1$, $g_{\min} \le g(E, T) \le 1$, and g decreases with E.*

The gate attenuates the whole sampled-action update. This is a design choice, not a consequence of the decomposition, and should be isolated from field-local alternatives by ablation.

The variational form gives a simple interpretation: ηE charges a coefficient for trusting a high-energy sample, while $T(\eta \log \eta - \eta + 1)$ penalizes moving that coefficient away from one. Thus T controls trust sensitivity and $g_{\min}$ prevents a permanent zero-gradient region. Unlike a hard clipping boundary, the gate remains continuous; unlike the directional importance factor below, it can only reduce update magnitude.

To retain directional importance correction while bounding its range, define

$$\psi_\Delta(u) = \Delta \tanh(u/\Delta), \qquad \overline{r}_\Delta(u) = \exp \psi_\Delta(u), \qquad \omega^{\text{GAUGE}} = \text{sg}[\overline{r}_\Delta(u^{\text{act}}) g^{\text{GAUGE}}]. \tag{20}$$

Lemma 8 (Bounded executable-action ratio). *ψ_Δ is odd, increasing, and 1-Lipschitz; $e^{-\Delta} < \overline{r}_\Delta(u) < e^{\Delta}$ and $\psi_\Delta(u) = u + \mathcal{O}(u^3/\Delta^2)$ near zero.*

The bounded ratio is directional, whereas $g^{\mathrm{GAUGE}} \in [g_{\min}, 1]$ is unsigned semantic attenuation; neither reproduces PPO hard clipping.

Proposition 9 (Detached trust-weighted update). *For $w_\theta = \overline{r}_\Delta(u^{\mathrm{act}}) g^{\mathrm{GAUGE}}$, $\ell_\theta = \log \widehat{\mu}_\theta(a \mid h)$, and fixed advantage A,*

$$\nabla_\theta \{\mathrm{sg}[w_\theta] A \ell_\theta\} = w_\theta A \nabla_\theta \ell_\theta, \tag{21}$$

$$\|\nabla_\theta \{\mathrm{sg}[w_\theta] A \ell_\theta\}\| \le e^{\Delta} |A| \|\nabla_\theta \ell_\theta\|. \tag{22}$$

The gate, bounded-ratio, and detached-update proofs are in Appendices M, N, and O, respectively.

Local behavior. At the behavior policy, all action and sampled-slot log-ratios vanish, and therefore

$$u^{\mathrm{act}} = u_c^{\mathrm{slot}} = 0 \quad \Longrightarrow \quad E^{\mathrm{GAUGE}} = 0, \qquad g^{\mathrm{GAUGE}} = \overline{r}_\Delta(0) = \omega^{\mathrm{GAUGE}} = 1. \tag{23}$$

For small drift, the energy is quadratic, so the gate departs from one only at second order, while $\psi_\Delta(u^{\mathrm{act}}) = u^{\mathrm{act}} + \mathcal{O}(|u^{\mathrm{act}}|^3/\Delta^2)$. Thus the *weighting coefficients* locally recover ordinary executable-action importance weighting on a fixed support. This does not by itself prove equality with the deployed executable-policy gradient: $\widehat{\mu}$ still carries the support-normalization term described above, and rollout actions originate from the deployed behavior policy. Full gradient consistency therefore additionally requires complete support or an explicit correction for omitted mass.

Scoring cost. For one history, the number of teacher-forced completions is approximately

$$N_{\mathrm{score}}(h) = \sum_{a \in \mathcal{A}_{\mathrm{cand}}(h)} |\widehat{\mathcal{E}}(a)| + \sum_c \sum_{v \in \mathcal{V}_c(h,a)} |\widehat{\mathcal{E}}(a[c \leftarrow v])|. \tag{24}$$

Caching shared prefixes reduces wall-clock overhead but does not remove the extra scored tokens relative to a sampled-token update. Candidate counts, alias counts, scored tokens, throughput, and GPU-hours are therefore required to distinguish equal-rollout from equal-compute comparisons. The current endpoint table uses the former and does not by itself establish an equal-compute advantage.

3.4 Training Objective

With any turn-level advantage $A_{i,k}$, GAUGE maximizes

$$\mathcal{J}_{\mathrm{GAUGE}}(\theta) = \frac{1}{\sum_i K_i} \sum_{i,k} \omega_{i,k}^{\mathrm{GAUGE}} A_{i,k} \log \widehat{\mu}_\theta(a_{i,k} \mid h_{i,k}) - \alpha D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}). \tag{25}$$

The quotient score and trust terms are invariant only to redistribution within the fixed enumerated rendering support. When $\alpha > 0$, the raw-text reference KL makes the full objective a hybrid and is not gauge-invariant. Moreover, because rollouts come from the deployed behavior policy while $\widehat{\mu}$ is support-conditioned, the objective is best viewed as a restricted conditional surrogate unless candidate coverage is complete. In the experiments, GAUGE uses the same group-relative advantage and testbed pipeline as the base comparison; it changes the score, ratio bounding, and semantic gating jointly. Appendix J gives the complete update algorithm.

4 Experiments

Setup. We evaluate ALFWorld (Shridhar et al., 2020), WebShop (Yao et al., 2022a), Sokoban (Schrader, 2018), and TIR Math (Xue et al., 2025) in the ARLArena testbed (Wang et al., 2026c). Comparisons include GRPO, $\mathrm{GRPO}_{\mathrm{ST}}$, SAPO, CISPO, GSPO, and DAPO. ALFWorld, WebShop, and Sokoban start from the testbed's SFT-tuned Qwen3-4B policy; TIR Math starts from Qwen3-4B-base. All methods use the same behavior-cloning protocol, structured-output format penalty, reference-policy KL with the k_3 estimator, trajectory-to-turn segmentation, and H100 hardware. GAUGE Base uses the standard group-relative advantage; task and optimization hyperparameters are in Appendix K.

For environments with admissible actions, candidate sets come from the environment; otherwise the implementation uses parser-valid sampled actions, canonical renderings, and contrastive perturbations. Candidate and alias coverage therefore varies by task, so the results evaluate the implemented finite-support estimator rather than an exact quotient over every possible query or program. The format penalty controls schema validity but does not replace executable-action trust among valid outputs.

Task	Executable object	Candidates / critical slots	Equivalence scope
ALFWorld	verb with object/location args.	admissible actions; object, source, target	parser canonicalization
WebShop	SEARCH, CLICK, BUY	page controls; query or target	parser canonicalization
Sokoban	MOVE(direction)	legal moves; direction	parser-valid wrappers
TIR Math	interpreter/tool call	sampled call + valid perturbations; args.	parser identity only

Table 2: Conservative task-level instantiation. Candidate support is finite and state-dependent.

Dimension	Method	ALFWorld		WebShop		Sokoban		TIR Math	
		Score	Success	Score	Success	Score	Success	AIME24	AIME25
Base	GRPO	3.56	62.21	67.48	51.56	–	–	–	–
Loss aggregation	GRPO_{ST}	4.41	72.61	64.57	51.29	3.03	68.73	27.55	21.63
	SAPO	0.80	25.16	73.85	52.10	−0.23	30.25	45.00	30.85
Importance sampling	CISPO	2.16	54.42	67.96	54.71	−0.47	26.02	36.53	30.87
	GSPO	5.19	78.61	85.29	72.48	5.22	82.22	51.29	37.95
Dynamic sampling	DAPO	1.95	49.58	62.43	46.17	5.16	82.40	54.66	38.97
GAUGE	Base	**5.41**	**80.54**	**88.46**	**75.09**	**6.09**	**85.67**	**56.32**	**39.03**

Table 3: Endpoint point estimates. Qwen3-4B is SFT-initialized for ALFWorld, WebShop, and Sokoban and base-initialized for TIR Math.

Executable equivalence. Table 2 makes the task interface explicit. WebShop and ALFWorld expose structured parsers, while Sokoban exposes a small legal move set. For TIR Math, GAUGE does not declare arbitrary Python programs equivalent merely because they return the same value: programs may differ in side effects, errors, or intermediate tool state. Only strings canonicalized to the same executor call are pooled; otherwise a sampled program remains a singleton rendering class and GAUGE relies on slot/call perturbations. This conservative choice avoids assuming semantic program equivalence, but also limits quotient coverage on open-ended code.

Results and scope. GAUGE improves WebShop success by 2.61 points over GSPO (75.09% vs. 72.48%) and Sokoban success by 3.27 points over DAPO (85.67% vs. 82.40%); it also has the highest point estimate in the remaining reported columns. These are endpoint comparisons, not uncertainty-aware rankings: the table lacks training-seed variation, and the 0.06-point AIME25 margin is not evidence of a reliable gain. Moreover, GAUGE changes quotient scoring, ratio bounding, and semantic gating jointly. Without matched component ablations and throughput or GPU-hour accounting, the current results do not isolate trust geometry or establish an equal-compute advantage.

5 CONCLUSION

We introduced GAUGE, a parser-aware trust-control framework motivated by an exact decomposition of textual drift into executable-action and within-action rendering components. GAUGE estimates a finite-support quotient policy, probes critical slots, and smoothly weights a differentiable policy update using behavioral drift. Real-rollout diagnostics expose the predicted representation mismatch, and four-task endpoint results are promising. The present evidence does not establish improved training stability, full-support gauge invariance, or component-level causality; those claims require multi-seed curves, coverage and compute reporting, and matched ablations. Overall, the results support evaluating trust in the space of actions an environment executes, while retaining explicit textual regularization where it is useful. More broadly, the parser-induced quotient offers a common language for comparing agent optimizers whose surface likelihoods are otherwise difficult to interpret. Extending this view to stochastic executors, partially observed side effects, and learned parsers is a natural next step toward trust control that follows operational behavior rather than syntax alone.

AI USE STATEMENT

Generative AI tools were used solely to polish the language of the manuscript. All research ideas, theoretical results, experiments, and analyses are the authors' own, and the authors take full responsibility for the content of this paper.

REFERENCES

Ömer Veysel Çağatan, Barış Akgün, Gözde Gül Şahin, and Xuandong Zhao. Clipping-free policy optimization for large language models. *arXiv preprint arXiv:2601.22801*, 2026.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Egor Denisov, Svetlana Glazyrina, Maksim Kryzhanovskiy, and Roman Ischenko. Smooth gate functions for soft advantage policy optimization. *arXiv preprint arXiv:2602.19345*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv e-prints*, pp. arXiv–2501, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Chenliang Li, Adel Elmahdy, Alex Boyd, Zhongruo Wang, Alfredo Garcia, Parminder Bhatia, Taha Kass-Hout, Cao Xiao, and Mingyi Hong. St-ppo: Stabilized off-policy proximal policy optimization for multi-turn agents training. *arXiv preprint arXiv:2511.20718*, 2025.

Yuanfan Li, Qi Zhou, Wenjing Duan, and Lu Chen. When denser credit is not enough: Evidence-calibrated policy optimization for long-horizon llm agent training. *arXiv preprint arXiv:2606.05885*, 2026.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Penghui Qi, Xiangxin Zhou, Zichen Liu, Tianyu Pang, Chao Du, Min Lin, and Wee Sun Lee. Rethinking the trust region in llm reinforcement learning. *arXiv preprint arXiv:2602.04879*, 2026.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Max-Philipp B. Schrader. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Qi Liu, and Enhong Chen. Steppo: Step-aligned policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2604.18401*, 2026a.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. *arXiv preprint arXiv:2510.14967*, 2025a.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon llm agents. *arXiv preprint arXiv:2509.09265*, 2025b.

Junzhe Wang, Zhiheng Xi, Yajie Yang, Hao Luo, Shihan Dou, Tao Gui, and Qi Zhang. Enhancing llm-based search agents via contribution weighted group relative policy optimization. *arXiv preprint arXiv:2604.14267*, 2026b.

Xiaoxuan Wang, Han Zhang, Haixin Wang, Yidan Shi, Ruoyan Li, Kaiqiao Han, Chenyi Tong, Haoran Deng, Renliang Sun, Alexander Taylor, et al. Arlarena: A unified framework for stable agentic reinforcement learning. *arXiv preprint arXiv:2602.21534*, 2026c.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Xucong Wang, Ziyu Ma, Yong Wang, Yuxiang Ji, Shidong Yang, Guanhua Chen, Pengkun Wang, and Xiangxiang Chu. Appo: Agentic procedural policy optimization. *arXiv preprint arXiv:2606.12384*, 2026d.

Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition. *arXiv preprint arXiv:2405.15821*, 2024.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, et al. Bapo: Stabilizing off-policy reinforcement learning for llms via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*, 2025.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1:2, 2025.

Jiazhen Yuan, Wei Zhao, and Zhengbiao Bai. Graph-enhanced policy optimization in llm agent training. *arXiv preprint arXiv:2510.26270*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025a.

Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, et al. Tool zero: Training tool-augmented llms via pure rl from scratch. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9135–9147, 2025b.

Yabo Zhang, Yihan Zeng, Qingyun Li, Zhen Hu, Kavin Han, and Wangmeng Zuo. Tool-r1: Sample-efficient reinforcement learning for agentic tool use. *arXiv preprint arXiv:2509.12867*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A Detailed Related Work

Policy Optimization as Likelihood-Space Trust Control. Trust-region methods are a standard way to stabilize policy optimization. TRPO (Schulman et al., 2015) constrains the KL divergence between the old and new policies, while PPO (Schulman et al., 2017) uses a clipped likelihood-ratio objective as a simpler approximation. These methods control policy updates by limiting how much the probability of sampled actions can change. This idea was later adopted in reinforcement learning from human feedback for language models. Early RLHF systems use PPO-style updates with reward models and KL regularization to keep the optimized language model close to a reference model (Stiennon et al., 2020; Ouyang et al., 2022). In this setting, policy drift is measured in token likelihood space: the model increases or decreases the probability of generated tokens according to their estimated advantages. Recent reasoning-oriented LLM training further simplifies this pipeline. DeepSeekMath introduces Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which removes the learned critic and normalizes rewards within a group of sampled responses. DeepSeek-R1 further popularizes this critic-free GRPO-style training recipe for large-scale reasoning models (Guo et al., 2025). Related REINFORCE-style methods also study critic-free policy optimization for LLM alignment (Hu, 2025). Overall, these methods define trust control in likelihood space, either through KL regularization or likelihood-ratio clipping. This is natural for ordinary text generation, but it may be insufficient for language agents, where the environment executes parsed actions rather than raw tokens.

Clipping and Importance-Ratio Design in LLM Reinforcement Learning. Likelihood-ratio clipping is a key stabilizer in PPO-style LLM reinforcement learning, but recent work shows that its design strongly affects training stability and learning efficiency. DAPO (Yu et al., 2025) improves large-scale reasoning RL with practical techniques such as dynamic sampling and asymmetric clipping. GSPO (Zheng et al., 2025) moves from token-level ratios to sequence-level ratios, aiming to reduce the high variance of token-level importance ratios. CISPO, introduced in MiniMax-M1 (Chen et al., 2025), clips importance-sampling weights rather than directly clipping token updates, which helps preserve useful gradients. Another line of work replaces or adapts hard clipping. SAPO (Gao et al., 2025) uses a smooth, temperature-controlled gate to attenuate off-policy updates instead of applying a hard clipping boundary. Follow-up work studies different smooth gate functions for more stable GRPO-style training (Denisov et al., 2026). BAPO (Xi et al., 2025) dynamically adjusts clipping bounds to balance positive and negative policy-gradient contributions in off-policy settings. Other recent methods further revisit the trust-region mechanism itself, including divergence-based alternatives to PPO clipping (Qi et al., 2026), clipping-free policy optimization (Çağatan et al., 2026), and value-augmented PPO variants for advanced reasoning tasks (Yue et al., 2025). These methods reveal that token-level hard clipping can be noisy, overly conservative, or unstable, and they improve the clipping rule, ratio granularity, or optimization objective. However, their trust control is still primarily defined in token or sequence likelihood space. In contrast, our work asks whether this is the right space for language agents, whose environments execute parsed actions rather than raw tokens.

Agentic Reinforcement Learning and the Step-Level Decision Unit. Language agents differ from standard single-turn generation because they interact with environments over multiple turns, where each turn may invoke a tool, issue an action, or change the environment state. ARLArena (Wang et al., 2026c) systematically studies this setting and identifies clipping, advantage estimation, and rollout filtering as key design dimensions for stable agentic policy optimization. Recent methods further argue that the step or turn is a more natural optimization unit than the individual token. ST-PPO (Li et al., 2025) introduces turn-level importance sampling, while StepPO (Wang et al., 2026a) formulates agent training as a step-level MDP. Related work studies turn-level credit assignment (Zeng et al., 2025a) and the discrepancy between action-level and token-level optimization (Wen et al., 2024). These works expose the mismatch between token-level optimization and multi-turn interaction structure. Our work instead focuses on the trust-control space: language-agent environments execute parsed actions, not raw text tokens.

Credit Assignment for Long-Horizon Language Agents. Long-horizon language agents often receive sparse outcome rewards, making it difficult to determine which intermediate decisions contributed to success or failure. Recent work addresses this issue by constructing finer-grained credit signals. GiGPO (Feng et al., 2026) builds step-level comparison groups from repeated anchor states. EMPG (Wang et al., 2025b) modulates policy gradients using step-wise uncertainty. GEPO (Yuan

et al., 2025) uses state-transition graphs to provide topology-aware credit signals. APPO (Wang et al., 2026d) assigns credit around fine-grained procedural branching points. ECPO (Li et al., 2026) calibrates dense step credit using canonical action evidence and variance gating. Other methods introduce per-round contribution weighting for search agents (Wang et al., 2026b), hindsight credit assignment (Tan et al., 2026), or information-gain based turn-level rewards (Wang et al., 2025a). These methods mainly improve how advantages are assigned across steps. Our work studies a different issue:where trust control should be defined when the environment executes parsed actions rather than raw text.

From Textual Outputs to Executable Actions. Language agents increasingly interact with external environments through textual commands, tool calls, web actions, or executable code. Early systems such as WebGPT (Nakano et al., 2021), ReAct (Yao et al., 2022b), and Toolformer (Schick et al., 2023) show that language models can use search, APIs, or environment feedback to solve tasks beyond static text generation. Interactive benchmarks such as ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a) further study agents whose text outputs are grounded in environment actions. Recent tool-use and code-agent work makes this connection more explicit. ToolLLM (Qin et al., 2024) studies large-scale API use, while CodeAct (Wang et al.) represents agent actions as executable Python code. Search-R1 (Jin et al., 2025), ToolRL (Qian et al., 2026), Tool-R1 (Zhang et al., 2025), and Tool-Zero (Zeng et al., 2025b) train models with reinforcement learning to issue search queries, tool calls, or code actions. These works show that language-agent outputs are not merely text: they are parsed or executed by an external system. This motivates our focus on trust control over executable actions rather than raw token sequences.

Behavioral Trust Misalignment and Parser-Induced Trust Regions. Several recent works are closely related to our view. POAD (Wen et al., 2024) studies the discrepancy between action-level and token-level optimization, while CodeAct (Wang et al.) represents language-agent actions as executable code. StepPO (Wang et al., 2026a) and ST-PPO (Li et al., 2025) move policy optimization from token-level updates toward step- or turn-level decision units. ECPO (Li et al., 2026) further uses canonical actions to calibrate step level credit. In parallel, recent trust-region methods revisit token-level ratio clipping, replacing it with divergence-based objectives (Qi et al., 2026) or clipping-free penalties (Çağatan et al., 2026). These works study action-token credit, step-level optimization, canonical action evidence, or likelihood-space trust regions. Our work focuses on a different mismatch: language-agent environments execute parsed actions, while standard trust control is defined over raw token or sequence likelihoods. We therefore study parser-induced trust regions that control behavioral drift in executable action space.

B BACKGROUND FOR GRPO-STYLE ALGORITHMS

Notation	Description
q	A task or prompt.
$\mathcal{D}$	The task distribution.
G	The number of trajectories sampled for the same task.
$\mathcal{G}(q)$	The rollout group for task q, i.e., $\mathcal{G}(q) = \{\tau_1, \ldots, \tau_G\}$.
i	The trajectory index in a rollout group, where $i = 1, \ldots, G$.
j	Another trajectory index used for group statistics, where $j = 1, \ldots, G$.
τ_i	The i-th sampled trajectory in the rollout group.
K_i	The number of agent turns in trajectory τ_i.
k	The turn index in trajectory τ_i, where $k = 1, \ldots, K_i$.
$h_{i,k}$	The interaction history before the k-th agent turn.
$\tilde{a}_{i,k}$	The raw text emitted by the language model at turn k.
$a_{i,k}$	The parsed environment action executed at turn k.
$o_{i,k}$	The observation returned by the environment after action $a_{i,k}$.
$L_{i,k}$	The number of tokens in the raw output $\tilde{a}_{i,k}$.
ℓ	The token index inside the k-th turn, where $\ell = 1, \ldots, L_{i,k}$.
$y_{i,k,\ell}$	The ℓ-th token of $\tilde{a}_{i,k}$.
$y_{i,k,<\ell}$	The tokens generated before token ℓ in the same turn.
θ	The parameters of the trainable policy.
π_θ	The trainable policy being optimized.
π_{old}	The behavior policy used to sample the current batch.
π_{ref}	The reference policy used for KL regularization.
$r_{i,k,\ell}(\theta)$	The token-level importance ratio.
$s_{i,k}(\theta)$	The length-normalized turn-level importance ratio.
R_i	The trajectory-level reward of τ_i.
R_j	The trajectory-level reward of the j-th trajectory in the same rollout group.
A_i	The trajectory-level group-relative advantage of τ_i.
$A_{i,k}$	The advantage assigned to the k-th agent turn.
ε_{adv}	A small numerical constant used in advantage normalization.
ϵ_{clip}	The PPO-style clipping range.
$\text{clip}(x, l, u)$	The clipping operator that truncates x into the interval $[l, u]$.
$\mathcal{S}^{\text{tok}}_{i,k,\ell}$	The token-level PPO-style surrogate term for token $y_{i,k,\ell}$.
$\mathcal{S}_{i,k,\ell}$	A generic PPO-style surrogate term for token $y_{i,k,\ell}$.
$\mathcal{N}$	The loss normalizer, e.g., number of tokens, turns, responses, or trajectories.
$\text{KL}(\pi_\theta \Vert \pi_{\text{ref}})$	The KL penalty that keeps the trainable policy close to the reference policy.
α	The KL regularization coefficient.
M_i	The total number of tokens in trajectory τ_i, i.e., $M_i = \sum_{k=1}^{K_i} L_{i,k}$.
$s_i^{\text{GSPO}}(\theta)$	The length-normalized sequence-level importance ratio for trajectory τ_i.
$x_{i,k}$	The environment anchor state before the k-th turn of trajectory τ_i.
$\mathcal{A}(x)$	The anchor group containing all trajectory-turn pairs that reach anchor state x.
$\bar{r}_{i,k}$	The environment reward observed at the k-th turn of trajectory τ_i.
$\widehat{R}_{i,k}$	The discounted return-to-go from turn k of trajectory τ_i.
$A^{\text{step}}_{i,k}$	The step-level relative advantage computed within an anchor group.
λ_{step}	The mixing weight for the step-level advantage in GiGPO.
γ	The discount factor used to compute return-to-go.

Table 4: Notation for GRPO-style policy optimization in multi-turn agentic RL.

B.1 OVERVIEW

We first clarify the notation used in this appendix. A language agent differs from a standard sequence generator in that the language model outputs text, while the environment executes an action parsed from that text. Therefore, we distinguish between two levels: the *agent-turn level*, where the model emits an action or response, and the *token level*, where the language model is actually optimized.

Tasks, rollouts, and rollout groups. Let $q \sim \mathcal{D}$ denote a task sampled from a task distribution $\mathcal{D}$. In an agentic RL setting, a *rollout* is one complete interaction episode produced by running the agent in the environment for task q. Starting from the initial task instruction, the agent repeatedly generates an action, the environment executes the action, and the resulting observation is appended

to the interaction history. The rollout ends when the agent reaches a terminal state, produces a final answer, succeeds or fails the task, or hits a maximum turn limit.

For each task q, GRPO-style methods usually sample multiple rollouts using the same behavior policy π_{old}. We call these rollouts a *rollout group*. Formally, a rollout group of size G is

$$\mathcal{G}(q) = \{\tau_1, \tau_2, \ldots, \tau_G\}, \tag{26}$$

where each τ_i is an independently sampled trajectory for the same task q. The trajectories in the same group differ because of stochastic decoding, environment randomness, or different action choices made by the policy. This group structure is important because GRPO computes relative advantages by comparing the rewards of trajectories within the same group.

The i-th trajectory in the rollout group is denoted by τ_i. It contains K_i agent turns:

$$\tau_i = (h_{i,1}, \tilde{a}_{i,1}, a_{i,1}, o_{i,1}, \ldots, h_{i,K_i}, \tilde{a}_{i,K_i}, a_{i,K_i}, o_{i,K_i}). \tag{27}$$

Here $h_{i,k}$ is the interaction history before the k-th turn. The language model first emits a raw text string $\tilde{a}_{i,k}$, such as `Search[waterproof backpack]`. A parser converts this string into a structured environment action $a_{i,k}$, which is executed by the environment. The environment then returns an observation $o_{i,k}$, which is appended to the history for the next turn.

Tokenization of an agent action. Although the environment executes the parsed action $a_{i,k}$, the policy optimization objective is defined over language-model tokens. We write the raw text output $\tilde{a}_{i,k}$ as a sequence of $L_{i,k}$ tokens:

$$\tilde{a}_{i,k} = (y_{i,k,1}, y_{i,k,2}, \ldots, y_{i,k,L_{i,k}}). \tag{28}$$

Thus, $y_{i,k,\ell}$ denotes the ℓ-th token generated within the k-th agent turn of trajectory τ_i.

Standard GRPO advantage. GRPO is a critic-free policy optimization method. Instead of learning a value function, it uses the rewards of multiple sampled trajectories for the same task to construct a group-relative baseline. Concretely, for a task q, suppose we sample a group of G trajectories

$$\{\tau_1, \tau_2, \ldots, \tau_G\}. \tag{29}$$

Each trajectory τ_j receives a trajectory-level reward

$$R_j = R(\tau_j), \qquad j = 1, \ldots, G. \tag{30}$$

Here R_j denotes the final outcome reward of the j-th trajectory in the group, such as a binary success indicator or a scalar task score.

For the i-th trajectory τ_i, GRPO computes its advantage by comparing its reward R_i against the rewards of the other trajectories sampled for the same task:

$$A_i^{\text{GRPO}} = \frac{R_i - \text{mean}_{j=1}^{G}(R_j)}{\text{std}_{j=1}^{G}(R_j) + \varepsilon_{\text{adv}}}. \tag{31}$$

The mean and standard deviation are computed over the G rewards $\{R_1, \ldots, R_G\}$ in the same rollout group, and ε_{adv} is a small constant used for numerical stability.

Thus, $A_i^{\text{GRPO}} > 0$ means that trajectory τ_i performs better than the group average, while $A_i^{\text{GRPO}} < 0$ means that it performs worse than the group average. In standard GRPO, this single trajectory-level advantage is broadcast to all turns and tokens in τ_i:

$$A_{i,k} = A_i^{\text{GRPO}}, \qquad k = 1, \ldots, K_i. \tag{32}$$

This design is simple and avoids a critic, but it is coarse for long-horizon agentic tasks because all actions in the same trajectory receive the same credit signal.

Token-level importance ratio. During optimization, the trajectories are fixed samples generated by π_{old}, while the current trainable policy is π_θ. For each token $y_{i,k,\ell}$, we compare its probability under the current policy with its probability under the behavior policy:

$$r_{i,k,\ell}(\theta) = \frac{\pi_\theta(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell})}{\pi_{\text{old}}(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell})}. \tag{33}$$

If $r_{i,k,\ell}(\theta) > 1$, the current policy assigns higher probability to this token than the behavior policy. If $r_{i,k,\ell}(\theta) < 1$, it assigns lower probability.

Turn-level importance ratio. Some methods control policy updates at the level of an entire agent turn rather than at the level of individual tokens. For the k-th turn of trajectory τ_i, we define a length-normalized turn-level ratio:

$$s_{i,k}(\theta) = \left(\prod_{\ell=1}^{L_{i,k}} r_{i,k,\ell}(\theta) \right)^{1/L_{i,k}} = \exp\left(\frac{1}{L_{i,k}} \sum_{\ell=1}^{L_{i,k}} \log r_{i,k,\ell}(\theta) \right). \tag{34}$$

Equivalently, $s_{i,k}(\theta)$ is the exponential of the average token log-ratio within the turn. Length normalization prevents longer actions from having systematically larger raw probability ratios.

Generic clipped surrogate. A token-level PPO-style surrogate uses an importance ratio and an advantage signal to update the probability of each generated token. For example, with a token-level ratio, the clipped surrogate can be written as

$$\mathcal{S}^{\text{tok}}_{i,k,\ell}(\theta; A_{i,k}) = \min\left(r_{i,k,\ell}(\theta) A_{i,k}, \text{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\, A_{i,k} \right). \tag{35}$$

A sequence- or turn-level variant replaces $r_{i,k,\ell}(\theta)$ with a turn-level ratio such as $s_{i,k}(\theta)$.

We write policy objectives in maximization form; in implementation, one may minimize the negative objective. Using a generic surrogate term $\mathcal{S}_{i,k,\ell}$, many GRPO-style algorithms can be summarized by the objective

$$\mathcal{L}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, \mathcal{G}(q)\sim\pi_{\text{old}}} \left[\frac{1}{\mathcal{N}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}(\theta; A_{i,k}) \right] - \alpha\, \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}}). \tag{36}$$

Here $\mathcal{N}$ specifies how the loss is normalized, for example by the number of tokens, turns, or trajectories. The KL term keeps the trainable policy close to the reference policy.

Four design axes. Following Wang et al. (2026c), GRPO-style algorithms mainly differ in four places.

1. **Importance ratio and clipping.** Methods differ in whether they use token-level ratios $r_{i,k,\ell}(\theta)$, turn- or sequence-level ratios $s_{i,k}(\theta)$, or modified clipping rules.
2. **Loss aggregation.** Methods differ in the choice of the normalizer $\mathcal{N}$, which determines whether the loss is averaged over tokens, turns, responses, or trajectories.
3. **Advantage estimation.** Methods differ in how they construct $A_{i,k}$. Standard GRPO uses a trajectory-level group-relative advantage and broadcasts it to all turns. Agentic variants may introduce turn-level, step-level, or environment-informed advantages.
4. **Data filtering.** Methods differ in whether they keep all sampled rollout groups or dynamically filter groups with weak training signal, such as groups where all trajectories succeed or all trajectories fail.

In the following sections, we take standard GRPO as the reference algorithm. For each subsequent variant, we describe which of the four axes it modifies.

B.2 GRPO

GRPO (Shao et al., 2024; Guo et al., 2025) is a critic-free PPO-style policy optimization method. For each task $q \sim \mathcal{D}$, the behavior policy π_{old} samples a rollout group

$$\mathcal{G}(q) = \{\tau_1, \tau_2, \ldots, \tau_G\}. \tag{37}$$

Each trajectory τ_i receives a trajectory-level final reward

$$R_i = R_{\text{final}}(\tau_i). \tag{38}$$

GRPO constructs a group-relative advantage by normalizing trajectory rewards within the same rollout group:

$$A_i^{\text{GRPO}} = \frac{R_i - \text{mean}_{j=1}^{G}(R_j)}{\text{std}_{j=1}^{G}(R_j) + \varepsilon_{\text{adv}}}. \tag{39}$$

Here R_j denotes the reward of the j-th trajectory in the same rollout group, and ε_{adv} is a small constant used for numerical stability. If $A_i^{\text{GRPO}} > 0$, trajectory τ_i performs better than the group average; if $A_i^{\text{GRPO}} < 0$, it performs worse than the group average.

Standard GRPO uses a single trajectory-level advantage and broadcasts it to all agent turns in the same trajectory:

$$A_{i,k}^{\text{GRPO}} = A_i^{\text{GRPO}}, \qquad k = 1, \ldots, K_i. \tag{40}$$

Since the language model is optimized at the token level, all tokens within the same agent turn share this turn-level advantage.

For the token $y_{i,k,\ell}$ in the k-th turn of trajectory τ_i, GRPO uses the token-level importance ratio $r_{i,k,\ell}(\theta)$. The corresponding clipped surrogate term is

$$\mathcal{S}_{i,k,\ell}^{\text{GRPO}}(\theta) = \min\left(r_{i,k,\ell}(\theta) A_{i,k}^{\text{GRPO}}, \text{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\, A_{i,k}^{\text{GRPO}}\right). \tag{41}$$

Using token-mean loss aggregation, the GRPO objective can be written as

$$\mathcal{L}_{\text{GRPO}}(\theta) = \frac{1}{\mathcal{N}^{\text{GRPO}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\text{GRPO}}(\theta) - \alpha\, \text{KL}(\pi_\theta \| \pi_{\text{ref}}), \tag{42}$$

where the token-mean normalizer is

$$\mathcal{N}^{\text{GRPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{43}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

This formulation fixes four structural choices of the standard GRPO baseline: (i) the importance ratio is token-level, $r_{i,k,\ell}(\theta)$; (ii) the advantage is trajectory-level and is broadcast uniformly to all turns and tokens in the same trajectory; (iii) the loss is token-mean normalized; and (iv) the rollout group $\mathcal{G}(q)$ is unfiltered. Subsequent GRPO-style methods can be described by modifying one or more of these choices.

B.3 GRPO$_{\text{ST}}$

GRPO$_{\text{ST}}$ (Wang et al., 2026c) keeps the GRPO group-relative advantage in Equation 39, the advantage broadcasting rule in Equation 40, and the token-level clipped surrogate in Equation 41. It only changes the loss aggregation rule. Instead of averaging over all tokens in the rollout group with a single token-mean normalizer, GRPO$_{\text{ST}}$ first averages the token-level surrogate terms within each trajectory and then averages across trajectories:

$$\mathcal{L}_{\text{GRPO-ST}}(\theta) = \frac{1}{G} \sum_{i=1}^{G} \frac{1}{\sum_{k=1}^{K_i} L_{i,k}} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\text{GRPO}}(\theta) - \alpha\, \text{KL}(\pi_\theta \| \pi_{\text{ref}}). \tag{44}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

Difference from GRPO. Standard GRPO uses token-mean aggregation,

$$\mathcal{N}^{\text{GRPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}, \tag{45}$$

so longer trajectories contribute more terms to the objective. In contrast, GRPO$_{\text{ST}}$ uses a sequence-mean-token-mean aggregation: each trajectory is first normalized by its own token count $\sum_{k=1}^{K_i} L_{i,k}$, and the resulting trajectory-level averages are then averaged with weight $\frac{1}{G}$. Thus, each trajectory contributes equal mass to the gradient regardless of length.

B.4 SAPO

SAPO (Gao et al., 2025) replaces the hard clipping rule in GRPO with a smooth temperature-controlled gate on the token-level importance ratio. It keeps the group-relative advantage construction but changes the surrogate term used for policy updates. For the token $y_{i,k,\ell}$ in the k-th turn of trajectory τ_i, define an advantage-dependent temperature

$$\tau_{i,k}^{\mathrm{SAPO}} = \begin{cases} \tau_{\mathrm{pos}}, & \text{if } A_{i,k}^{\mathrm{GRPO}} > 0, \\ \tau_{\mathrm{neg}}, & \text{if } A_{i,k}^{\mathrm{GRPO}} \le 0. \end{cases} \tag{46}$$

The SAPO soft-gating function is

$$f_{i,k}^{\mathrm{SAPO}}(x) = \frac{4}{\tau_{i,k}^{\mathrm{SAPO}}} \sigma\big(\tau_{i,k}^{\mathrm{SAPO}}(x-1)\big), \qquad \sigma(u) = \frac{1}{1+\exp(-u)}. \tag{47}$$

Using the token-level ratio $r_{i,k,\ell}(\theta)$, the SAPO surrogate term is

$$\mathcal{S}_{i,k,\ell}^{\mathrm{SAPO}}(\theta) = f_{i,k}^{\mathrm{SAPO}}(r_{i,k,\ell}(\theta))\, A_{i,k}^{\mathrm{GRPO}}. \tag{48}$$

With token-mean aggregation, the corresponding objective can be written as

$$\mathcal{L}_{\mathrm{SAPO}}(\theta) = \frac{1}{\mathcal{N}^{\mathrm{SAPO}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\mathrm{SAPO}}(\theta) - \alpha\, \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{49}$$

where, under token-mean aggregation,

$$\mathcal{N}^{\mathrm{SAPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{50}$$

Difference from GRPO. GRPO uses the hard clipped surrogate in Equation 41, which truncates updates once the token-level ratio leaves the clipping band $[1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}]$. SAPO replaces the hard clipping rule with a smooth sigmoid-based surrogate. The derivative of this surrogate is largest at the on-policy point $r_{i,k,\ell}(\theta) = 1$, where it matches the local slope of the unclipped ratio. As the ratio moves away from 1, the sigmoid saturates and the resulting gradient is smoothly attenuated, avoiding the abrupt zero-gradient regions introduced by hard clipping. The use of separate temperatures τ_{pos} and τ_{neg} allows SAPO to control positive- and negative-advantage updates asymmetrically.

B.5 CISPO

CISPO (Chen et al., 2025) modifies the way importance ratios are used in the policy-gradient objective. Unlike GRPO, which clips the PPO-style surrogate itself, CISPO clips the importance-sampling weight and applies a stop-gradient operation to the clipped weight. The resulting clipped weight is then used as a multiplicative coefficient on the log-probability gradient.

For token $y_{i,k,\ell}$ in the k-th turn of trajectory τ_i, define the clipped importance weight

$$\widehat{r}_{i,k,\ell}^{\mathrm{CISPO}}(\theta) = \mathrm{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\mathrm{IS,lo}}, 1+\epsilon_{\mathrm{IS,hi}}), \tag{51}$$

where $\epsilon_{\mathrm{IS,lo}}$ and $\epsilon_{\mathrm{IS,hi}}$ are the lower and upper clipping hyperparameters for the importance-sampling weight. CISPO uses this clipped weight under stop-gradient:

$$\widetilde{w}_{i,k,\ell}^{\mathrm{CISPO}}(\theta) = \mathrm{sg}\big[\widehat{r}_{i,k,\ell}^{\mathrm{CISPO}}(\theta)\big], \tag{52}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator.

Using the GRPO group-relative advantage $A_{i,k}^{\mathrm{GRPO}}$, the CISPO surrogate term is

$$\mathcal{S}_{i,k,\ell}^{\mathrm{CISPO}}(\theta) = \widetilde{w}_{i,k,\ell}^{\mathrm{CISPO}}(\theta)\, A_{i,k}^{\mathrm{GRPO}}\, \log \pi_\theta\left(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell}\right). \tag{53}$$

With token-mean aggregation, the CISPO objective can be written as

$$\mathcal{L}_{\mathrm{CISPO}}(\theta) = \frac{1}{\mathcal{N}^{\mathrm{CISPO}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\mathrm{CISPO}}(\theta), \tag{54}$$

where

$$\mathcal{N}^{\mathrm{CISPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{55}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective. If one uses the same KL-regularized master objective as in Equation 36, an additional $-\alpha \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}})$ term can be added. The original MiniMax-M1 CISPO formulation omits this KL penalty.

Difference from GRPO. GRPO clips the PPO-style surrogate $\min(rA, \mathrm{clip}(r)A)$. When a token is clipped by this surrogate, its policy-gradient contribution can be suppressed. CISPO instead clips the importance-sampling weight, stops gradients through that weight, and routes the gradient through the log-probability term $\log \pi_\theta(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell})$. Therefore, even tokens with large importance ratios can still contribute gradients, while the magnitude of their contribution is bounded by the clipped stop-gradient weight. In this notation, CISPO modifies the importance-ratio/clipping axis while keeping the GRPO-style group-relative advantage.

B.6 GSPO

GSPO (Zheng et al., 2025) moves the importance ratio and clipping operation from the token level to the sequence level. For trajectory τ_i, define its total token length as

$$M_i = \sum_{k=1}^{K_i} L_{i,k}. \tag{56}$$

The length-normalized sequence-level importance ratio is

$$s_i^{\mathrm{GSPO}}(\theta) = \left(\prod_{k=1}^{K_i} \prod_{\ell=1}^{L_{i,k}} r_{i,k,\ell}(\theta) \right)^{1/M_i} = \exp\left(\frac{1}{M_i} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \log r_{i,k,\ell}(\theta) \right). \tag{57}$$

This is the trajectory-level analogue of the turn-level ratio $s_{i,k}(\theta)$ defined in Equation 34.

GSPO keeps the GRPO group-relative advantage A_i^{GRPO} in Equation 39, but applies clipping to the sequence-level ratio instead of individual token-level ratios:

$$\mathcal{S}_i^{\mathrm{GSPO}}(\theta) = \min\left(s_i^{\mathrm{GSPO}}(\theta) A_i^{\mathrm{GRPO}}, \mathrm{clip}\big(s_i^{\mathrm{GSPO}}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\big) A_i^{\mathrm{GRPO}} \right). \tag{58}$$

The GSPO objective is

$$\mathcal{L}_{\mathrm{GSPO}}(\theta) = \frac{1}{G} \sum_{i=1}^{G} \mathcal{S}_i^{\mathrm{GSPO}}(\theta) - \alpha\, \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}). \tag{59}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

Difference from GRPO. Standard GRPO uses token-level ratios $r_{i,k,\ell}(\theta)$ and clips each token-level surrogate separately. GSPO instead uses one sequence-level ratio $s_i^{\mathrm{GSPO}}(\theta)$ for the entire trajectory. Therefore, the clipping decision is shared by all tokens in τ_i. When the sequence-level surrogate enters a clipped region, the gradient contribution of the whole trajectory can be suppressed together. This removes the high variance caused by unequal token-level importance weights, but it also makes the update coarser because all tokens in the trajectory are controlled by the same sequence-level ratio.

GSPO-token. The GSPO paper also introduces a token-level variant, GSPO-token, for settings where finer-grained advantages are desired. In our notation, a token-level GSPO-token ratio can be written as

$$s_{i,k,\ell}^{\text{GSPO-tok}}(\theta) = \mathrm{sg}\big[s_i^{\mathrm{GSPO}}(\theta)\big] \cdot \frac{\pi_\theta(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell})}{\mathrm{sg}\big[\pi_\theta(y_{i,k,\ell} \mid h_{i,k}, y_{i,k,<\ell})\big]}, \tag{60}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. This ratio is numerically equal to the sequence-level ratio $s_i^{\mathrm{GSPO}}(\theta)$, but its gradient flows through the token-level log-probability. The corresponding token-level surrogate is

$$\mathcal{S}_{i,k,\ell}^{\text{GSPO-tok}}(\theta) = \min\left(s_{i,k,\ell}^{\text{GSPO-tok}}(\theta)A_{i,k}, \mathrm{clip}\big(s_{i,k,\ell}^{\text{GSPO-tok}}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\big)\, A_{i,k}\right). \tag{61}$$

When $A_{i,k} = A_i^{\mathrm{GRPO}}$ for all turns, GSPO-token is numerically equivalent to GSPO in the objective and clipping condition. Its benefit is that it can support finer-grained turn-level or token-level advantages while keeping the sequence-level clipping decision.

B.7 GiGPO

GiGPO (Feng et al., 2026) modifies the advantage-estimation axis of GRPO. It keeps the token-level clipped surrogate structure, but augments the trajectory-level GRPO advantage with a step-level relative advantage computed from state-aligned subgroups. The method is motivated by the observation that long-horizon agent trajectories may revisit the same environment state across different rollouts, allowing local action quality to be compared within such anchor-state groups.

Let $x_{i,k}$ denote the environment anchor state before the k-th agent turn in trajectory τ_i. For an anchor state x, define its anchor group as

$$\mathcal{A}(x) = \{(i,k) : x_{i,k} = x\}. \tag{62}$$

Thus, $\mathcal{A}(x)$ contains all trajectory-turn pairs that reach the same environment anchor state x.

Let $\bar{r}_{i,m}$ denote the reward assigned to turn m of trajectory τ_i. In sparse-reward agentic tasks, this is typically zero for all non-terminal turns and equals the final outcome reward at the terminal turn: $\bar{r}_{i,m} = \mathbf{1}[m = K_i]R_{\mathrm{final}}(\tau_i)$. The discounted return-to-go from turn k is

$$\widehat{R}_{i,k} = \sum_{m=k}^{K_i} \gamma^{m-k}\bar{r}_{i,m}, \tag{63}$$

where $\gamma \in [0,1]$ is the discount factor. For each turn (i,k), GiGPO computes a step-level relative advantage by normalizing $\widehat{R}_{i,k}$ within the anchor group of the current state:

$$A_{i,k}^{\mathrm{step}} = \frac{\widehat{R}_{i,k} - \mathrm{mean}_{(j,m)\in\mathcal{A}(x_{i,k})}\left(\widehat{R}_{j,m}\right)}{\mathrm{std}_{(j,m)\in\mathcal{A}(x_{i,k})}\left(\widehat{R}_{j,m}\right) + \varepsilon_{\mathrm{adv}}}. \tag{64}$$

If an anchor group contains too few samples, one may set $A_{i,k}^{\mathrm{step}} = 0$ or skip the step-level correction for that group.

GiGPO combines this local step-level advantage with the trajectory-level GRPO advantage:

$$A_{i,k}^{\mathrm{GiGPO}} = A_i^{\mathrm{GRPO}} + \lambda_{\mathrm{step}} A_{i,k}^{\mathrm{step}}, \qquad \lambda_{\mathrm{step}} \geq 0. \tag{65}$$

The resulting token-level surrogate is obtained by replacing $A_{i,k}^{\mathrm{GRPO}}$ in Equation 41 with $A_{i,k}^{\mathrm{GiGPO}}$:

$$\mathcal{S}_{i,k,\ell}^{\mathrm{GiGPO}}(\theta) = \min\left(r_{i,k,\ell}(\theta)A_{i,k}^{\mathrm{GiGPO}}, \mathrm{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\, A_{i,k}^{\mathrm{GiGPO}}\right). \tag{66}$$

With token-mean aggregation, the GiGPO objective can be written as

$$\mathcal{L}_{\mathrm{GiGPO}}(\theta) = \frac{1}{\mathcal{N}^{\mathrm{GiGPO}}} \sum_{i=1}^{G}\sum_{k=1}^{K_i}\sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\mathrm{GiGPO}}(\theta) - \alpha\,\mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{67}$$

where

$$\mathcal{N}^{\mathrm{GiGPO}} = \sum_{i=1}^{G}\sum_{k=1}^{K_i} L_{i,k}. \tag{68}$$

Difference from GRPO. GRPO uses a single trajectory-level group-relative advantage A_i^{GRPO} and broadcasts it to all turns and tokens in τ_i. GiGPO augments this broadcast advantage with a state-aligned step-level relative term $A_{i,k}^{\mathrm{step}}$. The step-level term is computed by comparing returns-to-go among turns that reach the same anchor state. Thus, GiGPO modifies the advantage-estimation axis: it preserves the critic-free group-based structure of GRPO while breaking the invariant that all turns in the same trajectory receive exactly the same credit.

B.8 EMPG

EMPG (Wang et al., 2025b) modifies the advantage-estimation axis of GRPO by using the policy's own uncertainty to re-calibrate the learning signal. It keeps the GRPO-style clipped surrogate form, but replaces the uniformly broadcast trajectory-level advantage with a turn-level entropy-modulated advantage.

For the k-th agent turn of trajectory τ_i, define the average policy entropy over the tokens in the raw output $\tilde{a}_{i,k}$ as

$$H_{i,k} = -\frac{1}{L_{i,k}} \sum_{\ell=1}^{L_{i,k}} \sum_{v \in \mathcal{V}} \pi_\theta \left(v \mid h_{i,k}, y_{i,k,<\ell}\right) \log \pi_\theta \left(v \mid h_{i,k}, y_{i,k,<\ell}\right), \tag{69}$$

where $\mathcal{V}$ is the vocabulary. A lower value of $H_{i,k}$ indicates that the policy is more confident when generating the k-th turn.

EMPG normalizes step entropies within the current training batch. Let

$$\mathcal{B} = \{(i,k) : i = 1, \ldots, G,\ k = 1, \ldots, K_i\}$$

denote the set of trajectory-turn pairs in the batch. We define

$$H^{\text{norm}}_{i,k} = \frac{H_{i,k} - \min_{(j,m)\in\mathcal{B}} H_{j,m}}{\max_{(j,m)\in\mathcal{B}} H_{j,m} - \min_{(j,m)\in\mathcal{B}} H_{j,m} + \varepsilon_H}, \tag{70}$$

where ε_H is a small numerical constant.

The first EMPG component is a self-calibrating gradient-scaling factor:

$$g_{i,k} = \frac{\exp\left(-\kappa H^{\text{norm}}_{i,k}\right)}{\frac{1}{|\mathcal{B}|}\sum_{(j,m)\in\mathcal{B}} \exp\left(-\kappa H^{\text{norm}}_{j,m}\right)}. \tag{71}$$

This normalization makes the batch average of $g_{i,k}$ equal to one. Thus, low-entropy confident turns receive larger scaling factors, while high-entropy uncertain turns receive smaller scaling factors.

The second EMPG component is a future clarity bonus, which encourages actions that lead to lower-entropy subsequent turns:

$$c^{\text{future}}_{i,k} = \begin{cases} \exp\left(-\kappa_f H^{\text{norm}}_{i,k+1}\right), & k < K_i, \\ 0, & k = K_i. \end{cases} \tag{72}$$

Here $\kappa_f > 0$ controls how strongly the bonus favors low-entropy future turns.

Using the GRPO trajectory-level advantage A^{GRPO}_i, EMPG defines a raw turn-level modulated advantage

$$A^{\text{EMPG,raw}}_{i,k} = A^{\text{GRPO}}_i \, g_{i,k} + \zeta \, c^{\text{future}}_{i,k}, \qquad \zeta \geq 0. \tag{73}$$

Finally, EMPG normalizes the resulting modulated advantages over the batch:

$$A^{\text{EMPG}}_{i,k} = A^{\text{EMPG,raw}}_{i,k} - \frac{1}{|\mathcal{B}|} \sum_{(j,m)\in\mathcal{B}} A^{\text{EMPG,raw}}_{j,m}. \tag{74}$$

The resulting token-level surrogate is obtained by replacing $A^{\text{GRPO}}_{i,k}$ in Equation 41 with $A^{\text{EMPG}}_{i,k}$:

$$\mathcal{S}^{\text{EMPG}}_{i,k,\ell}(\theta) = \min\left(r_{i,k,\ell}(\theta) A^{\text{EMPG}}_{i,k}, \text{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\, A^{\text{EMPG}}_{i,k}\right). \tag{75}$$

With token-mean aggregation, the EMPG objective can be written as

$$\mathcal{L}_{\text{EMPG}}(\theta) = \frac{1}{\mathcal{N}^{\text{EMPG}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}^{\text{EMPG}}_{i,k,\ell}(\theta) - \alpha \, \text{KL}(\pi_\theta \| \pi_{\text{ref}}), \tag{76}$$

where

$$\mathcal{N}^{\mathrm{EMPG}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{77}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

Difference from GRPO. GRPO broadcasts the same trajectory-level advantage A_i^{GRPO} to all turns and tokens in τ_i. EMPG instead constructs a turn-level advantage using two entropy-based terms: a self-calibrating scaling factor $g_{i,k}$ that amplifies confident low-entropy turns and attenuates uncertain high-entropy turns, and a future clarity bonus $c_{i,k}^{\mathrm{future}}$ that rewards transitions toward more predictable next turns. Thus, EMPG modifies the advantage-estimation axis while leaving the clipped surrogate form and loss aggregation otherwise unchanged.

B.9 DAPO

DAPO (Yu et al., 2025) is a GRPO-style method that introduces dynamic sampling and asymmetric clipping, often referred to as *Clip-Higher*. In the original DAPO recipe, these are combined with token-level policy-gradient loss and overlong reward shaping. In this appendix, we focus on the policy optimization components that modify the GRPO-style objective: dynamic sampling and asymmetric clipping.

Dynamic sampling. For each task q, the behavior policy π_{old} samples a candidate rollout group

$$\mathcal{G}(q) = \{\tau_1, \ldots, \tau_G\}.$$

Let $R_i = R_{\mathrm{final}}(\tau_i)$ denote the final reward of trajectory τ_i. DAPO keeps only rollout groups with non-degenerate rewards. A simple acceptance condition is

$$\mathcal{G}(q) \in \Omega_{\mathrm{DAPO}} \quad \Longleftrightarrow \quad \min_{1\le i\le G} R_i < \max_{1\le i\le G} R_i. \tag{78}$$

For binary rewards, this condition is equivalent to requiring that the group contains at least one successful trajectory and at least one failed trajectory:

$$0 < \sum_{i=1}^{G} \mathbf{1}[R_i = 1] < G.$$

Groups in which all trajectories succeed or all trajectories fail provide weak or zero group-relative training signal and are discarded. The optimizer samples candidate groups until the acceptance condition is satisfied.

Asymmetric clipping. DAPO keeps the GRPO group-relative advantage construction, but replaces the symmetric clipping interval $[1\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}]$ with an asymmetric interval:

$$[1-\epsilon_{\mathrm{lo}},\, 1+\epsilon_{\mathrm{hi}}], \qquad 0 \le \epsilon_{\mathrm{lo}} < \epsilon_{\mathrm{hi}}.$$

For token $y_{i,k,\ell}$, the DAPO clipped surrogate is

$$\mathcal{S}_{i,k,\ell}^{\mathrm{DAPO}}(\theta) = \min\left(r_{i,k,\ell}(\theta) A_{i,k}^{\mathrm{DAPO}}, \mathrm{clip}(r_{i,k,\ell}(\theta), 1-\epsilon_{\mathrm{lo}}, 1+\epsilon_{\mathrm{hi}})\, A_{i,k}^{\mathrm{DAPO}}\right). \tag{79}$$

The advantage term can be instantiated in different ways. For the GRPO-based variant,

$$A_{i,k}^{\mathrm{DAPO}} = A_i^{\mathrm{GRPO}}, \tag{80}$$

whereas for a GiGPO-based variant one can use

$$A_{i,k}^{\mathrm{DAPO}} = A_{i,k}^{\mathrm{GiGPO}}. \tag{81}$$

With token-mean aggregation, the DAPO objective over an accepted rollout group can be written as

$$\mathcal{L}_{\mathrm{DAPO}}(\theta) = \frac{1}{\mathcal{N}^{\mathrm{DAPO}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\mathrm{DAPO}}(\theta) - \alpha\, \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{82}$$

where

$$\mathcal{N}^{\mathrm{DAPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{83}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

Difference from GRPO. DAPO modifies two main axes of the standard GRPO baseline. First, it changes the clipping rule from a symmetric interval to an asymmetric one, allowing larger upward movement for positive-advantage low-probability tokens through $\epsilon_{\mathrm{hi}} > \epsilon_{\mathrm{lo}}$. Second, it changes the data filtering axis by conditioning policy updates on non-degenerate rollout groups, i.e., groups that contain reward variation. Thus, groups with $\mathrm{std}_{j=1}^{G}(R_j) = 0$, such as all-success or all-failure groups under binary rewards, are filtered out before reaching the optimizer.

B.10 SAMPO

SAMPO combines three design choices identified by ARLArena (Wang et al., 2026c) as important for stable agentic RL: sequence-level clipping, fine-grained advantage estimation, and dynamic filtering. Unlike GRPO, which clips token-level ratios and broadcasts a single trajectory-level advantage to all turns, SAMPO uses a sequence-level importance ratio and a turn-level advantage.

For trajectory τ_i, let

$$M_i = \sum_{k=1}^{K_i} L_{i,k} \tag{84}$$

denote the total number of generated tokens. SAMPO uses the same length-normalized sequence-level importance ratio as GSPO:

$$s_i^{\mathrm{SAMPO}}(\theta) = \left(\prod_{k=1}^{K_i} \prod_{\ell=1}^{L_{i,k}} r_{i,k,\ell}(\theta) \right)^{1/M_i} = \exp\left(\frac{1}{M_i} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \log r_{i,k,\ell}(\theta) \right). \tag{85}$$

SAMPO augments the trajectory-level GRPO advantage with a step-level advantage:

$$A_{i,k}^{\mathrm{SAMPO}} = A_i^{\mathrm{GRPO}} + \lambda_{\mathrm{step}} A_{i,k}^{\mathrm{step}}, \qquad \lambda_{\mathrm{step}} \geq 0. \tag{86}$$

Here $A_{i,k}^{\mathrm{step}}$ is the fine-grained step-level advantage defined by the advantage-design component, such as the anchor-state relative advantage used in GiGPO.

Using the sequence-level ratio and the turn-level advantage, the SAMPO surrogate term for token $y_{i,k,\ell}$ is

$$\mathcal{S}_{i,k,\ell}^{\mathrm{SAMPO}}(\theta) = \min\left(s_i^{\mathrm{SAMPO}}(\theta) A_{i,k}^{\mathrm{SAMPO}}, \mathrm{clip}\big(s_i^{\mathrm{SAMPO}}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\big) A_{i,k}^{\mathrm{SAMPO}} \right). \tag{87}$$

The clipping decision is therefore shared across the whole trajectory through $s_i^{\mathrm{SAMPO}}(\theta)$, while the advantage can still vary across agent turns through $A_{i,k}^{\mathrm{SAMPO}}$.

SAMPO also applies dynamic filtering. In the binary-reward case, a rollout group is accepted only if it contains both successful and failed trajectories:

$$\mathcal{G}(q) \in \Omega_{\mathrm{SAMPO}} \quad \Longleftrightarrow \quad 0 < \sum_{i=1}^{G} \mathbf{1}[R_i = 1] < G. \tag{88}$$

Equivalently, the group is discarded if all trajectories have the same final outcome. For scalar rewards, the same idea can be written as

$$\min_{1 \leq i \leq G} R_i < \max_{1 \leq i \leq G} R_i.$$

With token-mean aggregation over accepted rollout groups, the SAMPO objective can be written as

$$\mathcal{L}_{\mathrm{SAMPO}}(\theta) = \mathbb{E}_{q \sim \mathcal{D},\, \mathcal{G}(q) \sim \pi_{\mathrm{old}} \,|\, \mathcal{G}(q) \in \Omega_{\mathrm{SAMPO}}} \left[\frac{1}{\mathcal{N}^{\mathrm{SAMPO}}} \sum_{i=1}^{G} \sum_{k=1}^{K_i} \sum_{\ell=1}^{L_{i,k}} \mathcal{S}_{i,k,\ell}^{\mathrm{SAMPO}}(\theta) \right] - \alpha\, \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \tag{89}$$

where

$$\mathcal{N}^{\mathrm{SAMPO}} = \sum_{i=1}^{G} \sum_{k=1}^{K_i} L_{i,k}. \tag{90}$$

We write the objective in maximization form; in implementation, one may minimize the negative objective.

Difference from GRPO. Standard GRPO uses token-level ratios $r_{i,k,\ell}(\theta)$, clips each token-level surrogate separately, broadcasts a single trajectory-level advantage A_i^{GRPO} to all turns, and trains on unfiltered rollout groups. SAMPO modifies three axes simultaneously. First, it replaces token-level clipping with sequence-level clipping through $s_i^{\mathrm{SAMPO}}(\theta)$. Second, it replaces the uniformly broadcast advantage with a turn-level advantage. Third, it applies dynamic filtering so that rollout groups with degenerate final outcomes are discarded before policy updates. Thus, SAMPO keeps the critic-free group-relative baseline of GRPO while combining sequence-aware update control, finer-grained credit assignment, and more informative rollout groups.

C Proof of Proposition 1

Proof. Fix a history h. Let $\widetilde{\mathcal{A}}$ denote the raw textual action space and let $\mathcal{A}$ denote the executable-action space. The parser $\mathrm{Parse} : \widetilde{\mathcal{A}} \to \mathcal{A}$ partitions the textual action space into parser-induced equivalence classes:

$$\mathcal{E}(a) = \{\tilde{a} \in \widetilde{\mathcal{A}} : \mathrm{Parse}(\tilde{a}) = a\}. \tag{91}$$

For each executable action $a \in \mathcal{A}$, the induced executable-action policy is

$$\mu_\theta(a \mid h) = \sum_{\tilde{a} \in \mathcal{E}(a)} \pi_\theta(\tilde{a} \mid h). \tag{92}$$

Whenever $\mu_\theta(a \mid h) > 0$, define the conditional rendering distribution inside the equivalence class $\mathcal{E}(a)$ as

$$\nu_\theta(\tilde{a} \mid a, h) = \frac{\pi_\theta(\tilde{a} \mid h)}{\mu_\theta(a \mid h)}, \qquad \tilde{a} \in \mathcal{E}(a). \tag{93}$$

This definition gives the factorization

$$\pi_\theta(\tilde{a} \mid h) = \mu_\theta(a \mid h)\, \nu_\theta(\tilde{a} \mid a, h), \qquad a = \mathrm{Parse}(\tilde{a}). \tag{94}$$

The old policy admits the analogous factorization:

$$\pi_{\mathrm{old}}(\tilde{a} \mid h) = \mu_{\mathrm{old}}(a \mid h)\, \nu_{\mathrm{old}}(\tilde{a} \mid a, h). \tag{95}$$

We assume $\pi_\theta(\cdot \mid h)$ is absolutely continuous with respect to $\pi_{\mathrm{old}}(\cdot \mid h)$, so that the likelihood ratios and KL divergences below are well-defined on the support of π_θ. Dividing Equation 94 by Equation 95 yields

$$\frac{\pi_\theta(\tilde{a} \mid h)}{\pi_{\mathrm{old}}(\tilde{a} \mid h)} = \frac{\mu_\theta(a \mid h)}{\mu_{\mathrm{old}}(a \mid h)} \cdot \frac{\nu_\theta(\tilde{a} \mid a, h)}{\nu_{\mathrm{old}}(\tilde{a} \mid a, h)}. \tag{96}$$

Taking logarithms gives the pointwise decomposition:

$$\log \frac{\pi_\theta(\tilde{a} \mid h)}{\pi_{\mathrm{old}}(\tilde{a} \mid h)} = \log \frac{\mu_\theta(a \mid h)}{\mu_{\mathrm{old}}(a \mid h)} + \log \frac{\nu_\theta(\tilde{a} \mid a, h)}{\nu_{\mathrm{old}}(\tilde{a} \mid a, h)}. \tag{97}$$

This proves Equation 3.

We now prove the KL decomposition. Starting from the definition of KL divergence and grouping raw textual actions by their parser-induced equivalence classes, we have

$$\begin{aligned} &D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) \\ &= \sum_{a \in \mathcal{A}} \sum_{\tilde{a} \in \mathcal{E}(a)} \pi_\theta(\tilde{a} \mid h) \log \frac{\pi_\theta(\tilde{a} \mid h)}{\pi_{\mathrm{old}}(\tilde{a} \mid h)}. \end{aligned} \tag{98}$$

Substituting the factorizations from Equation 94 and Equation 95 gives

$$\begin{aligned} &D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) \\ &= \sum_{a \in \mathcal{A}} \sum_{\tilde{a} \in \mathcal{E}(a)} \mu_\theta(a \mid h) \nu_\theta(\tilde{a} \mid a, h) \log \frac{\mu_\theta(a \mid h) \nu_\theta(\tilde{a} \mid a, h)}{\mu_{\mathrm{old}}(a \mid h) \nu_{\mathrm{old}}(\tilde{a} \mid a, h)}. \end{aligned} \tag{99}$$

Using

$$\log \frac{xy}{uv} = \log \frac{x}{u} + \log \frac{y}{v}, \tag{100}$$

we can split Equation 99 into two terms:

$$\begin{aligned} &D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) \\ &= \sum_{a \in \mathcal{A}} \sum_{\tilde{a} \in \mathcal{E}(a)} \mu_\theta(a \mid h) \nu_\theta(\tilde{a} \mid a, h) \log \frac{\mu_\theta(a \mid h)}{\mu_{\mathrm{old}}(a \mid h)} \\ &\quad + \sum_{a \in \mathcal{A}} \sum_{\tilde{a} \in \mathcal{E}(a)} \mu_\theta(a \mid h) \nu_\theta(\tilde{a} \mid a, h) \log \frac{\nu_\theta(\tilde{a} \mid a, h)}{\nu_{\mathrm{old}}(\tilde{a} \mid a, h)}. \end{aligned} \tag{101}$$

We simplify the first term. Since $\log \frac{\mu_\theta(a|h)}{\mu_{\text{old}}(a|h)}$ depends only on a and not on $\tilde{a}$, it can be taken outside the inner sum:

$$\begin{aligned}&\sum_{a\in\mathcal{A}}\sum_{\tilde{a}\in\mathcal{E}(a)}\mu_\theta(a\mid h)\nu_\theta(\tilde{a}\mid a,h)\log\frac{\mu_\theta(a\mid h)}{\mu_{\text{old}}(a\mid h)}\\&=\sum_{a\in\mathcal{A}}\mu_\theta(a\mid h)\log\frac{\mu_\theta(a\mid h)}{\mu_{\text{old}}(a\mid h)}\sum_{\tilde{a}\in\mathcal{E}(a)}\nu_\theta(\tilde{a}\mid a,h).\end{aligned}\tag{102}$$

Because $\nu_\theta(\cdot\mid a,h)$ is a conditional distribution over $\mathcal{E}(a)$, we have

$$\sum_{\tilde{a}\in\mathcal{E}(a)}\nu_\theta(\tilde{a}\mid a,h)=1.\tag{103}$$

Actions with $\mu_\theta(a\mid h)=0$ contribute zero. Therefore, the first term becomes

$$\sum_{a\in\mathcal{A}}\mu_\theta(a\mid h)\log\frac{\mu_\theta(a\mid h)}{\mu_{\text{old}}(a\mid h)}=D_{\text{KL}}\left(\mu_\theta(\cdot\mid h)\|\mu_{\text{old}}(\cdot\mid h)\right).\tag{104}$$

We now simplify the second term. Factoring out $\mu_\theta(a\mid h)$ from the inner sum gives

$$\begin{aligned}&\sum_{a\in\mathcal{A}}\sum_{\tilde{a}\in\mathcal{E}(a)}\mu_\theta(a\mid h)\nu_\theta(\tilde{a}\mid a,h)\log\frac{\nu_\theta(\tilde{a}\mid a,h)}{\nu_{\text{old}}(\tilde{a}\mid a,h)}\\&=\sum_{a\in\mathcal{A}}\mu_\theta(a\mid h)\sum_{\tilde{a}\in\mathcal{E}(a)}\nu_\theta(\tilde{a}\mid a,h)\log\frac{\nu_\theta(\tilde{a}\mid a,h)}{\nu_{\text{old}}(\tilde{a}\mid a,h)}.\end{aligned}\tag{105}$$

The inner sum is the KL divergence between the two conditional rendering distributions:

$$\sum_{\tilde{a}\in\mathcal{E}(a)}\nu_\theta(\tilde{a}\mid a,h)\log\frac{\nu_\theta(\tilde{a}\mid a,h)}{\nu_{\text{old}}(\tilde{a}\mid a,h)}=D_{\text{KL}}\left(\nu_\theta(\cdot\mid a,h)\|\nu_{\text{old}}(\cdot\mid a,h)\right).\tag{106}$$

Thus the second term is

$$\mathbb{E}_{a\sim\mu_\theta(\cdot|h)}D_{\text{KL}}\left(\nu_\theta(\cdot\mid a,h)\|\nu_{\text{old}}(\cdot\mid a,h)\right).\tag{107}$$

Combining Equation 104 and Equation 107, we obtain

$$\begin{aligned}&D_{\text{KL}}\left(\pi_\theta(\cdot\mid h)\|\pi_{\text{old}}(\cdot\mid h)\right)\\&=D_{\text{KL}}\left(\mu_\theta(\cdot\mid h)\|\mu_{\text{old}}(\cdot\mid h)\right)+\mathbb{E}_{a\sim\mu_\theta(\cdot|h)}D_{\text{KL}}\left(\nu_\theta(\cdot\mid a,h)\|\nu_{\text{old}}(\cdot\mid a,h)\right).\end{aligned}\tag{108}$$

This proves Equation 4. □

D PROOF OF PROPOSITION 3

Proof. Fix a history h. Let $\widetilde{\mathcal{A}}$ be the raw textual action space, let $\mathcal{A}$ be the executable-action space, and let $\mathrm{Parse} : \widetilde{\mathcal{A}} \to \mathcal{A}$ be the parser. For each executable action $a \in \mathcal{A}$, define its parser-induced equivalence class

$$\mathcal{E}(a) = \{\tilde{a} \in \widetilde{\mathcal{A}} : \mathrm{Parse}(\tilde{a}) = a\}. \tag{109}$$

Assume there exists an executable action $a_0 \in \mathcal{A}$ such that $\mathcal{E}(a_0)$ contains at least two distinct textual renderings. Choose two of them and denote them by

$$\tilde{a}_1, \tilde{a}_2 \in \mathcal{E}(a_0), \qquad \tilde{a}_1 \neq \tilde{a}_2. \tag{110}$$

Let $\mu(\cdot \mid h)$ be any executable-action policy satisfying

$$p := \mu(a_0 \mid h) > 0. \tag{111}$$

We will construct two textual policies $\pi_\theta(\cdot \mid h)$ and $\pi_{\mathrm{old}}(\cdot \mid h)$ that have the same pushforward executable-action policy $\mu(\cdot \mid h)$, but whose raw-text KL divergence is arbitrarily large.

For each executable action a, choose conditional rendering distributions $\nu_\theta(\cdot \mid a, h)$ and $\nu_{\mathrm{old}}(\cdot \mid a, h)$ supported on $\mathcal{E}(a)$. For all $a \neq a_0$, set

$$\nu_\theta(\cdot \mid a, h) = \nu_{\mathrm{old}}(\cdot \mid a, h). \tag{112}$$

For the special action a_0, let $\epsilon \in (0, 1/2)$ and define

$$\nu_{\mathrm{old}}(\tilde{a}_1 \mid a_0, h) = 1 - \epsilon, \qquad \nu_{\mathrm{old}}(\tilde{a}_2 \mid a_0, h) = \epsilon, \tag{113}$$

and

$$\nu_\theta(\tilde{a}_1 \mid a_0, h) = \epsilon, \qquad \nu_\theta(\tilde{a}_2 \mid a_0, h) = 1 - \epsilon. \tag{114}$$

All other renderings in $\mathcal{E}(a_0)$, if any, are assigned probability zero under both conditional rendering distributions. This preserves absolute continuity on the support of ν_θ for every fixed $\epsilon > 0$.

Now define the textual policies by

$$\pi_{\mathrm{old}}(\tilde{a} \mid h) = \mu(a \mid h)\, \nu_{\mathrm{old}}(\tilde{a} \mid a, h), \qquad a = \mathrm{Parse}(\tilde{a}), \tag{115}$$

and

$$\pi_\theta(\tilde{a} \mid h) = \mu(a \mid h)\, \nu_\theta(\tilde{a} \mid a, h), \qquad a = \mathrm{Parse}(\tilde{a}). \tag{116}$$

Both are valid probability distributions over $\widetilde{\mathcal{A}}$. Their pushforwards through the parser are identical. Indeed, for any $a \in \mathcal{A}$,

$$\begin{aligned} \sum_{\tilde{a} \in \mathcal{E}(a)} \pi_\theta(\tilde{a} \mid h) &= \sum_{\tilde{a} \in \mathcal{E}(a)} \mu(a \mid h) \nu_\theta(\tilde{a} \mid a, h) \\ &= \mu(a \mid h) \sum_{\tilde{a} \in \mathcal{E}(a)} \nu_\theta(\tilde{a} \mid a, h) \\ &= \mu(a \mid h), \end{aligned} \tag{117}$$

and the same calculation gives

$$\sum_{\tilde{a} \in \mathcal{E}(a)} \pi_{\mathrm{old}}(\tilde{a} \mid h) = \mu(a \mid h). \tag{118}$$

Therefore,

$$\mu_\theta(\cdot \mid h) = \mu_{\mathrm{old}}(\cdot \mid h) = \mu(\cdot \mid h). \tag{119}$$

It remains to show that the raw-text KL divergence can be made arbitrarily large. By the KL chain-rule decomposition, or directly from the construction, we have

$$\begin{aligned} D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) = {} & D_{\mathrm{KL}}\left(\mu(\cdot \mid h) \| \mu(\cdot \mid h)\right) \\ & + \mathbb{E}_{a \sim \mu(\cdot \mid h)} D_{\mathrm{KL}}\left(\nu_\theta(\cdot \mid a, h) \| \nu_{\mathrm{old}}(\cdot \mid a, h)\right). \end{aligned} \tag{120}$$

The first term is zero. The conditional rendering distributions are identical for all $a \neq a_0$, so only a_0 contributes to the second term. Hence,

$$\begin{aligned} D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) &= p\, D_{\mathrm{KL}}\left(\nu_\theta(\cdot \mid a_0, h) \| \nu_{\mathrm{old}}(\cdot \mid a_0, h)\right) \\ &= p\left[\epsilon \log \frac{\epsilon}{1-\epsilon} + (1-\epsilon) \log \frac{1-\epsilon}{\epsilon}\right]. \end{aligned} \tag{121}$$

Since $p > 0$ and

$$\epsilon \log \frac{\epsilon}{1-\epsilon} + (1-\epsilon) \log \frac{1-\epsilon}{\epsilon} \longrightarrow +\infty \qquad \text{as } \epsilon \to 0, \tag{122}$$

for any constant $C > 0$ we can choose ϵ sufficiently small such that

$$D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h) \| \pi_{\mathrm{old}}(\cdot \mid h)\right) > C. \tag{123}$$

Thus the two textual policies induce exactly the same executable-action policy but can have arbitrarily large raw-text KL divergence. This proves the proposition. □

E Proof of Proposition 4

Proof. Let r_ℓ denote the token-level likelihood ratio of the ℓ-th token in the action. By assumption, there exists one semantically critical slot token $c \in \{1, \dots, L\}$ such that

$$r_c = R > 1, \qquad r_\ell = 1 \quad \text{for all } \ell \neq c. \tag{124}$$

The length-normalized sequence ratio is the geometric mean of the token-level ratios:

$$s = \left(\prod_{\ell=1}^{L} r_\ell \right)^{1/L} = \exp\left(\frac{1}{L} \sum_{\ell=1}^{L} \log r_\ell \right). \tag{125}$$

Substituting Equation 124 into Equation 125 gives

$$s = \left(R \cdot \prod_{\ell \neq c} 1 \right)^{1/L} = R^{1/L}. \tag{126}$$

Since $R > 1$ and $\epsilon_{\text{clip}} > 0$, both $\log R$ and $\log(1 + \epsilon_{\text{clip}})$ are positive. By monotonicity of the logarithm,

$$s \leq 1 + \epsilon_{\text{clip}} \tag{127}$$

is equivalent to

$$\log s \leq \log(1 + \epsilon_{\text{clip}}). \tag{128}$$

Using Equation 126, we have

$$\log s = \frac{1}{L} \log R. \tag{129}$$

Therefore,

$$s \leq 1 + \epsilon_{\text{clip}} \quad \Longleftrightarrow \quad \frac{\log R}{L} \leq \log(1 + \epsilon_{\text{clip}}). \tag{130}$$

Because $\log(1 + \epsilon_{\text{clip}}) > 0$, Equation 130 is equivalent to

$$L \geq \frac{\log R}{\log(1 + \epsilon_{\text{clip}})}. \tag{131}$$

Thus, whenever the stated condition holds, the length-normalized sequence ratio remains within the upper clipping threshold, even though the critical slot token itself has likelihood ratio R. This proves the proposition. □

F PROOF OF PROPOSITION 5

Proof. Let z_t denote the executable history before step t, i.e., the history obtained by replacing each previous raw textual action $\tilde{a}_s$ with its parsed executable action $a_s = \mathrm{Parse}(\tilde{a}_s)$. We assume that the environment only observes executable actions: after action a_t, it generates an observation according to a transition kernel

$$o_t \sim P_{\mathrm{env}}(\cdot \mid z_t, a_t), \tag{132}$$

and the next executable history is obtained by a deterministic update

$$z_{t+1} = \mathrm{Update}(z_t, a_t, o_t). \tag{133}$$

The reward is also assumed to depend only on the executable trajectory.

By assumption, the two textual policies induce the same executable-action policy at every executable history:

$$\mu_\theta(\cdot \mid z) = \mu_{\theta'}(\cdot \mid z), \qquad \text{for all executable histories } z. \tag{134}$$

We show by induction that the two policies induce the same distribution over executable histories at every step. The base case is immediate, since both policies start from the same initial executable history distribution. Suppose that the distributions over executable histories before step t are identical under the two policies. Then, for any possible next executable history z_{t+1}, the distribution induced by π_θ is

$$\mathbb{P}_\theta(z_{t+1}) = \sum_{z_t} \mathbb{P}_\theta(z_t) \sum_{a_t} \mu_\theta(a_t \mid z_t) \sum_{o_t} P_{\mathrm{env}}(o_t \mid z_t, a_t) \mathbf{1}\left[z_{t+1} = \mathrm{Update}(z_t, a_t, o_t)\right]. \tag{135}$$

By the induction hypothesis,

$$\mathbb{P}_\theta(z_t) = \mathbb{P}_{\theta'}(z_t), \tag{136}$$

and by Equation 134,

$$\mu_\theta(a_t \mid z_t) = \mu_{\theta'}(a_t \mid z_t). \tag{137}$$

The environment transition kernel and the update rule are the same for bot policies. Therefore, the right-hand side of Equation 135 is identical to the corresponding quantity under $\pi_{\theta'}$, and hence

$$\mathbb{P}_\theta(z_{t+1}) = \mathbb{P}_{\theta'}(z_{t+1}). \tag{138}$$

This completes the induction.

Thus, the two policies induce the same distribution over executable trajectories,

$$(z_1, a_1, o_1, \ldots, z_T, a_T, o_T). \tag{139}$$

Since the reward depends only on the executable trajectory, the expected return is identical:

$$J(\pi_\theta) = J(\pi_{\theta'}). \tag{140}$$

This proves the proposition. □

G REAL-ROLLOUT PARSER-EQUIVALENT ALIASING AUDIT

This section describes our real-rollout parser-equivalent aliasing audit. The goal is to test whether behavioral trust misalignment appears in naturally collected language-agent trajectories, rather than in manually constructed alias examples. Unlike controlled alias-shift settings, all histories, model outputs, and parsed actions used in this diagnostic are collected from real agent-environment rollouts under trained policy checkpoints.

G.1 GOAL

The audit tests the first failure mode of likelihood-space trust control: raw textual likelihood drift can be large even when executable behavior changes little. We use the same decomposition as in Theorem 1, but estimate all quantities on the finite empirical support of raw actions observed in real rollouts.

For a fixed real history h, let

$$\mathcal{U}(h) = \mathcal{R}_\theta(h) \cup \mathcal{R}_{\text{old}}(h) \tag{141}$$

denote the observed support of raw textual actions sampled from the current and old policies. We treat $\mathcal{U}(h)$ as a deduplicated support of unique raw textual actions.

For each policy, we define the finite-support projection of the raw textual policy:

$$\widehat{\pi}_\theta^{\mathcal{U}}(\tilde{a} \mid h) = \frac{\exp\left(\log \pi_\theta(\tilde{a} \mid h)\right)}{\sum_{\tilde{a}' \in \mathcal{U}(h)} \exp\left(\log \pi_\theta(\tilde{a}' \mid h)\right)}, \qquad \tilde{a} \in \mathcal{U}(h), \tag{142}$$

and define $\widehat{\pi}_{\text{old}}^{\mathcal{U}}$ analogously.

The parser induces a projected executable-action policy:

$$\widehat{\mu}_\theta^{\mathcal{U}}(a \mid h) = \sum_{\tilde{a} \in \mathcal{U}(h): \text{Parse}(\tilde{a}) = a} \widehat{\pi}_\theta^{\mathcal{U}}(\tilde{a} \mid h), \tag{143}$$

with $\widehat{\mu}_{\text{old}}^{\mathcal{U}}$ defined analogously. The observed parser-induced equivalence class of a is

$$\widehat{\mathcal{E}}_{\mathcal{U}}(a, h) = \{\tilde{a} \in \mathcal{U}(h) : \text{Parse}(\tilde{a}) = a\}. \tag{144}$$

Within this class, we define the projected conditional rendering distribution:

$$\widehat{\nu}_\theta^{\mathcal{U}}(\tilde{a} \mid a, h) = \frac{\widehat{\pi}_\theta^{\mathcal{U}}(\tilde{a} \mid h)}{\widehat{\mu}_\theta^{\mathcal{U}}(a \mid h)}, \qquad \tilde{a} \in \widehat{\mathcal{E}}_{\mathcal{U}}(a, h), \tag{145}$$

with $\widehat{\nu}_{\text{old}}^{\mathcal{U}}$ defined analogously.

We then define the three projected drift measures:

$$\widehat{D}_{\text{text}}^{\mathcal{U}}(h) = D_{\text{KL}}\left(\widehat{\pi}_\theta^{\mathcal{U}}(\cdot \mid h) \| \widehat{\pi}_{\text{old}}^{\mathcal{U}}(\cdot \mid h)\right), \tag{146}$$

$$\widehat{D}_{\text{exec}}^{\mathcal{U}}(h) = D_{\text{KL}}\left(\widehat{\mu}_\theta^{\mathcal{U}}(\cdot \mid h) \| \widehat{\mu}_{\text{old}}^{\mathcal{U}}(\cdot \mid h)\right), \tag{147}$$

and

$$\widehat{D}_{\text{gauge}}^{\mathcal{U}}(h) = \sum_a \widehat{\mu}_\theta^{\mathcal{U}}(a \mid h) D_{\text{KL}}\left(\widehat{\nu}_\theta^{\mathcal{U}}(\cdot \mid a, h) \| \widehat{\nu}_{\text{old}}^{\mathcal{U}}(\cdot \mid a, h)\right). \text{why U?} \tag{148}$$

By the same KL chain rule as in Theorem 1, these finite-support quantities satisfy

$$\widehat{D}_{\text{text}}^{\mathcal{U}}(h) = \widehat{D}_{\text{exec}}^{\mathcal{U}}(h) + \widehat{D}_{\text{gauge}}^{\mathcal{U}}(h), \tag{149}$$

up to smoothing and numerical precision.

The audit asks whether real rollout histories exhibit the signature

$$\widehat{D}_{\text{text}}^{\mathcal{U}}(h) \gg \widehat{D}_{\text{exec}}^{\mathcal{U}}(h), \qquad \widehat{D}_{\text{gauge}}^{\mathcal{U}}(h) \text{ is large.} \tag{150}$$

This is precisely the regime where the policy changes how it renders actions in text, while the parsed executable-action distribution changes little.

Importantly, this diagnostic does not rely on hand-written alias shifts. The renderings used in the audit are actual model-generated actions observed in real rollout histories. All divergences above are finite-support projected drift measures on $\mathcal{U}(h)$, rather than exact divergences over the full open-ended textual action space.

G.2 Rollout Collection

We collect rollout histories from real agentic tasks, including WebShop, Search-style agents, and ALFWorld. For each environment, we save trajectories from multiple stages of training under the same standardized testbed used in our main experiments. We consider the following policy pairs:

$$(\pi_{\mathrm{old}}, \pi_\theta) \in \{(\pi_{\mathrm{SFT}}, \pi_{\mathrm{early}}), (\pi_{\mathrm{early}}, \pi_{\mathrm{mid}}), (\pi_{\mathrm{mid}}, \pi_{\mathrm{late}}), (\pi_{\mathrm{behavior}}, \pi_{\mathrm{current}})\}. \tag{151}$$

For each pair, we replay the same real histories h and sample multiple raw textual actions from both policies using the same decoding configuration:

$$\mathcal{R}_\theta(h) = \{\tilde{a}_\theta^{(1)}, \ldots, \tilde{a}_\theta^{(N)}\}, \qquad \mathcal{R}_{\mathrm{old}}(h) = \{\tilde{a}_{\mathrm{old}}^{(1)}, \ldots, \tilde{a}_{\mathrm{old}}^{(N)}\}. \tag{152}$$

These sets contain actual model-generated actions, not synthetic aliases. This fixed-history replay protocol isolates conditional action-distribution drift from changes in state visitation. It does not assume that the two policies visit identical trajectory distributions on-policy; instead, it measures how their conditional rendering and executable-action distributions differ at the same real environment histories.

G.3 Canonicalization and Natural Equivalence Classes

Throughout the audit, Parse denotes the environment-specific parser-canonicalizer that maps a raw textual action to the canonical executable action observed by the environment. For WebShop, it maps different surface forms of a search, click, or buy command to structured executable actions such as Search(query) or Click(item). For ALFWorld, it maps text commands to structured action types and arguments, such as verb, object, source, and target location. Unparseable outputs are mapped to a distinguished invalid executable action $a = \perp$; we report sensitivity analyses both including and excluding invalid actions.

For each executable action a, its naturally observed rendering set at history h is the finite-support equivalence class

$$\widehat{\mathcal{E}}_{\mathcal{U}}(a, h) = \{\tilde{a} \in \mathcal{U}(h) : \mathrm{Parse}(\tilde{a}) = a\}. \tag{153}$$

We call a a naturally aliased executable action at history h if

$$\left|\widehat{\mathcal{E}}_{\mathcal{U}}(a, h)\right| \geq 2. \tag{154}$$

This criterion ensures that the audit focuses on parser-equivalent renderings that actually appear in model rollouts. Equivalently, $\widehat{\mathcal{E}}_{\mathcal{U}}(a, h) = \mathcal{E}(a) \cap \mathcal{U}(h)$ is the finite observed analogue of the parser-induced equivalence class $\mathcal{E}(a)$ used in the main text.

G.4 Estimating Raw-Text Drift

For every raw action $\tilde{a} \in \mathcal{U}(h)$, we compute sequence log-probabilities under both policies:

$$\ell_\theta(\tilde{a}, h) = \log \pi_\theta(\tilde{a} \mid h), \qquad \ell_{\mathrm{old}}(\tilde{a}, h) = \log \pi_{\mathrm{old}}(\tilde{a} \mid h). \tag{155}$$

These scores define the finite-support projected textual policies $\widehat{\pi}_\theta^{\mathcal{U}}$ and $\widehat{\pi}_{\mathrm{old}}^{\mathcal{U}}$ in Equation 142. The projected raw-text drift is $\widehat{D}_{\mathrm{text}}^{\mathcal{U}}(h)$ in Equation 146. We apply a small smoothing constant to empirical probabilities when computing KL divergences, and report Jensen–Shannon divergence as a support-robust check.

G.5 Estimating Executable-Action Drift

We push the projected raw-text policy through the parser to obtain the finite-support executable-action policy $\widehat{\mu}_\theta^{\mathcal{U}}(a \mid h)$ in Equation 143, with $\widehat{\mu}_{\mathrm{old}}^{\mathcal{U}}$ defined analogously. The projected executable-action drift is $\widehat{D}_{\mathrm{exec}}^{\mathcal{U}}(h)$ in Equation 147. This quantity measures how much the policy changes over actions actually observed by the environment, after parser-equivalent textual renderings are identified.

G.6 Estimating Gauge Drift

Within each observed parser-induced equivalence class $\widehat{\mathcal{E}}_{\mathcal{U}}(a,h)$, we compute the finite-support conditional rendering distributions $\widehat{\nu}^{\mathcal{U}}_{\theta}(\cdot \mid a,h)$ and $\widehat{\nu}^{\mathcal{U}}_{\mathrm{old}}(\cdot \mid a,h)$ as in Equation 145. The projected gauge drift is $\widehat{D}^{\mathcal{U}}_{\mathrm{gauge}}(h)$ in Equation 148. This quantity measures how much the policy changes its textual rendering preferences while staying within the same parsed executable action.

The identity in Equation 149 is the finite-support analogue of Theorem 1. When all projected distributions are computed on the same support $\mathcal{U}(h)$, the decomposition holds up to smoothing and numerical precision. We therefore report the residual

$$\Delta^{\mathcal{U}}_{\mathrm{res}}(h) = \left| \widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h) - \widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h) - \widehat{D}^{\mathcal{U}}_{\mathrm{gauge}}(h) \right| \tag{156}$$

as a sanity check for support handling, smoothing, and parser consistency.

G.7 Clipping and Trust-Response Metrics

For each real sampled action $\tilde{a} = (y_1, \ldots, y_L)$, we compute token-level likelihood ratios

$$r_\ell(\theta) = \frac{\pi_\theta(y_\ell \mid h, y_{<\ell})}{\pi_{\mathrm{old}}(y_\ell \mid h, y_{<\ell})}. \tag{157}$$

The token clipping fraction is

$$\mathrm{ClipFrac}_{\mathrm{tok}}(\tilde{a}) = \frac{1}{L} \sum_{\ell=1}^{L} \mathbf{1}\left[r_\ell(\theta) \notin [1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}] \right]. \tag{158}$$

The length-normalized sequence ratio is

$$s(\tilde{a}) = \exp\left(\frac{1}{L} \sum_{\ell=1}^{L} \log r_\ell(\theta) \right), \tag{159}$$

and the sequence clipping indicator is

$$\mathrm{Clip}_{\mathrm{seq}}(\tilde{a}) = \mathbf{1}\left[s(\tilde{a}) \notin [1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}] \right]. \tag{160}$$

For GAUGE, we compute the projected executable-action log drift

$$u^{\mathrm{exec}}_{\mathcal{U}}(a,h) = \log \frac{\widehat{\mu}^{\mathcal{U}}_{\theta}(a \mid h) + \epsilon_\mu}{\widehat{\mu}^{\mathcal{U}}_{\mathrm{old}}(a \mid h) + \epsilon_\mu}. \tag{161}$$

For this aliasing audit, we report the action-level GAUGE gate computed from projected executable-action drift only:

$$g^{\mathrm{GAUGE}}_{\mathcal{U}}(a,h) = g_{\min} + (1 - g_{\min}) \exp\left(-\frac{\phi_\delta(u^{\mathrm{exec}}_{\mathcal{U}}(a,h))}{T} \right), \qquad \phi_\delta(u) = \delta^2 \log\cosh(u/\delta). \tag{162}$$

Slot-sensitive trust responses are evaluated separately in the semantic-slot drift audit. We compare whether token-level clipping, sequence-level clipping, and the action-level GAUGE gate respond differently to naturally occurring gauge drift.

G.8 Reported Quantities

For the real-rollout audit, let $\mathcal{M}$ denote the set of evaluated environments, such as WebShop, Search-style agents, and ALFWorld. For each environment $e \in \mathcal{M}$, let $\mathcal{P}_e$ denote the set of checkpoint pairs evaluated in that environment. Each checkpoint pair $p \in \mathcal{P}_e$ is an ordered pair

$$p = \left(\pi^{(p)}_{\mathrm{old}}, \pi^{(p)}_{\theta} \right), \tag{163}$$

such as $(\pi_{\mathrm{SFT}}, \pi_{\mathrm{early}})$, $(\pi_{\mathrm{early}}, \pi_{\mathrm{mid}})$, or $(\pi_{\mathrm{mid}}, \pi_{\mathrm{late}})$.

For each environment e and checkpoint pair p, let $\mathcal{T}_{e,p}$ denote the set of audited real rollout trajectories. We split each trajectory at agent decision boundaries: after an environment observation has been received and immediately before the model generates the next raw textual action. For a trajectory $\tau \in \mathcal{T}_{e,p}$, let K_τ be the number of audited agent turns, and let $h_{\tau,k}$ denote the real interaction history before the k-th model action generation. Equivalently, $h_{\tau,k}$ is the prefix of trajectory τ ending at the observation immediately preceding the k-th action.

Unless otherwise specified, we report trajectory-normalized averages:

$$\mathrm{Avg}_{e,p}[f] = \frac{1}{|\mathcal{T}_{e,p}|} \sum_{\tau \in \mathcal{T}_{e,p}} \frac{1}{K_\tau} \sum_{k=1}^{K_\tau} f(h_{\tau,k}). \tag{164}$$

Here $f(h_{\tau,k})$ denotes any per-history diagnostic quantity, such as $\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h_{\tau,k})$, $\widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h_{\tau,k})$, $\widehat{D}^{\mathcal{U}}_{\mathrm{gauge}}(h_{\tau,k})$, or an indicator for the high-gauge / low-exec regime. This averaging scheme first averages over agent turns within each trajectory and then averages over trajectories, thereby preventing longer rollouts from dominating the reported statistics.

For a fixed history h, define the set of naturally aliased executable actions as

$$\mathcal{A}_{\mathrm{alias}}(h) = \left\{ a : \left| \widehat{\mathcal{E}}_{\mathcal{U}}(a,h) \right| \geq 2 \right\}. \tag{165}$$

We also define the union of real textual renderings belonging to naturally aliased executable actions:

$$\mathcal{U}_{\mathrm{alias}}(h) = \bigcup_{a \in \mathcal{A}_{\mathrm{alias}}(h)} \widehat{\mathcal{E}}_{\mathcal{U}}(a,h). \tag{166}$$

Natural alias rate. The natural alias rate measures how often a real rollout history contains at least one executable action with two or more observed real textual renderings. For environment e and checkpoint pair p, we define

$$\mathrm{AliasRate}_{e,p} = \mathrm{Avg}_{e,p}\left[\mathbf{1}\left[|\mathcal{A}_{\mathrm{alias}}(h)| \geq 1 \right] \right]. \tag{167}$$

A high value means that parser-equivalent textual renderings occur naturally in model rollouts, rather than only in hand-written examples.

Rendering support size. The rendering support size measures how many distinct real textual renderings are observed per naturally aliased executable action. We average over executable-action equivalence classes that actually have at least two renderings:

$$\mathrm{RenderSize}_{e,p} = \frac{\mathrm{Avg}_{e,p}\left[\sum_{a \in \mathcal{A}_{\mathrm{alias}}(h)} \left| \widehat{\mathcal{E}}_{\mathcal{U}}(a,h) \right| \right]}{\mathrm{Avg}_{e,p}\left[|\mathcal{A}_{\mathrm{alias}}(h)| \right]}. \tag{168}$$

This statistic is reported only when the denominator is nonzero. It distinguishes environments where aliasing is rare from environments where the same executable action naturally appears with many surface renderings.

Mean projected raw-text drift. The projected raw-text drift at a history h is $\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h)$, defined in Equation 146. We report its trajectory-normalized mean:

$$\overline{D}^{\mathcal{U}}_{\mathrm{text}}(e,p) = \mathrm{Avg}_{e,p}\left[\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h) \right]. \tag{169}$$

This measures the average drift in the finite-support raw textual action distribution induced by real model samples.

Mean projected executable drift. The projected executable-action drift at history h is $\widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h)$, defined in Equation 147. We report

$$\overline{D}^{\mathcal{U}}_{\mathrm{exec}}(e,p) = \mathrm{Avg}_{e,p}\left[\widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h) \right]. \tag{170}$$

This measures how much the policy changes over parser-induced executable actions, after parser-equivalent textual renderings have been identified.

Mean projected gauge drift. The projected gauge drift at history h is $\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h)$, defined in Equation 148. We report

$$\overline{D}^{\mathcal{U}}_{\text{gauge}}(e,p) = \text{Avg}_{e,p}\left[\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h)\right]. \tag{171}$$

This measures the average amount of drift that occurs within parser-induced equivalence classes, i.e., changes in textual rendering preference that do not necessarily change the parsed executable action.

High-gauge / low-exec rate. We define a high-gauge / low-exec history as one satisfying

$$\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h) > \tau_{\text{gauge}}, \qquad \widehat{D}^{\mathcal{U}}_{\text{exec}}(h) < \tau_{\text{exec}}. \tag{172}$$

Let

$$\mathbf{1}_{\text{HGLE}}(h) = \mathbf{1}\left[\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h) > \tau_{\text{gauge}} \ \wedge \ \widehat{D}^{\mathcal{U}}_{\text{exec}}(h) < \tau_{\text{exec}}\right]. \tag{173}$$

The high-gauge / low-exec rate is

$$\text{HGLE}_{e,p} = \text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h)\right]. \tag{174}$$

This is the main diagnostic region: textual rendering drift is large, while the parsed executable-action drift is small.

Token clip fraction on gauge drift. For a sampled real action $\tilde{a} = (y_1, \ldots, y_L)$, let $\text{ClipFrac}_{\text{tok}}(\tilde{a})$ be the token clipping fraction defined in Equation 158. To measure whether token-level trust responds to gauge drift, we average token clipping over real renderings inside naturally aliased equivalence classes on high-gauge / low-exec histories:

$$\text{TokClipOnGauge}_{e,p} = \frac{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h) \cdot \frac{1}{|\mathcal{U}_{\text{alias}}(h)|}\sum_{\tilde{a}\in\mathcal{U}_{\text{alias}}(h)} \text{ClipFrac}_{\text{tok}}(\tilde{a})\right]}{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h)\right]}. \tag{175}$$

Histories with $\mathcal{U}_{\text{alias}}(h) = \emptyset$ are excluded from this conditional average. A large value indicates that token-level clipping is often triggered in the regime where drift is mostly gauge drift.

Sequence clip rate on gauge drift. Let $\text{Clip}_{\text{seq}}(\tilde{a})$ be the sequence clipping indicator defined in Equation 160. We report the conditional sequence clipping rate on high-gauge / low-exec histories:

$$\text{SeqClipOnGauge}_{e,p} = \frac{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h) \cdot \frac{1}{|\mathcal{U}_{\text{alias}}(h)|}\sum_{\tilde{a}\in\mathcal{U}_{\text{alias}}(h)} \text{Clip}_{\text{seq}}(\tilde{a})\right]}{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h)\right]}. \tag{176}$$

This measures how often length-normalized sequence-level trust flags updates in histories where executable-action drift is low but gauge drift is high.

GAUGE gate on gauge drift. For a high-gauge / low-exec history, we evaluate the action-level GAUGE gate over naturally aliased executable actions. To match the weighting used in $\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h)$, we weight each executable action by its projected current-policy executable-action probability:

$$w^{\mathcal{U}}_{\theta}(a \mid h) = \frac{\widehat{\mu}^{\mathcal{U}}_{\theta}(a \mid h)}{\sum_{a'\in\mathcal{A}_{\text{alias}}(h)} \widehat{\mu}^{\mathcal{U}}_{\theta}(a' \mid h)}, \qquad a \in \mathcal{A}_{\text{alias}}(h). \tag{177}$$

The GAUGE gate on gauge drift is

$$\text{GateOnGauge}_{e,p} = \frac{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h)\sum_{a\in\mathcal{A}_{\text{alias}}(h)} w^{\mathcal{U}}_{\theta}(a \mid h) g^{\text{GAUGE}}_{\mathcal{U}}(a,h)\right]}{\text{Avg}_{e,p}\left[\mathbf{1}_{\text{HGLE}}(h)\right]}. \tag{178}$$

The desired behavior is that this quantity remains close to one in the high-gauge / low-exec regime, indicating that GAUGE does not suppress behavior-preserving rendering drift.

Unless otherwise specified, we set τ_{gauge} to the 75th percentile of $\widehat{D}^{\mathcal{U}}_{\text{gauge}}(h)$ and τ_{exec} to the 25th percentile of $\widehat{D}^{\mathcal{U}}_{\text{exec}}(h)$ within each environment. These thresholds are fixed across methods and checkpoint pairs within the same environment. We report sensitivity to these choices in additional analyses.

G.9 Sanity Checks

We perform the following sanity checks before interpreting the audit:

1. **Parser consistency:** each raw action included in $\widehat{\mathcal{E}}_{\mathcal{U}}(a, h)$ must be parsed into the same executable action a.
2. **Natural support:** all renderings included in $\widehat{\mathcal{E}}_{\mathcal{U}}(a, h)$ must be generated by at least one policy on the real history h.
3. **Decomposition residual:** the residual $\Delta^{\mathcal{U}}_{\mathrm{res}}(h)$ in Equation 156 should be small relative to $\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h)$.
4. **Gate behavior:** histories with low projected executable-action drift should have high action-level GAUGE gates even when projected textual drift is large.
5. **Invalid-action sensitivity:** including or excluding unparseable actions mapped to $a = \perp$ should not qualitatively change the high-gauge / low-exec pattern.

G.10 Expected Evidence and Visualization

The audit supports behavioral trust misalignment if a nontrivial fraction of real rollout histories satisfy

$$\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h) \gg \widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h), \qquad \widehat{D}^{\mathcal{U}}_{\mathrm{gauge}}(h) \text{ is large.} \tag{179}$$

The key region is the high-gauge / low-exec regime: these are real rollout states where the policy changes how it renders actions, while the parsed executable-action distribution changes little.

We visualize the audit with three plots:

- a scatter plot of $\widehat{D}^{\mathcal{U}}_{\mathrm{text}}(h)$ versus $\widehat{D}^{\mathcal{U}}_{\mathrm{exec}}(h)$, highlighting high-gauge / low-exec histories;
- a bar plot comparing token clipping, sequence clipping, and $g^{\mathrm{GAUGE}}_{\mathcal{U}}$ on high-gauge / low-exec histories;
- a breakdown of natural alias rate, rendering support size, and projected gauge drift by environment, action type, and training stage.

The central claim supported by this experiment is that, on real language-agent rollouts, token- and sequence-level likelihood trust can respond strongly to textual rendering drift even when executable-action drift is small. GAUGE, by contrast, keeps the action-level trust gate high in this regime because the action executed by the environment is nearly unchanged.

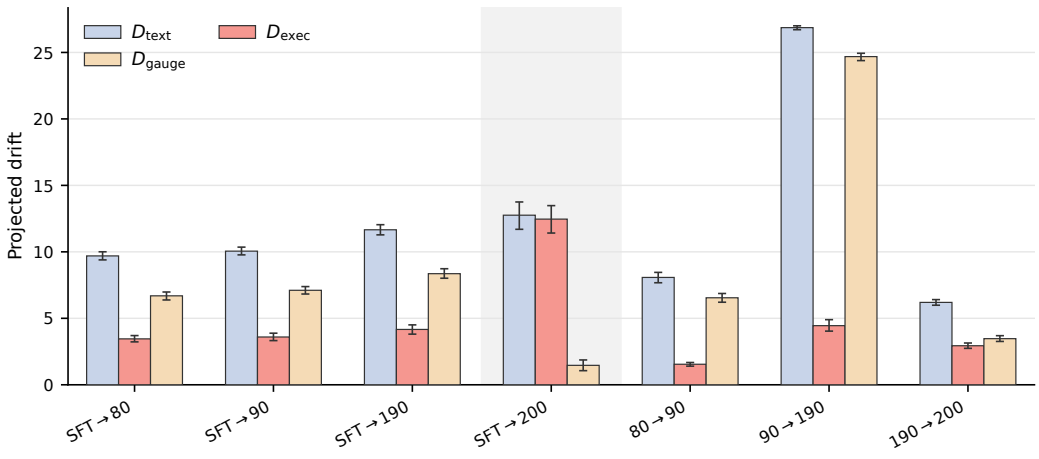


Figure 4: Real-rollout parser-equivalent aliasing audit on WebShop. We compare projected raw-text drift (D_{text}), executable-action drift (D_{exec}), and gauge drift (D_{gauge}) across checkpoint pairs from the same GRPO training trajectory. All metrics are computed on finite empirical supports induced by model samples and averaged with trajectory normalization. For most checkpoint pairs, raw-text drift substantially exceeds executable-action drift, and the excess is captured by gauge drift within parser-induced equivalence classes. The shaded SFT→200 pair is a contrast case where drift is dominated by executable-action changes rather than parser-equivalent rendering changes.

H Main-Paper Result Block: Real-Rollout Aliasing Audit

We include a real-rollout parser-equivalent aliasing audit to test whether the first failure mode of likelihood-space trust control appears in naturally collected language-agent trajectories. The audit uses WebShop rollouts generated by the trained GRPO policy at step 200. We evaluate seven checkpoint pairs on the same fixed histories: SFT→80, SFT→90, SFT→190, SFT→200, 80→90, 90→190, and 190→200. For each fixed history, we sample raw textual actions from both policies, deduplicate the raw text support, score every support element under both policies, parse each raw output into a WebShop executable action, and decompose projected raw-text drift into executable-action drift and gauge drift.

The main result is that naturally occurring WebShop histories contain a large amount of parser-equivalent aliasing. For most checkpoint pairs, the natural alias rate is above 99%, and the projected raw-text drift is substantially larger than executable-action drift. The strongest positive example is 90→190, where $D_{\text{text}} = 26.87$, $D_{\text{exec}} = 4.45$, and $D_{\text{gauge}} = 24.69$. This indicates that most of the raw likelihood drift occurs within parser-induced equivalence classes rather than between executable actions.

Recommended main-paper visualization: a grouped bar plot over checkpoint pairs with three bars per pair: D_{text}, D_{exec}, and D_{gauge}. This should be the primary figure for the real-rollout aliasing audit. (Figure 4)

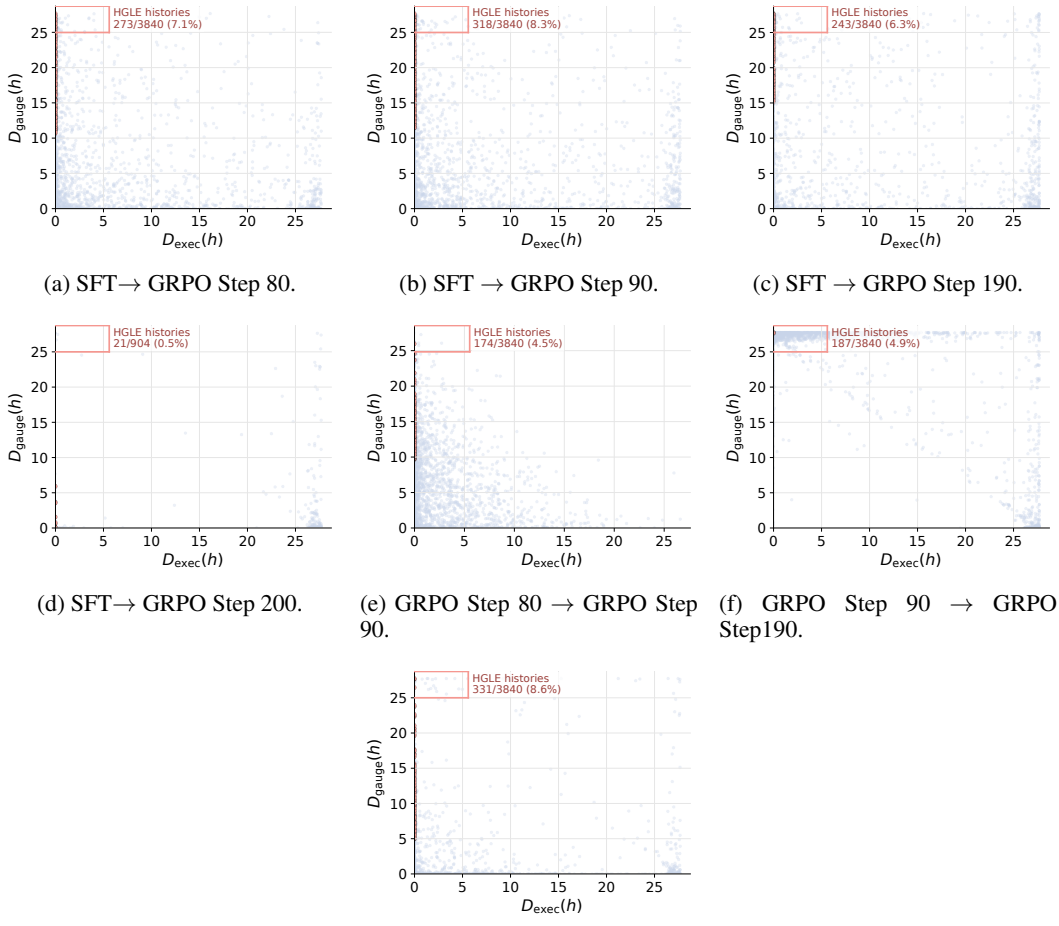
The high-gauge/low-exec (HGLE) subset isolates the most direct evidence for behavioral trust misalignment: histories where gauge drift is high while executable-action drift is near zero. In this region, raw textual drift is almost entirely gauge drift. For example, on 90→190, the HGLE subset has $D_{\text{text}} = 27.62$, $D_{\text{exec}} \approx 0$, and $D_{\text{gauge}} = 27.62$. Token-level and sequence-level clipping can still be nontrivial in this region, although the executable action distribution is essentially unchanged.

Recommended main-paper visualization: a scatter plot with $x = D_{\text{exec}}$ and $y = D_{\text{gauge}}$ for individual histories, highlighting the HGLE region. This figure directly shows that high-gauge/low-exec histories are not synthetic examples but naturally occur in real rollouts. (Figure 5)

Recommended main-paper visualization: a trust-response plot on the HGLE subset comparing token clip fraction, sequence clip rate, and GAUGE gate. This is the cleanest visual connection between the diagnostic and the GAUGE trust-control motivation. (Figure 6)

Policy pair	D_{text}	D_{exec}	D_{gauge}	Natural alias	HGLE rate
SFT→80	9.69	3.45	6.69	0.995	0.071
SFT→90	10.05	3.60	7.11	0.999	0.083
SFT→190	11.66	4.16	8.36	0.998	0.063
SFT→200	12.76	12.46	1.46	0.090	0.005
80→90	8.08	1.55	6.55	0.999	0.045
90→190	26.87	4.45	24.69	1.000	0.049
190→200	6.20	2.94	3.47	0.999	0.086

Table 5: Trajectory-normalized real-rollout aliasing audit on WebShop. Metrics are computed on the finite empirical support induced by real model samples. The primary numbers exclude invalid/unparseable outputs mapped to `BOT`.



(a) SFT→ GRPO Step 80.

(b) SFT → GRPO Step 90.

(c) SFT → GRPO Step 190.

(d) SFT→ GRPO Step 200.

(e) GRPO Step 80 → GRPO Step 90.

(f) GRPO Step 90 → GRPO Step190.

(g) GRPO Step 190 → GRPO Step 200.

Figure 5: Real-rollout high-gauge/low-exec (HGLE) scatter plots across checkpoint pairs on WebShop. Each point is one fixed WebShop history, with the x-axis showing executable-action drift $D_{\text{exec}}(h)$ and the y-axis showing gauge drift $D_{\text{gauge}}(h)$. The red callout marks the high-gauge/low-exec region and reports the number and fraction of HGLE histories for each checkpoint pair. These plots show that high gauge drift with near-zero executable-action drift occurs naturally in real rollouts, rather than only in synthetic aliasing examples.

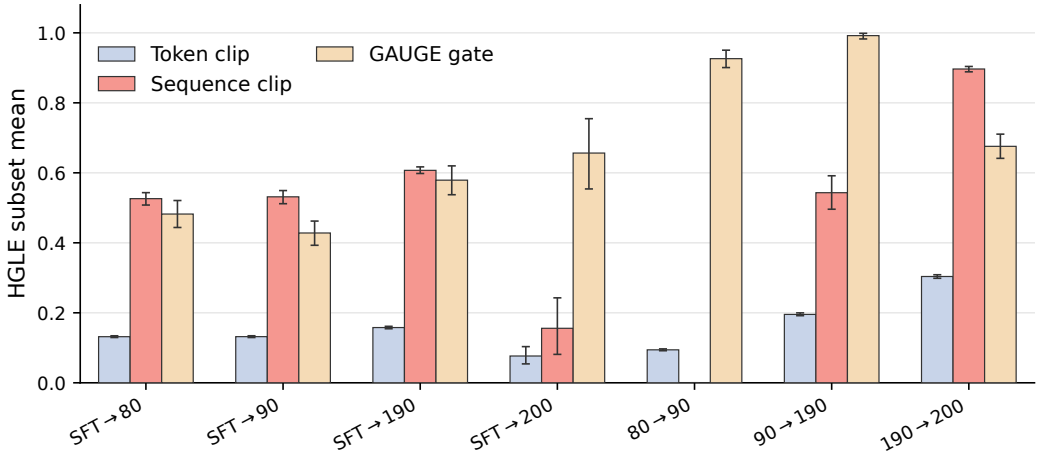


Figure 6: Trust diagnostics on the high-gauge low-executable-drift (HGLE) subset. For each checkpoint pair, we report the average token-level clipping fraction, sequence-level clipping rate, and GAUGE gate value over HGLE histories, with confidence intervals computed under trajectory-normalized averaging. The GAUGE gate remains high on several HGLE-heavy transitions, especially GRPO step 80→90 and step 90→190, indicating that the diagnostic assigns high trust to histories where raw-text drift is large but executable-action drift is small. In contrast, SFT→200 has a much smaller HGLE subset, so its larger uncertainty reflects the small number of qualifying histories rather than a broad aliasing pattern.

Policy pair	HGLE n	D_{text}	D_{exec}	D_{gauge}	Token clip	Seq. clip
SFT→80	273	17.71	9.4×10^{-10}	17.71	0.132	0.526
SFT→90	318	17.78	-7.2×10^{-15}	17.78	0.132	0.531
SFT→190	243	22.78	-2.4×10^{-14}	22.78	0.158	0.607
SFT→200	21	0.98	-2.0×10^{-15}	0.98	0.077	0.156
80→90	174	12.77	1.7×10^{-7}	12.77	0.094	0.000
90→190	187	27.62	1.0×10^{-9}	27.62	0.195	0.543
190→200	331	10.42	-4.0×10^{-15}	10.42	0.304	0.897

Table 6: High-gauge/low-exec histories. In this subset, the executable-action drift is numerically zero or near zero, while raw textual drift remains large.

I Appendix: Real-Rollout Parser-Equivalent Aliasing Audit

I.1 Experimental Setting

Dataset and environment. We run the audit on WebShop (Yao et al., 2022a), an interactive shopping environment where the language agent observes a product-search page, emits a raw textual action, and the environment parser maps this text into executable actions such as `search[...]` or `click[...]`. WebShop is a natural setting for parser-equivalent aliasing because many different surface forms can map to the same executable search query or click target. The evaluation histories are collected from the held-out WebShop text split using the same prompt format as the agentic-RL training pipeline, including the task instruction, previous observations/actions, current observation, and admissible action list.

Base SFT model. The SFT reference policy is the WebShop-tuned Qwen3-4B model used to initialize the base GRPO run. We denote this policy as SFT in all tables. The SFT model is not retrained for this audit; it is only used as one endpoint in the SFT-to-GRPO checkpoint comparisons and for teacher-forcing likelihood scoring on the fixed empirical supports.

GRPO checkpoints. The GRPO checkpoints come from a single base WebShop GRPO training run initialized from the SFT policy. We use four checkpoints from this run: global steps 80, 90, 190, and 200. Before the audit, the selected actor checkpoints are converted into standard HuggingFace-format model checkpoints for generation and likelihood scoring. Thus, all GRPO comparisons are along one training trajectory rather than across independently trained models.

Goal. This audit tests whether likelihood-space drift can be dominated by behavior-preserving textual rendering changes in real agent-environment rollouts. Unlike controlled alias diagnostics, all histories and raw outputs used here are produced by trained WebShop agents. The audit is therefore a fixed-history conditional diagnostic: it isolates conditional action-distribution drift at the same histories, rather than comparing the full on-policy state-visitation distributions of two policies.

Histories and trajectory normalization. We collect 256 WebShop trajectories from the step-200 GRPO policy and keep all 3840 turn-level histories. Every history contains a trajectory identifier, turn index, and episode length. For each metric $m(h_{\tau,k})$, where τ indexes a trajectory and k indexes a turn in that trajectory, we report the trajectory-normalized average

$$\bar{f} = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \frac{1}{K_\tau} \sum_{k=1}^{K_\tau} f(h_{\tau,k}). \tag{180}$$

This prevents longer trajectories from dominating the aggregate. We do not use a naive turn-level average because WebShop trajectories can have different numbers of decision points; a turn-level average would overweight longer or harder episodes.

Rollout collection. We collect the fixed histories by rolling out the step-200 GRPO checkpoint on WebShop with a maximum of 15 environment steps per trajectory. Generation uses a maximum prompt length of 4096, a maximum response length of 512, temperature 0.6, and sampling enabled. The resulting rollout contains 256 trajectories and 3840 turn-level histories. Each history stores trajectory metadata, so the final summaries can use trajectory-normalized averaging.

Policy pairs. We evaluate seven policy pairs: SFT→80, SFT→90, SFT→190, SFT→200, 80→90, 90→190, and 190→200. Steps 80, 90, 190, and 200 are checkpoints from the same base GRPO training run. The first four pairs compare the SFT initialization to different RL checkpoints, while the last three compare adjacent or late-stage GRPO checkpoints.

Fixed-history replay setting. For every policy pair, we reuse the same fixed WebShop histories collected from the step-200 rollout. This means the audit does not estimate how the two policies differ in their on-policy state visitation distributions. Instead, it asks a narrower conditional question: given the same history, how different are the two policies over raw textual actions and over parser-induced executable actions? This fixed-history design is intentional because it isolates parser-equivalent aliasing from changes in trajectory distribution.

Empirical support construction. For each fixed history h, we sample 16 outputs per policy, producing at most 32 raw textual outputs before deduplication. We then construct a finite empirical support $\mathcal{U}(h)$ by deduplicating exact raw text outputs. All drift metrics in this audit are projected finite-support quantities. They are not exact KL divergences over the full open-ended language action space.

Generation and scoring alignment. For support construction, each policy is sampled on the exact chat-template prompt stored in the metadata rollout. We use maximum prompt length 4096, maximum new tokens 512, and 16 samples per policy. After deduplication, every unique raw output is scored under both policies by teacher forcing on the same prompt-token sequence used for generation. This alignment is important: the drift metrics compare policy probabilities on the same finite raw-output support, not outputs generated under one prompt format and scored under another.

Parsing and invalid outputs. Each raw output is parsed into a canonical WebShop executable action. Valid actions include search actions and click actions over admissible WebShop controls, products, tabs, and buy buttons. Invalid or unparseable outputs are mapped to a distinguished invalid

action BOT. We compute the drift decomposition both including and excluding BOT; the main tables use the exclude-BOT version because BOT is not an executable environment action.

Drift decomposition. For each fixed history, the raw-text policy is defined by normalizing policy likelihoods over $\mathcal{U}(h)$. The executable-action policy is obtained by pushing the raw-text distribution through the parser. Gauge drift measures conditional rendering drift within each parser-induced executable-action equivalence class. We report

$$D_{\text{text}}(h), \qquad D_{\text{exec}}(h), \qquad D_{\text{gauge}}(h), \tag{181}$$

with the intended decomposition

$$D_{\text{text}}(h) \approx D_{\text{exec}}(h) + D_{\text{gauge}}(h). \tag{182}$$

The decomposition residual is

$$\Delta_{\text{res}}(h) = \left| D_{\text{text}}(h) - D_{\text{exec}}(h) - D_{\text{gauge}}(h) \right|. \tag{183}$$

Residuals are reported as a sanity check because smoothing, finite support, invalid-action filtering, and numerical normalization can introduce small deviations.

HGLE histories. We define high-gauge/low-exec (HGLE) histories using empirical thresholds on D_{gauge} and D_{exec}. These histories are the most direct diagnostic region for parser-equivalent aliasing: the model changes its raw textual distribution, but the executable-action distribution is unchanged or nearly unchanged.

I.2 Experimental Results

Complete trajectory-normalized summary. Table 7 reports the full trajectory-normalized summary used in the real-rollout aliasing audit. The natural alias rate is above 99% for all pairs except SFT→200. The pairs SFT→80, SFT→90, SFT→190, 80→90, 90→190, and 190→200 all show substantial D_{gauge} relative to D_{exec}.

Pair	Alias	Supp.	Act. supp.	D_{text}	D_{exec}	D_{gauge}	Resid.	Tok. clip	Seq. clip	Gate
SFT→80	0.995	16.35	3.19	9.69	3.45	6.69	0.45	0.141	0.565	0.465
SFT→90	0.999	16.36	3.02	10.05	3.60	7.11	0.65	0.141	0.576	0.479
SFT→190	0.998	16.36	3.03	11.66	4.16	8.36	0.86	0.166	0.633	0.665
SFT→200	0.090	1.70	1.33	12.76	12.46	1.46	1.17	0.072	0.156	0.912
80→90	0.999	31.77	3.57	8.08	1.55	6.55	0.01	0.097	0.000	0.591
90→190	1.000	31.77	3.69	26.87	4.45	24.69	2.27	0.205	0.635	0.572
190→200	0.999	17.09	3.50	6.20	2.94	3.47	0.21	0.312	0.911	0.719

Table 7: Complete trajectory-normalized summary for the exclude-BOT audit. "Supp." is the mean unique raw-output support size per history. "Act. supp." is the mean executable-action support size after parsing. "Gate" is the mean action-level GAUGE gate diagnostic.

HGLE subset. Table 8 reports the HGLE subset. For all positive pairs except SFT→200, the HGLE histories have large D_{text} and D_{gauge}, while D_{exec} is numerically zero or near zero. This supports the claim that a nontrivial subset of real WebShop decision points exhibit large raw likelihood drift without executable behavior drift.

Pair	HGLE n	Traj.	D_{text}	D_{exec}	D_{gauge}	Tok. clip	Seq. clip	Gate
SFT→80	273	114	17.71	9.4×10^{-10}	17.71	0.132	0.526	0.482
SFT→90	318	119	17.78	-7.2×10^{-15}	17.78	0.132	0.531	0.428
SFT→190	243	133	22.78	-2.4×10^{-14}	22.78	0.158	0.607	0.579
SFT→200	21	20	0.98	-2.0×10^{-15}	0.98	0.077	0.156	0.656
80→90	174	83	12.77	1.7×10^{-7}	12.77	0.094	0.000	0.926
90→190	187	84	27.62	1.0×10^{-9}	27.62	0.195	0.543	0.992
190→200	331	135	10.42	-4.0×10^{-15}	10.42	0.304	0.897	0.676

Table 8: HGLE subset summary. HGLE histories isolate high gauge drift and low executable-action drift.

Include-BOT versus exclude-BOT sensitivity. Table 9 compares the decomposition when invalid outputs are included as a distinguished BOT action versus excluded. The main conclusions are stable for the GRPO-to-GRPO pairs 80→90 and 90→190. The SFT-to-GRPO pairs are more sensitive because the SFT policy has a different invalid-output profile. For main behavioral conclusions we therefore report exclude-BOT results and treat include-BOT as a diagnostic for format and parsing failures.

Pair	Mode	Supp.	D_{text}	D_{exec}	D_{gauge}	Resid.	Alias
SFT→80	include BOT	28.00	10.69	7.54	4.96	1.81	0.995
SFT→80	exclude BOT	16.35	9.69	3.45	6.69	0.45	0.995
SFT→90	include BOT	28.00	10.63	7.40	5.30	2.07	0.999
SFT→90	exclude BOT	16.36	10.05	3.60	7.11	0.65	0.999
SFT→190	include BOT	28.06	8.65	5.29	4.82	1.46	0.998
SFT→190	exclude BOT	16.36	11.66	4.16	8.36	0.86	0.998
SFT→200	include BOT	15.38	9.89	1.83	8.23	0.16	0.090
SFT→200	exclude BOT	1.70	12.76	12.46	1.46	1.17	0.090
80→90	include BOT	31.97	8.06	1.55	6.53	0.01	0.999
80→90	exclude BOT	31.77	8.08	1.55	6.55	0.01	0.999
90→190	include BOT	31.99	26.79	4.44	24.61	2.26	1.000
90→190	exclude BOT	31.77	26.87	4.45	24.69	2.27	1.000
190→200	include BOT	19.44	2.66	2.35	0.71	0.40	0.999
190→200	exclude BOT	17.09	6.20	2.94	3.47	0.21	0.999

Table 9: Include-BOT versus exclude-BOT sensitivity. Main reported results exclude BOT because invalid outputs are not true executable WebShop actions.

Malformed click fix. During the audit we found a parser edge case: raw outputs of the form `click[id]]` contain an extra trailing bracket. Treating these strings as valid canonical clicks would overstate executable-action support. The aligned parser maps such outputs to BOT with parse error `malformed_click_extra_trailing_bracket`. Table 10 reports the post-fix counts. After the fix, these malformed clicks are invalid and are excluded from the main executable-action metrics.

Pair	Unique support	Malformed unique	Unique rate	Malformed samples	Sample rate
SFT→80	107512	0	0.000	0	0.000
SFT→90	107523	0	0.000	0	0.000
SFT→190	107759	0	0.000	0	0.000
SFT→200	59065	1700	0.029	1764	0.014
80→90	122774	0	0.000	0	0.000
90→190	122830	0	0.000	0	0.000
190→200	74667	1711	0.023	1763	0.014

Table 10: Malformed extra-bracket click outputs after parser correction. These outputs are mapped to invalid BOT and are excluded in the main metrics. The issue appears only in pairs involving step 200 as the sampled GRPO policy.

I.3 Experimental Analysis

Main positive evidence. The clearest positive evidence comes from 80→90 and 90→190. In both cases, the natural alias rate is essentially one and the raw-text drift is much larger than executable-action drift. For 80→90, $D_{\text{text}} = 8.08$, $D_{\text{exec}} = 1.55$, and $D_{\text{gauge}} = 6.55$. For 90→190, $D_{\text{text}} = 26.87$, $D_{\text{exec}} = 4.45$, and $D_{\text{gauge}} = 24.69$. The HGLE subsets make the phenomenon sharper: D_{exec} is numerically zero, while D_{text} and D_{gauge} remain large. This supports the claim that real WebShop rollouts naturally contain parser-equivalent textual drift that likelihood-space trust would treat as policy drift.

Support size and action support size. The support statistics explain why aliasing is visible. In the GRPO-to-GRPO pairs, each history has roughly 17–32 unique raw outputs but only about 3.5 executable actions after parsing. This compression from raw text support to executable-action support is exactly the parser-induced quotient structure used by GAUGE. The SFT-to-GRPO pairs have smaller valid support because more SFT/GRPO outputs are invalid or filtered, but still show about three executable actions per history for SFT→80, SFT→90, and SFT→190.

Decomposition residual. The decomposition residual is small for 80→90 and 190→200, moderate for the SFT-to-early-GRPO pairs, and larger for 90→190. The largest residual appears for 90→190, where the raw and gauge drifts are also largest. We interpret the residual as a numerical and finite-support diagnostic rather than a separate behavioral quantity. The HGLE subset has near-zero residual in most positive cases because D_{exec} is near zero and the finite-support mass is concentrated within a small number of executable equivalence classes.

SFT→200 special case. SFT→200 should not be used as the primary evidence for parser-equivalent aliasing. It is a useful contrast case. In the exclude-`BOT` summary, SFT→200 has $D_{\mathrm{text}} = 12.76$, $D_{\mathrm{exec}} = 12.46$, and $D_{\mathrm{gauge}} = 1.46$, with natural alias rate only 0.090. This means the late GRPO checkpoint differs from SFT mostly through real executable-action drift rather than gauge drift. The support size is also much smaller after excluding `BOT`, only 1.70 unique valid raw outputs per history on average. This pair therefore reflects a late-stage policy that is behaviorally far from SFT and has less valid parser-equivalent support for the audit.

Effect of malformed click outputs. Malformed extra-bracket clicks are concentrated in pairs involving step 200. After the parser fix, these outputs are mapped to `BOT` and removed from exclude-`BOT` metrics. This is important because treating `click[id]]` as a valid click would create artificial executable actions and inflate action support. The corrected results show that 190→200 still has substantial natural aliasing and HGLE evidence, whereas SFT→200 remains a special behavior-drift contrast case.

Interpretation for the paper. The main text should emphasize the positive real-rollout finding: in naturally collected WebShop histories, raw textual likelihood drift often decomposes into a much smaller executable-action drift plus a large gauge drift. The appendix should carry the sensitivity checks: include-`BOT` versus exclude-`BOT`, malformed click handling, decomposition residuals, support sizes, and the SFT→200 caveat. This separation keeps the main paper focused on the behavioral trust misalignment claim while making the audit reproducible and defensible.

J Detailed Appendix Pseudocode

Algorithm 1 GAUGE: Gauge-Invariant Policy Optimization

Require: Rollout batch $\mathcal{B}$ sampled by behavior policy π_{old}; current policy π_θ; reference policy π_{ref}; parser Parse; renderer Render; advantage estimates $\{A_{i,k}\}$.
Require: Hyperparameters δ_{act}, $\{\delta_c\}$, $\{w_c\}$, λ_{slot}, T, $g_{\min}$, ϵ_μ, ϵ_q, Δ, and KL coefficient α.
Initialize objective accumulator $\mathcal{J}_{\text{GAUGE}} \leftarrow 0$.
for each sampled turn (i,k) in $\mathcal{B}$ **do**
  Parse the raw textual action:

$$a_{i,k} \leftarrow \text{Parse}(\tilde{a}_{i,k}).$$

  Extract semantic slots $\mathcal{C}(a_{i,k})$.
  Construct candidate executable actions $\mathcal{A}_{\text{cand}}(h_{i,k})$ and finite rendering sets $\widehat{\mathcal{E}}(a)$.
  Estimate quotient-policy probabilities $\widehat{\mu}_\theta(a \mid h_{i,k})$ and $\widehat{\mu}_{\text{old}}(a \mid h_{i,k})$.
  Compute executable-action drift:

$$u^{\text{act}}_{i,k} = \log \frac{\widehat{\mu}_\theta(a_{i,k} \mid h_{i,k}) + \epsilon_\mu}{\widehat{\mu}_{\text{old}}(a_{i,k} \mid h_{i,k}) + \epsilon_\mu}.$$

  Compute slot drifts $\{u^{\text{slot}}_{i,k,c}\}_{c \in \mathcal{C}(a_{i,k})}$.
  Compute smooth energy:

$$E^{\text{GAUGE}}_{i,k} = \phi_{\delta_{\text{act}}}(u^{\text{act}}_{i,k}) + \lambda_{\text{slot}} \sum_{c \in \mathcal{C}(a_{i,k})} w_c \phi_{\delta_c}(u^{\text{slot}}_{i,k,c}), \qquad \phi_\delta(u) = \delta^2 \log\cosh(u/\delta).$$

  Compute trust gate:

$$g^{\text{GAUGE}}_{i,k} = g_{\min} + (1 - g_{\min}) \exp\left(-E^{\text{GAUGE}}_{i,k}/T\right).$$

  Compute stop-gradient trust weight:

$$\omega^{\text{GAUGE}}_{i,k} = \text{sg}\left[\exp\big(\Delta \tanh(u^{\text{act}}_{i,k}/\Delta)\big)\, g^{\text{GAUGE}}_{i,k}\right].$$

  Accumulate quotient-policy objective:

$$\mathcal{J}_{\text{GAUGE}} \leftarrow \mathcal{J}_{\text{GAUGE}} + \omega^{\text{GAUGE}}_{i,k} A_{i,k} \log \widehat{\mu}_\theta(a_{i,k} \mid h_{i,k}).$$

end for
Normalize $\mathcal{J}_{\text{GAUGE}}$ by the number of sampled agent turns.
Add KL regularization:

$$\mathcal{J}_{\text{GAUGE}} \leftarrow \mathcal{J}_{\text{GAUGE}} - \alpha \text{KL}(\pi_\theta \| \pi_{\text{ref}}).$$

Update θ by maximizing $\mathcal{J}_{\text{GAUGE}}$.

For each rollout batch sampled by π_{old}, GAUGE performs:

1. Parse each raw textual action $\tilde{a}_{i,k}$ into an executable action $a_{i,k}$ and semantic slots.
2. Construct a candidate executable-action set $\mathcal{A}_{\text{cand}}(h_{i,k})$ and finite rendering sets $\widehat{\mathcal{E}}(a)$.
3. Estimate quotient-policy probabilities $\widehat{\mu}_\theta(a \mid h)$ and $\widehat{\mu}_{\text{old}}(a \mid h)$.
4. Compute executable-action drift $u^{\text{act}}_{i,k}$.
5. Construct slot candidate sets and compute slot drift $u^{\text{slot}}_{i,k,c}$.
6. Convert action and slot drifts into energy $E^{\text{GAUGE}}_{i,k}$ and gate $g^{\text{GAUGE}}_{i,k}$.
7. Compute the stop-gradient trust weight $\omega^{\text{GAUGE}}_{i,k}$.
8. Update the policy using the quotient-policy objective in Eq. equation 25.

K Detailed Experimental Setup

Evaluation under the ARLArena standardized testbed. Following the standardized testbed protocol of ARLArena (Wang et al., 2026c), we evaluate GAUGE under the same stabilized agentic-RL training pipeline used for prior GRPO-style baselines. Specifically, all methods are initialized from a behavior-cloned policy trained on high-quality multi-turn interaction traces, so that the initial policy can produce valid environment actions and interact with the target tasks within a reasonable behavioral manifold. During RL training, we keep the same format-penalty mechanism that enforces structured outputs with explicit `<think>` and `<action>` tags, reducing malformed responses and invalid rollouts. We also apply the auxiliary KL regularization to the same reference policy using the standard k_3 estimator, which prevents excessive drift from the pretrained model and keeps different methods comparable under a shared stability constraint. Finally, following the ARLArena evaluation protocol, we perform method-specific hyperparameter tuning for each policy optimization algorithm, including our method, until training reaches a stable regime according to the variance of late-stage success rates. Importantly, these four components are treated as testbed-level stabilizers rather than contributions of GAUGE; our method focuses on the trust-control geometry used during policy optimization, replacing raw token- or sequence-level likelihood trust with parser-induced executable-action and semantic-slot trust while keeping the rest of the standardized training pipeline unchanged.

Tasks and environments. Following the ARLArena testbed (Wang et al., 2026c), we evaluate GAUGE on four representative agentic tasks: ALFWorld (Shridhar et al., 2020), WebShop (Yao et al., 2022a), Sokoban (Schrader, 2018), and TIR Math (Xue et al., 2025). ALFWorld evaluates long-horizon embodied decision making in text-based household environments, requiring agents to plan and execute sequential actions from natural-language observations. WebShop is an interactive web-shopping environment where agents search, inspect, and select products according to user instructions. Sokoban tests spatial planning and multi-step control in a grid-world puzzle environment. TIR Math evaluates multi-turn tool-integrated mathematical reasoning, where agents use a Python interpreter to perform intermediate computation before producing final answers. Together, these environments cover embodied interaction, web navigation, visual/spatial planning, and tool-augmented reasoning, providing a diverse testbed for evaluating policy optimization methods for language agents.

L Proof of Lemma 6

For $\delta > 0$,

$$\phi_\delta(u) = \delta^2 \log\cosh(u/\delta).$$

Because $\cosh(x) \geq 1$, we have $\log\cosh(x) \geq 0$, so $\phi_\delta(u) \geq 0$. Since $\cosh$ is even, ϕ_δ is even. Smoothness follows from smoothness and strict positivity of $\cosh$.

Differentiating gives

$$\phi'_\delta(u) = \delta^2 \frac{\sinh(u/\delta)}{\cosh(u/\delta)} \frac{1}{\delta} = \delta \tanh(u/\delta), \tag{184}$$

$$\phi''_\delta(u) = \delta \operatorname{sech}^2(u/\delta) \frac{1}{\delta} = \operatorname{sech}^2(u/\delta). \tag{185}$$

For every finite u, $\operatorname{sech}^2(u/\delta) > 0$, so ϕ_δ is strictly convex. Also,

$$0 < \operatorname{sech}^2(u/\delta) \leq 1, \qquad |\tanh(u/\delta)| \leq 1,$$

which proves

$$\phi''_\delta(u) \in (0, 1], \qquad |\phi'_\delta(u)| \leq \delta.$$

For the local behavior, use the Taylor expansion

$$\log\cosh x = \frac{x^2}{2} - \frac{x^4}{12} + \mathcal{O}(x^6) \qquad \text{as } x \to 0.$$

Substituting $x = u/\delta$ and multiplying by δ^2 yields

$$\phi_\delta(u) = \frac{u^2}{2} - \frac{u^4}{12\delta^2} + \mathcal{O}\left(\frac{u^6}{\delta^4}\right),$$

which implies Equation 18.

For the tail behavior, write

$$\cosh x = \frac{e^{|x|} + e^{-|x|}}{2} = \frac{e^{|x|}}{2}\left(1 + e^{-2|x|}\right).$$

Therefore,

$$\log\cosh x = |x| - \log 2 + \log\left(1 + e^{-2|x|}\right).$$

Since

$$\log(1 + z) = \mathcal{O}(z) \qquad \text{as } z \to 0,$$

we obtain

$$\log\cosh x = |x| - \log 2 + \mathcal{O}(e^{-2|x|}).$$

Setting $x = u/\delta$ and multiplying by δ^2 gives

$$\phi_\delta(u) = \delta|u| - \delta^2 \log 2 + \mathcal{O}\left(\delta^2 e^{-2|u|/\delta}\right),$$

which proves Equation 19 and completes the proof.

M PROOF OF PROPOSITION 7

Fix $E \geq 0$ and $T > 0$, and define

$$F(\eta) = \eta E + T\left(\eta \log \eta - \eta + 1\right), \qquad \eta > 0.$$

Its first and second derivatives are

$$F'(\eta) = E + T \log \eta, \qquad F''(\eta) = \frac{T}{\eta} > 0.$$

Hence F is strictly convex and can have at most one minimizer. The stationary condition $F'(\eta) = 0$ gives

$$E + T \log \eta = 0 \quad \Longleftrightarrow \quad \eta = \exp(-E/T).$$

Because $F''(\eta) > 0$, this stationary point is the unique minimizer:

$$\eta^\star(E, T) = \exp(-E/T).$$

Now define

$$g(E, T) = g_{\min} + (1 - g_{\min}) \exp(-E/T).$$

At $E = 0$,

$$g(0, T) = g_{\min} + (1 - g_{\min}) = 1.$$

Since $E \geq 0$,

$$0 < \exp(-E/T) \leq 1,$$

and therefore

$$g_{\min} < g(E, T) \leq 1$$

for finite E, with

$$\lim_{E \to \infty} g(E, T) = g_{\min}.$$

Finally,

$$\frac{\partial g(E, T)}{\partial E} = -\frac{1 - g_{\min}}{T} \exp(-E/T) = -\frac{g(E, T) - g_{\min}}{T} \leq 0.$$

This proves all stated properties.

N PROOF OF LEMMA 8

Let

$$\psi_\Delta(u) = \Delta \tanh(u/\Delta), \qquad \Delta > 0.$$

Because $\tanh$ is odd, ψ_Δ is odd. Differentiating gives

$$\psi'_\Delta(u) = \Delta \operatorname{sech}^2(u/\Delta)\frac{1}{\Delta} = \operatorname{sech}^2(u/\Delta).$$

Thus,

$$0 < \psi'_\Delta(u) \leq 1.$$

It follows that ψ_Δ is strictly increasing and 1-Lipschitz.

For finite u, $|\tanh(u/\Delta)| < 1$, so

$$|\psi_\Delta(u)| = \Delta|\tanh(u/\Delta)| < \Delta.$$

Exponentiating gives

$$e^{-\Delta} < \exp(\psi_\Delta(u)) < e^{\Delta}.$$

For the local expansion, use

$$\tanh x = x - \frac{x^3}{3} + \mathcal{O}(x^5) \qquad \text{as } x \to 0.$$

Substituting $x = u/\Delta$ gives

$$\psi_\Delta(u) = \Delta\left[\frac{u}{\Delta} - \frac{1}{3}\left(\frac{u}{\Delta}\right)^3 + \mathcal{O}\left(\frac{u^5}{\Delta^5}\right)\right] = u - \frac{u^3}{3\Delta^2} + \mathcal{O}\left(\frac{u^5}{\Delta^4}\right).$$

Hence,

$$\psi_\Delta(u) - u = \mathcal{O}\left(\frac{u^3}{\Delta^2}\right).$$

Therefore,

$$\begin{aligned}
\overline{r}_\Delta(u) &= \exp(\psi_\Delta(u)) && (186)\\
&= e^u \exp(\psi_\Delta(u) - u) && (187)\\
&= e^u\left[1 + \mathcal{O}\left(\frac{|u|^3}{\Delta^2}\right)\right], && (188)
\end{aligned}$$

where the last line uses $\exp(z) = 1 + \mathcal{O}(z)$ as $z \to 0$. This completes the proof.

O PROOF OF PROPOSITION 9

Recall

$$w_\theta = \overline{r}_\Delta\left(u_{i,k}^{\mathrm{act}}\right) g_{i,k}^{\mathrm{GAUGE}}, \qquad \ell_\theta = \log \widehat{\mu}_\theta(a_{i,k} \mid h_{i,k}),$$

and consider

$$\mathcal{L}_{i,k}(\theta) = \mathrm{sg}[w_\theta]\, A_{i,k} \ell_\theta.$$

During a policy update, $A_{i,k}$ is treated as fixed. By definition of the stop-gradient operator,

$$\nabla_\theta \,\mathrm{sg}[w_\theta] = 0, \qquad \mathrm{sg}[w_\theta] = w_\theta$$

in the forward pass. Applying the product rule therefore gives

$$\nabla_\theta \mathcal{L}_{i,k}(\theta) = A_{i,k} \nabla_\theta \left(\mathrm{sg}[w_\theta] \ell_\theta\right) \tag{189}$$

$$= A_{i,k} \left[\underbrace{\nabla_\theta \,\mathrm{sg}[w_\theta]}_{=0} \ell_\theta + \mathrm{sg}[w_\theta] \nabla_\theta \ell_\theta \right] \tag{190}$$

$$= w_\theta A_{i,k} \nabla_\theta \ell_\theta, \tag{191}$$

which proves Equation 21.

Without stop-gradient, the per-turn objective would be

$$\widetilde{\mathcal{L}}_{i,k}(\theta) = w_\theta A_{i,k} \ell_\theta.$$

Its gradient is

$$\nabla_\theta \widetilde{\mathcal{L}}_{i,k}(\theta) = A_{i,k} \nabla_\theta (w_\theta \ell_\theta) \tag{192}$$

$$= w_\theta A_{i,k} \nabla_\theta \ell_\theta + A_{i,k} \ell_\theta \nabla_\theta w_\theta. \tag{193}$$

The second term is the additional gradient path through the adaptive trust estimator. It allows the objective to change by modifying the trust coefficient itself, rather than only through the quotient-policy score.

It remains to prove the gradient bound. By Lemma 8,

$$\overline{r}_\Delta\left(u_{i,k}^{\mathrm{act}}\right) < e^{\Delta}.$$

By Equation 17,

$$0 < g_{i,k}^{\mathrm{GAUGE}} \leq 1.$$

Consequently,

$$0 < w_\theta < e^{\Delta}.$$

Using Equation 21 and absolute homogeneity of the norm,

$$\|\nabla_\theta \mathcal{L}_{i,k}(\theta)\| = w_\theta |A_{i,k}| \, \|\nabla_\theta \ell_\theta\| \tag{194}$$

$$\leq e^{\Delta} |A_{i,k}| \, \|\nabla_\theta \ell_\theta\| \,. \tag{195}$$

This proves Equation 22 and completes the proof.