
Do Modules Stay in Their Lane? Role Drift in Compound LLM Systems

Xiaoyang Cao
Massachusetts Institute of Technology
xycao@mit.edu

Sidharth Srinivasan
Harvard University
ssrinivasan@seas.harvard.edu

Michiel A. Bakker
Massachusetts Institute of Technology
bakker@mit.edu

Abstract

End-to-end reinforcement learning can improve the accuracy of compound LLM systems, but it does not constrain how modules divide labor internally. We identify *Role Drift*, a failure mode in which modules preserve or improve end-task performance while deviating from their assigned roles through role-violating shortcuts that remain invisible to system-level evaluation. To make such drift observable and controllable, we propose *Role Anchor*, a regularizer derived from an exponential-tilt view of prompted distributions. The key idea is to preserve the log-ratio between role-prompted and neutral-prompted predictions, which serves as a proxy for the module’s intended function during training. Experiments on two compound LLM pipelines reveal drift that accuracy alone fails to detect: a decomposer that embeds answers into sub-questions instead of decomposing, and a retrieval-augmented reader that ignores retrieved evidence in favor of parametric memory. In the decomposer setting, 86% of the RL gain vanishes under role constraints, indicating that much of the apparent improvement comes from role-violating shortcuts. Across both pipelines, Role Anchor mitigates drift at a tunable accuracy cost that varies by pipeline and anchor strength. Additional gradient analysis suggests that the regularizer reduces alignment with drift-associated update directions rather than simply suppressing learning.

1 Introduction

Compound LLM systems divide a task among specialized modules: a retriever and a reader, a planner and an executor, a decomposer and a solver. This division of labor is not just an engineering convenience; it encodes the designer’s intent about *what each module should do*. The reader should answer from retrieved evidence, not from memory. The decomposer should break problems into sub-questions, not solve them directly. Each module carries an implicit *role utility* that the designer took for granted when partitioning the system.

End-to-end reinforcement learning optimizes these pipelines against a single terminal reward. Terminal accuracy improves routinely, and the community has largely treated this as validation that the system is working as intended [Khatab et al., 2024, Yuksekgonul et al., 2024, Wang et al., 2025]. But terminal reward measures only whether the final answer is correct; it is agnostic to *which module performed which computation*. When a module can improve the terminal reward by violating its assigned role, outcome-only RL can find and exploit that shortcut, because the role utility is nowhere in the objective.

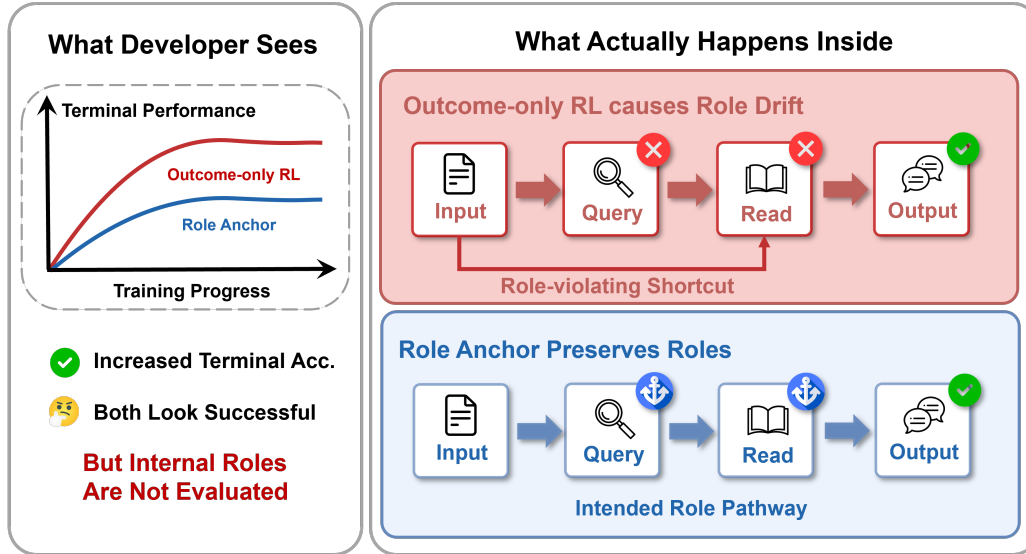


Figure 1: **Same terminal success, different computation.** Outcome-only RL can improve terminal accuracy while silently reallocating computation across modules via role-violating shortcuts (top right). Because only terminal performance is evaluated, this internal drift is invisible (left). Role Anchor preserves the intended division of labor by anchoring each module to its assigned role (bottom right).

We call this failure *Role Drift*: a module’s effective behavior diverges from its assigned role while the system’s terminal accuracy is preserved or improves. Consider a concrete example from our experiments (Section 4.2): a Decomposer–Solver pipeline where the Decomposer should write abstract sub-questions and the Solver should answer them. Under outcome-only RL, the Decomposer learns to read the answer from the passages and embed it directly into the sub-questions, letting the Solver copy it through. Terminal accuracy doubles. But when we enforce the role boundary, 86% of that gain disappears, revealing that what looked like learning was largely the Decomposer absorbing the Solver’s job. This is not an isolated case: on a separate RAG pipeline (Section 4.2), the Reader similarly learns to ignore retrieved evidence and answer from parametric memory, while terminal accuracy continues to improve.

If a role prompt’s effect on a module’s predictions changes during training, the module is no longer doing what the role prompt asks. We propose **Role Anchor**, a regularizer that preserves how a role prompt changes a module’s predictions relative to a neutral prompt, anchored to a pre-RL reference. The method adds one auxiliary term to the policy-gradient objective with no architectural change (Section 3). Because Role Anchor operates on the contrast between role and neutral prompts, it applies to LLM-based modules whose behavior is steered by natural-language instructions; non-LLM components (e.g., a fixed retriever) fall outside its scope.

Contributions. (i) **Identification:** we name *Role Drift* as a failure mode of compound-system RL that is invisible to terminal accuracy by construction, and demonstrate it on two pipelines: a retrieval-augmented reader on HotpotQA [Yang et al., 2018] and a decomposer-solver on MuSiQue [Trivedi et al., 2022]. (ii) **Diagnosis and mitigation:** we propose Role Anchor, a regularizer that serves as both a diagnostic for detecting Role Drift and a tunable constraint for controlling it, applicable to any LLM-based module steered by natural-language instructions. (iii) **Mechanism:** gradient analysis suggests that Role Anchor mitigates drift not by attenuating learning, but by redirecting parameter updates away from the drift direction.

2 Related Work

Compound-system optimization. Recent compound-AI optimizers treat an LLM pipeline as a program or graph whose prompts, demonstrations, and component policies can be optimized

against an end-task metric: DSPy [Khattab et al., 2024], TextGrad [Yuksekgonul et al., 2024], AFlow [Zhang et al., 2024], Trace/OPTO [Cheng et al., 2024], SysDPO [Wang et al., 2025], OptiMAS [Wu et al., 2026]. These methods supply our optimization context but do not encode module-level role fidelity: a module can change what job it performs if the final metric improves. Role Drift is a failure mode this leaves open.

Underspecification and shortcut learning. Underspecification means many models can match validation performance via different internal mechanisms [D’Amour et al., 2022]. Role Drift shares this spirit at the compound-system level: because terminal reward depends only on the final output, it does not constrain how modules divide labor internally, leaving room for role allocations that differ from the designer’s intent. In this sense, Role Drift can be viewed as a module finding a shortcut to improve terminal reward that bypasses its assigned role [Geirhos et al., 2020], though the shortcut arises from the multi-module structure rather than from spurious data correlations.

Reward hacking and process supervision. Reward hacking [Gao et al., 2023, Skalse et al., 2022] arises when a proxy reward has exploitable gaps relative to the true objective. Role Drift is a different pathology: the terminal reward is not wrong (the final answer *is* correct), but *incomplete*, as it specifies only system-level performance without any component-level constraint on how modules should contribute. At matched terminal accuracy, reward hacking is null while Role Drift can be large. A role-fidelity probe at matched performance separates the two. The insufficiency of outcome-only supervision is also recognized in single-model reasoning: process reward models [Lightman et al., 2023, Uesato et al., 2022] provide step-level feedback rather than relying solely on final-answer correctness. Role Anchor extends this intuition to compound systems, providing component-level supervision derived from the role prompt contrast rather than from human-labeled intermediate steps.

Inference-time role and faithfulness degradation. Even without training, LLMs can fail to maintain assigned behavior. Liu et al. [2024] show that models neglect evidence from the middle of long contexts; Laban et al. [2025] find a 39% performance drop in multi-turn versus single-turn settings, with models making premature assumptions and failing to recover; and Abdulhai et al. [2025] document persona drift in role-playing dialogue, where LLMs contradict their assigned persona over successive turns. Sycophancy [Sharma et al., 2023] similarly causes models to abandon correct answers under conversational pressure. These findings characterize role and faithfulness degradation as inference-time phenomena. Our work identifies an analogous failure induced by *training*: outcome-only RL actively pushes modules away from their assigned roles because the reward is agnostic to internal role allocation. Role Anchor provides a training-time mitigation complementary to inference-time interventions.

Context distillation. Role Anchor is closest in spirit to context distillation [Snell et al., 2022, Askell et al., 2021], which transfers a prompted teacher’s distribution into model weights offline and drops the prompt at inference. Constitutional AI [Bai et al., 2022] trains with fixed principles via self-critique; prompt internalization [Choi et al., 2022, Shin et al., 2024] transfers fixed prompts offline. RLHF and preference objectives regularize against a reference [Ziegler et al., 2019, Ouyang et al., 2022, Rafailov et al., 2023, Ethayarajh et al., 2024] without preserving how the role prompt changes predictions relative to a neutral prompt. The key distinction is what gets preserved: context distillation absorbs the prompt into weights (destroying the role-vs-neutral contrast), while Role Anchor preserves precisely this contrast online during RL (Section 3).

3 Role Drift and Role Anchor

We model a compound LLM system as a directed graph of modules with assigned roles (V_R). Each module $i \in V_R$ emits $p_\theta(v \mid h, s)$ conditioned on history h and prompt s ; the role prompt $s_r^{(i)}$ specifies intended behavior and the neutral prompt $s_0^{(i)}$ strips it.

Exponential-tilt assumption. We assume the role-conditioned distribution is an exponential tilt of the neutral distribution:

$$p_\theta(v \mid h, s_r) \propto p_\theta(v \mid h, s_0) \exp\{\beta u_r(h, v)\}, \tag{1}$$

where $u_r: \mathcal{H} \times \mathcal{V} \rightarrow \mathbb{R}$ is a latent role utility that captures how the role prompt reshapes the module’s predictions, and the proportionality constant is the normalizer $Z_\theta(h) = \sum_{v'} p_\theta(v' | h, s_0) \exp\{\beta u_r(h, v')\}$. This form is standard in reinforcement learning as inference [Levine, 2018, Todorov, 2007], DPO [Rafailov et al., 2023], and maximum-entropy IRL [Ziebart et al., 2008], where the log-ratio between two distributions reveals the underlying utility.

Role Drift. Under this view, the role prompt induces a utility u_r that steers the module toward its intended behavior. We define **Role Drift** as the failure mode in which u_r shifts during end-to-end optimization: *a module’s role utility departs from its pre-training value, so the module no longer performs the job assigned by its role prompt, even though the compound system’s terminal performance is preserved or improved.* In our retrieval-augmented reader, this means the reader becomes less sensitive to retrieved evidence; in our decomposer-solver pipeline, the decomposer begins to place answer information in its intermediate questions instead of decomposing the problem. Because terminal reward depends only on the final output, it cannot detect changes in u_r : two systems with identical terminal accuracy can have entirely different role utilities. We use *role fidelity* to refer to the degree to which a module’s behavior conforms to its assigned role; Role Drift is the erosion of role fidelity during training. We measure it via task-specific probes (Section 4.2).

Role Anchor loss. To detect and control Role Drift, we need a computable proxy for u_r . Define the log-ratio between role-prompted and neutral-prompted predictions as $\delta_\theta(v | h) = \log p_\theta(v | h, s_r) - \log p_\theta(v | h, s_0)$, and its mean-centered version over a candidate token set \mathcal{C} (the top- k tokens under the current model):

$$\bar{\delta}_\theta(v | h) = \delta_\theta(v | h) - \frac{1}{|\mathcal{C}|} \sum_{v' \in \mathcal{C}} \delta_\theta(v' | h). \quad (2)$$

Under Eq. 1, $\bar{\delta}_\theta(v | h) = \beta(u_r(h, v) - \bar{u}_r(h))$ where $\bar{u}_r(h) = |\mathcal{C}|^{-1} \sum_{v'} u_r(h, v')$; mean-centering removes the normalizer exactly, making $\bar{\delta}_\theta$ a direct proxy for the relative role utility (derivation in Appendix A.1).

We freeze the pre-RL model as a reference and compute $\bar{\delta}_{\text{ref}}$ from it using the same prompts. Role Anchor penalizes the squared difference between the current and reference centered log-ratios, averaged over token positions t :

$$\mathcal{L}_{\text{role}}(\theta) = \mathbb{E}_t \left[\frac{1}{|\mathcal{C}_t|} \sum_{v \in \mathcal{C}_t} (\bar{\delta}_\theta^{(t)}(v) - \bar{\delta}_{\text{ref}}^{(t)}(v))^2 \right], \quad (3)$$

added to the policy-gradient objective as $\max_\theta \mathbb{E}[R(y)] - \lambda \sum_{i \in V_R} \mathcal{L}_{\text{role}}^{(i)}(\theta)$, where $R(y)$ is the terminal reward and λ controls the trade-off between task performance and role preservation.

Proposition 1 (Role utility preservation). *Under the exponential-tilt assumption (Eq. 1), $\mathcal{L}_{\text{role}}(\theta) = 0$ if and only if the trained model’s role utility equals the reference’s up to a position-dependent constant: $u_{r,\theta}(h, v) = u_{r,\text{ref}}(h, v) + c(h)$ for all v, h . (Proof in Appendix A.2.)*

Proposition 1 implies that when the reference model’s role utility is reliable (i.e., the pre-RL model faithfully follows its role prompt), Role Anchor provides a principled component-level supervision signal: it constrains each module to preserve its original role behavior throughout end-to-end optimization, without requiring any task-specific probe design. To verify this precondition, we run an inference-only check before training: compare the base model’s behavior under role and neutral prompts on the drift indicator. If the role prompt produces meaningfully more role-faithful behavior than the neutral prompt, the reference is suitable for anchoring; if the two prompts are indistinguishable, the role prompt should be redesigned before proceeding. Both benchmarks pass this check (details in Appendix B). This is conceptually related to context distillation [Snell et al., 2022, Askeel et al., 2021], which absorbs the prompt-induced behavior into model weights and discards the prompt at inference. Role Anchor differs in a key way: rather than collapsing the role prompt into the model, it explicitly preserves the prompt’s differential effect ($\bar{\delta}_\theta \approx \bar{\delta}_{\text{ref}}$) and keeps the prompt alive at inference. The model’s absolute predictions are free to change (allowing RL to improve accuracy); only the role-prompt-induced *tilt* is anchored.

4 Experiments

4.1 Experimental Setup

We instantiate Role Drift on two compound pipelines. For each, we describe the system architecture, the drift pattern we aim to detect, and the dataset. Probes that quantify each drift pattern are introduced alongside results below.

RAG. The pipeline has three components: *QueryGen* rewrites the user question into a search query, *Retriever* (a frozen term-matching retriever) retrieves a supporting passage from the corpus, and *Reader* produces a yes/no answer conditioned on both the question and the retrieved passage. QueryGen and Reader are both Qwen2.5-3B-Instruct [Qwen Team, 2024]. The Reader’s assigned role is to answer *from the retrieved evidence* rather than from the question alone or its parametric prior.

Drift pattern. The Reader becomes less responsive to the retrieved passage: its answer ceases to change when the supporting passage is swapped to one implying the opposite answer.

Dataset. We train and evaluate on HotpotQA [Yang et al., 2018].

Decomposer–Solver (DEC). The pipeline has two components: *Decomposer* receives the question and supporting passages and produces a sequence of abstract sub-questions; *Solver* answers each sub-question in order and returns the final answer. Decomposer is Qwen2.5-7B-Instruct and Solver is Qwen2.5-0.5B-Instruct. We deliberately choose a much weaker Solver: this mirrors practical workflows in which a capable model plans or decomposes while a smaller model executes each sub-task, and it amplifies drift pressure because the Decomposer can easily absorb the Solver’s role. The Decomposer’s assigned role is to *decompose* the reasoning while leaving the task-solving entirely to the Solver.

Drift pattern. The Decomposer encodes answer information directly into the sub-questions (e.g. embedding the gold-answer entity), pre-empting the computation the Solver is intended to perform.

Dataset. We train and evaluate on MuSiQue-Ans [Trivedi et al., 2022].

Training. All trainable modules use independent LoRA adapters [Hu et al., 2022] (rank 16). Both pipelines are optimized with REINFORCE [Williams, 1992] and a group baseline ($k=8$ for RAG, $k=4$ for DEC; learning rates 2×10^{-5} and 1×10^{-5} ; 10 RL epochs; 3 seeds each). RAG initializes from a 3-epoch SFT checkpoint; DEC trains directly from the base model. The core comparison is *RL alone* (no anchor) vs. *RL + Role Anchor*, with λ sweeps over $\{0.05-2.0\}$ (RAG) and $\{0.05-1.0\}$ (DEC). Full hyperparameters and prompts appear in Appendix C.¹

4.2 Main Results

We evaluate both pipelines under outcome-only RL (no anchor) and RL with Role Anchor, using three seeds each. For each pipeline, we track both terminal accuracy and a task-specific role-fidelity probe: evidence-following accuracy for RAG and answer-entity insertion rate for DEC. Figure 2 summarizes the results; Table 1 reports the numerical values at matched operating points.

Outcome-only RL induces Role Drift invisible to terminal accuracy. On both pipelines, terminal accuracy improves steadily under outcome-only RL (Figure 2a, c). Yet the role-fidelity probes tell a different story.

For RAG, we test whether the Reader still relies on retrieved evidence by swapping the supporting passage to one implying the opposite answer. We call the resulting metric *evidence-following accuracy*: a Reader that genuinely uses the passage should change its answer when the evidence changes (illustrative example in Appendix D.1). Under outcome-only RL, evidence-following accuracy drops from 0.86 to 0.54 (Figure 2b), nearly reaching the 0.506 chance floor of a Reader that ignores passages entirely. The decline is gradual and monotonic, suggesting that the Reader progressively shifts from evidence-based to memory-based answering as RL proceeds.

¹Role Anchor requires log-probability access and trainable weights; API-only or non-LLM components (e.g., the frozen retriever in RAG) fall outside its scope.

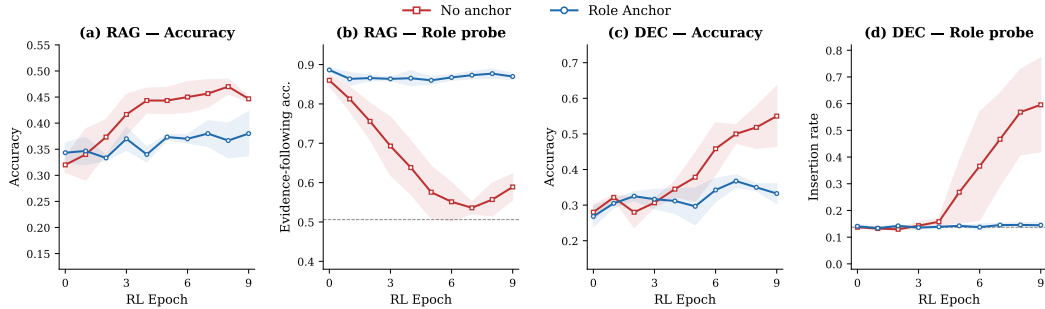


Figure 2: **Outcome-only RL improves accuracy while eroding role fidelity; Role Anchor restores it.** (a, c) Terminal accuracy; (b, d) role-fidelity probes. Red: no anchor; blue: Role Anchor. Without anchor, the RAG Reader stops following retrieved evidence (b) and the DEC Decomposer begins embedding answers into sub-questions (d), even as accuracy rises (a, c). Role Anchor keeps both probes near pre-RL levels. The accuracy gap between arms reflects the cost of preserving the intended division of labor.

For DEC, we test whether the Decomposer respects its role boundary by measuring the *answer-entropy insertion rate*: the fraction of sub-questions that contain the gold-answer entity, indicating that the Decomposer is solving the task itself rather than delegating to the Solver (concrete example in Appendix D.2). Under outcome-only RL, the insertion rate rises from 0.14 to 0.60 (Figure 2d). Unlike RAG’s gradual decline, DEC exhibits an abrupt phase transition after epoch 4, where the insertion rate triples in a single epoch. This suggests that once RL discovers the leakage shortcut, it rapidly commits to it. The pattern is consistent across all seeds, with the insertion rate increasing by a factor of two to five.

The two pipelines thus exhibit qualitatively different drift dynamics (gradual erosion on RAG versus abrupt onset on DEC), but the same structural pattern: terminal accuracy improves while role fidelity degrades, and the degradation is not captured by terminal evaluation alone.

Role Anchor recovers role-following behavior. Under Role Anchor (blue in Figure 2), both probes remain flat throughout training: the RAG Reader continues to follow retrieved evidence (0.87, Figure 2b), and the DEC Decomposer maintains its pre-RL insertion rate (~ 0.14 , Figure 2d). Crucially, the modules still improve at the task (Figure 2a, c), indicating that Role Anchor does not freeze learning but channels it through the intended roles.

Two additional probes corroborate the picture (Table 1). A *random-passage probe* on RAG replaces the retrieved passage with an unrelated one: the anchor Reader’s accuracy drops to 0.03 (correctly refusing to answer without evidence), while the no-anchor Reader retains 0.19, confirming it has learned to answer from parametric memory regardless of what the passage says. A *passage-removal probe* on DEC removes all passages: accuracy collapses to near zero for both arms (0.05 and 0.08), confirming that both systems depend on passages. The difference is *how* they use them: the unanchored Decomposer extracts answers and embeds them in sub-questions; the anchored Decomposer writes abstract sub-questions and lets the Solver reason over the passages.

The accuracy cost of role preservation. Preserving role fidelity comes at an accuracy cost that varies markedly between the two pipelines (Table 1). On RAG, the cost is modest: -0.067 in accuracy for a $+0.280$ gain in evidence-following. The Reader becomes slightly less accurate overall, but the answers it produces are genuinely grounded in retrieved evidence rather than recalled from parametric memory, a property that matters whenever the retrieval corpus is updated or the question falls outside the model’s training distribution.

On DEC, the cost is large, but this is itself the key finding. Unanchored RL improves accuracy by $+0.310$ above the base, but under Role Anchor the improvement is only $+0.057$: across seeds, $86\% \pm 19\%$ of the apparent RL gain vanishes under role constraints. This asymmetry between RAG and DEC reflects a difference in how much “genuine” learning each pipeline has beyond the drift shortcut. RAG’s Reader has a legitimate improvement pathway (learning to extract answers from evidence more effectively), so most of its gain survives the anchor. DEC’s Decomposer, paired with

Pipeline	Metric	No anchor	Role Anchor
RAG	Accuracy	0.447	0.380
	Evidence-following acc.	0.589	0.869
	Random-passage acc.	0.187	0.030
DEC	Accuracy	0.550	0.297
	Insertion rate	0.596	0.143
	Acc. without passages	0.050	0.080

Table 1: **Accuracy and role-fidelity probes** at epoch 9 (3-seed mean). *Evidence-following acc.*: Reader accuracy after swapping the passage to one implying the opposite answer (higher means the Reader follows evidence). *Random-passage acc.*: Reader accuracy when the passage is replaced with an unrelated one (lower means the Reader relies on evidence, not memory). *Insertion rate*: fraction of Decomposer sub-questions containing the gold-answer entity (lower means the Decomposer respects its role boundary). *Acc. without passages*: accuracy when passages are removed (near-zero for both arms suggests accuracy is passage-mediated).

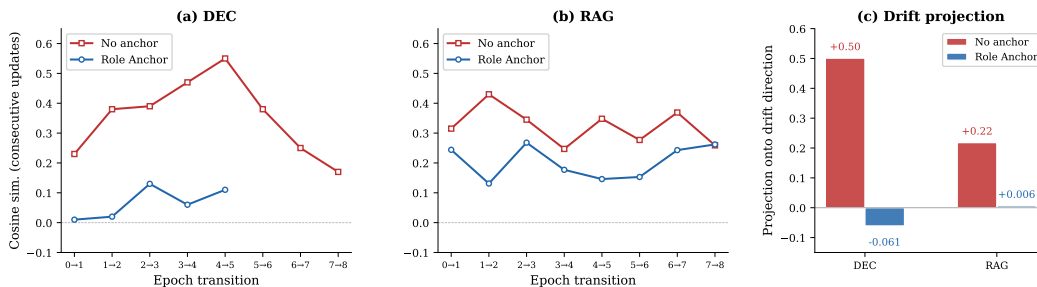


Figure 3: **Role Anchor reduces alignment with the drift direction rather than suppressing learning overall.** (a, b) Cosine similarity between consecutive LoRA updates over training epochs. Without anchor (red), updates cohere along a consistent direction; with anchor (blue), coherence drops sharply on DEC (0.36 \rightarrow 0.07) but only modestly on RAG (0.32 \rightarrow 0.20), reflecting the different amounts of legitimate learning available beyond the drift shortcut. (c) Projection of each method’s updates onto the drift direction. Unanchored updates align positively with drift (DEC: +0.50, RAG: +0.22); anchored updates project to near zero (DEC: -0.06 , RAG: +0.006).

a much weaker 0.5B Solver, has little room for genuine improvement and relies heavily on bypassing the Solver. The accuracy cost is not a limitation of Role Anchor but a diagnostic: it quantifies how much of the system’s improvement depends on role violation.

4.3 How Does Role Anchor Mitigate Drift?

A natural hypothesis is that Role Anchor mitigates drift by shrinking parameter updates. The gradient geometry reveals a more nuanced picture that also illuminates why drift affects the two pipelines differently (Figure 3).

We extract LoRA adapter weights at each epoch and compute the cosine similarity between consecutive update vectors $\Delta\theta_t = \theta_t - \theta_{t-1}$. On DEC (Figure 3a), the unanchored Decomposer’s updates show a characteristic arc: coherence rises from 0.23 to a peak of 0.55 around epoch 4–5, then decays as drift saturates. This peak coincides with the onset of the insertion-rate surge (Figure 2d), suggesting that directional commitment in parameter space precedes and drives task-level drift. Under Role Anchor, DEC’s cosine drops to ~ 0.07 , nearly indistinguishable from random high-dimensional vectors. This near-zero coherence is consistent with the main results: 86% of DEC’s unanchored accuracy gain depended on drift (Section 4.2), so once drift is removed, the model has little consistent direction left to follow.

RAG tells a complementary story (Figure 3b). Without anchor, updates fluctuate around 0.32; with anchor, coherence decreases to 0.20, a smaller reduction than DEC. But this is not a weakness of anchor on RAG. It reflects the fact that RAG’s Reader has a legitimate improvement pathway (learning to use passage evidence more effectively) that persists even after drift is removed. This is

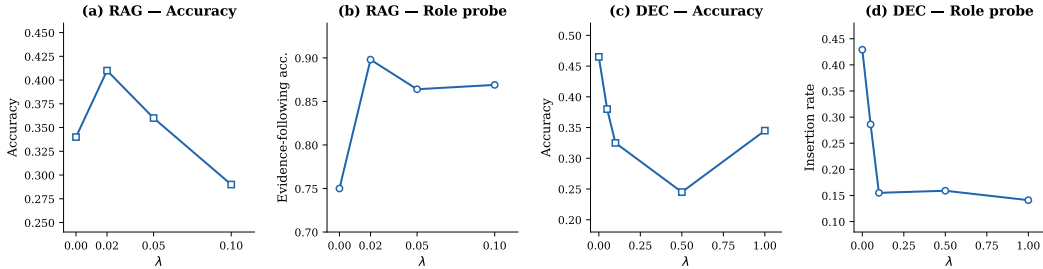


Figure 4: **A small λ suffices to suppress most drift.** The anchor strength λ provides a continuous trade-off between accuracy (a, c) and role fidelity (b, d). On RAG, even $\lambda=0.02$ closes most of the evidence-following gap. On DEC, insertion rate decreases nearly monotonically with λ .

consistent with RAG’s modest accuracy cost (-0.067 , Table 1): the anchor Reader keeps learning, just not through the parametric-prior shortcut.

To test whether the anchor specifically blocks the drift direction rather than reducing learning in general, we define a *drift direction* as the mean unanchored update vector during peak drift epochs (ep4–6 for DEC, ep2–6 for RAG) and project each method’s per-epoch updates onto it (Figure 3c). Unanchored updates align with the drift direction ($+0.50$ on DEC, $+0.22$ on RAG), while anchored updates project to approximately zero (-0.06 on DEC, $+0.006$ on RAG). On both pipelines, Role Anchor appears to reduce the alignment of parameter updates with the drift direction, allowing continued task learning with less directional commitment to the role-violating shortcut.

4.4 Ablation Study

We ablate the anchor strength λ to understand the accuracy–role-fidelity trade-off (Figure 4).

The most striking finding is that a small λ already captures most of the role-fidelity benefit. On RAG, $\lambda=0.02$ raises evidence-following accuracy from 0.75 to 0.90 (Figure 4b) while also *improving* task accuracy to 0.41, the highest value across all tested settings including the unanchored baseline (0.34). This suggests that mild role enforcement can help rather than hurt performance when the drift shortcut (parametric memory) is noisier than the intended pathway (passage evidence). Beyond $\lambda=0.02$, evidence-following saturates while accuracy declines.

On DEC, the trade-off is qualitatively different: accuracy decreases in the mid-range ($\lambda=0.10$ – 0.50) but partially recovers at $\lambda=1.00$ (0.35), while insertion rate drops nearly monotonically (Figure 4d). The recovery at high λ suggests that a sufficiently strong anchor forces the Decomposer to discover a legitimate decomposition strategy that intermediate settings do not fully commit to.

The contrast between the two pipelines reinforces the diagnostic value of the λ sweep: on RAG, where legitimate learning exists beyond the shortcut, even a small anchor yields a favorable trade-off; on DEC, where most of the gain is drift-dependent, stronger anchoring is needed and the accuracy cost is correspondingly larger.

4.5 Why Do Role Boundaries Matter?

A natural objection is that Role Drift improves terminal accuracy, so why prevent it? The answer depends on what the deployment requires beyond a correct final answer.

On RAG, the Reader’s role is to answer *from retrieved evidence*. A Reader that ignores passages and answers from parametric memory may produce correct answers on familiar questions, but it is vulnerable to outdated information and hallucination on questions where the parametric prior is wrong or stale. The retrieval stage exists precisely to ground answers in current, verifiable sources; a drifted Reader makes the entire retrieval pipeline functionally irrelevant.

On DEC, the Decomposer’s role is to *direct* the Solver by producing abstract sub-questions, not to solve the task itself. A key motivation for decomposition is efficiency: a powerful Decomposer plans the work while one or more Solvers execute sub-tasks in parallel. When the Decomposer absorbs the Solver’s role by embedding answer entities into sub-questions, this parallelization benefit disappears

because the Decomposer is doing all the work, and the Solver is reduced to a copy mechanism. Beyond efficiency, auditability suffers: downstream stakeholders cannot verify whether the system genuinely reasoned through the problem or short-circuited it.

In both cases, the accuracy gain from drift is fragile: it depends on a task-specific shortcut that may not transfer to new downstream modules or out-of-distribution queries. The role boundary is a property of the deployment, not an arbitrary constraint. We do not claim Role Anchor improves terminal accuracy in general; rather, it makes Role Drift measurable and tunable, letting practitioners choose a position on the role-fidelity–accuracy frontier based on their deployment requirements.

5 Conclusion

We identify Role Drift as a failure mode of compound-system RL in which modules deviate from their assigned roles while terminal accuracy continues to improve. On two pipelines with different architectures and drift patterns, task-specific probes reveal role-fidelity degradation that standard accuracy evaluation does not capture. We propose Role Anchor, a regularizer that preserves each module’s role-prompt-induced behavior during training. Its anchor strength λ provides a continuous trade-off between accuracy and role fidelity, and the accuracy cost under anchoring serves as a diagnostic for how much of the system’s improvement depends on role violation.

Gradient analysis suggests that Role Anchor works by reducing the alignment of parameter updates with the drift direction rather than by slowing learning overall. This is consistent with the observation that anchored systems continue to improve at the task through their intended roles. We believe these findings point to a broader principle: when optimizing compound systems end-to-end, it is worth evaluating not only whether the final answer improves, but whether the improvement arrives through the module pathway the system was designed around.

6 Limitations

Role Anchor requires log-probability access under both role and neutral prompts, a frozen reference model, and trainable weights for backpropagation. It is therefore only applicable to LLM-based components with weight access, and does not extend to API-only modules, non-LLM components (e.g., retrievers), or prompt-optimized systems. Our empirical evaluation is limited to two compound pipelines (RAG and Decomposer–Solver) at 3B/7B scale with LoRA adapters. While these pipelines exhibit structurally different drift patterns, they do not represent the full diversity of compound AI systems, and the generality of both the drift phenomenon and the anchor’s effectiveness to other architectures, scales, and task domains remains an open question. On potential negative impacts: role-preservation objectives could make harmful or poorly specified pipelines more robust against corrective drift, and incomplete role probes may create false confidence in the fidelity of deployed systems.

References

- Marwa Abdulhai, Ryan Cheng, Donovan Clay, Tim Althoff, Sergey Levine, and Natasha Jaques. Consistently simulating human personas with multi-turn reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.
- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization with rich feedback, execution traces, and llms. *Advances in Neural Information Processing Systems*, 37:71596–71642, 2024.

- Eunbi Choi, Yongrae Jo, Joel Jang, and Minjoon Seo. Prompt injection: Parameterization of fixed inputs. *arXiv preprint arXiv:2206.11349*, 2022.
- Alexander D’Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.
- Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.
- Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askill, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2023.
- Haebin Shin, Lei Ji, Yeyun Gong, Sungdong Kim, Eunbi Choi, and Minjoon Seo. Generative prompt internalization. *arXiv preprint arXiv:2411.15927*, 2024.
- Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *NeurIPS*, 2022.
- Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

- Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, 2007.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl.a.00475.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- Xiangwen Wang, Yibo Jacky Zhang, Zhoujie Ding, Katherine Tsai, Haolun Wu, and Sanmi Koyejo. Aligning compound AI systems via system-level DPO. *arXiv preprint arXiv:2502.17721*, 2025.
- Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.
- Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenec Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, Diyi Yang, James Zou, and Jure Leskovec. Optimas: Optimizing compound AI systems with globally aligned local rewards. In *International Conference on Learning Representations (ICLR)*, 2026.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.
- Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic differentiation via text. *arXiv preprint arXiv:2406.07496*, 2024.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.
- Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.
- Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Theoretical Details

A.1 Exponential-Tilt Derivation

This subsection derives the Role Anchor loss from the exponential-tilt assumption stated in Section 3.

Exponential-tilt assumption. For each role-bearing module $i \in V_R$ with role prompt s_r and neutral prompt s_0 , we assume the role-conditioned distribution is an exponential tilt of the neutral distribution:

$$p_\theta(v | h, s_r) = \frac{1}{Z_\theta(h)} p_\theta(v | h, s_0) \exp\{\beta u_r(h, v)\}, \quad (4)$$

where $u_r: \mathcal{H} \times \mathcal{V} \rightarrow \mathbb{R}$ is a smooth role utility and $Z_\theta(h) = \sum_v p_\theta(v | h, s_0) \exp\{\beta u_r(h, v)\}$ is the partition function. The existence of *some* u_r satisfying Eq. 1 is trivial when the two distributions share support; the substantive content is that u_r be smooth and low-dimensional relative to the full output space.

Log-ratio identifies relative role utility. Taking the log-ratio of Eq. 1,

$$\delta_\theta(v | h) = \log p_\theta(v | h, s_r) - \log p_\theta(v | h, s_0) = \beta u_r(h, v) - \log Z_\theta(h). \quad (5)$$

Since $\log Z_\theta(h)$ is constant across tokens v , for any tokens v, v' ,

$$\delta_\theta(v | h) - \delta_\theta(v' | h) = \beta [u_r(h, v) - u_r(h, v')]. \quad (6)$$

Pairwise log-ratio differences identify the relative role utility up to scale β ; the partition function cancels.

Mean-centering and the Role Anchor loss. To obtain a computable proxy, we mean-center δ_θ over a candidate token set \mathcal{C} , which removes the partition-function offset:

$$\bar{\delta}_\theta(v | h) = \delta_\theta(v | h) - \frac{1}{|\mathcal{C}|} \sum_{v' \in \mathcal{C}} \delta_\theta(v' | h) = \beta \left(u_r(h, v) - \frac{1}{|\mathcal{C}|} \sum_{v' \in \mathcal{C}} u_r(h, v') \right). \quad (7)$$

Role Anchor penalizes departure of the centered log-ratio from the frozen reference’s: $\mathcal{L}_{\text{role}} = |\mathcal{C}|^{-1} \sum_{v \in \mathcal{C}} (\bar{\delta}_\theta(v | h) - \bar{\delta}_{\text{ref}}(v | h))^2$.

Behavior at and away from the reference. At $\theta = \theta_{\text{ref}}$, $\bar{\delta}_\theta = \bar{\delta}_{\text{ref}}$ and the loss is identically zero. If RL improves terminal performance *without changing* the role utility ($u_{r,\theta} \approx u_{r,\text{ref}}$), the loss remains near zero. If RL changes the role utility, the loss grows in proportion to the change.

What the assumption does not imply. Eq. 1 only states that, at any given checkpoint, the role-conditioned distribution can be written as an exponential tilt with *some* role utility u_r . It does not by itself imply that u_r is preserved across training. An RL update can perfectly satisfy Eq. 1 at every checkpoint while the role utility shifts substantially. The Role Anchor loss *detects* this shift; it does not become zero by virtue of the assumption alone.

A.2 Proof of Proposition 1

Proof. Under the exponential-tilt assumption, the mean-centered log-ratio at any checkpoint is (Eq. 2): $\bar{\delta}_\theta(v | h) = \beta(u_{r,\theta}(h, v) - \bar{u}_{r,\theta}(h))$, where $\bar{u}_{r,\theta}(h) = |\mathcal{C}|^{-1} \sum_{v'} u_{r,\theta}(h, v')$.

(\Rightarrow) If $\mathcal{L}_{\text{role}}(\theta) = 0$, then $\bar{\delta}_\theta(v | h) = \bar{\delta}_{\text{ref}}(v | h)$ for all v, h . Expanding: $\beta(u_{r,\theta}(h, v) - \bar{u}_{r,\theta}(h)) = \beta(u_{r,\text{ref}}(h, v) - \bar{u}_{r,\text{ref}}(h))$. This gives $u_{r,\theta}(h, v) - u_{r,\text{ref}}(h, v) = \bar{u}_{r,\theta}(h) - \bar{u}_{r,\text{ref}}(h)$ for all v . The right-hand side depends only on h , so setting $c(h) = \bar{u}_{r,\theta}(h) - \bar{u}_{r,\text{ref}}(h)$ yields $u_{r,\theta}(h, v) = u_{r,\text{ref}}(h, v) + c(h)$.

(\Leftarrow) If $u_{r,\theta}(h, v) = u_{r,\text{ref}}(h, v) + c(h)$, then $\bar{u}_{r,\theta}(h) = \bar{u}_{r,\text{ref}}(h) + c(h)$, so $u_{r,\theta}(h, v) - \bar{u}_{r,\theta}(h) = u_{r,\text{ref}}(h, v) - \bar{u}_{r,\text{ref}}(h)$, giving $\bar{\delta}_\theta = \bar{\delta}_{\text{ref}}$ and $\mathcal{L}_{\text{role}} = 0$. \square

We therefore treat the loss as a soft constraint: under finite λ and a terminal reward that conflicts with role preservation, the optimum is an interior trade-off (the anchor trade-off analysis), not a zero-loss point.

B Pre-Training Calibration

Before anchor training, we verify that the role prompt has a measurable effect on the base model by comparing its behavior under role and neutral prompts on the drift indicators defined in Table 1. The pipelines are described in Section 4.1. Table A1 confirms that the role prompt improves role fidelity on both benchmarks relative to the neutral prompt, validating the reference model for use in Role Anchor.

Pipeline	Base model	Indicator	Role	Neutral	Margin
RAG	Qwen2.5-3B-Instruct	Evidence-following acc. (higher is better)	0.659	0.580	+0.079
RAG	Qwen2.5-3B-Instruct	Random-passage acc. (lower is better)	0.205	0.272	-0.067
DEC	Qwen2.5-7B-Instruct	Insertion rate (lower is better)	0.130	0.210	-0.080

Table A1: Pre-training calibration on the base model. The role prompt produces meaningfully more role-faithful behavior than the neutral prompt on all indicators. Role and neutral prompts are recorded verbatim in Appendix C.

C Hyperparameters and Prompts

Table A2 summarizes the key hyperparameters for both pipelines. Full experimental setup is in Section 4.1.

	RAG	DEC
Base model	Qwen2.5-3B-Instruct	Qwen2.5-7B-Instruct (Decomposer) Qwen2.5-0.5B-Instruct (Solver)
LoRA rank	16	16
LoRA alpha	32	32
Adapter sharing	Independent	Independent
SFT initialization	3 epochs	None (base model)
RL algorithm	REINFORCE + group baseline	REINFORCE + group baseline
Rollouts per question (k)	8	4
Learning rate	2×10^{-5}	1×10^{-5}
RL epochs	10	10
Seeds	42, 43, 44	42, 43, 44
Anchor λ (main result)	0.05	1.00
Anchor λ (sweep)	{0, 0.02, 0.05, 0.10}	{0, 0.05, 0.10, 0.50, 1.00}
Anchor top- k tokens	5	5
Evaluation set size	176 (yes/no slice)	200 (MuSiQue-Ans dev)
GPU	1 \times A100 80 GB	1 \times A100 80 GB
Approx. wall-clock per run	\sim 2 h (no anchor), \sim 2.5 h (anchor)	\sim 4 h (no anchor), \sim 4.7 h (anchor)

Table A2: Hyperparameters for the two pipelines.

Compute budget. All experiments were run on single NVIDIA A100 80 GB GPUs. Each RAG run (10 RL epochs, 3B Reader + 3B QueryGen) takes approximately 2 hours without anchor and 2.5 hours with Role Anchor (\sim 22% overhead from two extra frozen-reference forward passes). Each DEC run (10 RL epochs, 7B Decomposer + 0.5B Solver) takes approximately 4 hours without anchor and 4.7 hours with Role Anchor (\sim 18% overhead). With 3 seeds \times 2 conditions (anchor/no-anchor) \times 2 pipelines, plus λ -sweep runs, the total compute for reported experiments is approximately 120 A100-hours. Preliminary and failed experiments roughly doubled this figure.

C.1 Verbatim Prompts

RAG Reader role prompt:

You are a careful Reader. Use the retrieved passages to answer the user’s question. If the passages clearly support an answer, give that answer; if they do not, say so. Do not rely on outside knowledge that conflicts with the passages.

RAG Reader neutral prompt:

Answer the user’s question. Format: a concise final answer.

DEC Decomposer role prompt:

You are a question decomposer for multi-hop questions. Given a complex question, produce a numbered list of 2-4 simpler sub-questions whose answers chain to answer the original. Example: Question: What is the population of the country where Mount Everest is located? 1. In which country is Mount Everest located? 2. What is the population of #1? Compare: GOOD: "What is the population of #1?" BAD: "What is the population of Nepal?" (do not write the answer entity from the passages; use #N instead) Return 2-4 numbered sub-questions. Use #N for chained answers.

DEC Decomposer neutral prompt:

Anchor reference model. We use the post-SFT checkpoint as the frozen anchor reference. In our LoRA setup, the reference is a frozen copy of the adapter loaded alongside the training adapter; no extra model copy is needed.

You are a question decomposer for multi-hop questions. Given a complex question, produce a numbered list of 2-4 simpler sub-questions whose answers chain to answer the original. Use #N notation to reference the answer of sub-question N (e.g. 'Where is #1 located?'). Output ONLY the numbered list, no commentary.

D Qualitative Examples

D.1 RAG: Evidence-Following Probe Example

The following example illustrates the counterfactual evidence-swap probe used on the RAG pipeline. By swapping the retrieved passage to one supporting the opposite answer, we test whether the Reader follows the evidence or relies on parametric memory:

Question: Are Mount Tamalpais and Mount Diablo in the same state?

Original passages (supports "yes"):

- [1] Mount Tamalpais: Located in California.
- [2] Mount Diablo: Located in California.

Counterfactual passages (supports "no"):

- [1] Mount Tamalpais: Located in California.
- [2] Mount Diablo: Located in Nevada.

Role-faithful Reader: no (follows counterfactual evidence)
Role-drifted Reader: yes (ignores evidence, relies on memory)

A role-faithful Reader changes its answer when the evidence changes; a drifted Reader answers from parametric memory regardless of what the passages say. The evidence-following accuracy reported in Table 1 measures how often the Reader's answer tracks the counterfactual evidence.

D.2 DEC: Answer-Entity Insertion Example

The following example illustrates the answer-entity insertion pattern in the Decomposer-Solver pipeline. The Decomposer receives a multi-hop question and supporting passages, and should produce abstract sub-questions using #N references:

Question: What other county does the county where Imperial is located share a border with?

Decomposer output:

- #1 Where is Imperial, Texas located?
- #2 Which counties are located near Pecos County, Texas?

#3 Which other county does Pecos County, Texas share a border with?

The gold entities are *Pecos County* and *Crockett County*. Sub-questions #2 and #3 contain *Pecos County, Texas*, which the Decomposer resolved from the passages instead of preserving the abstract reference #1. This is Role Drift in miniature: the Decomposer reads the answer from the passages and embeds it into the sub-questions, bypassing the Solver's intended role.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: The abstract and introduction (Section 1) state three contributions—identification of Role Drift, the Role Anchor regularizer, and empirical demonstration on two pipelines—which are directly supported by Sections 3–4.2. Scope limitations (two pipelines, 3B/7B scale, LoRA only) are noted in the introduction and detailed in Section 6.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Section 6 discusses the key limitations: Role Anchor requires LLM weight access (excluding API-only or non-LLM components), and the empirical evaluation covers only two compound pipelines at 3B/7B scale, which does not represent the full diversity of compound AI systems.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The exponential-tilt assumption is stated as Assumption 1 in Section 3. Proposition 1 is stated with full assumptions and proved in Appendix A.2. The derivation of the Role Anchor loss appears in Appendix A.1.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4.1 specifies model identifiers, LoRA rank, learning rates, number of seeds, and RL algorithm. Appendix C provides a complete hyperparameter table and verbatim prompts. Both datasets (HotpotQA, MuSiQue-Ans) are publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Code is not included with the anonymous submission. We plan to release the training and evaluation code upon acceptance. Both datasets used (HotpotQA, MuSiQue-Ans) are publicly available.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4.1 describes the training setup including optimizer, learning rates, number of rollouts, and seeds. Appendix C provides a comprehensive hyperparameter table. Data splits follow the standard splits of HotpotQA and MuSiQue-Ans.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Main results use 3 random seeds (42, 43, 44) with shaded ± 1 standard deviation bands in trajectory plots (Figures 2). Table 1 reports 3-seed means.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Section 4.1 and Appendix C specify the model sizes, LoRA configuration, training epochs, GPU type and memory, approximate runtime per run, and total compute used for the reported experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics [https://neurips.cc/public/EthicsGuidelines?](https://neurips.cc/public/EthicsGuidelines)

Answer: [Yes]

Justification: The research uses publicly available datasets and open-weight models, involves no human subjects, and conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Section 5 discusses positive impacts on the reliability and interpretability of compound AI systems. Section 6 notes possible negative impacts: role-preservation objectives could make harmful or poorly specified pipelines more robust against corrective drift, and incomplete role probes may create false confidence in deployed systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release new models or datasets. All experiments use publicly available open-weight models and established benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the original papers for HotpotQA (CC BY-SA 4.0) and MuSiQue-Ans (CC BY 4.0). Qwen2.5 models are cited via their technical report; the 0.5B and 7B variants are released under Apache 2.0, and the 3B variant is released under the Qwen Research License.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: No new datasets or pre-trained models are released with this submission. Code will be released upon acceptance.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs (Qwen2.5 family) are the core components of the compound systems studied. Their use as pipeline modules is described in detail in Section 4.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.