Compositional Skill Acquisition in Agentic Pipelines via Reinforcement Learning and Knowledge Distillation

Akshaykumar [1] **Tadisetty Sai Yashwanth** [1]

Abstract

We study compositional skill acquisition in small language models (SLMs) trained for agentic pipeline tasks via reinforcement learning with verifiable rewards (RLVR) and knowledge distillation (KD). On a benchmark of 100 agentic coding tasks spanning four categories with varying compositional structure, RLVR achieves near-perfect generalization on structured categories (error-recovery: 0.000→1.000; multi-file feature: 0.667→1.000) but scores 0.000 on sparse-reward categories where the base model rarely succeeds. KD on 100 tasks achieves the strongest overall result ($+0.62$ held-out improvement), uniquely solving the sparse-reward tool/API category that RLVR cannot. The two methods fail on complementary categories for structurally distinct reasons, motivating a staged KD→RLVR training recipe.

1. Introduction

Compositionality, defined as the construction of complex behavior from reusable components, is widely regarded as a prerequisite for systematic generalization in artificial agents (Lake & Baroni, 2018; Fodor & Pylyshyn, 1988). A compositionally capable agent does not merely memorize training instances; it extracts transferable patterns that apply to new inputs drawn from the same distribution of tasks.

Modern agentic pipelines are themselves compositional structures: a sequence of specialized reasoning nodes, each consuming the output of the prior stage and producing a well-defined artifact. Yet these nodes are typically implemented with general-purpose frontier model APIs, raising two questions that are central to this workshop:

1. **Do models learn compositionally structured skills, or do they memorize?** Does training an SLM on agentic pipeline tasks produce transferable compositional patterns, or only task-specific overfitting?
2. **Which training paradigm better acquires compositional skills?** RLVR via self-play, or knowledge distillation from a teacher?

We address these questions through **Prism**, an open-source framework for training SLMs to replace frontier API calls at individual pipeline nodes. We evaluate two training methods on a benchmark of 100 agentic coding tasks spanning four categories that differ in their degree of compositional structure. Our key findings are:

- **RLVR acquires compositional skills.** On categories with transferable pattern structure (error-recovery, multi-file feature), RLVR achieves near-perfect held-out generalization from 50 training examples. The model learns the *pattern*, not the *instance*.
- **KD bypasses the sparse-reward barrier.** On categories where the base model rarely produces a correct output, RLVR receives no gradient signal and fails entirely. KD (100 tasks) solves these categories by providing teacher supervision regardless of the student's initial success rate.
- **Compositional failure modes are complementary.** RLVR fails where reward is sparse; KD fails where data is insufficient. A staged KD→RLVR pipeline that bootstraps all categories to a working baseline before applying self-play is the natural synthesis.

2. The Prism Framework

Task interface. Every task encodes a single observe-act cycle: the model observes a broken pipeline state and must produce one corrective action. A structured prompt presents the current state (file contents, stack traces, test failures, or API payloads), a precise success criterion, and a single action instruction. The reward function executes the action in a sandboxed namespace and returns a weighted partial-

[1]Turi labs. Correspondence to: Akshaykumar <akshayk@turilabs.in>, Tadisetty Sai Yashwanth <sai@turilabs.in>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

credit score:

$$r = \sum_k w_k \cdot \mathbf{1}[\text{condition}_k], \quad \sum_k w_k = 1.$$

This setup enforces outcome-verifiable evaluation: no rubric, no LLM judge, no string matching. A partial-credit structure (e.g., 0.4 on the specific broken case, 0.3 on regression, 0.3 on an edge case) gives GRPO a gradient signal that distinguishes "almost correct" from "completely wrong."

RLVR pathway. Prism trains with GRPO (Shao et al., 2024): for each step, $G = 4$ rollouts are generated from the current policy and scored via the verifiable reward. Group-normalized advantages:

$$A_i = \frac{r_i - \mu(r_{1:G})}{\sigma(r_{1:G}) + \delta}$$

drive a clipped importance-sampling update. No reward model is trained; the reward function *is* the specification.

KD pathway. A teacher model (GPT-4o) is prompted to briefly explain the underlying fault in 2–4 sentences and then produce corrected code, providing a chain-of-thought rationale that signals *why* the fix is correct. Teacher traces are scored with the same reward functions used in RLVR and filtered at score ≥ 0.5 (≈80% acceptance rate across runs). The student is then trained via standard next-token SFT loss on the filtered traces.

The two pathways share the same `Task` abstraction, enabling controlled comparison under identical tasks and scoring logic.

3. Benchmark

3.1. Task Categories and Compositional Structure

We construct 100 training tasks and 10 held-out evaluation tasks across four categories (Table 1), each corresponding to a common agentic pipeline stage. The categories differ substantially in their degree of compositional structure.

Table 1. Task distribution across training versions and held-out set.

Category	Train v1	Train v2	Held-out
Bug fix	15	28	3
Error recovery	11	23	2
Multi-file feature	12	29	3
Tool/API integration	12	20	2
Total	**50**	**100**	**10**

Error recovery tasks present functions that crash on edge cases; the model must add exception handling without breaking the happy path. This category is *highly compositional*: the skill "wrap in try/except and return a sensible default" is a reusable module that transfers across any function signature and exception type.

Multi-file feature tasks present two related files; the model must add a feature that spans both. OOP patterns (add a method, update the corresponding interface) are compositionally structured: the model learns a schema for cross-file edits that applies broadly.

Bug fix tasks present a broken function and failing tests. Compositionality is *partial*: some bug types (e.g., mutable defaults, off-by-ones) are pattern-like and transfer; others require case-specific reading and understanding of specific code logic.

Tool/API integration tasks present a broken API client; the model must fix a specific failure mode. This category is *low compositional density* from the perspective of RLVR: the base model's pass@1 is ≈8%, producing near-zero reward groups and sparse gradient signal. The underlying skills (HTTP status handling, pagination, response parsing) do have reusable structure, but RLVR cannot discover that structure without an initial foothold.

3.2. Reward Design

All reward functions follow a four-step pipeline: extract code (strip markdown fences), execute in an isolated namespace, call the function against test inputs, and return the weighted partial-credit score. For API tasks, `unittest.mock.MagicMock` simulates HTTP responses, avoiding real network calls. Timeout guards via `SIGALRM` terminate non-terminating code paths, directly relevant for bug-fix tasks where the bug itself may be an infinite loop.

4. Experiments and Results

4.1. RLVR: Compositional Skill Acquisition

We train `meta-llama/Llama-3.1-8B` with LoRA rank 16, group size 4, learning rate 2×10^{-5}, KL penalty 0.0, for 50 steps (one epoch over 50 tasks).

Training dynamics. Reward increases from 0.25 to 0.50 between steps 33 and 46. `frac_mixed` $= 1.0$ throughout, confirming that GRPO always received an informative advantage signal. KL divergence decreases from 0.0022 to 0.0009 and entropy from 1.56 to 1.32, indicating stable convergence. Trajectory inspection at step 46 reveals the character of learning: on a tool/API task, 3 of 4 rollouts attempt the correct structural approach; failures are syntax errors (missing colons) and undefined variables, not conceptual errors. The model has acquired the *compositional intent*; training refines execution precision.

Held-out results. Table 2 shows held-out scores. RLVR achieves perfect generalization on error-recovery (0.000→1.000) and multi-file feature (0.667→1.000), the two categories with the most compositional structure. Once the model acquires "wrap in try/except" or "add method and update interface," the skill composes with any new function or codebase. Bug-fix generalizes partially (0.000→0.267), consistent with its mixed compositional character. Tool/API integration remains at 0.000 held-out: the base model's near-zero success rate means GRPO groups collapse to zero advantage, producing no parameter update.

4.2. Knowledge Distillation: Data-Driven Compositional Bootstrapping

KD-50 (40 filtered traces, 5 epochs) yields near-zero held-out improvement (+0.020 overall). Approximately 200 gradient steps are insufficient to shift an 8B model's weights toward compositional generalization.

KD-100 (≈80 filtered traces, 5 epochs) yields the strongest result across all methods: +0.620 overall, including 1.000 on tool/API integration, the exact category where RLVR scores 0.000. KD bypasses the sparse-reward barrier because teacher supervision is always available; the student does not need to discover correct API patterns via self-play. The chain-of-thought explanations in teacher traces provide compositional signal: not just the correct code, but the reasoning structure behind it.

The scaling gap between KD-50 (+0.020) and KD-100 (+0.620) is substantial. Doubling training traces from 40 to 80 roughly doubles gradient coverage, crossing a threshold for compositional generalization to emerge.

4.3. Comparative Analysis

The compositional failure modes are complementary (Table 2). RLVR excels where reward density enables self-play but is structurally blocked by near-zero reward groups. KD exceeds RLVR overall when sufficient traces are available, and uniquely solves the sparse-reward category by providing teacher supervision regardless of student competence. Neither method generalizes multi-file feature beyond the base model's 0.667 score, likely because the 3-task held-out set captures a single OOP pattern that the base model already handles.

The KD→RLVR recipe. These results motivate a staged compositional training strategy: (1) KD to bootstrap the model past zero on all categories, establishing compositional patterns from teacher demonstrations; (2) RLVR to refine from the KD checkpoint via reward-aligned self-play. This mirrors the warm-start strategy in DeepSeek-R1 (DeepSeek-AI, 2025) and is empirically justified by our complementarity finding: the KD checkpoint provides the initialized policy that RLVR needs to build on.

5. Discussion

Compositionality as a predictor of generalization. Our results show a clear correspondence between a category's compositional structure and its generalization behavior under RLVR. Error-recovery (highly compositional: one exception-handling pattern applies universally) generalizes perfectly from 11 training examples. Bug-fix (partially compositional: some patterns transfer, others don't) generalizes partially. Tool/API integration (compositionally rich but sparse in reward density) requires teacher bootstrapping to unlock. This suggests that task compositionality is a meaningful predictor of training method efficacy, a dimension that benchmark design should explicitly track.

Interpretability of learned skills. The training dynamics and trajectory analysis provide interpretable evidence of compositional learning. At step 46, the model's tool/API rollouts show that the intent is compositionally correct (use `if key not in resp.json()`) while execution is imprecise. This separation of compositional structure (what to do) from execution precision (how to do it syntactically) is itself evidence that RLVR acquires modular, interpretable skills.

Safety implications. Compositionally specialized SLMs fail *predictably*: a model trained on error-recovery patterns will apply those patterns to new functions reliably. This predictability is a safety property. By contrast, general frontier models may hallucinate plausible but incorrect solutions when applied outside their most common training contexts. Prism's verifiable reward design makes generalization failures directly observable during training, enabling controlled deployment under known reliability bounds.

Limitations. Tasks are single-turn; real agentic coding is iterative. The 10-task held-out set has high per-category variance (2–3 tasks each). All experiments use Llama-3.1-8B; behavior at 1B–3B or with different families (Qwen, Mistral) is unknown. A combined KD→RLVR run confirming the staged recipe is deferred.

6. Conclusion

We have studied compositional skill acquisition in agentic pipeline SLMs under RLVR and KD. RLVR acquires transferable compositional patterns in categories with sufficient reward density, achieving near-perfect held-out generalization on structured tasks from 50 training examples. KD bootstraps compositional knowledge from teacher demonstrations, uniquely solving the sparse-reward category that RLVR cannot crack. The key structural insight is that the two methods fail on complementary categories: sparse re-

Table 2. Held-out evaluation (10 unseen tasks). RLVR and KD fail on *complementary* categories: sparse rewards block RLVR; insufficient data blocks KD-50. KD-100 covers all categories.

Method	Overall	Bug Fix	Error Rec.	Multi-file	Tool/API
Base model	0.200	0.000	0.000	0.667	0.000
RLVR (50 tasks, 1 epoch)	0.580	0.267	1.000	1.000	0.000
KD (50 tasks, 5 epochs)	0.220	0.067	0.000	0.667	0.000
KD (100 tasks, 5 epochs)	**0.820**	**0.733**	**1.000**	**0.667**	**1.000**

wards block RLVR; insufficient data blocks KD. A staged KD→RLVR pipeline that bootstraps all categories before applying self-play refinement is the natural synthesis. Our benchmark, with task categories that differ explicitly in compositional structure, provides a reusable scaffold for studying compositional generalization in agentic settings, directly supporting the goals of this workshop.

References

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1–2): 3–71, 1988.

Kim, N. and Linzen, T. COGS: A compositional generalization challenge based on semantic interpretation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *Proceedings of the International Conference on Machine Learning*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the Conference on Robot Learning*, 2021.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

A. Additional Related Work

Compositional generalization benchmarks. SCAN (Lake & Baroni, 2018) and COGS (Kim & Linzen, 2020) established systematic tests for compositional generalization in sequence-to-sequence models. Our benchmark extends this spirit to executable agentic tasks, where compositionality is assessed via held-out generalization of learned *behavioral* patterns rather than linguistic rules.

Modular and skill-based agents. Prior work on modular architectures (Andreas et al., 2016; Shazeer et al., 2017) and skill-conditioned agents (Pertsch et al., 2021) studied compositional structure at the *architecture* level. Our work studies compositional generalization at the *training* level: does a monolithic SLM, trained on a set of agentic tasks, acquire compositionally structured skills that transfer to new instances?

RLHF and verifiable rewards. Standard RLHF (Ouyang et al., 2022) trains a reward model on human preferences, introducing a second learning problem. RLVR (DeepSeek-AI, 2025) replaces this with deterministic task checks, which in our setting are executable reward functions. This choice is particularly natural for studying compositional generalization: verifiable rewards make the generalization criterion precise and observable.

B. LLM/Agent Usage Disclosure

Experimental system (subject of study). GPT-4o was used as the teacher model in all KD experiments. Its outputs were scored with the same verifiable reward functions used in RLVR and filtered at score ≥ 0.5. This is the direct subject of investigation, not an auxiliary tool.

Benchmark construction. Training and evaluation tasks were written by the authors. LLM assistance was used for drafting initial task descriptions and reward function templates, subsequently reviewed and verified manually. No task or reward function was accepted without human verification of all test assertions.

Writing assistance. LLM tools were used for grammar checking and phrasing suggestions. All technical content, experimental design, result interpretation, and conclusions are the authors' own.