RL UNLOCKS COMPOSITIONAL REASONING VIA SOFT TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) acquire reasoning capabilities in post-training via methods such as supervised fine-tuning (SFT) on demonstrations and reinforcement learning (RL) with verifiable rewards, but continuously extending these capabilities without forgetting previously acquired ones remains challenging. Skill neologisms (SNs) have recently been proposed as a way to learn new skills in modular fashion via semantically integrated soft tokens. However, prior work has demonstrated SNs only for direct input–output skills, whereas soft token methods are known to struggle with multi-step reasoning. It therefore remains unclear whether SNs can encode reasoning procedures that compose with other capabilities as effectively as skills internalized in model weights. We study this question by controlling which reasoning skills are available to a base model, training new skills via SNs and weight-based fine-tuning (FT), and evaluating their composition abilities with in- and out-of-distribution skills. Beyond preventing forgetting of existing skills by construction, we find that the effectiveness of reasoning SNs depends strongly on both model capability and training method. Under SFT, more capable models increasingly close the gap to internalized skills on in-distribution compositions, but remain weak on out-of-distribution composition. Training SNs with RL significantly improves performance, matching or exceeding internalized skills on in- and out-of-distribution compositions, and unlocking composition between independently learned SNs. We extend our findings to a medical reasoning task, where RL-trained neologisms significantly improve transfer to OOD compositions over SFT. Our results show that RL-trained skill neologisms are a promising path for continually extending the reasoning capabilities of strong LLMs.

1 INTRODUCTION

Large language models (LLMs) have shown the ability to acquire new reasoning capabilities through various methods such as supervised fine-tuning (SFT) from demonstrations and reinforcement learning (RL) with verifiable rewards (Shao et al., 2024; Guo et al., 2025a). However, continuously acquiring new reasoning abilities remains an open challenge (Zheng et al., 2025b). Though RL fine-tuning has generally been found to be more robust to forgetting than SFT (Chen et al., 2025; Shenfeld et al., 2026b), it requires atomic skills to be present in the base model (Yuan et al., 2026), and still incurs forgetting of previous tasks (Guo et al., 2025c; Li et al., 2025; Lai et al., 2025; Shenfeld et al., 2026b). Continually extending a model's reasoning skills therefore requires acquiring new, composable capabilities without disrupting existing ones.

Skill neologisms (SNs) (Berthon et al., 2026) have recently been proposed as an alternative way to teach LLMs new skills without weight updates, by learning semantically integrated soft tokens on a training set focused on the skill of interest. Prior work demonstrated this approach for direct input–output skills. However, reasoning skills require the learned representation to guide a multi-step computation, and soft token methods have been found to struggle with chain-of-thought reasoning (Fan et al., 2025). It therefore remains unclear whether SNs can encode generally composable reasoning skills, and whether they can approach the performance of skills internalized in model weights.

Studying this question requires clear definitions of atomic and compositional reasoning skills, and the ability to control precisely which skills are available to the base model. This is difficult in practice, as

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.

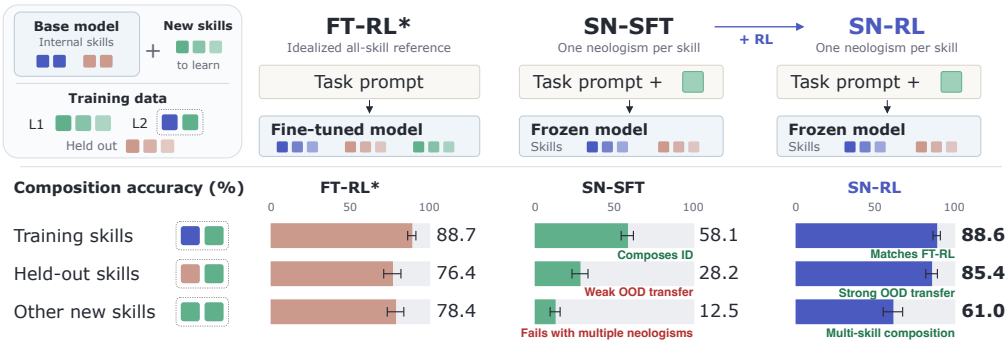

Figure 1: **Overview of our study and key findings**.(Top) New skills are learned alongside internal skills via fine-tuning or skill neologisms. *FT-RL is an idealized baseline initialized by jointly fine-tuning on all 25 atomic skills before compositional RL. (Bottom) RL-trained neologisms substantially improve composition over SFT, matching or exceeding the fully internalized reference on internal skills and unlocking inter-neologism composition.

stronger models will generally have a greater range of existing skills. We therefore use the controlled string-manipulation framework introduced by (Yuan et al., 2026), which defines 25 atomic skills as non-decomposable string transformations instantiated as Python functions. Skill compositions are defined as nested function calls, and anonymized skill identifiers (e.g., `func_0`) are used to prevent contamination of skills before they have been formally learned by the model.

As a basis for our analysis, we construct models with different levels of capability using Qwen2.5 models (Qwen et al., 2025) from 0.5B to 7B parameters and Qwen3.5 models (Yang et al., 2025) at 4B and 9B. Using a fixed partition across all models, we train on 15 skills and their compositions through SFT and RL, and learn the remaining 10 skills either as independently learned SNs or through in-weight fine-tuning. This design lets us measure atomic skill mastery and composition with training-time internal skills, held-out internal skills, and independently acquired new skills.

Figure 1 summarizes our experimental framework and main findings. Our contributions are:

- We provide a controlled study of whether soft tokens can encode composable reasoning skills, while preserving existing capabilities by keeping the backbone frozen.
- We show that SFT-trained SNs become increasingly competitive with weight-based skills as model capability grows, but remain weak on out-of-distribution composition.
- We find that RL substantially improves SN composability: on Qwen3.5-9B, SN-RL matches or exceeds an idealized FT-RL reference on compositions with internal skills and unlocks composition between independently learned SNs.
- We demonstrate transfer beyond the synthetic setting: on MedCalc-Bench, RL-training of SNs significantly improves performance on OOD composition over SFT.

Together, our results show that skill neologisms are a promising approach for learning new reasoning skills in a scalable, modular fashion, but that their composability depends critically on both the capability of the frozen backbone and the learning algorithm.

2 Background

Fine-tuning LLMs for reasoning Fine-tuning LLMs for reasoning tasks has gathered a lot of interest since recent works (Guo et al., 2025a; Pan et al., 2025; Zeng et al., 2025) demonstrated that pretrained models could autonomously develop reasoning capabilities via reinforcement learning with verifiable rewards (RLVR) or distillation (i.e. SFT on high-quality reasoning traces). While part of the community argues that RL merely amplifies reasoning capabilities already present in the base model (Yue et al., 2026; Gandhi et al., 2025; Liu et al., 2025), Yuan et al. (2026) demonstrated that RL fine-tuning teaches LLMs genuinely new skills by composing existing ones. Our work focuses

on the natural follow-up question: how can a model's set of atomic reasoning skills—the building blocks for such compositions—be expanded over time without disrupting the ones it already has?

Continual learning for LLMs Continual learning refers to the ability to train on new tasks while avoiding *catastrophic forgetting*, where previous model capabilities are lost (McCloskey & Cohen, 1989; French, 1999; Kirkpatrick et al., 2017). For LLMs specifically, continual learning has been an active field of research in recent years (Wu et al., 2024; Luo et al., 2025a; Shi et al., 2025; Guo et al., 2025b). Existing approaches are generally split between architecture-based methods (Madotto et al., 2021; Wang et al., 2023; Ren et al., 2024; Zheng et al., 2025a), which directly intervene on the model to prevent forgetting; regularization-based methods (Ke et al., 2022; Jiang et al., 2025; Luo et al., 2025b), which use more robust learning objectives; and replay-based methods (Scialom et al., 2022; Wang et al., 2024), which periodically re-expose the model to previous data to mitigate forgetting. In the LLM reasoning literature, on-policy methods such as RL fine-tuning have emerged as a powerful regularization-based approach, with many works finding that RL suffers from significantly less forgetting than SFT (Chu et al., 2025; Shenfeld et al., 2026a;b). Shenfeld et al. (2026b) argue that the degree of forgetting is closely related to the KL-divergence between the fine-tuned and base model evaluated on the new task, and that on-policy RL fine-tuning is implicitly biased toward KL-minimal solutions.

Skill neologisms Rather than mitigating forgetting via architectural, regularization-based, or replay-based methods, skill neologisms (Berthon et al., 2026) aim to extend a model's capabilities by learning soft tokens that are semantically integrated in the prompt and targeted at specific procedural knowledge. Leveraging the ability of pretrained LLMs to flexibly compose skills (Arora & Goyal, 2023; Yu et al., 2024), skill neologisms are trained similarly to prompt tuning (Lester et al., 2021) on datasets centered on a target skill but mixed with other skills already held by the model, inducing a representation that composes naturally with existing capabilities. In contrast to standard prompt tuning, semantically positioned soft tokens allow independently learned neologisms to compose zero-shot. However, soft prompts have been shown to struggle in multi-step reasoning (Fan et al., 2025), so it is unclear if the extend to which the composition ability of soft prompts extends reasoning settings. We investigate how reasoning composability compares with internalized skills with a controlled study varying model scale and training methods. We find that standard SFT-based neologisms largely fail to compose OOD, and that both more capable backbones and RL-training are needed to unlock the composition abilities of soft tokens in reasoning settings.

3 Research Framework

3.1 String Manipulation Environment

Investigating whether skill neologisms can acquire new reasoning skills requires precise control over which reasoning skills the model has, whether any new skills are genuinely new or already known to the model, and how composition between skills is defined and evaluated. Realistic reasoning benchmarks rarely satisfy all three: skills are fuzzy and entangled, contamination from pretraining is common, and "compositional generalization" is usually defined ad hoc. We therefore adopt the synthetic string manipulation testbed of Yuan et al. (2026), which defines 25 atomic skills as deterministic Python functions over strings, and provides three properties that map to our study's needs: each skill is a non-decomposable Python function with deterministic behavior, giving an unambiguous notion of skill mastery and a trivial exact-match reward signal; skills compose naturally as nested function calls (we refer to a depth-k composition as a Level-k or Lk query); and anonymized identifiers (`func_0`, ..., `func_24`) prevent the model from inferring functionality from naming, so a skill can only be mastered if explicitly trained.

Skill partition. We partition the 25 functions into 15 *internal* (used in pretraining) and 10 *new* (targets of skill acquisition). The 15 internal functions further split into 10 *internal-train*, used as composition partners when training new skills, and 5 *internal-holdout*, withheld from all skill acquisition training to test composition with unseen internal skills at evaluation. Exact function assignments are in Appendix C.

3.2 Pretrained Base Models

Our experiments require base models that have internalized a known set of reasoning skills as well as the ability to compose them. Following the two-stage protocol of Yuan et al. (2026), we construct such models by fine-tuning Qwen2.5-Instruct (Qwen et al., 2025) (0.5B, 1.5B, 3B, 7B) and Qwen3.5-Instruct (4B, 9B) models. **Stage 1** performs SFT on correct L1 reasoning traces over the 15 internal functions, sampled from Qwen2.5-14B-Instruct, internalizing each atomic skill in model weights. **Stage 2** train with GRPO with binary correctness rewards on a mix of L1 and L2 queries over the same 15 internal functions, teaching the model to compose them. Table C2 confirms that the resulting models substantially master the internal skills both atomically (L1) and compositionally (L2), while remaining near zero on new functions. We additionally train a Qwen2.5-14B-Instruct model with the same protocol and use it to generate L1 and L2 demonstrations for skill acquisition (Sections 4–6); full hyperparameters are in Appendix C.1.

3.3 Compositional Generalization Conditions

Pretrained LLMs are known to flexibly compose their skills, including in combinations not seen during training (Arora & Goyal, 2023; Yu et al., 2024; Yuan et al., 2026). In order to measure the extend to which skills learn via SN or FT show this property, we define three Level-2 metrics: τ_{ID} on new$\times$internal-train pairs (compositions seen in training); $\tau_{\text{OOD}-\text{ho}}$ on new$\times$internal-holdout pairs (composition with held-out internal skills); and $\tau_{\text{OOD}-\text{new}}$ on new$\times$new pairs (inter-neologism compositions).

Controlling for atomic skill mastery. L2 accuracy on a pair is upper-bounded by L1 accuracy on its constituent skills, so direct τ comparisons across methods can be confounded by differences in atomic skill mastery. To isolate models' composition abilities, we additionally report each τ filtered to pairs where both functions exceed $p_{L_1} > 0.6$ under the same model and method, and confirm in Appendix A.3 that the qualitative results are robust to variations of this threshold.

3.4 Skill Acquisition Methods

We compare two families of methods. *Skill neologisms* (SN) learn one set of soft-token embeddings per new function with all base model weights frozen, following Berthon et al. (2026). We set neologism lengths to $l = 50$ and initialize soft tokens randomly (Gaussian, $\sigma = 0.2$). At training and inference, the soft tokens are inserted at the start of the prompt under a small header listing the neologisms used in the query (e.g. "`- func_3: <|func_3-0|>...<|func_3-49|>`"), keeping them semantically integrated with the rest of the prompt. We mask the soft tokens out of the next-token distribution during generation, as unmasked decoding occasionally produces soft tokens in the CoT and destabilizes training. *Fine-tuning* (FT) baselines instead update model weights via LoRA adapters ($r = \alpha = 32$). Table C1 summarizes the model configurations used across our experiments (full hyperparameters shown in Appendix C.3).

4 Motivation: Learning New Skills Without Forgetting

We start by investigating our first research question (**RQ1**): *can a model acquire new atomic reasoning skills via skill neologisms while preserving previously internalized ones?* We train one skill neologism per new skill via SFT on 5,000 Level-1 demonstrations, with each of the 10 neologisms trained independently while the model remains frozen (SN-SFT$_{\text{L1}}$). We compare against a LoRA adapter jointly fine-tuned on the same 50k demonstrations (FT-SFT$_{\text{L1}}$), evaluating acquisition and retention across all 25 skills.

Figure 2 shows the results across methods and model sizes. FT-SFT$_{\text{L1}}$ successfully learns the new skills, but at the cost of catastrophic forgetting of previously internalized ones. SN-SFT$_{\text{L1}}$ retains all of the pretrained model's performance on internalized skills by construction, while approaching the fine-tuned baseline on new skills. Full experimental details, per-skill results, and additional per-skill LoRA and continual learning experiments are provided in Appendices A.1 and A.2.

Skill neologisms provide a way to learn new skills in scalable fashion without risking forgetting of internalized ones, but how does their composability compare to that of internalized skills?



Figure 2: **L1 accuracy on internal and new skills after SFT training on L1 demonstrations of new skills only.** Fine-tuning (FT-SFT) learns new skills but forgets internalized ones; skill neologisms (SN-SFT) preserve internalized accuracy while approaching fine-tuning on new skills, particularly in stronger models. Error bars show 95% confidence intervals. We omit the "L1" suffix in model labels.

5 To what extent can reasoning skills learned by neologism compose with other skills?

An essential property of internalized reasoning skills is that they compose flexibly with other skills, even when they have never been trained jointly (Yuan et al., 2026). This naturally leads to our second research question (**RQ2**): *how well can atomic reasoning skills learned via neologisms compose with other skills compared to internalized skills?*

Setup For each new skill, we collect 5,000 correct reasoning demonstrations for both Level 1 and 2 queries, where Level 2 queries combine the new skill with internal-train skills. We train skill neologisms for each skill via SFT on these demonstrations and refer to the resulting model as **SN-SFT**$_{\text{L1+2}}$.

Baselines To investigate composition abilities of neologisms, we would ideally compare against new skills internalized in the model weights via SFT+RL, similarly to pretraining. However, the forgetting observed in §4 means that SFT on the new skills' demonstrations alone would significantly degrade performance on internal skills. We therefore consider the following two baselines:

- **FT-SFT**$_{\text{L1+2}}$: Fine-tune a LoRA adapter with SFT on the same L1+2 demonstrations used for SN-SFT$_{\text{L1+2}}$, mixing new skills with internal-train skills.
- **FT-RL**$_{\text{L1+2}}$: We first train with SFT on 5k L1 demonstrations for **all 25 skills**, providing initial knowledge of all atomic skills simultaneously. We then train with GRPO on the same L1+2 data as the other models. Since this requires prior L1 training on all skills simultaneously, this is not strictly comparable to neologisms and FT-SFT$_{\text{L1+2}}$ (where new skills are learned entirely independently from internal-holdout skills). Instead, it constitutes an idealized baseline for the ID and OOD composition abilities of new skills if they were internalized via RL fine-tuning.

Evaluation We evaluate L2 composition accuracy across the three conditions of §3.3 over 11,700 L2 samples comprising at least one new skill ($\sim$30 samples per function pair). Given weaker models do not manage to learn some atomic skills (see Figure A1), we control for L1 performance by additionally reporting L2 accuracy over functions that satisfy $p_{L_1} > 0.6$. A sensitivity analysis of the threshold is provided in Figure A3 and confirms that results are qualitatively consistent across thresholds.

Results Figure 3 shows the results across methods, model sizes, and composition types.

- **More capable backbones narrow the SFT composition gap on training-distribution pairs.** On mastery-controlled new$\times$internal-train pairs, SN-SFT rises from near-zero performance on the smallest Qwen2.5 models to 61.7% on Qwen2.5-7B, 63.7% on Qwen3.5-4B, and 70.4% on Qwen3.5-9B. Qwen3.5-4B outperforming Qwen2.5-7B despite its smaller parameter count indicates that this trend reflects backbone capability rather than parameter count alone.
- **SFT-trained SNs generalize to held-out internal skills, but a substantial OOD gap remains.** SN-SFT$_{\text{L1+2}}$ composes with internal-holdout skills that never appear during new-skill training. In contrast, FT-SFT$_{\text{L1+2}}$ achieves near-zero raw accuracy because in-weight SFT substantially degrades these held-out skills. Nevertheless, SN-SFT remains substantially weaker on held-out than training-time internal skills, and this OOD accuracy does not improve monotonically across

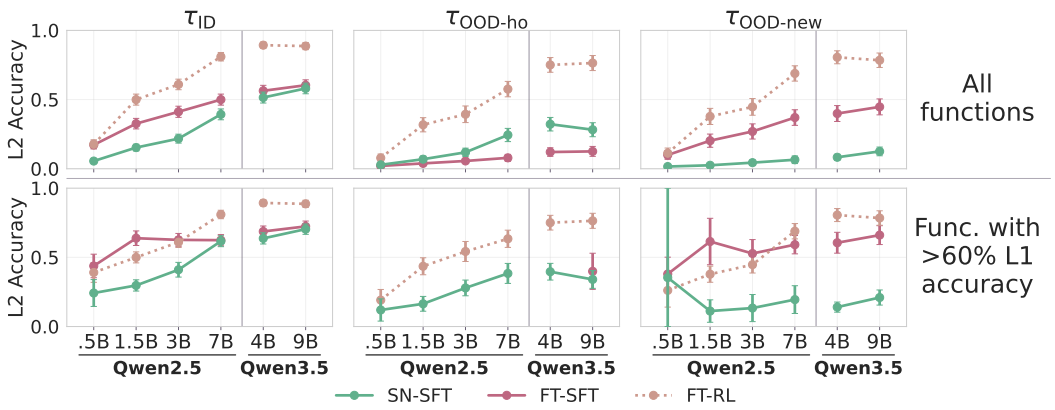

Figure 3: **Level 2 accuracy on ID and OOD skill combinations.** Accuracy is computed per function pair and averaged within each pair type. Top row: all function pairs. Bottom row: only function pairs where both atomic skills satisfy $p_{L_1} > 0.6$ under the same model and method, controlling for atomic skill mastery. Note that FT-SFT$_{\text{L1+2}}$ disappears from $\tau_{\text{OOD}-\text{ho}}$ in the bottom row because all internal-holdout functions fail the mastery threshold. Error bars show 95% Student's t intervals over pair-level accuracies. We omit the "L1+2" suffix in model labels.

the strongest backbones. Stronger models therefore narrow the in-distribution gap, but do not by themselves solve OOD composition.

- **Inter-neologism composition is limited.** Compositions between two skill neologisms does not exceed ∼15–20% across models, while fine-tuned models achieve ∼55–70%. This indicates that base models struggle to compose multiple SNs for reasoning when learned SFT, even for stronger backbones.

Takeaway

Skill neologisms learn reasoning skills that compose with internalized skills both in-distribution and out-of-distribution. On new×internal-train pairs, the gap to weight-based skill internalization disappears with more capable models.

6 RL Unlocks Inter-Neologism Composition

Section 5 demonstrated that skill neologisms can learn reasoning skills that compose with internalized skills, but also showed that a performance gap remains compared to fully-internalized skills. We also see with our FT-SFT$_{\text{L1+2}}$ and FT-RL$_{\text{L1+2}}$ baselines that RL seems to produce more generalizable skills than SFT, which has been noted in several recent works (Chu et al., 2025; Chen et al., 2025; Shenfeld et al., 2026b; Yuan et al., 2026). This motivates our third research question (**RQ3**): *does training with on-policy methods like RL also benefit skill neologisms?*

Setup We further train skill neologisms initialized from SN-SFT$_{\text{L1+2}}$ using GRPO (Shao et al., 2024) on the same L1+2 query distribution as previously, with binary rewards based on output correctness. The resulting model is **SN-RL$_{\text{L1+2}}$**. We focus on the Qwen3.5-9B model, for which training succeeds for all 10 new skills [1]. We compare against SN-SFT$_{\text{L1+2}}$ and FT-RL$_{\text{L1+2}}$ on the same L2 test set as in §5.

Results Figure 4 shows the L2 accuracy of SN-RL$_{\text{L1+2}}$, SN-SFT$_{\text{L1+2}}$, and FT-RL$_{\text{L1+2}}$ across the three composition conditions. Training neologisms with RL significantly improves performance compared to SFT, with accuracy gains of $+30.6\%$ for new×internal-train pairs and $+57.2\%$ for new×internal-heldout pairs, matching or exceeding FT-RL$_{\text{L1+2}}$ on new×internal pairs. Notably, RL

[1] RL training on Qwen2.5 ≤3B models generally resulted in unstable training and reward collapse. On Qwen2.5-7B, training succeeded for 6 out of 10 new skills. We discuss this further in Section 8.

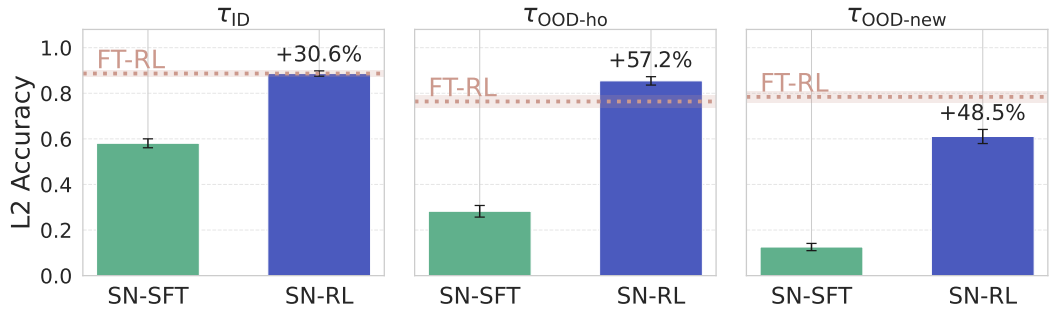

Figure 4: **Effect of learning algorithm on Level 2 accuracy.** Performance of skill neologisms trained via SFT (SN-SFT) and RL (SN-RL) compared to the idealized RL-fine-tuned baseline (FT-RL), for Qwen3.5-9B. All methods are trained on the same L1+2 query distribution. RL training substantially improves SN composition, with SN-RL matching the FT-RL baseline on τ_{ID} and $\tau_{\text{OOD-ho}}$, and approaching it for $\tau_{\text{OOD-new}}$. We omit the "L1+2" suffix in model labels. Error bars show 95% confidence intervals.

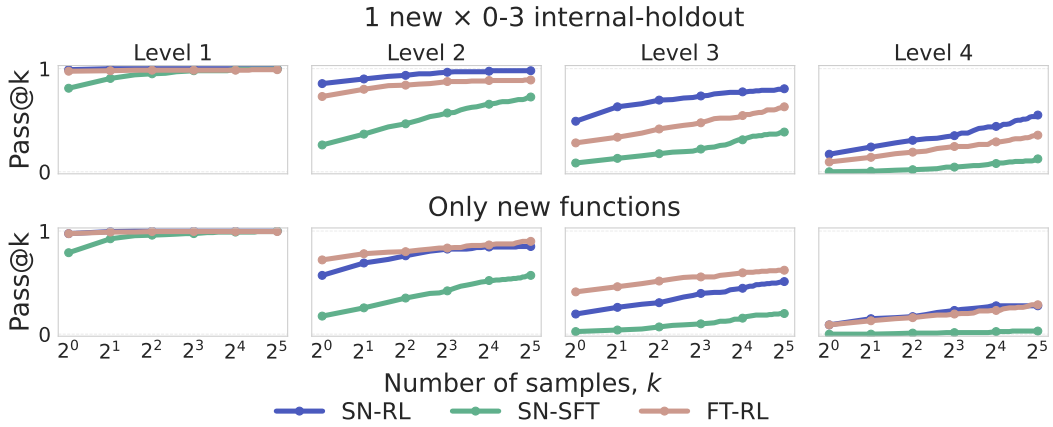

Figure 5: **Pass@k accuracy by composition depth** for Qwen3.5-9B on new×internal-holdout (top) and new-only compositions (bottom).

training unlocks performance on new×new pairs with a $+48.5\%$ gain over SFT-trained neologisms, and approaching the FT-RL$_{\text{L1+2}}$ baseline.

Pass@k analysis across composition depths. We further evaluate whether the SN-RL advantage persists with additional samples and at greater composition depth. We use two 800-example test sets spanning Levels 1–4: one with exactly one new skill composed with zero to three internal-holdout skills, while the other contains only new skills. Each set contains 200 examples per level.

The results are shown in Figure 5. SN-RL consistently outperforms SN-SFT across sampling budgets and composition depths. On compositions with internal-holdout skills, it also outperforms FT-RL across Levels 2–4. On new-only compositions, SN-RL substantially narrows the gap to FT-RL and reaches comparable pass@k at Level 2 and 4, while SN-SFT remains near zero at Levels 3–4. These results show that RL-trained neologisms generalize significantly better than SFT-trained ones, and rivals the composition abilities of fully internalized skills. Furthermore, they suggest that RL training is critical to unlock the ability to compose multiple independently learned neologisms zero-shot.

Takeaway

RL training substantially improves the OOD compositional generalization of skill neologisms, unlocking the ability to combine multiple neologisms zero-shot, and rivaling the composition performance of fully internalized skills.

6.1 How do RL-trained neologisms differ?

To understand why RL-trained neologisms compose more successfully than their SFT-trained counterparts, we analyze the failure modes of L2 compositions and investigate the attention dynamics towards soft tokens over the course of a reasoning trace.

Failure mode analysis We decompose the accuracy of Level-2 nested compositions by distinguishing the inner accuracy from *follow-through* accuracy (Table 1; details in Appendix A.4). SN-SFT errors are more often due to follow-through errors: on new×new compositions, follow-through accuracy is only 28.5% compared to 44.7% inner accuracy.

Table 1: **Intermediate accuracy and follow-through on Qwen3.5-9B.** Each cell reports follow-through; the gray annotation gives joint inner-and-final accuracy / inner accuracy (%).

Method	New×internal		New×new
	Inner	Outer	
SN-SFT	70.9 = $\frac{50.6}{71.4}$	59.4 = $\frac{46.0}{77.5}$	28.5 = $\frac{12.8}{44.7}$
SN-RL	**94.3** = $\frac{90.8}{96.3}$	**89.0** = $\frac{85.9}{96.5}$	**74.2** = $\frac{60.3}{81.3}$

Comparing new×new with new×internal compositions suggests that the presence of a second neologism interferes with both stages of computation. Under SN-SFT, inner accuracy decreases from 71.4–77.5% with one neologism to 44.7% with two, while follow-through accuracy drops even more sharply from 59.4–70.9% to 28.5%. SN-RL largely mitigates this interference, increasing inner accuracy on new×new compositions from 44.7% to 81.3% and follow-through accuracy from 28.5% to 74.2%. This suggests that RL makes neologisms more robust to the presence of other neologisms, improving both the computation of intermediate results and their subsequent use by the outer skill.

Attention dynamics Prior work have observed that soft prompts can derail reasoning by remaining active in the attention of tokens throughout the trajectory Fan et al. (2025). We investigate whether a similar phenomenon could explain the interference between multiple neologisms observed in the previous section.

We collect rollouts from 35 prompts each from newxinternal-holdout and newxnew queries under both SN-RL and SN-SFT. We then run both model on all rollouts and compute the share of attention directed to each neologism across the completion. Finally, we compute ratio between the highest-attended 10% of positions divided by the mean over the complete rollouts, to measure how diffuse or concentrated is the attention to neologisms across the completion (details in Appendix A.5).

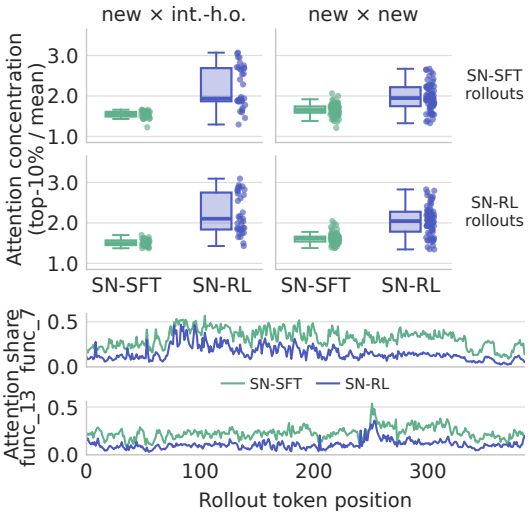

Figure 6: **RL-trained neologisms receive more temporally concentrated attention under matched trajectories.** Top: attention concentration on matched rollouts from newxinternal-holdout and newxnew queries. Bottom: attention shares for two neologisms across one rollout.

Figure 6 shows that, RL-trained neologisms receive more temporally concentrated attention under both trajectory sources and in both composition types. As illustrated in the bottom panel, SN-SFT neologisms tend to remain active across the entire reasoning trace. In contrast, RL-learned neologisms are attended too more sharply at specific positions of the trajectory, corresponding to where the underlying skill is active (see Figure A4).

7 Transfer to Medical Calculation Tasks

To study the broader applicability of these insights beyond a controlled synthetic setting, we experiment on MedCalc-Bench (Khandekar et al., 2024), a benchmark of real-world medical calculation tasks from patient narratives. Its tasks involve identifiable procedures, both standalone and as components of more complex calculations, providing a natural testbed for skill acquisition and compositional transfer.

We evaluate Qwen3.5-4B on the benchmark and select two poorly mastered procedures with sufficient training and evaluation data: *delta gap* and *adjusted body weight*, with base accuracies of 21.3% and 32.5%, respectively. Keeping the backbone frozen, we independently learn one neologism per skill using SFT, then refine it with RL, both on atomic examples only. We evaluate these skills both standalone and in untrained downstream calculations: delta ratios and albumin-corrected variants for delta gap, and the adjusted-weight branch of Cockcroft–Gault for body weight. At evaluation, all methods use a maximum generation length of 1,024 tokens; if reasoning reaches this cap, we close the reasoning trace and force an answer with up to 128 additional tokens. Full experimental details are provided in Appendix B.

Table 2: **MedCalc-Bench accuracy (%).** Gray: question-level SEM. SFT and RL train on atomic examples only. "Albumin-corr."=Albumin-corrected.

	Base	SN-SFT	SN-RL
Trained atomic skills (L1)			
Delta gap	21.3 ±5.5	56.3 ±6.0	**85.0** ±6.1
Adjusted body weight	32.5 ±4.8	57.5 ±4.8	**97.5** ±1.7
Downstream compositions (L2+ analogue)			
Delta ratio	40.0 ±6.9	58.8 ±6.6	**77.5** ±5.1
Albumin-corr. delta gap	5.0 ±3.4	12.5 ±3.4	**56.3** ±5.4
Albumin-corr. delta ratio	5.0 ±2.3	18.8 ±4.8	**63.8** ±7.1
Cockcroft–Gault†	12.5 ±4.6	21.3 ±5.2	**46.3** ±7.8

Results. Table 2 shows that SFT improves both atomic skills, reaching 56.3% on delta gap and 57.5% on adjusted body weight. However, transfer remains uneven: while delta-ratio accuracy improves, SFT reaches only 12.5–18.8% on albumin-corrected calculations and 21.3% on Cockcroft–Gault. RL refinement substantially improves both atomic accuracy and downstream performance. The largest downstream gains occur on the two albumin-corrected tasks, which improve by 43.8 and 45.0 percentage points over SFT; Cockcroft–Gault improves by 25.0 points. While these end-to-end improvements may partly reflect stronger atomic competence, the substantial gains on untrained downstream calculations suggest that RL-trained neologisms support more effective out-of-distribution composition. This echoes our controlled findings in §6, providing evidence beyond the synthetic setting that RL unlocks better compositional generalization for reasoning soft tokens.

8 Discussion

Skill neologisms extend a model's capabilities in a modular fashion without the risk of overwriting existing skills. However, for reasoning skills, we find that their effectiveness depends critically on two factors: the capability of the underlying model and the learning algorithm. More capable models learn increasingly composable SNs, while RL substantially improves their out-of-distribution generalization and unlocks zero-shot composition between independently learned SNs.

Lester et al. (2021)'s seminal work found that prompt tuning grows more competitive with in-weight fine-tuning as models scale. Our results echo these findings and pushes them further: not only are more capable models more easily *tuned* via soft prompts, they also better *generalize* to out-of-distribution compositions. Moreover, while soft prompts have mostly been trained with SFT in the past, we find that training soft prompts with RL significantly improves their OOD generalization performance, and is critical to unlock zero-shot composition of independently learned soft tokens in our reasoning setting.

Limitations and Future Work. The string manipulation testbed is essential for cleanly defining atomic skills and composition, controlling for pretraining contamination, and providing exact-match rewards—but it is a controlled abstraction of real reasoning. Our MedCalc-Bench experiments provide evidence of downstream transfer beyond this abstraction, but cover only two procedures on one backbone; investigating generalization to less explicitly decomposable reasoning is an exiting direction for future work. Whether SN-based skill acquisition transfers to domains where reasoning skills are fuzzier and more entangled is an important open question. RL optimization remains sensitive to the backbone and training configuration. Runs on Qwen2.5 models at 3B and below frequently collapsed, and the Qwen2.5-7B protocol produced usable RL neologisms for only 6 of 10 skills. In contrast, training succeeded for all 10 skills on Qwen3.5-9B. We hypothesize this reflects the empirical fragility of RL fine-tuning at small scales observed in prior works (Zeng et al., 2025; Liu et al., 2025). Important future directions include methods for more robust inter-neologism composition

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementation, clean and reformat datasets, and provide feedback on the reporting and interpretation of results. We have not used generative AI tools to design the research methodology or experiments. Tasks involving the formulation or proof of mathematical claims are not applicable to this work. Additionally, we used generative AI tools to create and modify scientific figures and polish the manuscript's writing. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to improve understanding of pretrained model capabilities and limitations. Evaluation results should be interpreted within the scope of the tasks, models, and experimental settings studied; benchmark accuracy alone does not establish safety, fairness, or reliability in real-world applications. Potential ethical concerns include biases and privacy risks inherited from pretrained models and their training data, misuse of improved capabilities, and the environmental costs of computation. Any downstream deployment should therefore involve application-specific evaluation and safeguards, particularly where errors could disproportionately affect vulnerable or underrepresented groups.

REFERENCES

Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023.

Antonin Berthon, Nicolas Astorga, and Mihaela van der Schaar. Skill neologisms: Towards skill-based continual learning, 2026. URL https://arxiv.org/abs/2605.04970.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dYur3yabMj.

Sinan Fan, Liang Xie, Chen Shen, Ge Teng, Xiaosong Yuan, Xiaofeng Zhang, Chenxi Huang, Wenxiao Wang, Xiaofei He, and Jieping Ye. Improving complex reasoning with dynamic prompt corruption: A soft prompt optimization approach. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=h7Qz1ulnvF.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. In *COLM*, 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 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Da-Han Wang, et al. Continual learning for generative ai: From llms to mllms and beyond. *arXiv preprint arXiv:2506.13045*, 2025b.

Haiyun Guo, ZhiYan Hou, Yandu Sun, Jinghan He, Yu Chen, Yuzhe Zhou, Yuheng Jia, Jinqiao Wang, and Tat-Seng Chua. Mllm-ctbench: A benchmark for continual instruction tuning with reasoning process diagnosis. *arXiv preprint arXiv:2508.08275*, 2025c.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Gangwei Jiang, Caigao Jiang, Zhaoyi Li, Siqiao Xue, Jun Zhou, Linqi Song, Defu Lian, and Ying Wei. Unlocking the power of function vectors for characterizing and mitigating catastrophic forgetting in continual instruction tuning. *arXiv preprint arXiv:2502.11019*, 2025.

Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. Continual training of language models for few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10205–10216, 2022.

Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina S Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad W Safranek, Abid Anwar, Andrew Jiaxing Zhang, Aidan Gilson, Maxwell B Singer, Amisha D Dave, R. Andrew Taylor, Aidong Zhang, Qingyu Chen, and Zhiyong Lu. Medcalc-bench: Evaluating large language models for medical calculations. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=VXohja0vrQ.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, et al. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, 2025.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *EMNLP*, 2021.

Yuetai Li, Zhangchen Xu, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, Xiang Yue, and Radha Poovendran. Temporal sampling for forgotten reasoning in LLMs. In *2nd AI for Math Workshop @ ICML 2025*, 2025. URL https://openreview.net/forum?id=1medQAgr1g.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025a.

Zheheng Luo, Xin Zhang, Xiao Liu, Haoling Li, Yeyun Gong, Qi Chen, and Peng Cheng. Velocitune: A velocity-based dynamic domain reweighting method for continual pre-training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16644–16656, 2025b.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul A Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7452–7467, 2021.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*, 2024.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, 2022.

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026a.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=7HNRYT4V44.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 663–677, 2024.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-mix: a flexible and expandable family of evaluations for ai models. In *ICLR*, 2024.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From f(x) and g(x) to f(g(x)): LLMs learn new skills in RL by composing old ones. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jt7oCtYqHE.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=4OsgYD7em5.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun MA, and Junxian He. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=vSMCBUgrQj.

Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35, 2025b.

APPENDIX

A DETAILED EXPERIMENTAL RESULTS

A.1 ATOMIC SKILL ACQUISITION (SECTION 4)

Setup For each of the 10 new skills, we generate 5,000 correct Level-1 reasoning demonstrations using our 14B teacher model (§3.2). We train one skill neologism per new skill via SFT on its 5k demonstrations, with each neologism trained independently. We refer to the resulting model equipped with all 10 neologisms as **SN-SFT**$_{\text{L1}}$. Note that RL training requires atomic skills to already be available in the model (Yuan et al., 2026), so all methods in this experiment are trained via SFT.

Baseline We compare against **FT-SFT**$_{\text{L1}}$: a LoRA adapter jointly fine-tuned on all 50k L1 demonstrations across the 10 new skills.

Evaluation We evaluate L1 accuracy on 200 test queries per skill across all 25 skills, measuring both the acquisition of new skills and the retention of internalized ones.

Per-skill results Figure A1 expands the aggregate results of Figure 2 to the per-skill level, showing L1 accuracy on every function before and after training across all four model sizes. The pattern is consistent across skills: **FT-SFT**$_{\text{L1}}$ acquires most new skills but loses substantial accuracy on previously internalized ones, while **SN-SFT**$_{\text{L1}}$ preserves internalized accuracy by construction and approaches the fine-tuned baseline on new skills. Skill difficulty varies noticeably across functions; see, e.g., `func_7` and `func_20` which remain harder even for the fine-tuned model.

Independently trained per-skill LoRA. Since neologisms are trained on a per-skill basis, we additionally evaluate a separate LoRA adapter for each new skill on Qwen2.5-7B and Qwen2.5-1.5B, using the same 5k L1 demonstrations per skill. Table A1 reports internal L1 accuracy before training and the mean accuracy with each of the 10 independently trained adapters active, together with the mean and range of accuracy drops across adapters.

Table A1: **Internal skill retention with independently trained per-skill LoRA adapters.** Post-SFT accuracy and mean forgetting are averaged across the 10 adapters. The range gives the minimum and maximum drops in average internal-skill accuracy across adapters, in percentage points (pp).

Model	Pre-SFT internal L1 acc. (%)	Post-SFT-per-skill internal L1 acc. (%)	Mean forgetting (pp, [min, max])
Qwen2.5-7B	98.7	65.3	33.4 [5.2, 88.9]
Qwen2.5-1.5B	85.7	60.9	24.8 [7.0, 51.3]

On average, a LoRA adapter trained on one new skill reduces accuracy on atomic internal skills by 33.4pp and 24.8pp for Qwen2.5-7B and Qwen2.5-1.5B respectively, with drops of up to 88.9pp and 51.3pp for learning a single new skill. Thus, training adapters independently does not eliminate interference with internalized skills while an adapter is active. These drops do not imply irreversible loss from the frozen backbone: disabling the adapter restores the base model.

A.2 CONTINUAL LEARNING WITH NEOLOGISMS

Setup and evaluation We compare against **FT-SFT**$_{\text{L1-CL}}$: the same LoRA adapter as in FT-SFT$_{\text{L1}}$, but trained sequentially on each skill's 5k demonstrations, simulating a continual learning scenario. We report L1 accuracy relative to the FT-SFT$_{\text{L1}}$ baseline to account for skill difficulty.

Results Figure A2 shows that sequential fine-tuning compounds forgetting: FT-SFT$_{\text{L1-CL}}$ forgets each previously learned new skill upon training on the next. In contrast, since each neologism is trained independently while the model remains frozen, SN-SFT$_{\text{L1}}$ preserves all skills throughout the training sequence.

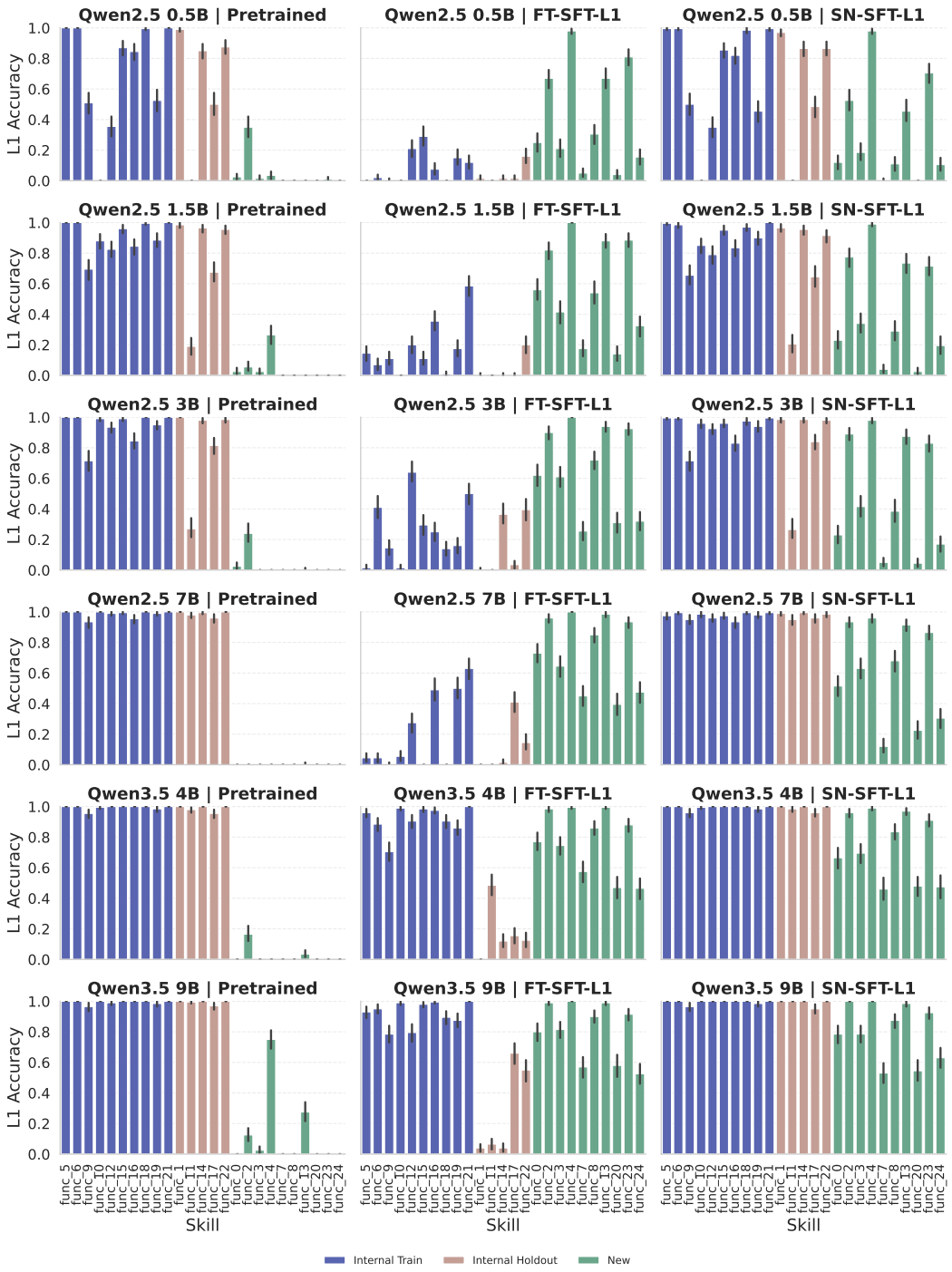

Figure A1: **Level-1 accuracy on all 25 skills before and after SFT training on L1 demonstrations of new skills, per model size.** Skills are grouped by type: internal-train (blue), internal-holdout (orange), and new (green). Fine-tuning successfully learns most new skills but incurs substantial forgetting of internalized ones; skill neologisms preserve all internalized performance by construction. Error bars show 95% confidence intervals ($N = 200$ per skill).

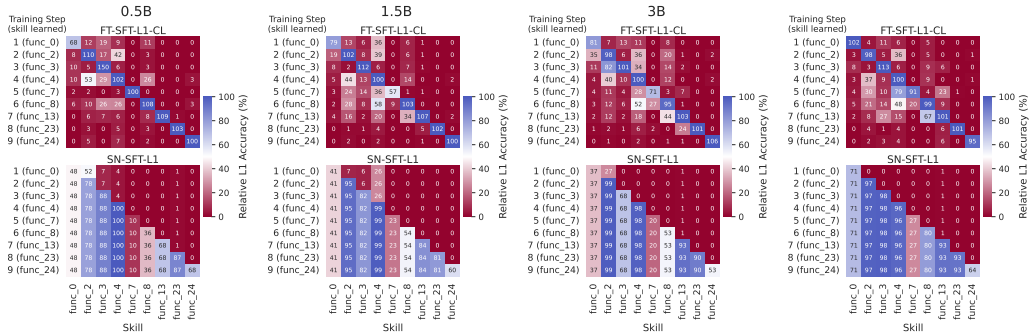

Figure A2: **Continual learning of new atomic skills on the Qwen2.5 0.5B, 1.5, 3B and 7B models (from left to right). Heatmaps show the L1 accuracy relative to the FT-SFT$_{\mathrm{L1}}$ baseline (trained jointly on all skills). New skills are learned one after the other via SFT on L1 demonstrations. The fine-tuned model forgets the previously learned skills, while neologisms scale by design.**

A.3 Compositional Generalization: p_{L_1} Threshold Sensitivity (Section 5)

To isolate compositional ability from differences in atomic skill mastery, the main text reports Level-2 accuracy filtered to pairs where both constituent skills satisfy $p_{L_1} > 0.6$. Figure A3 shows that all qualitative conclusions are robust to the choice of threshold across the range $\{0.5, 0.6, 0.7, 0.8\}$. Across every threshold: SN-SFT$_{\mathrm{L1+2}}$ generalizes on τ_{ID} and $\tau_{\mathrm{OOD\text{-}ho}}$ with a gap that shrinks with model scale; new×new compositions remain weak across all methods; and FT-SFT$_{\mathrm{L1+2}}$ disappears from the $\tau_{\mathrm{OOD\text{-}ho}}$ filtered view because internal-holdout functions fail the mastery threshold (the model forgets them during training on new skills).

A.4 Intermediate Accuracy and Follow-Through in Level-2 Compositions

For a nested composition $f_{\mathrm{outer}}(f_{\mathrm{inner}}(x))$, we estimate the intermediate accuracy by looking at how often the exact reference output of appears f_{inner} anywhere in the completion. Let I denote this event and F denote final-answer correctness. We report *inner accuracy*, $\Pr(I)$, and *follow-through accuracy*, the conditional probability that the final answer is correct given that the intermediate is observed:

$$p_{\mathrm{follow}} = \Pr(F \mid I) = \frac{\Pr(F \cap I)}{\Pr(I)}. \tag{1}$$

A.5 Attention Dynamics of Skill Neologisms

Experimental setup. We compare Qwen3.5-9B SN-SFT-L1+2 and SN-RL-L1+2 on 35 prompts each for new×internal-holdout and new×new. Each prompt's SFT- and RL-generated rollouts are replayed in full under both embedding sets to ensure a fair comparison. Within each comparison, the backbone and token sequence are fixed; only the neologism embeddings differ.

Attention extraction. Let $A_{t,j}^{(\ell,h)}$ be the post-softmax attention from rollout query t to key j. Its denominator includes all prompt tokens and rollout tokens up to and including t; future positions are masked. For skill s, we sum over its 50 learned token positions B_s and average over the $H = 16$ heads and $L = 8$ full-attention layers:

$$a_{B_s}(t) = \frac{1}{LH} \sum_{\ell=1}^{L} \sum_{h=1}^{H} \sum_{j \in B_s} A_{t,j}^{(\ell,h)}, \tag{2}$$

Qwen3.5-9B's 24 linear-attention layers are excluded. Values correspond to processing the displayed token, not predicting it.

Temporal attention concentration. For a rollout of T tokens, let $m = \lceil 0.1T \rceil$ and let $\mathcal{I}_s$ index the m largest values of $a_{B_s}(t)$. We measure the concentration of the attention share over the reasoning

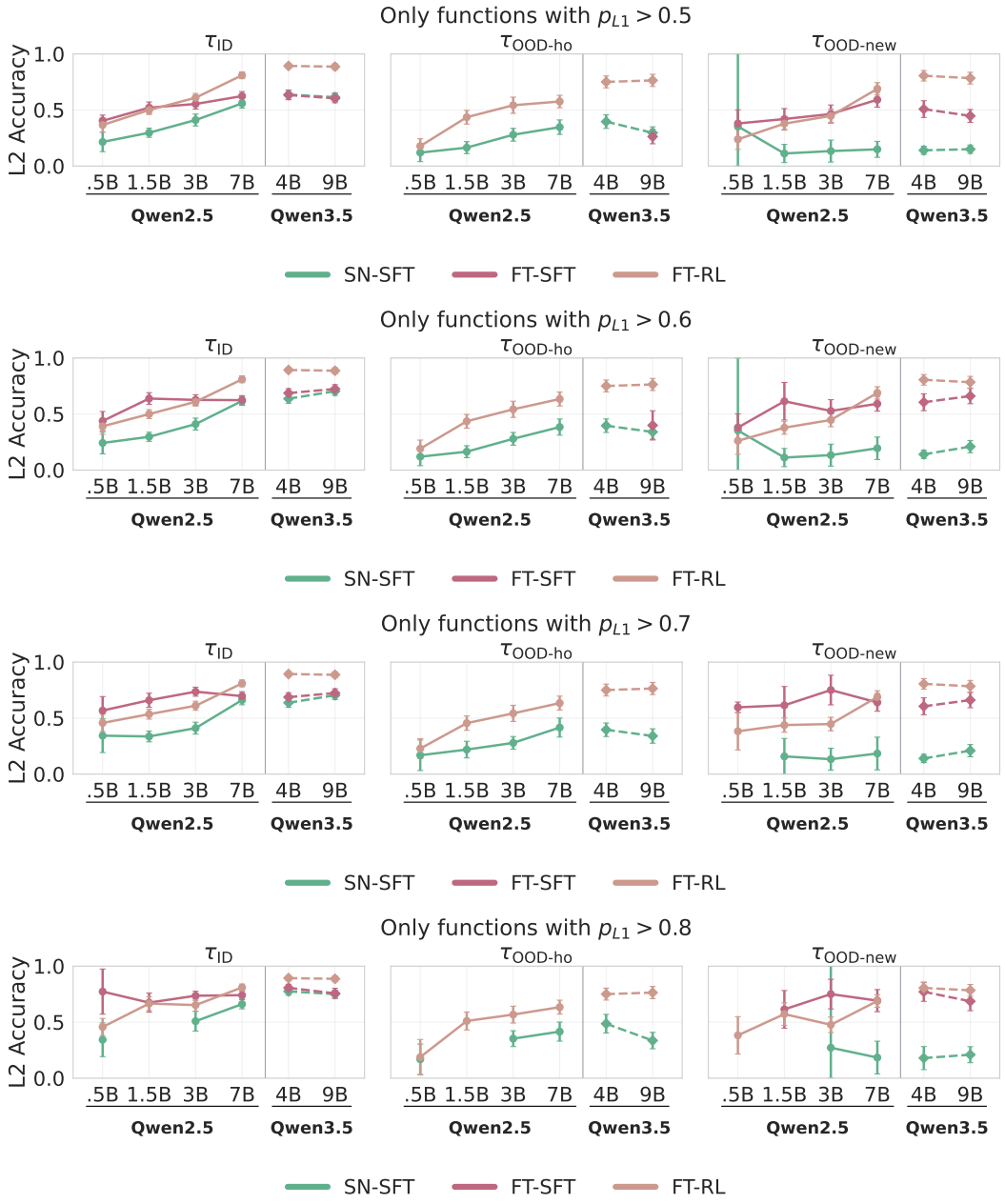

Figure A3: **Sensitivity of Level-2 compositional accuracy to the atomic mastery threshold p_{L_1}.** Rows correspond to thresholds 0.5, 0.6, 0.7, and 0.8; columns to the three composition conditions (τ_{ID}, $\tau_{\text{OOD-ho}}$, $\tau_{\text{OOD-new}}$). The threshold used in the main text (0.6) is the second row; qualitative trends are consistent across all values. Error bars show 95% Student-t intervals over pair-level accuracies. Wide error bars indicate that most pairs have been filtered, leading to noisy estimates of average accuracy across pairs.

trace by computing

$$C_{10}(s) = \frac{\frac{1}{m}\sum_{t \in \mathcal{I}_s} a_{B_s}(t)}{\frac{1}{T}\sum_{t=1}^{T} a_{B_s}(t)}. \tag{3}$$

A constant positive trace gives $C_{10} = 1$; larger values indicate greater temporal concentration, independent of uniform rescaling.

Matched-trajectory example. Figure A4 visualizes the same RL-generated example as the main figure.

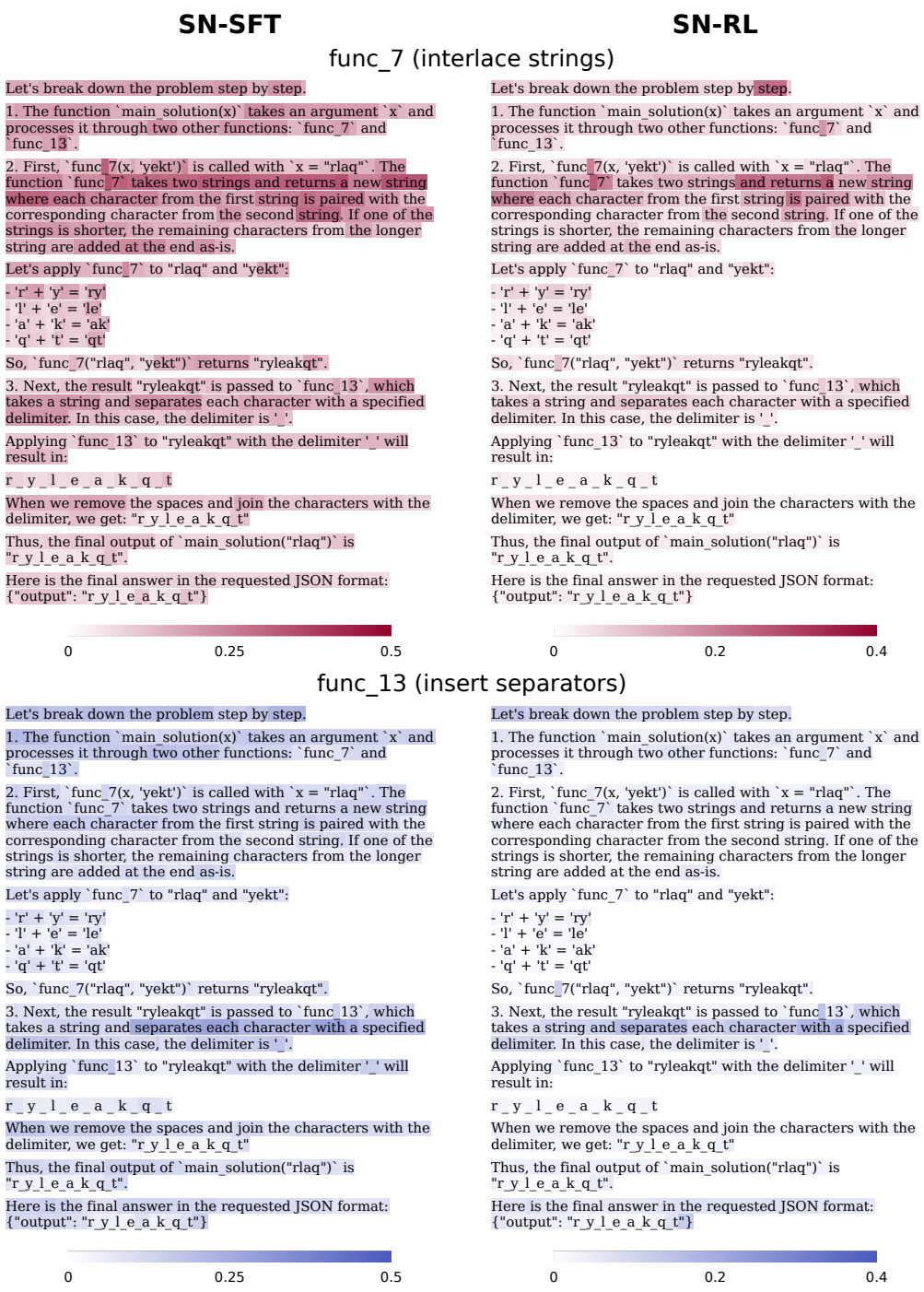

Figure A4: **Attention share to individual neologisms on an identical reasoning trajectory.**

B MEDICAL CALCULATION EXPERIMENTS

Data and training. We use MedCalc-Bench Verified v1.0.8,[2] (Khandekar et al., 2024) preserving the original patient notes, questions, reference explanations, and numerical bounds. All cases in Table 2 are labeled *Extracted* patient narratives. We focus on two calculator skills: *delta-gap* and *adjusted body weight*, for which the base Qwen3.5-4B model has accuracy $\tilde{2}$0-30%. Delta-gap is a common acid-base calculation, while adjusted body weight is a frequently used weight estimate for obese patients. The two training sets are disjoint and contain 241 delta-gap and 233 adjusted-body-weight examples. With Qwen3.5-4B frozen, each skill learns 50 embedding rows. SFT uses original worked solutions for one/two epochs, respectively (AdamW, batch 16, learning rates 10^{-3}/5×10^{-4} for delta-gap/adjusted-body-weight respectively). Each final SFT checkpoint initializes GRPO on the same training set, with binary numerical-correctness rewards, eight rollouts per question, 64 completions per update, learning rate 5×10^{-4}, and no KL penalty. We evaluate both neologisms for 70 steps.

Evaluation and uncertainty. Each table row uses 20 official-test questions with four sampled responses per question ($T = 1$, top-$p = 1$). Native tasks activate only the corresponding atomic neologism; base prompts contain no skill header. All conditions receive 1,024 generated tokens plus at most one 128-token continuation for length-capped traces in order to force an answer even when the model reaches the token limit while reasoning. RL training does not use this continuation and only scores based on the first 1,024 tokens. Accuracy averages the four binary scores within each question, then averages across questions. For question means $\bar{y}_i$, we report $\mathrm{SEM} = s(\bar{y}_1, \ldots, \bar{y}_n)/\sqrt{n}$, with sample standard deviation s and $n = 20$, rather than treating 80 responses as independent cases. Each calculator row has 20 distinct recorded patient groups, but groups may recur across calculators.

[2] https://github.com/nikhilk7153/MedCalc-Bench-Verified/tree/v1.0.8

Table C1: **Overview of skill acquisition methods used in the experiments.**

Family	Method	Training data	Algorithm	Section
Skill Neologisms	SN-SFT$_{\text{L1}}$	L1 (new)	SFT	§4
	SN-SFT$_{\text{L1+2}}$	SN-SFT$_{\text{L1}}$ initialization → L1+2 (new×int-train)	SFT	§5
	SN-RL$_{\text{L1+2}}$	SN-SFT$_{\text{L1+2}}$ initialization → L1+2 (new, int-train)	SFT → GRPO	§6
Fine-tuning (LoRA)	FT-SFT$_{\text{L1}}$	L1 (new)	SFT	§4
	FT-SFT$_{\text{L1+2}}$	L1+2 (new×int-train)	SFT	§5
	FT-RL$_{\text{L1+2}}$	L1 (all 25) → L1+2 (new, int-train)	SFT → GRPO	§6

r

Table C2: **Pretrained base models accuracy.**

	Level 1		Level 2
	Internal (N=3k)	New (N=2k)	Internal (N=6k)
Qwen2.5-0.5B	68.8	4.3	24.4
Qwen2.5-1.5B	85.7	3.7	55.8
Qwen2.5-3B	89.8	2.7	65.2
Qwen2.5-7B	98.7	0.1	89.0
Qwen3.5-4B	99.1	2.0	92.1
Qwen3.5-9B	99.4	11.8	94.6

C Experimental Details

C.1 Skill Partition and Base Model Pretraining

We use the 25-function string manipulation task of Yuan et al. (2026), with Level N denoting composition depth. The 25 functions are assigned as follows (sampled randomly):

- **Internal-train** (10): `func_5`, `func_6`, `func_9`, `func_10`, `func_12`, `func_15`, `func_16`, `func_18`, `func_19`, `func_21`
- **Internal-holdout** (5): `func_1`, `func_11`, `func_14`, `func_17`, `func_22`
- **New** (10): `func_0`, `func_2`, `func_3`, `func_4`, `func_7`, `func_8`, `func_13`, `func_20`, `func_23`, `func_24`

Exact function implementations are listed in Appendix C.6. All experiments use Qwen2.5-Instruct at 0.5B, 1.5B, 3B, and 7B unless stated otherwise. Base models are pretrained with the same two-stage protocol as Yuan et al. (2026): **Stage 1** applies SFT on 50,000 correct L1 trajectories over the 15 internal functions (collected from Qwen2.5-14B-Instruct); **Stage 2** applies GRPO on 50,000 L1+2 queries over the 15 internal functions, teaching compositional generalization. Hyperparameters for both stages are in Table C4. A separate Qwen2.5-14B-Instruct model trained with the same protocol is used as the trajectory collector for all skill-acquisition datasets (Sections 4–6).

C.2 Datasets

Table C3 lists all datasets used for training and evaluation. All training trajectories for new functions are generated by the 14B collector model described above and filtered for exact-match correctness before use.

C.3 Training Hyperparameters

All fine-tuned models use a LoRA adapter (Hu et al., 2022) with rank $r = \alpha = 32$ applied to all attention projection matrices. All skill neologisms use $l = 50$ soft tokens with prefix-map insertion (Section 3.4); only the embedding rows corresponding to neologism tokens are updated during training, with gradient masking applied to all other embedding rows.

Table C3: **Datasets used in the experiments.** Methods refer to the identifiers in Table C1. "Base models" refers to the pretrained base models described in Section 3.2, which are the starting point for all skill-acquisition methods.

Phase	Dataset	Methods	Size & construction
Training	Base pretraining, Stage 1	Base models	50k correct L1 trajectories; 15 internal functions only.
	Base pretraining, Stage 2	Base models	50k L1+2 queries (25k for each level); 15 internal functions.
	L1 demonstrations, new skills	SN-SFT$_{\text{L1}}$, FT-SFT$_{\text{L1}}$, FT-SFT$_{\text{L1-CL}}$, SN-RL$_{\text{L1+2}}$ (SFT warmup)	5k correct L1 trajectories per new function (50k total).
	L1+2 demonstrations, new $\times$ int-train	SN-SFT$_{\text{L1+2}}$, FT-SFT$_{\text{L1+2}}$, SN-RL$_{\text{L1+2}}$ (w/ binary rewards), FT-RL$_{\text{L1+2}}$ (w/ binary rewards)	5k correct L1 + 5k correct L2 per new function (100k total); L2 queries compose each new function with internal-train partners.
	L1 demonstrations, all 25 functions	FT-RL$_{\text{L1+2}}$ (SFT warmup)	5k correct L1 trajectories per function, all 25 functions (125k total).
Evaluation	L1 test set	All methods	200 samples per function $\times$ 25 functions (5k total).
	L2 test set	All methods	$\approx$30 samples per function pair (20k total).
	Pass@k test set	§6 analysis	800 new$\times$internal-holdout compositions, Levels 1–4 (200 per level); up to $k_{\max} = 32$ generations per sample.

Table C4: **Training hyperparameters for all methods.**

Method	Hyperparameters
Base Stage 1 (SFT)	epochs=2, lr=2×10^{-5}, max length=3072, completion-only loss, effective batch=16, bf16.
Base Stage 2 (GRPO)	lr=10^{-5}, max steps=600, generations=8, max completion length=800, $\beta = 0$, effective batch=128, constant LR schedule, bf16.
FT-SFT$_{\text{L1}}$	epochs=2, lr=2×10^{-5}, max length=3072, completion-only loss, effective batch=16.
FT-SFT$_{\text{L1-CL}}$	Same as FT-SFT$_{\text{L1}}$; trained sequentially one function at a time.
FT-SFT$_{\text{L1+2}}$	epochs=1, lr=2×10^{-5}, max length=3072, completion-only loss, effective batch=16.
FT-RL$_{\text{L1+2}}$ (SFT warmup)	epochs=1, lr=2×10^{-5}, max length=3072, effective batch=16.
FT-RL$_{\text{L1+2}}$ (GRPO)	lr=10^{-5}, max steps=600, generations=8, max completion length=800, $\beta = 0$, effective batch=128, constant LR schedule.
SN-SFT$_{\text{L1}}$	skill length=50, random init ($\mathcal{N}(0, 0.04)$), epochs=2, lr=5×10^{-3}, max length=3072, completion-only loss, effective batch=16.
SN-SFT$_{\text{L1+2}}$	Initialized from SN-SFT$_{\text{L1}}$ embeddings; epochs=2, lr=5×10^{-3}, max length=3072, effective batch=16.
SN-RL$_{\text{L1+2}}$	Initialized from SN-SFT$_{\text{L1+2}}$ embeddings; lr=5×10^{-4}, max steps=150, generations=8, max completion length=800, $\beta = 0$, effective batch=128, constant LR schedule.

C.4 Evaluation Protocol

A single evaluation protocol is used across all methods and model sizes. Decoding uses temperature 1.0, top-p 1.0, and a maximum of 1024 new tokens (sampling enabled); prompts are rendered

as single-turn user messages with the model's standard chat template. Predictions are extracted by parsing the first valid JSON object in the response and reading the `"output"` key; responses that fail to yield a valid parse are scored as incorrect.

The primary metric is exact match between the parsed output and the reference string. Level-2 results are reported across the three composition slices τ_{ID} (new×internal-train), $\tau_{\text{OOD-ho}}$ (new×internal-holdout), and $\tau_{\text{OOD-new}}$ (new×new), computed over ≈30 samples per function pair and averaged within each slice. The filtered-τ view further restricts to pairs where both functions satisfy $p_{L_1} > 0.6$ under the same model and method; error bars throughout report 95% Student-t intervals over pair-level means. For the pass@k analysis (Section 6), each test item receives up to $k_{\max} = 32$ independent generations and pass@k is computed cumulatively over the sorted draw.

C.5 Prompts and Skill Insertion

Demonstration collection prompt. Following (Yuan et al., 2026), function definitions are provided in the prompt when collecting reasoning traces to building demonstration datasets. The function definitions are then omitted from the collected demonstrations during training.

```
You are given a code:

{function_definitions}

{code}

Can you predict the output of `main_solution("{input}")` without writing any code?
Please think step-by-step and put your final answer in the following json format:
{"output": <your output>}, where <your output> should be the final string.
```

Forward prediction prompt. The same prompt template is used for all methods and levels of composition.

```
You are given a code:

{code}

Can you predict the output of `main_solution("{input}")` without writing any code?
Please think step-by-step and put your final answer in the following json format:
{"output": <your output>}, where <your output> should be the final string.
```

Skill neologism insertion. For skill neologism methods, a header listing the relevant soft tokens is prepended to the prompt. Only functions that appear in the query have their neologisms inserted.

```
Here are skills for different functions:
- func_23: <|func_23-0|><|func_23-1|>...
- func_8:  <|func_8-0|><|func_8-1|>...

<original prompt text>
```

During generation, soft tokens are masked out of the next-token distribution to prevent them from appearing in the chain-of-thought, which we found necessary to stabilize training.

C.6 Atomic Function Definitions

We use the same 25 functions as Yuan et al. (2026). The implementations below are used for all data generation.

```
from math import gcd

# func_0
def func_0(s):
    length = len(s)
    if length == 0:
```

```
        return s
    multiplier = 3
    while gcd(multiplier, length) != 1:
        multiplier += 2
    return "".join(s[(i * multiplier) % length] for i in range(length))

# func_1
def func_1(s, n):
    return s * n

# func_2
def func_2(s):
    vowels = "aeiouAEIOU"
    return "".join(ch for ch in s if ch not in vowels)

# func_3
def func_3(s):
    return "".join(sorted(s))

# func_4
def func_4(s):
    words = s.split()
    return " ".join(reversed(words))

# func_5
def func_5(s, pre):
    return pre + s

# func_6
def func_6(s, suf):
    return s + suf

# func_7
def func_7(s1, s2):
    result = []
    for i in range(max(len(s1), len(s2))):
        if i < len(s1):
            result.append(s1[i])
        if i < len(s2):
            result.append(s2[i])
    return "".join(result)

# func_8
def func_8(s, n):
    if not s:
        return s
    n = n % len(s)
    return s[n:] + s[:n]

# func_9
def func_9(s):
    return s + s[::-1]

# func_10
def func_10(s):
    return "".join(ch.lower() if i % 2 == 0 else ch.upper() for i, ch in enumerate(s))

# func_11
def func_11(s, shift):
    def shift_char(ch):
        if "a" <= ch <= "z":
            return chr((ord(ch) - ord("a") + shift) % 26 + ord("a"))
        if "A" <= ch <= "Z":
            return chr((ord(ch) - ord("A") + shift) % 26 + ord("A"))
        return ch
    return "".join(shift_char(ch) for ch in s)

# func_12
def func_12(s):
    mapping = {"a":"1","e":"2","i":"3","o":"4","u":"5",
               "A":"1","E":"2","I":"3","O":"4","U":"5"}
    return "".join(mapping.get(ch, ch) for ch in s)

# func_13
def func_13(s, sep):
    return sep.join(s)

# func_14
def func_14(s):
    return "".join(ch * 2 for ch in s)
```

```
# func_15
def func_15(s):
    return "".join("\\u00ab" + ch + "\\u00bb" for ch in s)

# func_16
def func_16(s):
    if not s:
        return s
    result = [s[0]]
    for ch in s[1:]:
        if ch != result[-1]:
            result.append(ch)
    return "".join(result)

# func_17
def func_17(s):
    if s == "":
        return s
    return func_17(s[1:]) + s[0]

# func_18
def func_18(s, n):
    result = ""
    for _ in range(n):
        result += s
    return result

# func_19
def func_19(s, n):
    count = 0
    while count < n and s:
        s = s[1:] + s[0]
        count += 1
    return s

# func_20
def func_20(s1, s2):
    if not s1 or not s2:
        return s1 + s2
    return s1[0] + s2[0] + func_20(s1[1:], s2[1:])

# func_21
def func_21(s):
    result = ""
    for ch in s:
        if ch.isalpha():
            result += ch
    return result

# func_22
def func_22(s):
    return s if len(s) % 2 == 0 else s[:-1]

# func_23
def func_23(s, depth):
    def has_digit(t):
        return any(ch.isdigit() for ch in t)

    transformations = [
        lambda t: t + "1",
        lambda t: "2" + t,
        lambda t: t.replace("a", "3"),
        lambda t: t[::-1],
    ]

    def helper(t, d):
        if has_digit(t):
            return t
        if d == 0:
            return None
        for trans in transformations:
            new_t = trans(t)
            res = helper(new_t, d - 1)
            if res is not None:
                return res
        return None

    result = helper(s, depth)
    return result if result is not None else s

# func_24
```

```
def func_24(s, depth):
    if s == s[::-1]:
        return s
    if depth <= 0:
        return s
    return func_24(s + s[::-1], depth - 1)
```

C.7 HARDWARE

Experiments were ran on two NVIDIA H100 GPUs (80GB VRAM each).