Reasoning with Neologisms: Can Soft Tokens Learn Composable Reasoning Skills Without Forgetting?

Antonin Berthon [1] **Mihaela van der Schaar** [1]

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 an open problem. Skill neologisms (SNs) have recently been proposed as a way to add procedural knowledge to a model without weight updates, by learning semantically integrated soft tokens. Whether SNs can acquire reasoning skills—and whether such skills compose with the model's existing capabilities—remains unknown. We systematically compare new reasoning skill acquisition via SFT/RL fine-tuning and via SNs in a controlled string-manipulation setting that prevents pretraining contamination and allows precise control over the base model's reasoning skills. We find that (i) naive fine-tuning catastrophically forgets previously internalized skills, while SNs preserve base-model behavior on all prompts not conditioned on learned neologism tokens, by construction; (ii) SN-learned skills compose with existing skills both in- and out-of-distribution, with the composition gap to fine-tuning shrinking with model scale; and (iii) RL-trained SNs match the fully-internalized RL baseline on compositions involving existing internal skills in the 7B setting, with pass@k analysis suggesting that this reflects genuine compositional skill acquisition rather than distribution sharpening. Together, these findings establish skill neologisms as a promising path for continually extending LLM reasoning capabilities.

[1]University of Cambridge, Cambridge, United Kingdom. Correspondence to: Antonin Berthon <armb3@cam.ac.uk>.

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

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. 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., 2025b; Li et al., 2025; Lai et al., 2025; Shenfeld et al., 2026b). Scaling continual learning to many new skills requires hard guarantees against forgetting, not just partial mitigation (Zheng et al., 2025b).

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. Mixing the new skill with existing model skills in the training set provides an inductive bias to learn a generally composable representation of the new skill, and LLMs' in-context learning abilities allow independently learned neologisms to be composed zero-shot. However, SNs have so far only been tested on non-reasoning, input-output skills. Whether they can be applied to reasoning skills (i.e., procedural knowledge that requires intermediate reasoning traces) and generally compose with other skills is unknown. In this work, we investigate whether SNs are a viable path towards continual learning of reasoning capabilities:

Central question

Given a model with existing reasoning capabilities, to what extent can SNs be used to learn new atomic and composable reasoning skills while preventing forgetting of previous ones?

Investigating this question requires (1) clear definitions of atomic and compositional reasoning skills, (2) controlling the reasoning skills available to the base model, and (3) preventing contamination of the new skills in the base model. To satisfy these requirements, we focus on a synthetic string

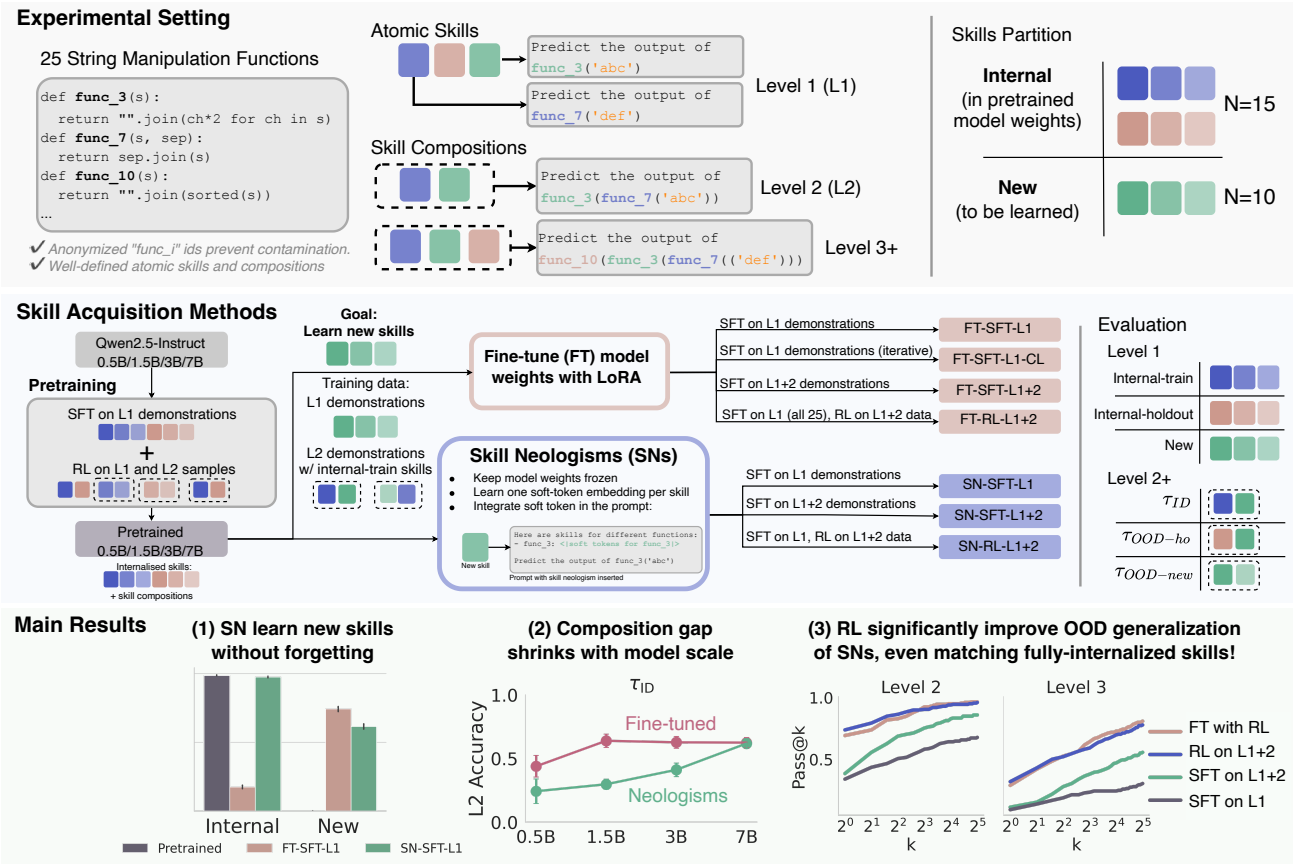

Figure 1. **Overview of our study and key findings.** (Top) We use the string manipulation setting introduced in Yuan et al. (2026), allowing for clear definitions of atomic and compositional reasoning skills. We separate the 25 skills into two clusters: 15 internal skills to be internalized in the base model weights, and 10 new skills to be learned. (Middle) We pretrain Qwen2.5 models of varying sizes with a two-stage SFT+RL similar to Yuan et al. (2026), learning the 15 internal skills and the ability to compose them. We then train to learn the new skills with either model fine-tuning (FT) via LoRA adapters (Hu et al., 2022) or Skill Neologisms (Berthon et al., 2026). (Bottom) We find that (1) learning new skills via model fine-tuning runs into catastrophic forgetting while SNs retain pretraining skills; (2) SN-learned skills compose with ID and OOD internal skills, with the gap to model fine-tuning shrinking with model scale; (3) RL significantly improves OOD generalization of SNs, matching the fully-internalized RL baseline on new × internal compositions at the 7B scale.

manipulation setting introduced in (Yuan et al., 2026). This setting defines 25 atomic skills as non-decomposable string transformations instantiated as Python functions, enabling natural skill compositions as nested function calls, and uses anonymized skill identifiers (e.g., `func_0`) to prevent contamination of skills before they have been formally learned by the model.

As a basis for our analysis, we train multiple Qwen2.5 models (Qwen et al., 2025) of various sizes (0.5B–7B) on a randomly selected subset of 15 functions, following the two-stage process used in (Yuan et al., 2026): first train with SFT on correct atomic skill trajectories to "internalize" each skill in model weights, then train with RL on skill compositions to acquire the ability to compose them. This creates models that have mastered most of the 15 atomic skills as well as the ability to compose them. We then study the acquisition of the remaining 10 atomic functions via either model fine-tuning or skill neologisms, training one set of soft tokens for each new atomic function. Specifically, we investigate the following three research questions:

RQ1. Can new atomic reasoning skills be acquired via skill neologisms without forgetting internalized skills?

RQ2. Can reasoning skills learned via neologisms compose with other skills?

RQ3. Does the learning algorithm used to learn skill neologisms affect their composability?

Figure 1 summarizes our study and main findings. We find that skill neologisms acquire new reasoning skills without the forgetting incurred by model fine-tuning, and compose with internalized skills in- and out-of-distribution—with the

gap to model fine-tuning shrinking with model scale. Most notably, RL-trained neologisms match fully-internalized skills on compositions involving existing capabilities, with pass@k analysis providing evidence that RL improves the availability of correct completions rather than merely reweighting existing ones. Together, these findings establish skill neologisms as a promising path for continually extending LLM reasoning capabilities.

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., 2025c). 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, the soft tokens are inserted in semantically meaningful positions in the prompt (e.g., in place of a specific string), which allows independently learned neologisms to be composed zero-shot at inference time via the model's in-context learning. Berthon et al. (2026) demonstrated this approach on non-reasoning input-output tasks; whether it can similarly enable continual acquisition of reasoning capabilities—where each skill requires multi-step intermediate chain-of-thought—is the central question we investigate in this work.

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 inter-

nal skills at evaluation. Exact function assignments are in Appendix B.

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) at four sizes (0.5B, 1.5B, 3B, 7B). **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** performs 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 1 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 B.1.

3.3. Compositional Generalization Conditions

Pretrained LLMs are known to flexibly compose skills, including in combinations not seen during training (Arora & Goyal, 2023; Yu et al., 2024; Yuan et al., 2026). In order to study the extent to which skills learned 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); $\tau_{\text{OOD-ho}}$ on new$\times$internal-holdout pairs (composition with held-out internal skills); and $\tau_{\text{OOD-new}}$ on new$\times$new pairs (composition between two newly acquired skills).

Formally, let $p_{L2}(i, j)$ denote the exact-match accuracy on Level-2 queries with outer function f_i and inner function f_j. For a set of ordered pairs $\mathcal{P}$, we define

$$\tau(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} p_{L2}(i, j)$$

where (i, j) and (j, i) are treated as distinct pairs. The three metrics correspond to $\mathcal{P}_{\text{ID}}$ (new $\times$ internal-train pairs), $\mathcal{P}_{\text{OOD-ho}}$ (new $\times$ internal-holdout pairs), and $\mathcal{P}_{\text{OOD-new}}$ (new $\times$ new pairs).

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.2 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 2 summarizes the model configurations used across our experiments (full hyperparameters shown in Appendix B.3).

4. Acquiring New Atomic Reasoning 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?* Note that RL training requires atomic skills to already be available in the model (Yuan et al., 2026), so all methods in this section are trained via SFT.

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-L1**.

Baselines We compare against two LoRA fine-tuning baselines on the same demonstration data: **FT-SFT-L1**: a LoRA adapter jointly fine-tuned on all 50k L1 demonstrations across the 10 new skills; **FT-SFT-L1-CL**: the same LoRA adapter, but trained sequentially on each skill's 5k demonstrations, simulating a continual learning scenario.

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. For the continual learning setting, we report L1 accuracy relative to the FT-SFT-L1 baseline to account for skill difficulty.

Results Figure 2 shows the results across methods and model sizes when training on all new skills jointly (per-

Table 1. **Pretrained base models accuracy.**

	Level 1		Level 2
	Internal (N=3k)	New (N=2k)	Internal (N=6k)
0.5B	68.8	4.3	24.4
1.5B	85.7	3.7	55.8
3B	89.8	2.7	65.2
7B	98.7	0.1	89.0

Table 2. **Overview of skill acquisition methods used in the experiments.**

Family	Method	Training data	Algorithm	Section
Skill Neologisms	SN-SFT-L1	L1 (new)	SFT	§4
	SN-SFT-L1+2	L1-2 (int-train, new)	SFT	§5
	SN-RL-L1+2	L1 (new) | L1-2 (int-train, new)	SFT | GRPO	§6
Fine-tuning (LoRA)	FT-SFT-L1	L1 (new)	SFT	§4
	FT-SFT-L1-CL	L1 (new), sequential	SFT	§4
	FT-SFT-L1+2	L1-2 (int-train, new)	SFT	§5
	FT-RL-L1+2	L1-2 (all 25) | L1-2 (int-train, new)	SFT | GRPO	§5

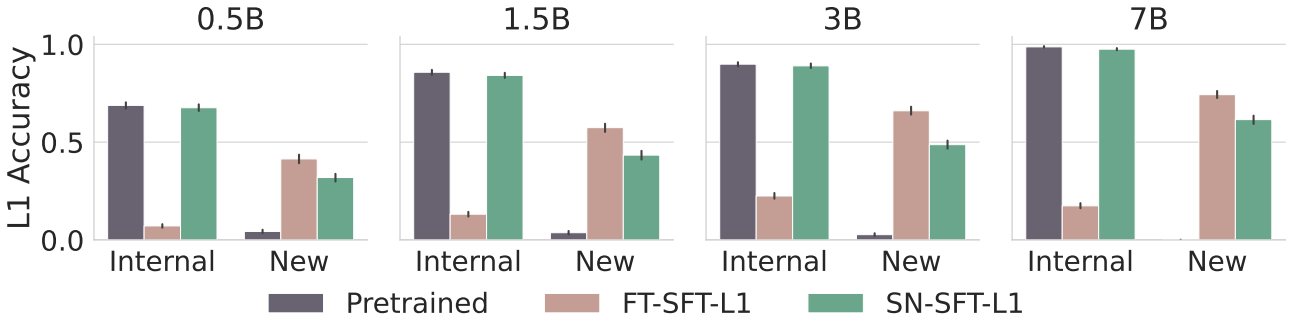

Figure 2. **L1 accuracy on internal and new skills after SFT training on L1 demonstrations of new skills only.** Fine-tuning successfully learns most new skills but loses substantial accuracy on previously internalized ones. Skill neologisms preserve internalized accuracy by construction while approaching the fine-tuned baseline on new skills. Error bars show 95% confidence intervals. Per-skill detailed results are shown in the Appendix Figure A1.

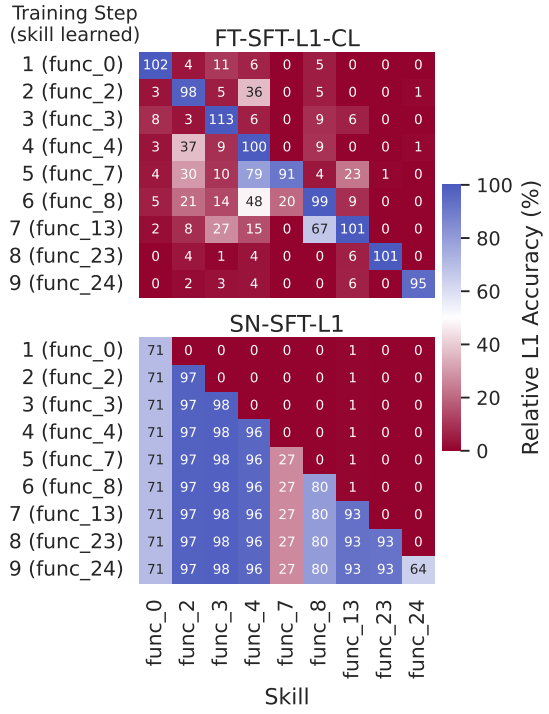

Figure 3. **Continual Learning of new atomic skills (7B model).** L1 accuracy relative to the FT-SFT-L1 baseline (trained jointly on all skills). New skills are learned one after the other via SFT on L1 demonstrations. The fine-tuned model forgets the previously learned skills, while neologisms scale by design.

skill detailed results are shown in Figure A1). FT-SFT-L1 successfully learns the new skills, but at the cost of catastrophic forgetting of previously internalized ones. SN-SFT-L1 retains all of the pretrained model's performance on internalized skills by construction, while approaching the fine-tuned baseline on new skills.

For the continual learning setting, Figure 3 shows that sequential fine-tuning compounds forgetting: FT-SFT-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-L1 preserves all skills throughout the training sequence.

Takeaway

Skill neologisms acquire new atomic reasoning skills while preserving all internalized model capabilities; by contrast, standard fine-tuning forgets both previously internalized and newly learned skills.

5. To What Extent Do Reasoning Skills Learned via Neologisms Compose with Other Skills?

The previous section demonstrated that skill neologisms can acquire new atomic reasoning skills without forgetting previously internalized ones. However, 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**): *to what extent can atomic reasoning skills learned via soft tokens compose with other atomic skills?*

Setup For each new skill, we collect 10k correct reasoning demonstrations of 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-L1+2**. We also compare with the **SN-SFT-L1** neologisms from §4, which are trained on L1 demonstrations only.

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

- **FT-SFT-L1+2**: Fine-tune a LoRA adapter with SFT on the same L1+2 demonstrations used for SN-SFT-L1+2, mixing new skills with internal-train skills. Since internal-holdout skills are not part of the training set, this model is expected to largely forget them during training; we therefore primarily use this baseline to evaluate ID and OOD (new×new) composition performance.
- **FT-RL-L1+2**: We first train with SFT on 5k L1 demonstrations for **all 25 skills**, providing initial knowledge of all atomic skills without running into catastrophic forgetting. We then train with GRPO on the same query distribution as SN-SFT-L1+2: L1+2 queries for the new skills, with *internal-train* skills in the L2 samples. Since this requires prior fine-tuning on all skills simultaneously, it is not strictly comparable to neologisms and FT-SFT-L1+2 (where new skills are learned entirely independently from internal-holdout skills). Nevertheless, 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 7.2k trajectories (∼30 samples per function pair), reporting both raw and $p_{L_1} > 0.6$-filtered accuracies. A sensitivity analysis of the threshold is provided in Figure A3 and confirms that results are qualitatively consistent across thresholds.

Results Figure 4 shows the results across methods, model sizes, and composition types.

- **Neologisms generalize to compose with internal skills.** SN-SFT-L1+2 achieves strong composition accuracy on both τ_{ID} (∼60% at 7B in the filtered view) and $\tau_{\text{OOD-ho}}$ (∼40%). At 7B, it matches FT-SFT-L1+2 on new×internal-train pairs and outperforms it on new×internal-holdout, since FT-SFT-L1+2 largely forgets internal-holdout skills. This shows that neologisms can learn composable representations of the underlying reasoning skill, and even compose with internal skills unseen during training.
- **The gap with internalized new skills shrinks with model scale.** On τ_{ID}, the gap between SN-SFT-L1+2 and FT-SFT-L1+2 reduces as model scale increases. SN-SFT-L1+2 at 0.5B is near 0% on most L2 compositions, while at 7B it rivals the fine-tuned models. On $\tau_{\text{OOD-ho}}$, only the 7B model reaches substantial L2 accuracy. This suggests stronger models can compose soft tokens more effectively.
- **Neologisms fail at new×new composition.** Compositions between two soft-token-encoded new skills plateau at ∼15–20% across model sizes in the filtered view, while fine-tuned models achieve ∼55–70%. This indicates that base models struggle to zero-shot compose multiple SNs for reasoning. While this is a clear empirical limitation of the current SN formulation, we discuss approaches to mitigate it in Section 7.
- **Compositional training data improves SN composability.** SN-SFT-L1+2 consistently outperforms SN-SFT-L1 across all conditions, showing that L2 demonstrations during training improve composability beyond what atomic-only training provides. This echoes Berthon et al. (2026): mixing the target skill with various other skills acts as an inductive bias on the neologism, leading to a more generally composable representation.

Takeaway

Skill neologisms learn reasoning skills that compose with internalized skills both in-distribution and out-of-distribution, and the gap to weight-based skill internalization shrinks with model size.

6. Does the Learning Algorithm Matter?

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. Recent works on reasoning fine-tuning have found that on-policy methods such as RL produce more generalizable skills (Chu et al., 2025; Chen et al., 2025; Shenfeld et al., 2026b; Yuan et al., 2026) than off-policy methods like SFT. This motivates our third research

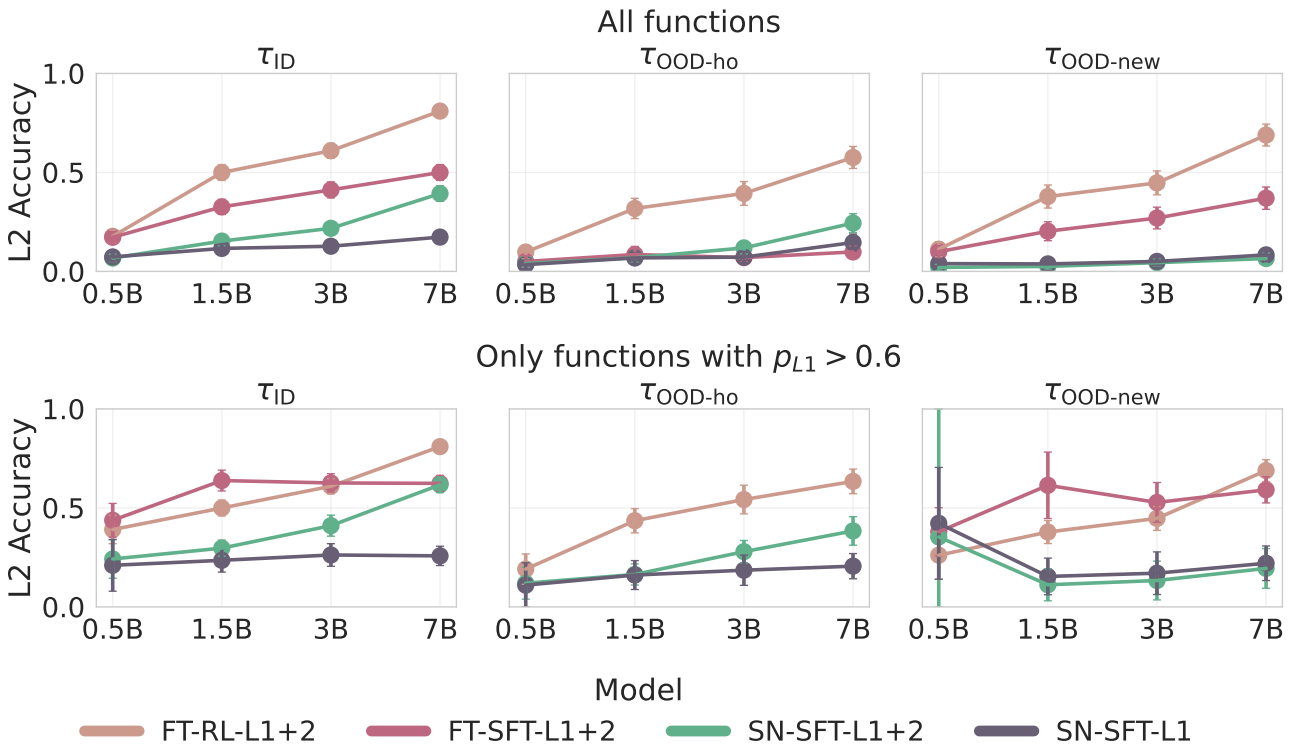

Figure 4. **Level 2 Accuracy on ID and OOD skill combinations.** Accuracy is computed per function pair and averaged within each pair type. Top row: all function pairs. Bottom row: only function pairs where both atomic skills satisfy $p_{L_1} > 0.6$ under the same model and method, controlling for atomic skill mastery. Note that FT-SFT-L1+2 disappears from $\tau_{\text{OOD-ho}}$ 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.

question (**RQ3**): *does this property extend to skill neologisms, where only the soft-token embeddings are updated and the model weights remain frozen?*

Setup We train skill neologisms via GRPO (Shao et al., 2024) using the same L1+2 query distribution as SN-SFT-L1+2, with binary rewards based on output correctness. The resulting model is **SN-RL-L1+2**. Since GRPO training is computationally intensive, we focus on the 7B model and on the 6 new skills that achieve $> 60\%$ Level 1 accuracy in SN-SFT-L1 (cf Figure A1 in Appendix)[1]. We initialize the soft tokens from SN-SFT-L1 to ensure the neologisms have atomic competence at the start of RL training. We compare against SN-SFT-L1+2 and FT-RL-L1+2 from §5, evaluated on the same L2 test set across the three composition types.

Results Figure 5 shows the L2 accuracy of SN-RL-L1+2, SN-SFT-L1+2, and FT-RL-L1+2 across the three composition conditions.

- **SN-RL matches fully-internalized skills on compositions with existing skills.** On both τ_{ID} and $\tau_{\text{OOD-ho}}$, SN-RL-L1+2 reaches accuracies comparable to the FT-RL-L1+2 upper bound, closing the gap between SN and internalized skills observed in §5. Moreover, SN-RL-L1+2 also significantly outperforms SN-SFT-L1+2 across all three conditions, with the largest absolute gain on $\tau_{\text{OOD-ho}}$ (+35.9%). This demonstrates that RL substantially improves the generalizability of the learned skill representations and even matches fully-internalized baselines.
- **The new×new gap persists.** On $\tau_{\text{OOD-new}}$, RL improves over SFT but a substantial gap to FT-RL-L1+2 remains. The limited ability of soft-token-encoded skills to compose with each other, observed in §5, is not fully resolved by RL and likely requires complementary approaches discussed in §7.

Pass@k analysis We investigate whether RL training learns genuinely stronger compositional skills, or merely sharpens the model's trajectory distribution. We compute pass@k accuracy for SN-RL-L1+2, SN-SFT-L1+2, SN-SFT-L1, and FT-RL-L1+2 on a test set of 800 new×internal-holdout compositions spread evenly across Levels 1 to 4 (sampling 1/2/3 internal-holdout skills for Levels 2/3/4).

[1]We experimented with RL training for smaller models, but this generally resulted in unstable training and reward collapse. We discuss this further in Section 7.

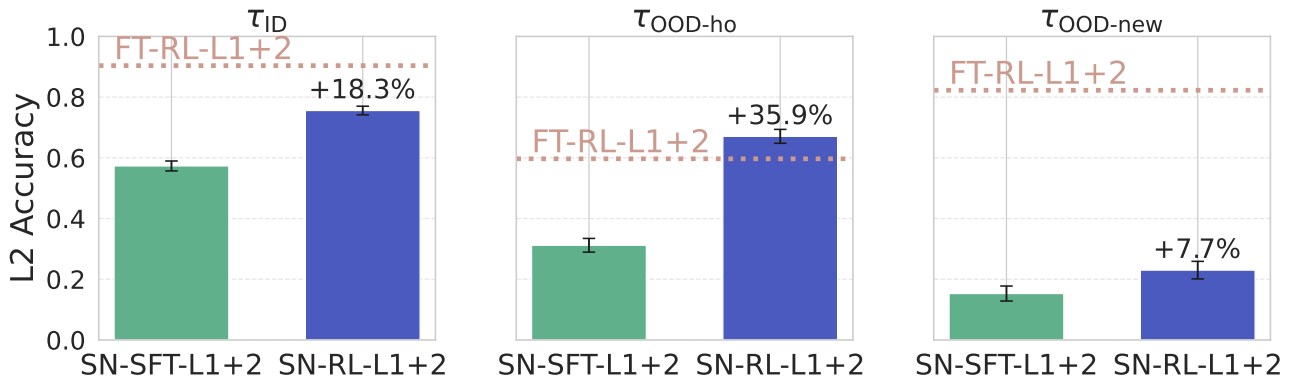

Figure 5. **Effect of learning algorithm on Level 2 accuracy.** Comparison of SN-SFT-L1+2 and SN-RL-L1+2 on the three composition conditions, with FT-RL-L1+2 (dotted) as the upper bound for internalized new skills. RL training substantially improves SN composition, with SN-RL-L1+2 matching the FT-RL-L1+2 baseline on τ_{ID} and $\tau_{\text{OOD-ho}}$, while improvements on $\tau_{\text{OOD-new}}$ remain limited. Results shown for the 7B model. Error bars show 95% confidence intervals.

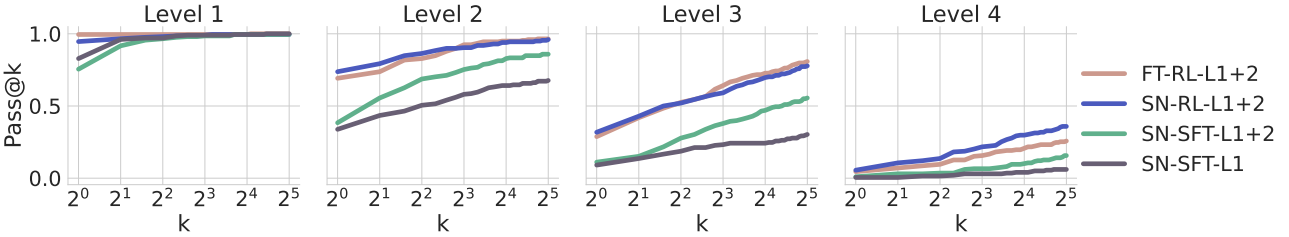

Figure 6. **Pass@k accuracy by composition depth** for the 7B model on new×internal-holdout compositions across Levels 1–4. SN-RL-L1+2 outperforms SN-SFT-L1+2 at all k, with the gap widening at higher composition depths, indicating that RL training expands the set of reachable compositional traces rather than merely re-weighting them.

The results are shown in Figure 6. The SN-RL-L1+2 advantage over SFT-based SNs persists at all k values for Levels 2–4. Notably, SN-RL-L1+2 also matches or exceeds FT-RL-L1+2 at higher depths and larger k. This indicates that RL training of neologisms learns skills that generalize significantly better than SFT-based training, rather than simply concentrating the model's neologism-conditioned output distribution onto existing high-scoring regions. This aligns with the conclusion of Yuan et al. (2026) for full-model fine-tuning that RL on compositional data leads to genuine skill acquisition through compositional generalization, and extends this result for skill neologisms.

Takeaway

RL training substantially improves the OOD compositional generalization of skill neologisms, closing the gap to the fully-internalized RL baseline on compositions involving existing internal skills. This effect persists at high pass@k, indicating genuine acquisition of compositional ability rather than re-weighting of behaviors already reachable by SFT-trained soft tokens.

7. Discussion

Our experiments demonstrate that skill neologisms can acquire new reasoning skills without the forgetting risks of model fine-tuning, while achieving strong ID and OOD composition with internal skills. Most notably, when trained with RL, soft tokens match the idealized RL-fine-tuned baseline on compositions involving existing internal skills—reaching the compositional generalization of weight-based skill acquisition while preserving all prior model capabilities by architectural construction. To the best of our knowledge, our work is the first to show that prompt-tuning-like soft tokens, optimized against verifiable rewards, reach internalized-skill levels of compositional generalization on reasoning tasks. Our results also echo Lester et al. (2021)'s finding that prompt tuning grows more competitive with full fine-tuning as models scale. We observe a complementary scaling effect: not only are larger models more easily *tuned* via soft prompts, they also better *generalize* from them, successfully composing learned neologisms with previously unseen skills at inference time.

Inter-neologism composition. Despite strong OOD generalization with internal skills, models struggle to compose

multiple neologisms zero-shot. Soft-token methods are known to struggle with multi-step reasoning (Fan et al., 2025), which likely compounds when multiple soft prompts must be composed zero-shot. A natural extension specific to SNs is to relax the frozen-weights constraint to allow a *one-time* weight update that internalizes *the ability to compose neologisms*, after which the model is frozen and new neologisms can be continually added.

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. Whether SN-based skill acquisition transfers to domains where reasoning skills are fuzzier and more entangled is an important open question. We also observed that RL training of neologisms was unstable below 7B, with runs frequently collapsing to degenerate solutions mid-training, while 7B runs were robust and required little tuning. 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, and scaling neologisms to realistic reasoning domains where the boundary between internal and new skills is less clearly defined.

References

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

Berthon, A., Astorga, N., and van der Schaar, M. Skill neologisms: Towards skill-based continual learning. In *ICML*, 2026.

Chen, H., Razin, N., Narasimhan, K., and Chen, D. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*, 2025.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *ICML*, 2025.

Fan, S., Xie, L., Shen, C., Teng, G., Yuan, X., Zhang, X., Huang, C., Wang, W., He, X., and Ye, J. Improving complex reasoning with dynamic prompt corruption: A soft prompt optimization approach. In *ICLR*, 2025.

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

Gandhi, K., Chakravarthy, A. K., Singh, A., Lile, N., and Goodman, N. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. In *COLM*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a.

Guo, H., Hou, Z., Sun, Y., He, J., Chen, Y., Zhou, Y., Jia, Y., Wang, J., and Chua, T.-S. Mllm-ctbench: A benchmark for continual instruction tuning with reasoning process diagnosis. *arXiv preprint arXiv:2508.08275*, 2025b.

Guo, H., Zeng, F., Zhu, F., Wang, J., Wang, X., Zhou, J., Zhao, H., Liu, W., Ma, S., Wang, D.-H., et al. Continual learning for generative ai: From llms to mllms and beyond. *arXiv preprint arXiv:2506.13045*, 2025c.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Jiang, G., Jiang, C., Li, Z., Xue, S., Zhou, J., Song, L., Lian, D., and Wei, Y. Unlocking the power of function vectors for characterizing and mitigating catastrophic forgetting in continual instruction tuning. *arXiv preprint arXiv:2502.11019*, 2025.

Ke, Z., Lin, H., Shao, Y., Xu, H., Shu, L., and Liu, B. Continual training of language models for few-shot learning. In *EMNLP*, 2022.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Lai, S., Zhao, H., Feng, R., Ma, C., Liu, W., Zhao, H., Lin, X., Yi, D., Zhang, Q., Liu, H., et al. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, 2025.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *EMNLP*, 2021.

Li, Y., Xu, Z., Jiang, F., Ramasubramanian, B., Niu, L., Lin, B. Y., Yue, X., and Poovendran, R. Temporal sampling for forgotten reasoning in LLMs. In *2nd AI for Math Workshop @ ICML 2025*, 2025.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. In *COLM*, 2025.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE*

Transactions on Audio, Speech and Language Processing, 2025a.

Luo, Z., Zhang, X., Liu, X., Li, H., Gong, Y., Chen, Q., and Cheng, P. 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.

Madotto, A., Lin, Z., Zhou, Z., Moon, S., Crook, P. A., Liu, B., Yu, Z., Cho, E., Fung, P., and Wang, Z. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7452–7467, 2021.

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

Pan, J., Zhang, J., Wang, X., Yuan, L., Peng, H., and Suhr, A. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025.

Ren, W., Li, X., Wang, L., Zhao, T., and Qin, W. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*, 2024.

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

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026a.

Shenfeld, I., Pari, J., and Agrawal, P. RL's razor: Why online reinforcement learning forgets less. In *ICLR*, 2026b.

Shi, H., Xu, Z., Wang, H., Qin, W., Wang, W., Wang, Y., Wang, Z., Ebrahimi, S., and Wang, H. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

Wang, X., Chen, T., Ge, Q., Xia, H., Bao, R., Zheng, R., Zhang, Q., Gui, T., and Huang, X.-J. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023.

Wang, Y., Liu, Y., Shi, C., Li, H., Chen, C., Lu, H., and Yang, Y. 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.

Wu, T., Luo, L., Li, Y.-F., Pan, S., Vu, T.-T., and Haffari, G. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

Yu, D., Kaur, S., Gupta, A., Brown-Cohen, J., Goyal, A., and Arora, S. Skill-mix: a flexible and expandable family of evaluations for ai models. In *ICLR*, 2024.

Yuan, L., Chen, W., Zhang, Y., Cui, G., Wang, H., You, Z., Ding, N., Liu, Z., Sun, M., and Peng, H. From f(x) and g(x) to f(g(x)): LLMs learn new skills in RL by composing old ones. In *ICLR*, 2026.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *NeurIPS*, 2026.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., MA, Z., and He, J. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *COLM*, 2025.

Zheng, J., Cai, X., Qiu, S., and Ma, Q. Spurious forgetting in continual learning of language models. In *ICLR*, 2025a.

Zheng, J., Qiu, S., Shi, C., and Ma, Q. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35, 2025b.

A. Detailed Experimental Results

A.1. Atomic Skill Acquisition (Section 4)

Figure A1 expands the aggregate results of Figure 3 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-L1** acquires most new skills but loses substantial accuracy on previously internalized ones, while **SN-SFT-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.

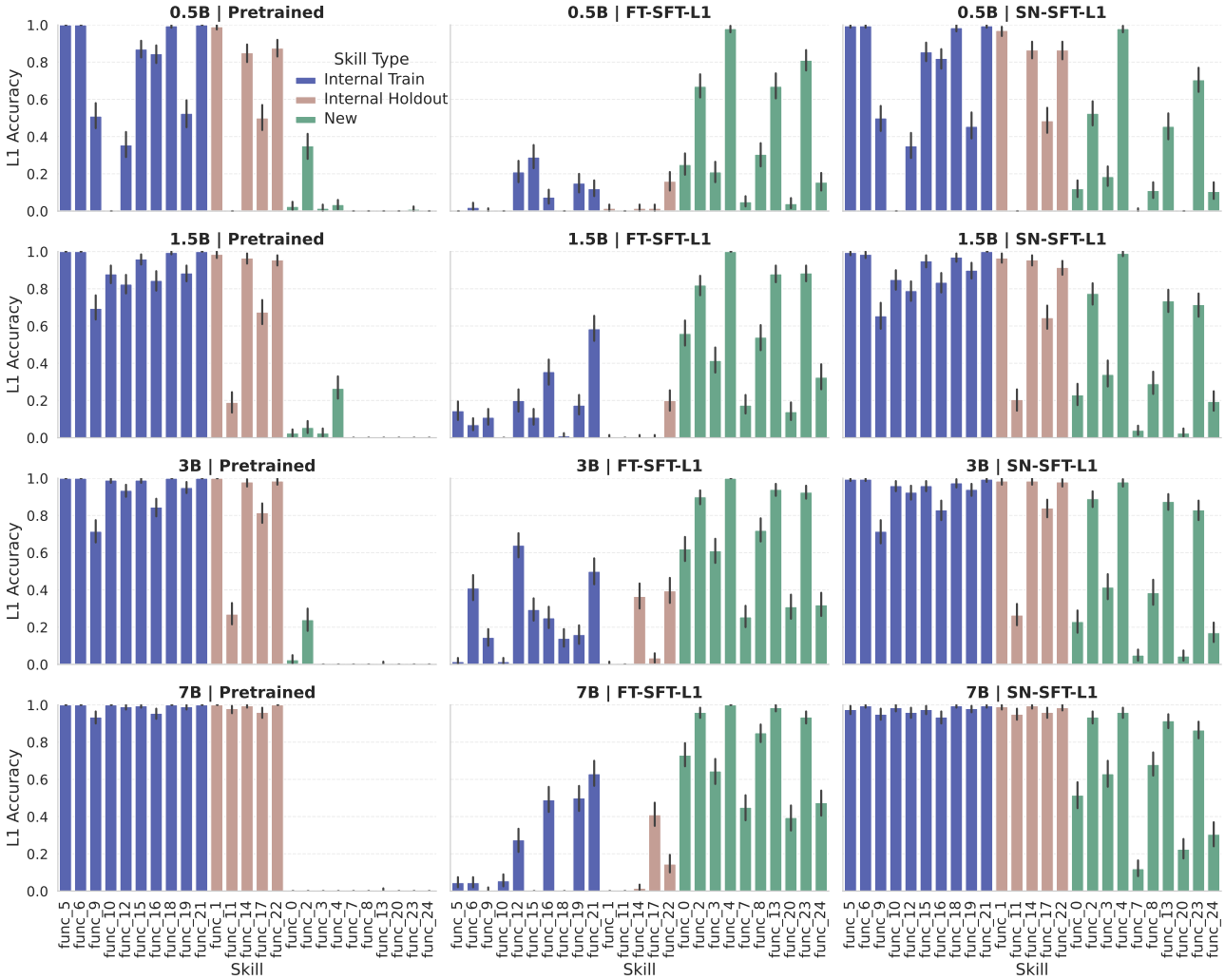

Figure A1. **Level-1 accuracy on all 25 skills before and after SFT training on L1 demonstrations of new skills, per model size.** Skills are grouped by type: internal-train (blue), internal-holdout (orange), and new (green). Fine-tuning successfully learns most new skills but incurs substantial forgetting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

Figure A2 extends the continual-learning analysis of Figure 3 to the remaining three model sizes (0.5B, 1.5B, 3B). At every scale, sequential fine-tuning compounds forgetting: previously learned new skills collapse as each successive skill is trained. Because each neologism is optimized independently with the base model frozen, **SN-SFT-L1** architecturally preserves all earlier skills throughout the training sequence.

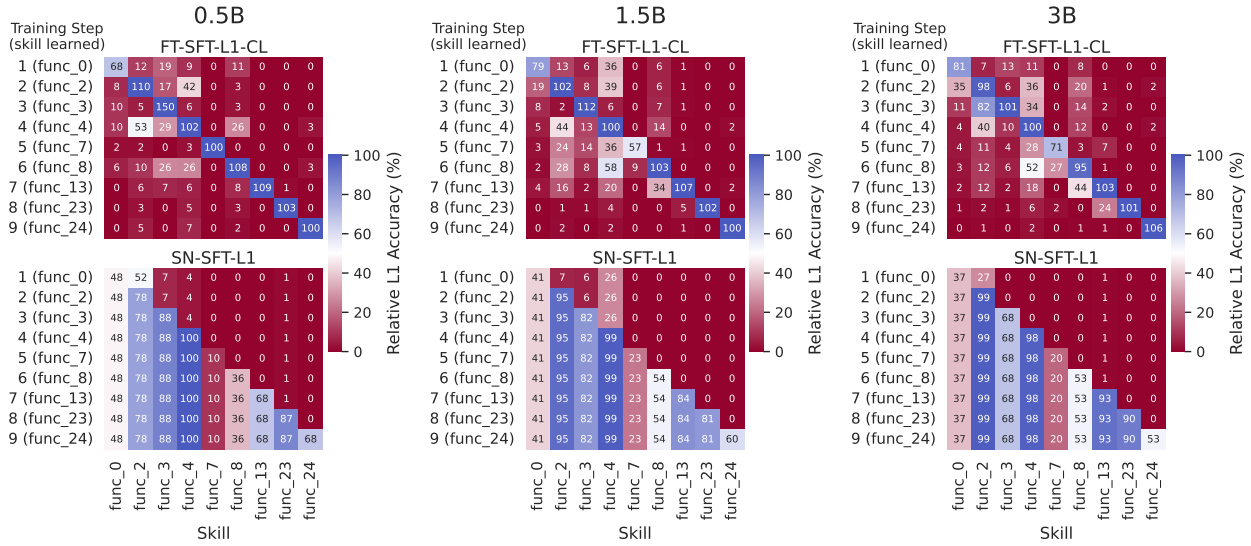

Figure A2. **Continual learning of new atomic skills at 0.5B, 1.5B, and 3B (cf. Figure 3 in the main text for the 7B model).**

A.2. 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-L1+2 generalizes on τ_{ID} and $\tau_{\text{OOD-ho}}$ with a gap that shrinks with model scale; new$\times$new compositions remain weak across all methods; and FT-SFT-L1+2 disappears from the $\tau_{\text{OOD-ho}}$ filtered view because internal-holdout functions fail the mastery threshold (the model forgets them during training on new skills).

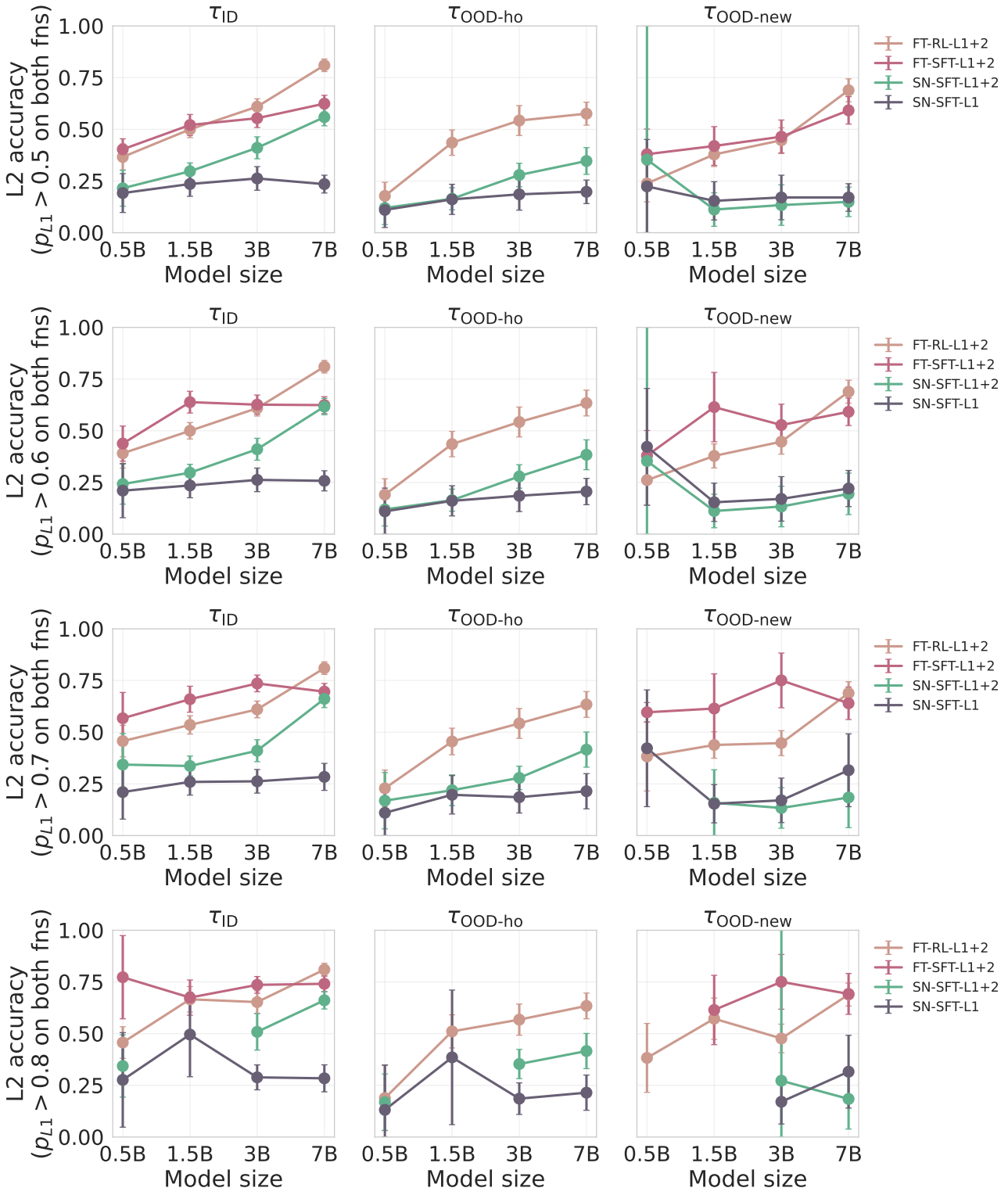

Figure A3. **Sensitivity of Level-2 compositional accuracy to the atomic mastery threshold p_{L_1}.** Rows correspond to thresholds 0.5, 0.6, 0.7, and 0.8; columns to the three composition conditions (τ_{ID}, $\tau_{\text{OOD-ho}}$, $\tau_{\text{OOD-new}}$). The threshold used in the main text (0.6) 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.

B. Experimental Details

B.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 B.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 B2. 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).

B.2. Datasets

Table B1 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.

Table B1. **Datasets used in the experiments.** Methods refer to the identifiers in Table 2. "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-L1, FT-SFT-L1, FT-SFT-L1-CL, SN-RL-L1+2 (SFT warmup)	5k correct L1 trajectories per new function (50k total).
	L1+2 demonstrations, new $\times$ int-train	SN-SFT-L1+2, FT-SFT-L1+2, SN-RL-L1+2 (w/ binary rewards), FT-RL-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-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.

B.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.

Section 6 setup. RL-based neologism training (SN-RL-L1+2) is applied only to the 7B model, on the six new functions that achieve $> 60\%$ L1 accuracy under SN-SFT-L1: `func_2, func_3, func_4, func_8, func_13, func_23`.

SN-RL-L1+2 soft tokens are initialized from the corresponding SN-SFT-L1 embeddings, ensuring atomic competence before RL training begins.

Table B2. **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-L1	epochs=2, lr=2×10^{-5}, max length=3072, completion-only loss, effective batch=16.
FT-SFT-L1-CL	Same as FT-SFT-L1; trained sequentially one function at a time.
FT-SFT-L1+2	epochs=1, lr=2×10^{-5}, max length=3072, completion-only loss, effective batch=16.
FT-RL-L1+2 (SFT warmup)	epochs=1, lr=2×10^{-5}, max length=3072, effective batch=16.
FT-RL-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-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-L1+2	Initialized from SN-SFT-L1 embeddings; epochs=2, lr=5×10^{-3}, max length=3072, effective batch=16.
SN-RL-L1+2	Initialized from SN-SFT-L1 embeddings; lr=5×10^{-4}, max steps=150, generations=8, max completion length=800, $\beta = 0$, effective batch=128, constant LR schedule.

B.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$\times$internal-train), $\tau_{\text{OOD-ho}}$ (new$\times$internal-holdout), and $\tau_{\text{OOD-new}}$ (new$\times$new), computed over $\approx$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.

B.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.

B.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)
```

B.7. Hardware

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