Notes to Self: Can LLMs Benefit from Experiential Abstractions?

Chang Liu, Xinyu Li, and Artur Dubrawski
Auton Lab, Carnegie Mellon University
{changl8,xinyul2,awd}@cs.cmu.edu

Abstract

Humans distill experience into reusable *abstractions*, e.g., strategies and cautionary reminders, and apply them to gradually solve problems more effectively. We study whether Large Language Models (LLMs) can similarly benefit from such experiential abstractions. From LLMs' solution traces on a training set, a stronger teacher or the LLMs themselves extract natural-language abstractions into a retrievable library. We explore two modes of abstraction usage: (1) inference-time retrieval and (2) reinforcement learning (RL) with abstraction-augmented training prompts. Experiential abstractions improve LLM performance on mathematical and logical reasoning benchmarks. Self-extracted abstractions match teacher-extracted ones, and our abstraction usage framework can transfer to other datasets and models. These findings suggest LLMs can extract and apply experiential abstractions much as humans leverage distilled experience.

1 Introduction

Humans actively filter, compress, and internalize past experiences into reusable *abstractions*, e.g., heuristics, strategies, and cautionary reminders (Zsambok and Klein, 2014), rather than solving every new problem from scratch or merely accumulating memory: a student compiles a cheatsheet of recurring tricks; a technician carries a mental catalog of techniques. These experiential abstractions act as cognitive toolkits for increasingly complex tasks (Gick and Holyoak, 1983; Polya, 1945). As large language models (LLMs) advance on complex reasoning tasks such as mathematics (Shao et al., 2024), we ask an analogous question: can they similarly benefit from experiential abstractions derived from their own reasoning experience?

Recent work explores experiential learning in LLMs (Zhao et al., 2024; Suzgun et al., 2026; Didolkar et al., 2025; Qu et al., 2025; Xia et al., 2026), but typically relies on large frontier models as the target or as a teacher supervising smaller students, leaving unclear whether student LLMs can themselves extract or use such abstractions.

We assess the opportunity of bridging this gap by deriving a library of experiential abstractions from a target LLM's own training traces, with extraction by either a stronger teacher model or the target LLM itself. We study two modes of abstraction use (Figure 1): (1) inference-time retrieval into the test prompt, and (2) RL fine-tuning with abstraction-augmented training prompts.

We find that experiential abstractions extracted from the `MATH` (Hendrycks et al., 2021) training set can improve LLM mathematical reasoning abilities on `MATH-500` (Lightman et al., 2024) for both modes of abstraction usage, with consistent gains from abstraction-augmented RL post-training. Moreover, self-extracted abstractions yield gains comparable to teacher-extracted ones. Additionally, our abstraction-usage pipeline can transfer to other datasets and domains (e.g., logical reasoning), and abstractions from one model's traces can benefit a different target. These results suggest that LLMs can effectively extract and apply experiential abstractions from their own reasoning experience, incrementally self-improving their abilities as their use expands. The code of this paper is available at `https://github.com/autonlab/Notes-to-self`.

2 Related work

Several recent works focusing on eliciting LLMs' ability to learn from experience heavily rely on large or frontier LLMs. ExpeL (Zhao et al., 2024) extracts insights from agent trajectories and applies them at inference to a frozen frontier LLM in agentic environments. Dynamic Cheatsheet (Suzgun et al., 2026) maintains a test-time-evolving memory for black-box frontier LLMs across rea-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 43634–43645
October 24-29, 2026 ©2026 Association for Computational Linguistics

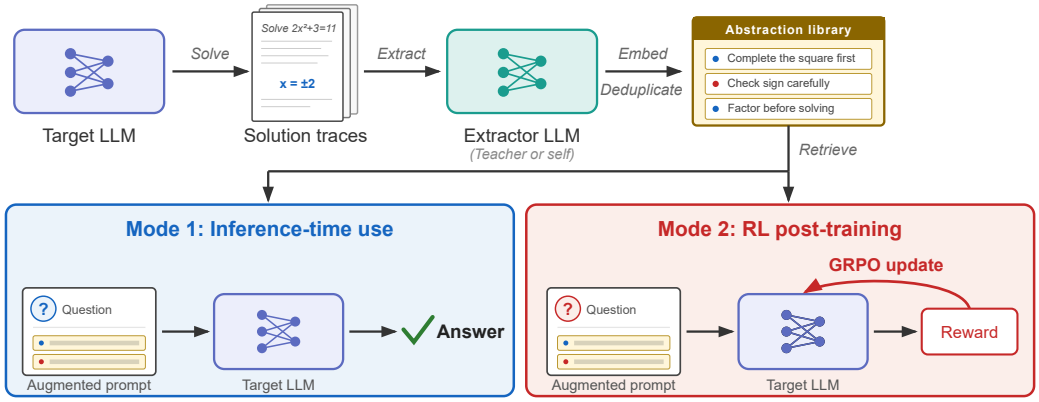


Figure 1: **Pipeline Overview.** An extractor LLM (frontier teacher or itself) distills the target LLM's training traces into a library of natural-language abstractions, used in two modes: (1) inference-time retrieval into the test prompt, and (2) GRPO (Shao et al., 2024) post-training on abstraction-augmented prompts.

soning queries. Metacognitive Reuse (Didolkar et al., 2025) extracts named behaviors from a 70B model's own traces and uses them in-context, in self-improvement loops, or as supervised fine-tuning on behavior-conditioned traces. RLAD (Qu et al., 2025) co-trains an abstraction generator alongside the solver via two-player RL, producing abstractions on the fly per problem, yet uses a strong teacher model to supervise the abstraction generator. SkillRL (Xia et al., 2026) maintains a hierarchical skill library compiled by a frontier teacher model, applies cold-start SFT to teach abstraction use, then runs RL with skills evolving during training, in agentic settings.

In this work, we ask whether small student LLMs themselves can extract and benefit from such abstractions through outcome-reward RL alone, without a frontier model at inference or teacher demonstrations for training. Moreover, our approach differs from reflection-style methods (Shinn et al., 2023; Madaan et al., 2023), which are a form of test-time scaling, leveraging multiple feedback and regeneration cycles to improve performance with additional test-time compute. Our approach instead emphasizes train-to-test knowledge transfer, distilling the insights from training into a library of abstractions reusable on unseen test instances.

3 Methods

We denote the target LLM by π_θ with parameters θ, its solution trace on a question q by $y \sim \pi_\theta(\cdot \mid q)$, and the abstraction library by $\mathcal{L} = \{a_1, \ldots, a_N\}$, where each a_i is a natural-language abstraction. A sentence embedder $\phi(\cdot)$ maps both questions and abstractions to a shared vector space for retrieval. We use outcome reward $r(y, y^*) \in \{0, 1\}$, set to one when the final answer in y matches the ground truth y^*. Figure 1 summarizes the full pipeline: we build a library of experiential abstractions from a target LLM's training traces and study two modes of using it.

3.1 Building the library of experiential abstractions

We first collect solution traces of the target LLM on a training set (e.g., the `MATH` (Hendrycks et al., 2021) training split of 7,500 problems) and sample one trace per problem from π_θ. Each trace is labeled correct or incorrect by comparing its final answer with the ground truth.

We then prompt an extractor, either the target LLM itself or a stronger teacher, to summarize each trace into one or two reusable abstractions. Given the problem, the solution trace, the ground-truth answer, and the correctness label, the extractor is instructed to produce single-sentence, reusable abstractions that are free of problem-specific numbers, variable names, or the answer itself.

Extraction is conditioned on correctness: for correctly solved problems the extractor identifies key reasoning moves worth reusing, producing *strategy* abstractions, while for failed problems it identifies mistake patterns to avoid, producing *caution* abstractions. We show the full extraction prompts in Appendix F.1.

To construct the library, we embed each abstraction with the `all-MiniLM-L6-v2` sentence encoder (Reimers and Gurevych, 2019) into 384-dimensional ℓ_2-normalized vectors. We remove near-duplicates by clustering abstractions with cosine-similarity union-find at a threshold of 0.85, keeping strategy and caution entries in separate

buckets, and storing the surviving entries. This deduplication scheme effectively reduces the library size by approximately 30%, as shown in Table 8 in Appendix C.

3.2 Inference-time use of abstractions

To use abstractions for inference, we retrieve relevant abstractions from the library for each test question and inject them into the prompt. Given the query representation $\tilde{q}$ of a question q, we retrieve top-k (k=6 in our experiments) similar library entries by cosine similarity:

$$\mathcal{R}_q = \underset{a \in \mathcal{L}}{\text{top-k}} \cos\big(\phi(\tilde{q}), \phi(a)\big). \quad (1)$$

We construct $\tilde{q}$ in one of two ways. The default query prepends the problem's subject tag to its text, "[{subject}] {question}", and falls back to the plain question when no subject is available. We additionally consider a rewritten query, obtained by prompting the LLM to restate the problem as a single abstraction-style sentence (Appendix F.2). Retrieval is score-gated: we retrieve with both queries and adopt the rewritten one only when its top-1 cosine score exceeds the original's by more than 0.02, breaking ties in favor of the original.

The retrieved set $\mathcal{R}_q$ is grouped by type (strategy/caution) and inserted into the prompt as a `<reference_notes>` block, placed after the question. The LLM then generates its solution from the augmented prompt $x = \text{Prompt}(q, \mathcal{R}_q)$ in a single forward pass (see Appendix F.3 for the prompts).

3.3 Abstraction-augmented RL post-training

Here, abstractions enter during RL fine-tuning rather than only at inference. We apply the above retrieval procedure to each training question q, obtaining its abstraction set $\mathcal{R}_q$, and fine-tune the target LLM with GRPO (Shao et al., 2024) so that the policy conditions on the pair $(q, \mathcal{R}_q)$ rather than on q alone.

For each training question, GRPO samples a group of candidate solutions $y \sim \pi_\theta(\cdot \mid q, \mathcal{R}_q)$, scores each with the outcome reward $r(y, y^*)$, and normalizes rewards within the group to form the advantage $\hat{A}(q, y)$. The policy is updated to maximize

$$\mathcal{J}(\theta) = \mathop{\mathbb{E}}_{\substack{q \sim \mathcal{D} \\ y \sim \pi_\theta(\cdot \mid q, \mathcal{R}_q)}} \big[\hat{A}(q, y)\, \log \pi_\theta(y \mid q, \mathcal{R}_q)\big], \quad (2)$$

where $\mathcal{D}$ is the set of training questions and $\pi_\theta(\cdot \mid q, \mathcal{R}_q)$ denotes the policy applied to the prompt that combines q with its retrieved abstractions (Section 3.2). In short, the reward signal and optimization match those of standard GRPO, while the policy is now conditioned on $(q, \mathcal{R}_q)$. After training, the model can be evaluated with or without abstraction-augmented prompts at test time.

4 Results

We evaluate two open-source LLMs `Llama-3.2-3B-Instruct` (Grattafiori et al., 2024) and `Qwen-2.5-1.5B-Instruct` (Qwen et al., 2025), whose quantitative reasoning abilities are less saturated than newer models. We adopt `DeepSeek-V4-Flash` (DeepSeek-AI et al., 2026) as the teacher model for abstraction extraction. We report `pass@1/pass@8`, averaged over 8 rollouts at temperature 0.6.

Experiential abstractions improve mathematical reasoning. We first evaluate on the `MATH-500` test set using `DeepSeek-V4-Flash`-extracted abstractions from the MATH training split and compare five configurations of abstraction usage: Baseline (no fine-tuning or abstractions); Inference$_{\text{abs}}$ (abstractions injected at test time only); GRPO (vanilla GRPO, no abstractions); GRPO$_{\text{train}}$ (GRPO with abstractions injected in training prompts only); and GRPO$_{\text{train+test}}$ (abstractions injected in both training and test prompts).

As shown in Table 1, inference-time injection alone can be helpful: it raises `Llama`'s `pass@1` by $+3.31$ and `pass@8` by $+2.23$ over the baseline, though slightly lowering both for `Qwen`, likely due to answer formatting and the quality of retrieved abstractions (Appendix B). Abstraction-augmented RL yields more consistent gains: GRPO$_{\text{train}}$ attains $+2.56$ over vanilla GRPO and $+6.34$ over baseline for `Llama` and $+0.89$ and $+3.21$ for `Qwen`, respectively. Notably, across all usage modes and models, self-extracted abstractions perform comparably with teacher-extracted ones, indicating that a small LLM can distill useful experiential abstractions from its own traces without supervision from a frontier model.

Interestingly, GRPO$_{\text{train+test}}$ underperforms GRPO$_{\text{train}}$, which we attribute to RL sharpening the model output distribution towards near-deterministic correct modes (Yue et al., 2025), on which the perturbation effect of abstractions can hurt performance (Appendix A).

We further perform an ablation study where models inherit the prompt structure of train-

Table 1: Performance (pass@1/pass@8) on MATH-500 across five configurations of abstraction usage, and teacher- vs. self- extracted abstractions.

Model	Abstraction extractor	Baseline	$\text{Inference}_{\text{abs}}$	GRPO	$\text{GRPO}_{\text{train}}$	$\text{GRPO}_{\text{train+test}}$
Llama	Teacher	43.11/70.57	46.42/**72.80**	46.89/70.54	**49.45**/71.98	47.65/70.23
	Self		46.30/71.66		49.07/71.14	47.67/70.69
Qwen	Teacher	50.23/76.47	49.14/75.09	52.55/75.55	53.44/76.34	52.66/75.76
	Self		49.44/74.92		**53.77/76.83**	52.72/75.22

Table 2: Ablation isolating prompt-template effects on MATH-500 (pass@1/pass@8), using Llama-3.2-3B-Instruct with teacher-extracted abstractions. *No notes* is the plain configuration; *Blank notes* inherits the same prompt structure but leaves the <reference_notes> block empty; + *Abstractions* fills it with retrieved abstractions.

Mode	No notes	Blank notes	+ Abstractions
Inference-time	Baseline	$\text{Inference}_{\text{blank}}$	$\text{Inference}_{\text{abs}}$
	43.11/70.57	46.02/70.00	46.42/72.80
RL (GRPO)	GRPO	$\text{GRPO}_{\text{blank}}$	$\text{GRPO}_{\text{train}}$
	46.89/70.54	48.55/70.98	49.45/71.98

ing/inference time injection but leave the <reference_notes> abstractions empty (Table 2). Both $\text{GRPO}_{\text{train}}$ and $\text{Inference}_{\text{abs}}$ outperform their blank controls on pass@1 and pass@8, showing the abstractions' unique advantage in extending model capabilities.

Abstraction-usage pipeline transfers to other datasets and models. We further assess how our abstraction-usage pipeline transfers to other datasets, domains, and models. We use the DeepSeek teacher for abstraction extraction in these experiments.

We first find that MATH-derived abstractions and model checkpoints induce gains on the harder OlympiadBench (He et al., 2024) but not the easier GSM8K (Cobbe et al., 2021) (Table 3), suggesting that abstractions help more on problems the model is less capable. This is also corroborated by further analyses on the MATH-500 results (Appendix A).

On the logical reasoning benchmark MuSR (Sprague et al., 2024), we observe that while MATH can only provide useful RLVR signals (pass@1: GRPO > $\text{GRPO}_{\text{train}}$ > Baseline), a logical reasoning dataset FOLIO (Han et al., 2024) can provide informative abstractions and extend model capability by inducing gains in pass@8 (79.37, $\text{GRPO}_{\text{train}}$, best configuration).

Lastly, Llama-derived abstractions can be transferred to Phi-3.5-mini-instruct (Abdin et al., 2024), but the blank template itself already harms Gemma models (Team et al., 2024, 2025) (Table 4), suggesting that transferability of abstractions is specific to the downstream model.

5 Conclusion

We studied whether LLMs benefit from experiential abstractions distilled from training traces, via either inference-time retrieval or abstraction-augmented RL post-training. Such abstractions extend model capability across mathematical and logical reasoning benchmarks. Moreover, small student LLMs extract abstractions with comparable effectiveness to frontier teachers, and abstractions can transfer across these target models. These findings indicate that experiential abstraction is a viable self-improvement signal for LLMs, mirroring how humans distill and reuse past experience.

Limitations

Our results depend on several fixed design choices: a particular teacher model, sentence encoder, retrieval cutoff k, and deduplication threshold, for which we do not provide a sensitivity analysis. Broader cross-model transfer claims would require additional pairs. RL post-training is conducted under a modest compute budget (one epoch, eight rollouts per prompt), and scaling the training could change the magnitude or ordering of the observed gains.

Our method targets mathematical problem solving and extends to logical reasoning, and we do not foresee direct societal risks. The main caveat is that an abstraction library can encode incorrect or misleading heuristics; outcome-reward RL mitigates this by reinforcing only verified-correct solutions.

Acknowledgments

This work was partially supported by the National Science Foundation awards 2406231 and 2530752,

Table 3: Out-of-distribution results (pass@1/pass@8) with Llama-3.2-3B-Instruct, spanning easier and harder math (GSM8K, OlympiadBench) and logical reasoning (MuSR). For MuSR, we report checkpoints trained (and abstractions extracted) on MATH or FOLIO.

Test set	Baseline	GRPO	$\text{GRPO}_{\text{blank}}$	$\text{GRPO}_{\text{train}}$
GSM8K	54.82/85.44	77.34/92.49	**81.30/93.33**	80.48/92.95
OlympiadBench	13.91/34.57	16.04/36.94	16.36/35.91	**17.34/37.24**
$\text{MuSR}_{\text{MATH}}$	40.94/78.70	**47.06**/75.79	45.21/76.59	46.20/75.93
$\text{MuSR}_{\text{FOLIO}}$		44.58/75.00	44.82/77.91	45.54/**79.37**

Table 4: Cross-model transfer of abstractions extracted from Llama-3.2-3B-Instruct traces (teacher-extracted), injected at inference into other model families on MATH-500 (pass@1/pass@8).

Target model	Baseline	$\text{Inference}_{\text{blank}}$	$\text{Inference}_{\text{abs}}$
gemma-2-2B-it	21.77/43.00	13.08/33.60	13.75/33.40
gemma-3-3B-it	44.88/62.40	42.02/62.40	40.15/60.00
Phi-3.5-mini -instruct	40.62/64.00	42.30/65.20	43.60/65.80

and by the US Department of Energy award DE-SC0026594.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, and 300 others. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. 2025. Metacognitive reuse: Turning recurring llm reasoning into concise behaviors. *arXiv preprint arXiv:2509.13237*.

Mary L Gick and Keith J Holyoak. 1983. Schema induction and analogical transfer. *Cognitive psychology*, 15(1):1–38.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,

Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

George Polya. 1945. *How to solve it: A new aspect of mathematical method*. Princeton university press.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. 2025. Rlad: Training llms to discover abstractions for solving reasoning problems. *arXiv preprint arXiv:2510.02263*.

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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, volume 2024, pages 14670–14728.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2026. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Caroline E Zsambok and Gary Klein. 2014. *Naturalistic decision making*. Psychology Press.

A Interpreting the effect of RL and abstractions

In the main text (Section 4) we observe that $\text{GRPO}_{\text{train+test}}$ consistently underperforms $\text{GRPO}_{\text{train}}$: re-injecting abstractions at test time hurts a model already fine-tuned with them. We attribute this to the interaction between abstractions and RL, and here we test two competing explanations. Under an *internalization* hypothesis, the fine-tuned model has already absorbed the abstractions and is merely distracted by their re-injection. Under an *exploration-vs-sharpening* account, abstractions perturb the output distribution to aid exploration, while RL sharpens that distribution toward near-deterministic correct modes, so perturbing an already-tuned model displaces probability mass away from correct solutions.

The internalization hypothesis does not hold. We evaluate three checkpoints—the base model (Baseline), plain GRPO (trained without abstractions), and $\text{GRPO}_{\text{train}}$—each with and without abstractions injected at test time (Table 5). Internalization predicts that test-time injection should harm only $\text{GRPO}_{\text{train}}$, the sole checkpoint exposed to abstractions during training. Instead, injection also degrades plain GRPO, which never encountered abstractions in training and therefore cannot have internalized them. Internalization alone thus cannot explain the drop.

Table 5: Effect of injecting abstractions at test time (`pass@1`) on `Llama-3.2-3B-Instruct` across three checkpoints. Injection degrades both RL-tuned checkpoints, including plain GRPO, which never saw abstractions during training.

Checkpoint	w/o abs.	w/ abs.	Δ
Baseline	43.11	46.42	+3.31
GRPO	46.89	45.76	−1.13
$\text{GRPO}_{\text{train}}$	49.45	47.65	−1.80

Abstractions help where the model is weak and hurt where it is strong. To locate the effect, we bucket each checkpoint's 500 `MATH-500` problems by their per-problem accuracy *before* injection, then measure the mean change in `pass@1` *after* inference-time injection within each bucket (Table 6). In the low-accuracy bucket (0–50%), where all three checkpoints struggle, abstractions help every checkpoint. In the mid bucket (50–100%), injection still helps the base model but hurts both RL-tuned checkpoints; in the fully-solved bucket (100%) it hurts all three. Abstractions thus aid the model on problems it is less capable of—encouraging exploration—but are detrimental once it is already competent. This directly corroborates the main-text observation that MATH-derived abstractions help the harder OlympiadBench (where the model is weak) but not the saturated GSM8K (where it is strong), in Table 3.

Table 6: Mean change in `pass@1` after inference-time abstraction injection, with problems bucketed by pre-injection per-problem accuracy. Abstractions help low-accuracy problems but hurt high-accuracy ones, increasingly so for RL-tuned checkpoints.

Accuracy bucket	Baseline	GRPO	$\text{GRPO}_{\text{train}}$
0–50%	+6.31	+3.62	+0.83
50–100%	+2.46	−5.22	−3.61
100%	−2.32	−5.29	−1.12

Exploration versus sharpening. These patterns are consistent with abstractions acting as a perturbation of the model's output distribution. For the untrained base model, whose distribution is comparatively diffuse, this perturbation broadens the search and raises coverage: injecting abstractions at test time lifts the base model's `pass@8` by +2.23 (Table 2). RLVR, however, is known to sharpen the output distribution toward near-deterministic correct modes (Yue et al., 2025). Once the model is RL-tuned, these modes already concentrate on the correct answers for problems it can solve, so perturbing them via test-time injection can only displace mass away from correct solutions—explaining why injection hurts RL-tuned checkpoints most on problems they already solve, and why `pass@8` gains from abstraction content shrink after RL tuning (e.g., $\textbf{GRPO}_{\text{train}}$ raises `pass@8` by +1.44 over vanilla **GRPO**, versus the base model's +2.23). Our logical-reasoning results reinforce this: on `MuSR` (Table 3), $\textbf{GRPO}_{\text{train}}$ is the only configuration that recovers the `pass@8` drop incurred by both **GRPO** and $\textbf{GRPO}_{\text{blank}}$.

B Quantitative analyses on `Qwen-2.5-1.5B-Instruct` results

Inference-time abstraction injection improves `Llama-3.2-3B-Instruct` but slightly degrades

Qwen-2.5-1.5B-Instruct (50.23 → 49.14 pass@1; Table 1). We analyze the trajectories of Qwen's Inference$_{abs}$ against its Baseline (standard chain-of-thought prompt) across all 500 MATH-500 problems, and find that degradation is driven by *format disruption and poor retrieval fit* rather than by retrieved cautions acting as negative constraints.

Retrieved cautions do not act as negative constraints. A natural hypothesis is that retrieved *caution* notes over-constrain Qwen and suppress its standard reasoning; if so, retrieving more cautions should worsen performance. We observe the opposite: the number of cautions retrieved is not positively correlated with degradation (Pearson $\rho = +0.069$, $p = 0.13$), and problems with more retrieved cautions are, if anything, less degraded (Table 7).

Table 7: Mean change in pass@1 (Inference$_{abs}$ − Baseline) for Qwen-2.5-1.5B-Instruct, grouped by the number of caution notes retrieved. More cautions do not induce more degradation.

# Cautions retrieved	# Problems	Mean Δ(pass@1)
0–1	143	−2.08
2–3	233	−1.15
4–6	124	+0.15
Total	500	−1.09

Format sensitivity. Comparing per-problem trajectories, abstraction injection improves Llama's answer-format compliance but harms Qwen's: the rate of missing \boxed{} answers falls for Llama (7.3% → 4.2%) but rises for Qwen (3.8% → 4.6%). This disruption alone accounts for Qwen's most-degraded problems. For instance, on Qwen's single most-degraded problem (id 1332), 24/32 **Inference$_{abs}$** samples derive the correct answer but omit the boxed format and are scored incorrect (Baseline: 11/32 correct); no cautions are retrieved and the chain of thought remains intact, isolating the failure to formatting.

Retrieval fit. Degradation also arises when retrieval surfaces abstractions that fit the query poorly. On a problem asking for the least possible sum of distinct positive integers with a fixed product (id 203), Llama retrieves an apt minimum-sum strategy ("distribute the prime factors so that the resulting numbers are as close in magnitude...") and adopts it near-verbatim, rising from 22% to 63%. Qwen instead retrieves a strategy for the *opposite* objective ("To maximize the sum of distinct positive integers with a fixed product..."), and its failing samples follow that prescription, dropping from 50% to 16%.

Together, these analyses indicate that Qwen's degradation under inference-time injection stems from disrupted answer formatting and occasional poorly-matched retrievals, not from cautions functioning as harmful constraints.

C Effect of library deduplication

Table 8 shows the effect of deduplicating the abstraction library as described in Section 3.1.

Table 8: The effect of library deduplication process. Each cell corresponds to the size of the library before→after deduplication, for a target model-extractor pair.

	Teacher	Self
Llama-3.2-3B-Instruct	10915 → 7676	6727 → 4232
Qwen-2.5-1.5B-Instruct	11332 → 8274	5647 → 3561

D Implementation details

D.1 Hardware compute

All experiments are run on an NVIDIA DGX Station A100 machine, using all 4 NVIDIA A100 80GB GPUs on the DGX Station. Each single training run on the MATH dataset takes about 2.5 hours. Each inference run on the MATH-500 dataset takes about 0.3 hours. Extracting MATH traces takes about 0.5 hours.

D.2 Teacher model access

We access the DeepSeek-V4-Flash teacher model via API granted by OpenRouter.

D.3 Hyperparameters

We report the training and inference hyperparameter configurations in Table 9 and 10. These configurations persist across all used models.

D.4 Packages

GRPO training is implemented with verl (Sheng et al., 2024). Final-answer correctness against the ground truth is determined by extracting the "\boxed" block and string-matching it against the ground truth. We did not perform a hyperparameter search; the values in Tables 9 and 10 follow standard GRPO settings.

D.5 Use of AI assistants

We used Claude (Anthropic) via the web interface for assistance with writing and editing, and the Cursor IDE (with an integrated LLM assistant) for coding support.

Table 9: Training hyperparameters (GRPO).

Hyperparameter	Value
Model & Data	
Max prompt length	1536
Max response length	3072
Training batch size	128
Optimization	
Learning rate	1×10^{-6}
LR schedule	Cosine
Warmup ratio	0.1
Epochs	1
PPO mini-batch size	128
Algorithm (GRPO)	
Advantage estimator	GRPO
KL loss coefficient	5×10^{-4}
KL loss type	Low-variance KL
KL in reward	×
Entropy coefficient	0
Rollouts per prompt	8

Table 10: Evaluation hyperparameters.

Hyperparameter	Value
Samples per problem (n)	8
Temperature	0.6
Top-p	0.95
Top-k	20
Max prompt length	1536
Max response length	3072

E Artifacts and licenses

The datasets, target and teacher models, and sentence encoder used in this work are publicly released research artifacts, used here solely for non-commercial research consistent with their intended use. Among the datasets, `MATH`, `GSM8K`, `OlympiadBench`, and `MuSR` are distributed under the MIT License; and `FOLIO` under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA 4.0). Among the models, `Llama-3.2-3B-Instruct` is used under the Llama 3.2 Community License; `Qwen-2.5-1.5B-Instruct`, `Phi-3.5-mini-instruct`, and `all-MiniLM-L6-v2` under Apache 2.0; and `gemma-2-2b-it` and `gemma-3-3b-it` under the Gemma Terms of Use. `DeepSeek-V4-Flash` is accessed through the OpenRouter API under the provider's terms of service. The datasets consist of competition mathematics, grade-school math, and logical-reasoning problems, and contain no personally identifying information or offensive content.

F Prompts used in the paper

F.1 Prompt for extracting abstractions from traces

System prompt

```
You are extracting reusable mathematical
    reasoning abstractions from a model
    trace.

Your job is to read one math problem, the
     ground-truth final answer, one model
    -generated trace, and a correctness
    label telling you whether the trace
    is correct or incorrect.

Produce 1 or 2 reusable abstractions only.
     Each abstraction must be:
- a single sentence in "principle"
- generic and reusable across similar
    problems
- free of problem-specific numbers,
    variable names, answer values, or
    story details
- phrased as reasoning advice, not as a
    restatement of the solution

Allowed abstraction types:
- "strategy": a positive reasoning move
    that should be reused
- "caution": a mistake pattern or failure
     mode that should be avoided

Output a JSON array with 1 or 2 objects.
    Each object must contain exactly
    these keys:
- "name"
- "type"
- "principle"
- "when_to_apply"
- "domain"

Important rules:
- Do not copy phrases from the trace
    unless necessary.
- Do not mention the final answer
    explicitly.
- Do not mention concrete numbers unless
    they are universal constants like 0,
    1, or pi.
```

```
- If the trace is correct, prefer
    strategy abstractions, but you may
    include one caution if the reasoning
    contains a risky pattern.
- If the trace is incorrect, prefer
    caution abstractions, but you may
    include one strategy if you can
    clearly infer a better reasoning move
     from the problem and the ground-
    truth answer.
- If there is only one strong abstraction,
     output only one.
```

User prompt for correct problems

```
Problem:
{problem_text}

Ground-truth final answer:
{ground_truth}

Model trace:
{trace_text}

Correctness label:
{label}

Task:
Extract 1 or 2 reusable one-sentence
    abstractions from this example.

Additional guidance:
This trace reached the correct final
    answer. Focus on the key reasoning
    move(s) worth reusing in future
    problems. Prefer strategy
    abstractions. You may include one
    caution only if the trace contains a
    potentially dangerous shortcut that
    happened to work here.

Return only a JSON array.
```

User prompt for incorrect problems

```
Problem:
{problem_text}

Ground-truth final answer:
{ground_truth}

Model trace:
{trace_text}

Correctness label:
{label}

Task:
Extract 1 or 2 reusable one-sentence
    abstractions from this example.

Additional guidance:
This trace did not reach the correct
    final answer. Focus on the main
    mistake pattern that should be
    avoided in future problems. Prefer
    caution abstractions. You may include
     one strategy only if a clearly
    better reasoning direction is
    recoverable from the problem and the
    ground-truth answer.

Return only a JSON array.
```

F.2 Prompt for rewriting input problems

System prompt

```
You are a math tutor writing retrieval
    abstractions. Output exactly one line
    , one sentence, in this format:
<Type>: <short abstraction>, where <Type>
     is either Strategy or Caution. For
    example \"Check extraneous roots\".
    Do not solve the problem and do not
    provide a final answer.
```

User prompt

```
Problem:
{text}

Abstraction (Strategy or Caution):
```

F.3 Prompt for problem solving

System prompt for abstraction-augmented problem-solving

```
    <reference_notes>
The following notes were retrieved from a
     library of lessons distilled
from past attempts on similar problems.
    They MAY OR MAY NOT apply to
the current problem.

How to use them:
- STRATEGIES are candidate approaches -
    adopt one only if it directly fits.
- CAUTIONS are common errors - before
    giving the final answer, verify
  you are not making any that apply here.
- If none apply, solve the problem
    normally.
- Properly cite the retrieved notes in
    your answer.

## Strategies
- When simplifying an expression, combine
     like terms by performing the
    indicated arithmetic operation on
    their coefficients.
  when: When simplifying linear algebraic
     expressions involving addition or
    subtraction of terms with the same
    variable and exponent.
```

```
- Factor quadratic equations when
    possible to find their roots.
  when: When a quadratic can be factored
    over the integers.

## Cautions (verify at the end)
- When solving inequalities involving
    fractions, ensure the denominator is
    not zero.
  when: When solving rational
    inequalities or equations with
    variable denominators.
</reference_notes>

Please reason step by step, and put your
    final answer within \boxed{}.
```

Vanilla system prompt

```
Please reason step by step, and put your
    final answer within \boxed{}.
```

User prompt

```
{Problem text}
```

G Example inference traces on MATH-500 with abstractions

Figure 2 shows two complete inference traces on `MATH-500`, with the retrieved abstraction block highlighted in yellow.

Example 1 — Divisors of 196

[SYSTEM] Cutting Knowledge Date: December 2023. Today Date: 16 May 2026.

[USER] How many positive whole-number divisors does 196 have?

<reference_notes> (retrieved abstractions)

Strategies

- When determining the number of divisors of a number, use its prime factorization to calculate the number of divisors.
- A number has exactly three positive divisors if and only if it is the square of a prime number.
- The next divisor must be a multiple of either 17 or 19, as 323 is a product of these two primes.

Cautions

- A number with more prime factors does not necessarily have more divisors.
- When finding divisors, consider both positive and negative factors.
- Ensure the prime factorization is correct before applying the divisor formula.

</reference_notes>

[ASSISTANT]

Step 1: Determine the prime factorization of 196.
To find the number of divisors, we first need the prime factorization of 196. Divide by the smallest primes until irreducible.

Step 2: Prime factorization of 196.
$196 \div 2 = 98,\ 98 \div 2 = 49,\ 49 = 7 \times 7.$

Step 3: Write the prime factorization.
$196 = 2^2 \times 7^2.$

Step 4: Apply the divisor-count formula.
$(2+1)(2+1) = 3 \times 3 = 9.$

Step 5: Verify cautions.
This problem does not involve comparing numbers by prime-factor count. It asks for positive divisors only, so negative factors are excluded. The prime factorization has been correctly identified.

The final answer is: $\boxed{9}$

Example 2 — Distance between two points

[SYSTEM] Cutting Knowledge Date: December 2023. Today Date: 16 May 2026.

[USER] What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form.

<reference_notes> (retrieved abstractions)

Strategies

- Use $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ to find the distance between two points in a coordinate plane.
- When given speed and time, use distance = speed $\times$ time.
- Use the distance formula for two points in 3D space.
- For equidistant points, set up an equation via the distance formula.
- When given a point on a circle, use the distance formula to find the radius.

Cautions

- In the complex plane, use $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ rather than $|z_2 - z_1|$.

</reference_notes>

[ASSISTANT]

Step 1: Identify the coordinates.
The two points are $(2, -6)$ and $(-4, 3)$.

Step 2: Apply the distance formula.
$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$

Step 3: Substitute the coordinates.
$d = \sqrt{(-4 - 2)^2 + (3 - (-6))^2}.$

Step 4: Simplify.
$d = \sqrt{(-6)^2 + (9)^2} = \sqrt{36 + 81} = \sqrt{117}.$

Step 5: Simplify the radical.
$\sqrt{117} = \sqrt{9 \cdot 13} = 3\sqrt{13}.$

The final answer is: $\boxed{3\sqrt{13}}$

Figure 2: Two inference examples in the MATH-500 test dataset. The yellow-shaded block in each prompt is the retrieved abstraction injected at inference time. Note that the user question is shown first for readability; the abstraction block follows.