Reasoning Depth and Environment Complexity: A Controlled Study of RLVR Data Allocation across Logical Reasoning Tasks

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become central to post-training reasoning models, yet a key limitation of existing studies is their narrow view of the reasoning space: difficulty is treated as reasoning depth alone, and reward is concentrated on forward deductive state tracking. We instead characterize the reasoning space along two dimensions. **Difficulty.** Beyond reasoning depth, we study environment complexity, where models must identify the correct path amid distractors and interacting structures. **Rewarded reasoning form.** We consider four abilities core to real-world reasoning: deductive state tracking, abductive recovery of hidden events or facts, inductive rule induction, and analogical transfer. To disentangle these factors, we construct a synthetic knowledge-graph environment with controlled pre- and post-training distributions, where each instance varies along depth, complexity, and task family. Three findings emerge: joint depth–complexity coverage outperforms single-axis recipes; reasoning families respond non-uniformly, with abductive reasoning degrading outside the RL-covered region and task correlations clustering into deductive–abductive and inductive–analogy pairs; and uniform mixing outperforms staged curricula under a fixed budget. We also find that recent off-the-shelf models exhibit the same deductive-over-abductive asymmetry, suggesting that this gap is not merely an artifact of our controlled setup.

1 Introduction

Reinforcement learning with verifiable rewards has become a central post-training paradigm for improving the reasoning ability of language models. By rewarding outputs that can be automatically verified, RLVR has enabled substantial progress on mathematical reasoning, code generation, STEM problem solving, and long-horizon reasoning (DeepSeek-AI, 2025; OpenAI, 2026b; Anthropic, 2026; OpenAI, 2025). However, the success of RLVR also exposes a more fundamental design question: **How should RL training examples be distributed across the reasoning space during post-training?** This distribution determines which reasoning behaviors are reinforced and generalized, yet how to shape it across difficulty levels and forms of reasoning remains poorly understood.

A natural dimension of this distribution is reasoning difficulty. Existing post-training studies have examined difficulty through a depth-oriented lens, emphasizing longer reasoning chains, harder compositional structures, longer reasoning horizons, or training near the model's competence boundary (Zhang et al., 2025a; Lu et al., 2025; Yang et al., 2025; Huang et al., 2026; Wang et al., 2026). However, as illustrated in Figure 1(a), a model can fail along two distinct axes: the target reasoning path may be long, requiring many sequential inferences, or the path could be buried in a complex environment of distractors and interacting structures. Difficulty in RLVR data thus has two distinct axes: *reasoning depth*, which stresses multi-step commitment, and *environment complexity*, which stresses relevant-structure selection. This motivates our first research question **RQ1: How should RL training examples be allocated across reasoning depth and environment complexity during RLVR post-training?** In particular, under a fixed post-training budget, should training data concentrate on one difficulty axis, jointly cover both axes, or follow a staged curriculum over the depth–complexity space?

Apart from difficulty, a second dimension is the form of reasoning that receives reward. Existing reasoning-oriented post-training mainly targets forward, deductive tasks such as mathematics and code (DeepSeek-AI, 2025; Abdin et al., 2025; Tang et al., 2024). However, logical reasoning is broader: abductive inference recovers hidden causes, inductive reasoning extracts rules from instances, and

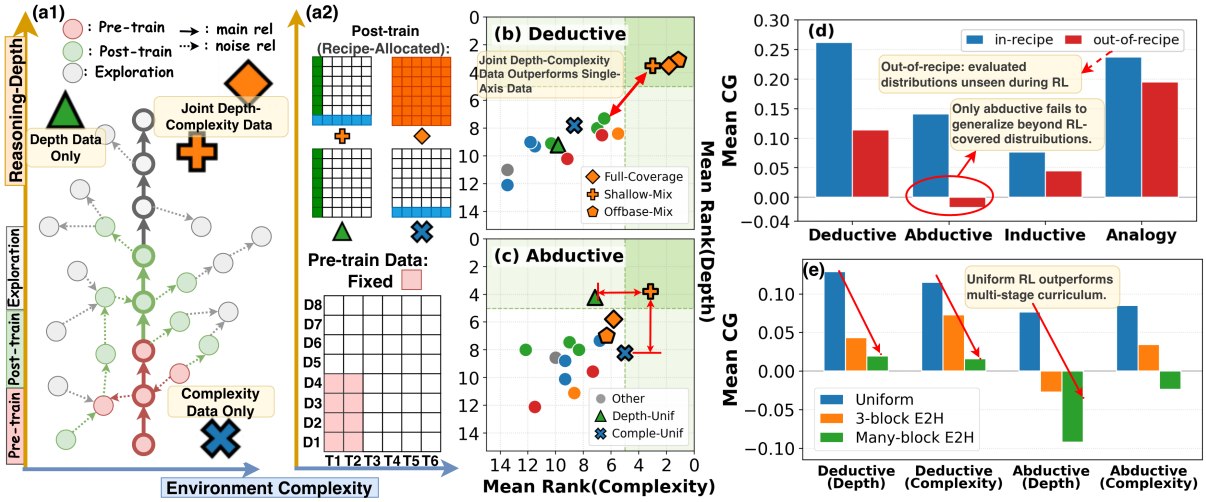


Figure 1: Two-axis data distribution and three main findings. (a) A knowledge-graph schematic shows reasoning depth and environment complexity. (b,c) Finding 1: joint depth–complexity coverage outperforms single-axis recipes. (d) Finding 2: abductive reasoning generalizes poorly outside the RL-covered data distribution. (e) Finding 3: uniform RL outperforms multi-stage curriculum under a fixed budget.

analogy transfers relational patterns across cases. This motivates our second research question **RQ2: Do different logical reasoning families behave uniformly under RLVR post-training?** We ask whether reasoning forms differ in how they use context, generalize beyond training coverage, and respond to the same post-training distribution.

Answering these questions with off-the-shelf models is difficult: their pre-training distributions are not observable, so post-training gains cannot be cleanly attributed to a specific RLVR data distribution. A controlled experimental setting is therefore necessary to isolate how that distribution shapes reasoning improvement. We construct a synthetic knowledge-graph reasoning environment in which both pre-training and post-training distributions are explicitly specified. Each data instance is grounded in a knowledge graph that combines a dynamic event-state component with a static kinship component. As shown in Figure 1(a), we vary two orthogonal dimensions of difficulty: reasoning depth D, which controls the length of the target reasoning chain, and environment complexity T, which controls the amount of distractor structure around that chain, including depth-focused, complexity-focused, jointly covered, or curriculum-based data recipes. From the same underlying world, we generate instances of four reasoning families: deductive state tracking, abductive recovery of missing events or facts, inductive rule induction, and analogy-based relation transfer. We first pre-train a model on shallow reasoning depth, low-complexity regions, and then apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024) post-training under controlled data recipes, that vary coverage over the $D \times T$ grid and reasoning families.

This controlled setting yields three main findings. **Joint depth–complexity coverage produces broader RLVR gains.** (Figure 1(b,c)) Jointly covered recipes produce more robust gains across the grid space over only reasoning depth or only environment complexity recipes. **Reasoning families exhibit distinct post-training behavior.** The four task families are not interchangeable RL targets: The task-pair correlations cluster into deductive–abductive and inductive–analogy pairs. Abductive reasoning is distribution sensitive and can even degrade outside the relevant training region (Figure 1(d)). Moreover, evaluation of off-the-shelf LLMs reproduces such deductive-vs.-abductive asymmetry, performing poorly on abductive reasoning instances. **Uniform mixing is stronger than staged curriculum under a fixed budget.** (Figure 1(e)) Easy-to-easy-to-hard schedules help harder regions but lose global coverage. Together, these findings show that RLVR data distribution shapes not only how much reasoning improves, but also where improvement occurs, how it transfers, and which reasoning forms benefit.

2 Methodology

We define a controlled KG reasoning framework. Appendices A.2 and A.3 give full data-generation and training details.

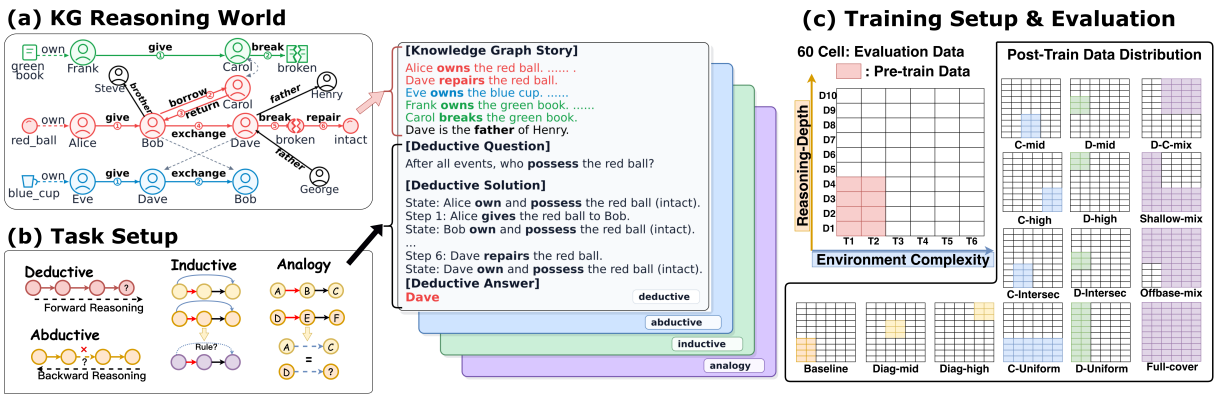


Figure 2: Controlled KGRL framework. The figure summarizes the synthetic KG world, four reasoning task families, and pre-training/RL post-training data distributions used to study depth–complexity coverage.

2.1 Controllable KG Reasoning World

Graph Definition Each reasoning instance is grounded in a knowledge graph

$$\mathcal{G} = (\mathcal{P}, \mathcal{O}, \mathcal{E}_{\text{dyn}}, \mathcal{E}_{\text{stat}}, s_0),$$

where $\mathcal{P}$ is a set of persons, $\mathcal{O}$ is a set of objects, $\mathcal{E}_{\text{dyn}}$ is a time-ordered sequence of dynamic events, $\mathcal{E}_{\text{stat}}$ is a set of static kinship relations over persons, and s_0 is the initial world state.

Dynamic and Static Worlds The generator separates each story into dynamic and static components. The dynamic event-state component updates object-centered facts such as ownership, possession, and integrity, while the static kinship component stores fixed relations among people. Figure 2(a) shows a toy instance: the red-ball chain tracks events from Alice's initial ownership to Dave's final possession, and the black kinship edges provide fixed relational facts in the same story.

2.2 Task Setup

Logical Reasoning Task We instantiate four logical task families from the same controlled graph world. Deductive and abductive tasks use the dynamic component: deductive tasks track observed events forward to a queried state value, while abductive tasks recover a missing event or initial fact consistent with the observed states. Inductive and analogy tasks use the static component: inductive tasks infer a shared relation rule from support examples, while analogy tasks transfer a relation pattern from one entity pair to another.

Reasoning Depth Reasoning depth counts dynamic operation events on the causal chain of the target object:

$$D(\mathcal{G}) = |\{e_i \in \mathcal{A}_{\text{op}},\ \text{obj}(e_i) = o^\star\}|\,.$$

Here $o^\star$ is the target object, $\text{obj}(e_i)$ returns the affected object, and $\mathcal{A}_{\text{op}}$ is the set of dynamic operation events. Figure 1(a) visualizes this as the vertical reasoning-depth axis.

Environment Complexity The environment complexity $T(\mathcal{G}) \in \{1, \ldots, 6\}$ indexes a structural parameter map:

$$\Theta(T) = (k_{\text{chains}}, n_{\text{persons}}, \rho_{\text{crossover}}, \rho_{\text{exchange}}).$$

Here $\Theta(T)$ controls the number of parallel chains, person scale, cross-chain entity overlap, and exchange frequency. Higher T adds distractors and cross-chain interactions around the target chain. Figure 1(a) provides a schematic example.

2.3 Multi-Grained Evaluation

Strict Correctness Following prior work (Zhang et al., 2025a), we evaluate every generated solution with a strict process-verified criterion. Given gold trace π, gold answer $a^\star$, predicted trace $\hat{\pi}$, and predicted answer $\hat{a}$, a solution is correct only when both process and answer match:

$$\text{correct}(\hat{\pi}, \hat{a} \mid \pi, a^\star) = \text{m}_{\text{P}}(\hat{\pi}, \pi) \wedge \text{m}_{\text{A}}(\hat{a}, a^\star).$$

This rule applies to all four task families. Here m_{P} is task-specific process matching, and m_{A} is answer matching. We report strict process-verified $\text{pass@}k$ for $k \in \{1, \ldots, 128\}$. Ceiling Gain (CG) is the post-training gain in sampled reasoning ceiling: $\text{CG} = \text{pass@128}_{\text{post}} - \text{pass@128}_{\text{pre}}$. Single Gain (SG) is the corresponding single-sample gain: $\text{SG} = \text{pass@1}_{\text{post}} - \text{pass@1}_{\text{pre}}$.

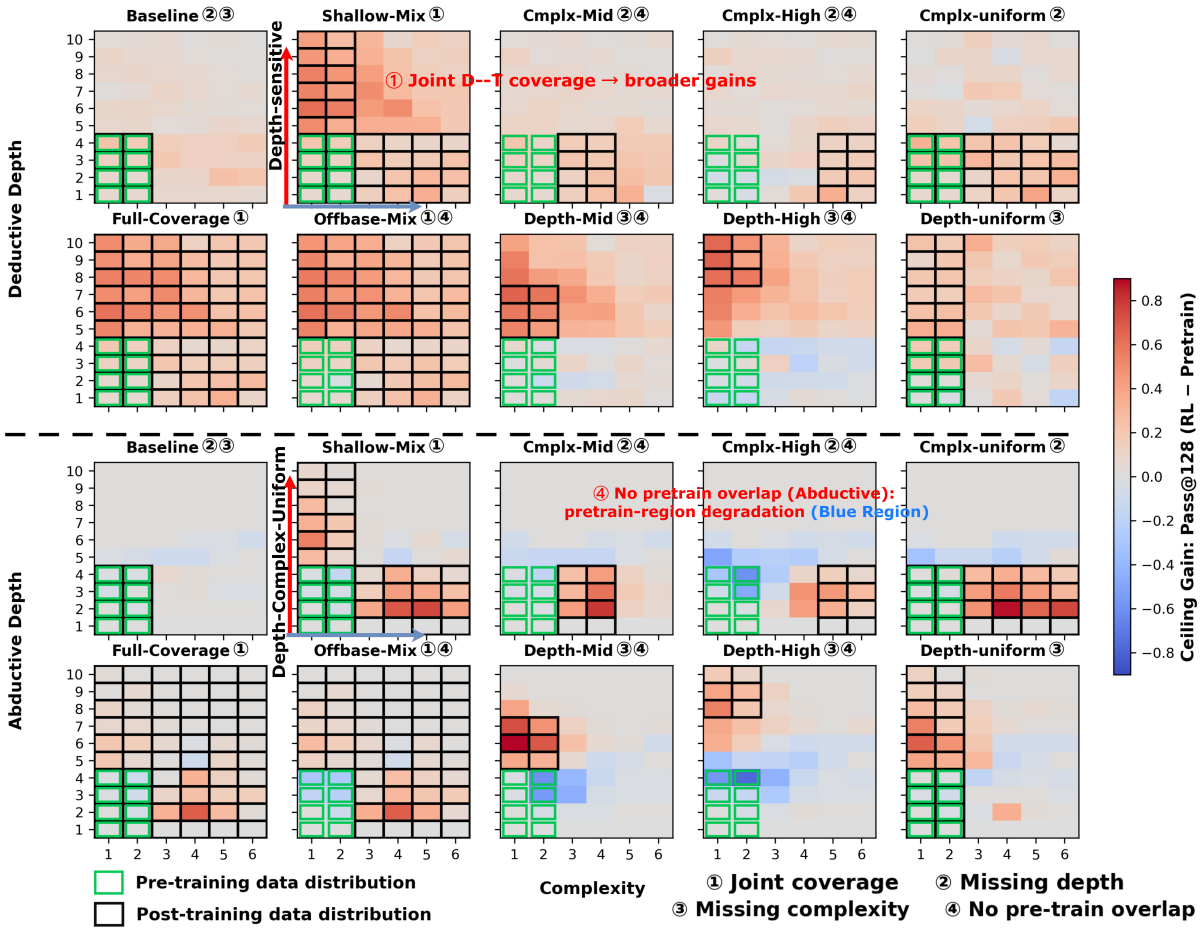


Figure 3: Cell-level CG heatmaps on the 60-cell (D, T) grid. Each panel corresponds to one RL recipe; warmer colors and larger CG values indicate stronger post-training gains, while blue indicates degradation below the pre-trained model.

Grid-Level Reporting The evaluation split contains 4,800 graphs, with 80 graphs in each of the 60 (D, T) cells. We report cell-level results before aggregating by depth, complexity, task family, or recipe family, since post-training can improve one region while suppressing another. Detailed verdict rules and coarser plane summaries are deferred to Appendices A.4 and A.5.

2.4 Train Setup

We conduct experiments with a 107M-parameter Qwen2.5-style decoder-only model (Qwen et al., 2025). All stages use the same controlled KG generator and differ only in (D, T) coverage and per-cell budgets. Full model, optimizer, recipe, and implementation details are given in Appendix A.3.

Pre-training The pre-training split covers the shallow and low-complexity block $D \in \{1, \ldots, 4\}$ and $T \in \{1, 2\}$, with 3.2 million graphs and 3.14B tokens in total. This region teaches the model basic reasoning ability in our domain while leaving deeper chains and higher-complexity environments unseen. This lets us test whether RL post-training extends the model beyond the pre-training distribution.

Post-Training Post-training starts from the pre-trained model and uses GRPO (Shao et al., 2024). We ask which (D, T) regions should provide RL signal after pre-training. We study 15 recipes: one baseline and 14 additional data-distribution recipes in four families, as detailed in Figure 2(c).

Each recipe uses a fixed budget of 220K tasks, distributed uniformly over its selected (D, T) cells. The budget is split into 140K, 60K, 10K, and 10K tasks for deductive, abductive, inductive, and analogy, respectively. This allocation reflects our intended reasoning emphasis: deductive receives the largest share because much real-world reasoning is deductive; abductive keeps a moderate share as the inverse dynamic counterpart of deductive state tracking; inductive and analogy receive smaller

shares because they operate mainly on the static component and are less affected by reasoning depth and environment complexity, making them less dependent on extensive RL training.

During GRPO, $A = \mathrm{m_A}(\hat{a}, a^\star) \in \{0, 1\}$ indicates answer match and $P = \mathrm{m_P}(\hat{\pi}, \pi) \in \{0, 1\}$ indicates full parsed-trace match. The outcome-gated reward is

$$R = \begin{cases} 0.8P + 0.2A, & A = 1, \\ 0, & A = 0. \end{cases}$$

Thus a response receives no reward when its final answer is wrong, even if the trace is partially plausible.

3 How Should Post-Training Data Be Distributed Across Reasoning Depth and Environment Complexity?

We study the question from global and local views. The main text focuses on deductive and abductive; inductive and analogy results are reported in Appendix A.6.2.

3.1 Task Setting

For each task family, we ask which RL data distribution gives the strongest global improvement over the 60-cell (D, T) grid, and which recipes improve or suppress each individual cell.

Task 1: Global Recipe Ranking We rank recipes along the depth and complexity axes using CG. For each task, recipes are ranked by mean CG within every depth or complexity slice, and the slice-level ranks are then averaged. For example, the depth slice $D = 1$ contains all cells with $T = 1, \ldots, 6$; averaging ranks over all 10 depth slices gives each recipe's mean depth-rank.

Task 2: Local Cell Improvement The local view keeps full grid resolution: for every task family and valid (D, T) cell, we compute CG against the pretrained model.

3.2 Observations

Task 1. Figure 1(a, b) shows that stronger recipes lie closer to the upper-right corner. For deductive, Shallow-Mix, Offbase-Mix, and Full-Coverage outrank all other RL recipes, including Cmplx-Uniform and Depth-Uniform, on both axes. For abductive, Shallow-Mix beats Cmplx-Uniform on depth and Depth-Uniform on complexity, while remaining competitive on each baseline's covered axis.

Task 2. Figure 3 shows that Shallow-Mix improves over Cmplx-Uniform and Depth-Uniform on nearly all deductive cells, with the largest margins on deeper low-complexity cells. On abductive, Shallow-Mix again gives broader gains than either single-axis recipe, though with smaller margins.

The heatmaps also reveal different axis sensitivities. Deductive gains are larger along the depth extension than along the complexity extension, and Depth-family recipes yield larger CG gains than Cmplx-family recipes. Abductive gains are more balanced across the two axes, with no clear preference for either.

3.3 Takeaways

Post-training data that jointly cover reasoning depth and environment complexity outperform data that cover only one axis.

Deductive benefits more from depth coverage, while Abductive has no clear axis preference.

3.4 Discussion

Joint depth-complexity coverage is necessary. Depth tests whether intermediate commitments can be preserved over longer chains, whereas complexity tests relevant-structure selection under distractors. Single-axis recipes reward only one demand, while joint coverage rewards both.

Figure 3 shows that Baseline and the Complexity-/Depth-High recipes are generally weaker than their Mid counterparts, consistent with the view that RL gains depend on headroom near the model's competence boundary (Zhang et al., 2025a). Our result extends this view from a single difficulty axis to a two-dimensional depth–complexity region.

Deductive is depth-sensitive, while Abductive is more balanced. A deductive answer applies the next operation forward from the current state. Abductive reasoning instead works backward: at each hop, the model must reconstruct adjacent states, match them, and infer the missing operation from their difference. Environment complexity makes this matching harder because distractor operations create many competing post-operation states. Environment-complexity training is therefore more necessary for abductive reasoning, whereas deductive reasoning mainly benefits from depth-oriented training and uses complexity for robustness.

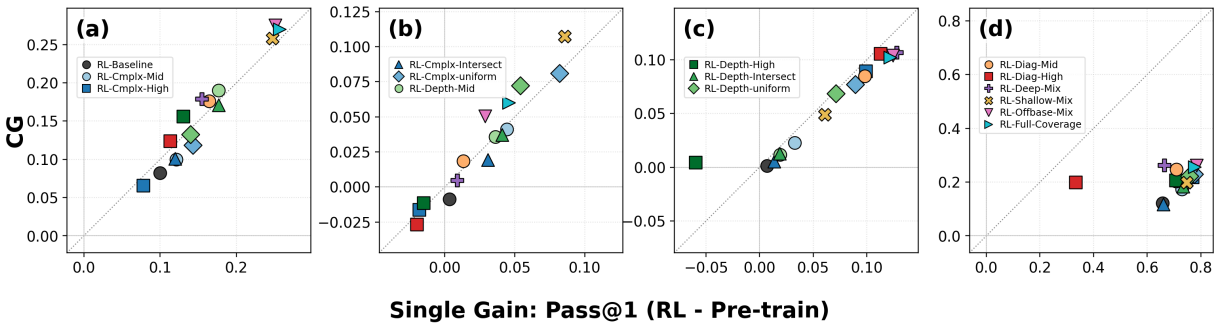


Figure 4: SG/CG plane across task families. Each point is one RL recipe. deductive, abductive, and inductive near the SG = CG diagonal, but analogy in the high-SG, low-CG region.

3.5 Practical Guidance

Post-training data should cover both reasoning depth and environment complexity. For hard deductive targets, allocation can lean toward deeper cells, but it should still retain complexity coverage rather than collapse to a depth-only recipe.

4 Do All Reasoning Tasks Behave Uniformly Under Post-training?

We next test whether the four reasoning families respond uniformly to post-training.

4.1 Task Setting

We evaluate out-of-recipe generalization, task-pair correlation, and reasoning-capability growth.

Task 1: Out-of-Recipe Generalization For each task family, we compute CG for 15 post-training recipes over 60 cells, yielding 900 recipe-cell units. Covered units are in-recipe; the rest are out-of-recipe. Under the Baseline recipe, for example, $D = 1, T = 2$ is in-recipe, whereas $D = 1, T = 3$ is out-of-recipe.

Task 2: Task Correlation For each recipe-cell unit, we obtain CG for all four task families and compute Pearson correlation for the six task pairs over the corresponding 900 CG pairs. Appendix A.4 gives the formal definition.

Task 3: SG/CG Diagnostic For each task family, we compute SG and CG for each of the 15 recipes on the full 60-cell grid.

4.2 Observations

Task 1. Figure 1(d) shows positive out-of-recipe CG for deductive, inductive, and analogy, about 50–70% of their in-recipe gains, whereas abductive is negative. Figure 3 shows the same abductive failure: without overlap between post-training and pre-training cells, blue regions near the pre-training block indicate degradation below the pre-trained model.

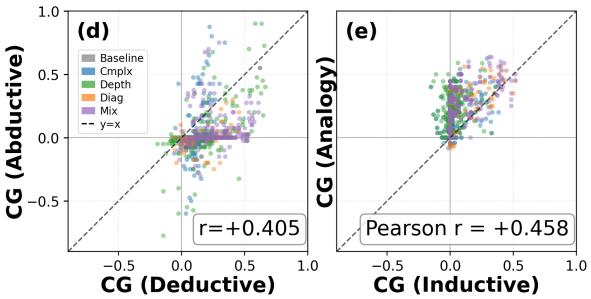


Figure 5: Strong task-pair CG correlations across the 60-cell grid and 15 RL recipes. (a) Deductive–abductive and (b) inductive–analogy are the strongest positive pairs.

Task 2. Figures 5 and 14 identify deductive–abductive and inductive–analogy as the strongest positive pairs; deductive–analogy is moderate, and the remaining three pairs are weak.

Task 3. Figure 4 places deductive, abductive, and inductive near the SG = CG diagonal, but analogy in the high-SG, low-CG region.

4.3 Takeaways

Abductive post-training shows almost no generalization beyond the data coverage, while the other three tasks generalize well.

Task correlations concentrate in Deductive–Abductive and Inductive–Analogy.

Deductive, Abductive, and Inductive exhibit healthy growth in reasoning ability during post-training, whereas Analogy shows a mode-collapse phenomenon.

4.4 Discussion

Abductive requires direct coverage. The weak out-of-recipe transfer of abductive reasoning is consistent with its reliance on multi-hop reverse inference, as discussed in Section 3. Prior work shows that reversal remains difficult for autoregressive language models without sufficient training support (Berglund et al., 2024; Lv et al., 2024; Zhu et al., 2024; Golovneva et al., 2024; Zhu et al., 2026); however, these studies mainly consider one-hop or fact-level reversal. Our setting imposes a stronger requirement: the model must reconstruct missing steps across multiple hops while resolving competing states induced by high environment complexity. Abductive gains therefore depend strongly on direct coverage of the target data distribution.

Task correlations reflect shared reasoning components. The deductive–abductive correlation arises from a shared dynamic transition system and from the deductive subproblem embedded in abductive reasoning: predicting the state after the previous operation is a forward state-prediction step. The inductive–analogy correlation reflects shared reliance on kinship relations and rule internalization, since solving an analogy instance requires extracting the relation pattern from a source case and transferring it to a target case. The weaker deductive–analogy coupling is consistent with their common entity/state-to-next-state prediction structure.

Analogy shows mode collapse. Deductive, abductive, and inductive gains require improving the underlying reasoning process, so pass@1 and the sampled candidate set tend to rise together. Analogy often admits a compact mapping pattern; RL can sharpen this dominant pattern and improve pass@1 without expanding the sampled candidate set, yielding high SG but low CG.

4.5 Practical Guidance

For abductive, directly cover the target evaluation region rather than assuming transfer beyond data coverage. For multi-task post-training, first pair deductive with abductive and inductive with analogy before testing broader mixtures.

5 Does Staged Curriculum Improve Over Uniform Post-Training?

We compare uniform mixing with staged curricula under a fixed post-training budget.

Task Setting All schedules use the same post-training budget and differ only in how target cells are partitioned. We split the target region along either the depth or complexity axis and train from easy to hard. For example, on the complexity axis, Cmplx-Uniform uses all 24 target cells in one stage; 3-block uses $T1$–$T2 \rightarrow T3$–$T4 \rightarrow T5$–$T6$; and 6-block uses one complexity level per stage.

Observations Figure 1(e) shows that, for both deductive and abductive tasks and along both the depth and complexity axes, uniform mixing outperforms 3-block E2H, which in turn outperforms many-block E2H.

Takeaway **Uniform mixing is the strongest default under fixed budget.**

Discussion **Uniform mixing is the strongest default under a fixed budget.** Each curriculum transition introduces forgetting and adaptation costs: updates on a new block can overwrite support for earlier blocks, and part of the budget is spent adapting to the next sub-distribution before useful learning resumes. These costs can be offset only when each block receives enough compute and later blocks build on earlier scaffolding, which is not guaranteed under a fixed post-training budget. Recent LLM-RL curricula, including VCRL (Jiang et al., 2025), Re-Schedule (Wang et al., 2025b), and the E2H Reasoner (Parashar et al., 2025), report gains over random or unscheduled RL baselines, but do not fix the total budget. Under fixed compute, per-transition forgetting (Ibrahim et al., 2024) can dominate. This is also consistent with curriculum theory (Bu et al., 2026), whose sample-complexity gains assume that per-block capacity is not budget-limited.

Practical Guidance Default to uniform mixing.

6 Benchmarking Off-the-Shelf Reasoning Models

We benchmark recent general-purpose and reasoning-oriented language models on the same 60-cell evaluation set to test whether the RL task-family asymmetries also appear in off-the-shelf and frontier models. Each model uses task- and

Model	Overall	Ded.	Abd.	Ind.	Ana.
Kimi-K2.6	**.748**	**.831**	**.262**	.746	.649
GPT-5.4	**.743**	**.827**	**.265**	.743	.617
GLM-5.1-FP8	**.742**	**.805**	**.264**	**.797**	.700
Offbase-mix	.585	.563	.177	**.822**	**.865**
Shallow-mix	.575	.559	.235	.759	.823
DeepSeek-R1	.560	.615	.135	.614	.522
Qwen3.6-35B-A3B	.558	.651	.089	.526	.402
Mixtral-8x22B-Instruct	.550	.607	.099	.581	.585
Kimi-K2-Thinking	.543	.578	.169	.609	.610
Qwen3.5-397B-A17B-FP8	.521	.552	.162	.567	.634
Cmplx-uniform	.510	.453	.227	**.787**	**.850**
Depth-uniform	.503	.452	.202	.770	**.835**
Qwen3.5-35B-A3B	.494	.539	.097	.565	.475
QwQ-32B-Preview	.475	.594	.078	.270	.347
Qwen2.5-32B-Instruct	.427	.430	.085	.603	.494
Qwen3-8B	.344	.391	.051	.357	.284
DS-R1-Distill-Qwen-32B	.301	.256	.078	.535	.544
Qwen2.5-72B-Instruct	.247	.145	.092	.689	.477
Qwen3.6-27B	.220	.209	.183	.269	.264
Qwen2.5-7B-Instruct	.203	.219	.032	.201	.302
Qwen3.5-122B-A10B-FP8	.176	.106	.013	.522	.316
DS-R1-Distill-Qwen-7B	.138	.180	.015	.063	.079

Table 1: Benchmark results on the 60-cell evaluation set with task- and cell-matched 3-shot prompting. Current LLMs remain weak on abductive tasks.

cell-matched 3-shot prompting: for a target task at depth D and environment complexity T, the prompt contains three random RL-data examples from the same task family and (D, T) cell.

Table 1 shows three patterns. Kimi-K2.6, GPT-5.4, and GLM-5.1-FP8 are the strongest overall models, with very close scores. Nearly all models are much stronger on deductive than abductive tasks; even the best abductive scores remain low, suggesting weaker post-training pressure on reverse inference. Despite their 107M scale, our RL checkpoints remain competitive: Offbase-mix and Shallow-mix rank highly overall, while Cmplx-uniform and Depth-uniform are especially strong on inductive and analogy tasks.

7 Related Work

Controlled Experiments from Scratch Controlled training from scratch with synthetic data studies language models under known data-generating processes, reducing opaque pre-training exposure and benchmark contamination as attribution confounds (Magar and Schwartz, 2022; Sainz et al., 2023; Roberts et al., 2023). Recent controlled studies use synthetic reasoning data to isolate the effects of pre-training, mid-training, RL, relational semantics, and small-model emergence (Zhang et al., 2025a; Zhu et al., 2026; Eldan and Li, 2023; Allen-Zhu and Li, 2025).

Depth-Oriented RLVR and Difficulty Allocation Recent RLVR and post-training work often frames difficulty through depth-like signals: longer chains, compositional depth, long horizons, competence boundaries, and curriculum schedules (Zhang et al., 2025a; Lu et al., 2025; Yang et al., 2025; Huang et al., 2026; Wang et al., 2026; Jiang et al., 2025; Wang et al., 2025b; Parashar et al., 2025; Bu et al., 2026; Kung et al., 2026). These studies clarify when RL sharpens or extends reasoning, but usually treat difficulty as scalar or depth-like. Controlled evaluations show that distractors, symbolic perturbations, and complexity-OOD shifts also affect reasoning (Mirzadeh et al., 2025; Paqaleh et al., 2025; Wang et al., 2026). We instead make environment complexity a post-training allocation axis, not only an evaluation stressor.

Logical Reasoning Tasks and Benchmarks Logical reasoning work evaluates LLMs on deductive and first-order benchmarks, proof-style tasks, and synthetic logic data (Parmar et al., 2024; Han et al., 2024; Tafjord et al., 2021; Liu et al., 2025a). Other studies examine abductive reasoning and reverse inference through abductive benchmarks (Bhagavatula et al., 2020; Cao et al., 2026; Han et al., 2026), reversal analyses (Berglund et al., 2024; Lv et al., 2024; Zhu et al., 2024) or study analogy transfer (Webb et al., 2023; Musker et al., 2025). Most use these families as benchmarks or study one form at a time. We use our four-task dataset for both RLVR training and benchmark evaluation.

8 Conclusion

RLVR recipe design is a data-allocation problem over both difficulty axes and reasoning forms. Using a controlled synthetic KG environment, we show that joint depth–complexity coverage is more reliable than single-axis recipes, that task families respond differently to the same post-training data, and that abductive reasoning is especially fragile outside the training distribution. Task correlations separate into deductive–abductive and inductive–analogy pairs, while uniform mixing outperforms staged curriculum under a fixed budget. Recent instruct/reasoning models show the same deductive-over-abductive gap, suggesting that RLVR post-training should optimize structured coverage rather than only aggregate benchmark gains.

9 Limitations

Synthetic KG setting. Our synthetic KG world provides strong control over pre-training, post-training, and evaluation distributions, but it simplifies real language environments. The setting does not fully cover semantic ambiguity, open-world knowledge, long-document context, or the linguistic diversity of natural reasoning tasks.

Controlled but narrow task families. Deductive, abductive, inductive, and analogy tasks are defined through explicit KG operations. These definitions make the task families comparable within the same controlled framework, but they represent only one controlled instantiation of each reasoning form and do not cover the full range of logical reasoning in the wild.

Finite depth–complexity grid. Our evaluation covers $D \in \{1, \ldots, 10\}$ and $T \in \{1, \ldots, 6\}$. The observed trends may change under longer reasoning horizons, denser distractor environments, or more complex structures beyond this grid.

Diagnostic benchmark comparison. The Table 1 comparison should be interpreted as diagnostic rather than as a controlled head-to-head evaluation. Our four internal models are only 107M parameters, but they receive domain-specific pre-training and RL post-training on the synthetic KG distribution. In contrast, off-the-shelf models are evaluated only with task- and cell-matched few-shot prompts that teach the answer format and basic story setting. Therefore, the table mainly tests whether the same deductive–abductive asymmetry appears in existing models, not whether our small models are fairly comparable to frontier systems.

Ethics Statement

This study complies with the ACL Ethics Policy.

References

Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Besmira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, Vibhav Vineet, Yue Wu, Safoora Yousefi, and Guoqing Zheng. 2025. Phi-4-reasoning technical report.

Zeyuan Allen-Zhu and Yuanzhi Li. 2025. Physics of Language Models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*.

Anthropic. 2026. Introducing Claude Opus 4.7.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The Reversal Curse: LLMs trained on "a is b" fail to learn "b is a". In *International Conference on Learning Representations*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference on Learning Representations*.

Dake Bu, Wei Huang, Andi Han, Atsushi Nitanda, Hau-San Wong, Qingfu Zhang, and Taiji Suzuki. 2026. Provable benefit of curriculum in transformer tree-reasoning post-training. In *Proceedings of the 43rd International Conference on Machine Learning*.

Pengfei Cao, Mingxuan Yang, Yubo Chen, Chenlong Zhang, Mingxuan Liu, Kang Liu, and Jun Zhao. 2026. Semeval-2026 task 12: Abductive event reasoning: Towards real-world event causal inference for large language models.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645(8081):633–638.

Dai Do, Manh Nguyen, Svetha Venkatesh, and Hung Le. 2025. SPaCe: Unlocking sample-efficient large language models training with self-pace curriculum learning.

Shihan Dou, Muling Wu, Jingwen Xu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Improving RL exploration for LLM reasoning through retrospective replay. In *Natural Language Processing and Chinese Computing (NLPCC 2025)*, pages 594–606.

Ronen Eldan and Yuanzhi Li. 2023. TinyStories: How small can language models be and still speak coherent English?

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 482–495. PMLR.

Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the Reversal Curse. In *Proceedings of the 1st Conference on Language Modeling*.

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, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 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.

Taolin Han, Shuang Wu, Jinghang Wang, Yuhao Zhou, Renquan Lv, Bing Zhao, and Wei Hu. 2026. MolQuest: A benchmark for agentic evaluation of abductive reasoning in chemical structure elucidation.

Yu Huang, Zixin Wen, Yuejie Chi, Yuting Wei, Aarti Singh, Yingbin Liang, and Yuxin Chen. 2026. The implicit curriculum: Learning dynamics in RL with verifiable rewards. In *Proceedings of the 43rd International Conference on Machine Learning*.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. 2024. Simple and scalable strategies to continually pre-train large language models.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. 2025. VCRL: Variance-based curriculum reinforcement learning for large language models.

Po-Nien Kung, Zhen Yang, Jeffrey Luo, Cheng-Fu Yang, Haikang Deng, Zi-Yi Dou, Yinfei Yang, Nanyun Peng, Zhe Gan, and Kai-Wei Chang. 2026. Learning structured reasoning via tractable trajectory control.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Mozhi Zhang, Pengyu Zhao, and Junxian He. 2025a. SynLogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. In *Advances in Neural Information Processing Systems*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025b. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems*.

Yi Lu, Jianing Wang, Linsen Guo, Wei He, Hongyin Tang, Tao Gui, Xuanjing Huang, Xuezhi Cao, Wei Wang, and Xunliang Cai. 2025. R-Horizon: How far can your large reasoning model really go in breadth and depth?

Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. 2024. An analysis and mitigation of the Reversal Curse. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13603–13615.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*.

Mistral AI. 2024. Cheaper, better, faster, stronger: Mixtral 8x22B.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moonshot AI. 2025. Introducing Kimi K2 Thinking.

Moonshot AI. 2026. Kimi-K2.6 model card.

Sam Musker, Alex Duchnowski, Raphaël Millière, and Ellie Pavlick. 2025. LLMs as models for analogical reasoning. *Journal of Memory and Language*, 145:104676.

Kangqi Ni, Zhen Tan, Zijie Liu, Pingzhi Li, and Tianlong Chen. 2025. Can GRPO help LLMs transcend their pretraining origin?

OpenAI. 2025. Introducing OpenAI o3 and o4-mini.

OpenAI. 2026a. Introducing GPT-5.4.

OpenAI. 2026b. Introducing GPT-5.5.

Mohammad Mahdi Samiei Paqaleh, Arash Marioriyad, Arman Tahmasebi-Zadeh, Mohamadreza Fereydooni, Mahdi Ghaznavai, and Mahdieh Soleymani Baghshah. 2025. Bridging reasoning to learning: Unmasking illusions using complexity out of distribution generalization.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. 2025. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13679–13707.

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

Qwen Team. 2024. QwQ-32B-Preview model card.

Qwen Team. 2025. Qwen3 technical report.

Qwen Team. 2026a. Qwen3.5-122B-A10B-FP8 model card.

Qwen Team. 2026b. Qwen3.5-35B-A3B model card.

Qwen Team. 2026c. Qwen3.5-397B-A17B-FP8 model card.

Qwen Team. 2026d. Qwen3.6-27B: Flagship-level coding in a 27B dense model.

Qwen Team. 2026e. Qwen3.6-35B-A3B: Agentic coding power, now open to all.

Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. 2023. Data contamination through the lens of time.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787.

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.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. MathScale: Scaling instruction tuning for mathematical reasoning. In *Proceedings of the 41st International Conference on Machine Learning*.

Chen Wang, Lai Wei, Yanzhi Zhang, Chenyang Shao, Zedong Dan, Weiran Huang, Yuzhi Zhang, and Yue Wang. 2025a. EFRame: Deeper reasoning via exploration-filter-replay reinforcement learning framework.

Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Qiang Lin, Hande Dong, and Jiawei Chen. 2025b. Scheduling your LLM reinforcement learning with reasoning trees.

Tianle Wang, Zhaoyang Wang, Guangchen Lan, Xinpeng Wei, Sipeng Zhang, Guanwen Qiu, and Abulhair Saparov. 2026. Can RL teach long-horizon reasoning to LLMs? expressiveness is key.

Taylor Webb, Keith J. Holyoak, and Hongjing Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Dongchun Xie, Hanhui Li, Yiwei Wang, Xiaodan Liang, and Jing Tang. 2025. Depth-breadth synergy in RLVR: Unlocking LLM reasoning gains with adaptive exploration.

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? In *Advances in Neural Information Processing Systems*.

Z.ai. 2026. GLM-5.1-FP8 model card.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. 2026. ExGRPO: Learning to reason from experience. In *The Fourteenth International Conference on Learning Representations*.

Charlie Zhang, Graham Neubig, and Xiang Yue. 2025a. On the interplay of pre-training, mid-training, and RL on reasoning language models.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. 2025b. RLEP: Reinforcement learning with experience replay for LLM reasoning.

Hanlin Zhu, Baihe Huang, Shaolun Zhang, Michael Jordan, Jiantao Jiao, Yuandong Tian, and Stuart Russell. 2024. Towards a theoretical understanding of the Reversal Curse via training dynamics. In *Advances in Neural Information Processing Systems*, volume 37, pages 90473–90513.

Yihua Zhu, Qianying Liu, Jiaxin Wang, Fei Cheng, Chaoran Liu, Akiko Aizawa, Sadao Kurohashi, and Hidetoshi Shimodaira. 2026. Memorization, emergence, and explaining reversal failures: A controlled study of relational semantics in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*.

A Appendix

Appendix Contents

A.1	Supplementary Material for ARR Submission	12
A.2	Data Generation Framework	12
A.2.1	Summary of Notation	12
A.2.2	Formal Graph Structure	12
A.2.3	Dynamic Event Semantics	13
A.2.4	Static Component	14
A.2.5	Reasoning Depth and Environment Complexity	15
A.2.6	Entity Sampling and Surface Rendering	15
A.2.7	Split Construction	15
A.2.8	Task-Specific Supervision	16
A.2.9	Worked Example	17
A.3	Training Setup	17
A.3.1	Model Architecture and Tokenizer	17
A.3.2	Pretraining Objective and Hyperparameters	17
A.3.3	GRPO Post-Training Implementation	18
A.3.4	Generated-Content Parsing and Matching	18
A.3.5	RL Recipes and Task Instantiation	18
A.4	Experiment Evaluation Metrics	18
–	Strict pass@k, SG, and CG	18
–	Global Mean Rank	18
–	Local Cell CG	18
–	Task-Pair Pearson Correlation	18
A.5	Ablation Study	19
A.5.1	Data Volume	19
A.5.2	Number of Rollouts	19
A.5.3	Task Mix	20
A.5.4	Process-Reward Ratio	20
A.6	Supplementary Experiments and Results	20
A.6.1	Full Marginal CG Heatmaps	22
A.6.2	Task-Family Supplementary Results	22
A.6.3	Pass@k Exploration Profile	25
A.6.4	Benchmark Supplementary Results	25
A.6.5	Best RL Recipe per Region	25
A.7	Curriculum Supplementary Experiments	28
A.7.1	Task Setting	28
A.7.2	Observation	28
A.7.3	Discussion	28
A.7.4	Full-Grid and Task-Family Extensions	30

A.1 Supplementary Material for ARR Submission

Unless otherwise stated, we report results from a single run due to computational constraints (pretraining from scratch and post-training are expensive and time-consuming under our GPU budget).

Moreover, all data are synthetically generated and do not contain personally identifying information. We also avoid generating offensive content by construction and manually spot-check a sample of the generated corpus. Finally, all datasets in this work are synthetically generated for research and reproducibility purposes. The released artifacts (code/data/model checkpoints, if any) are intended for research use only.

Our experiments were facilitated by leveraging PyTorch, Hugging Face, and NumPy as essential tools. Furthermore, we used ChatGPT, Codex, and Claude Code in our paper writing and programming.

We will release the code under the MIT License. The generated synthetic datasets (and trained checkpoints, if released) will be distributed under the same license terms specified in the repository. We use standard open-source libraries and frameworks, including PyTorch, NumPy, Hugging Face Transformers/Datasets/Tokenizers, LLaMA-Factory, verl, vLLM, pandas, and Matplotlib, and comply with their respective licenses.

A.2 Data Generation Framework

Scope of This Appendix This appendix provides the formal details of the data generation framework introduced in Section 2. The framework is organized around a single symbolic graph, two reasoning components, and two orthogonal setup axes. The dynamic component supports object-state reasoning over time, while the static component supports kinship-rule reasoning over relation paths. The same graph can therefore produce deductive, abductive, inductive, and analogy tasks with process-verifiable supervision.

A.2.1 Summary of Notation

Data Generation Notation Table 2 summarizes the notation introduced in this appendix and used in the main text.

A.2.2 Formal Graph Structure

Generated Graph Instance Each instance is a five-tuple

$$\mathcal{G} = (\mathcal{P}, \mathcal{O}, \mathcal{E}_{\text{dyn}}, \mathcal{E}_{\text{stat}}, s_0),$$

where $\mathcal{P}$ is a finite set of person entities, $\mathcal{O}$ is a finite set of object entities, $\mathcal{E}_{\text{dyn}}$ is a time-ordered sequence of object events, $\mathcal{E}_{\text{stat}}$ is a finite set of static binary kinship relations, and s_0 is the initial world state. The graph carries two components, defined below.

Dynamic Object States The dynamic event-state component is

$$\mathcal{C}_{\text{dyn}} = (\mathcal{E}_{\text{dyn}}, s_0, \delta, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{op}}).$$

It describes how object-centered states evolve. Object states live in the space

$$\mathcal{S} = \{\, s \mid s : \mathcal{O} \to \mathcal{P} \times \mathcal{P} \times \{\text{intact}, \text{broken}\} \,\}.$$

Category	Symbol	Type	Meaning
Graph	$\mathcal{G} = (\mathcal{P}, \mathcal{O}, \mathcal{E}_{\text{dyn}}, \mathcal{E}_{\text{stat}}, s_0)$	tuple	A reasoning instance
Graph	$\mathcal{P}$, $\mathcal{O}$	sets	Person and object entities
Graph	s_0	state	Initial world state
Dynamic	$\mathcal{C}_{\text{dyn}}$	component	Dynamic event-state component
Dynamic	$\mathcal{E}_{\text{dyn}}$	sequence	Dynamic events on objects
Dynamic	$\mathcal{A}_{\text{init}}$	operator set	Initialization declarations: own and create
Dynamic	$\mathcal{A}_{\text{op}}$	operator set	Operation events counted by reasoning depth
Dynamic	$\mathcal{S}$	state space	Object-state mapping space
Dynamic	$s_t(o)$	tuple	Owner, possessor, and integrity of object o
Dynamic	δ	function	Deterministic state transition
Dynamic	$o^\star$	object	Target object queried by a task
Dynamic	$\mathrm{Obj}(e_i)$	set	Objects affected by event e_i
Static	$\mathcal{C}_{\text{stat}}$	component	Static kinship component
Static	$\mathcal{E}_{\text{stat}}$	relation set	Static binary kinship facts
Static	$\mathcal{W}$	vocabulary	Basic kinship labels
Static	$\mathcal{R}$	rule set	Composite kinship rules
Static	r	rule	A composite kinship rule
Static	$\gamma(r)$	path	Basic-relation path realizing r
Static	$\mathrm{hop}(r)$	count	Length of the path $\gamma(r)$
Static	$\hat{P}$	person	Matched target person in an analogy task
Dynamic	T_{final}, T_q, k_q	indices	Final time, queried time, and target-chain state index
Complexity	$D(\mathcal{G})$	scalar	Target-chain reasoning depth
Complexity	$T(\mathcal{G})$	level index	Environment-complexity level
Complexity	$\Theta(T)$	parameter map	Structural complexity preset
Complexity	k_{chains}	distribution	Complexity-specific chain-count distribution
Complexity	n_{persons}	range	Person-count range for entity sampling
Complexity	$\rho_{\text{crossover}}$	probability	Cross-chain person-overlap rate
Complexity	ρ_{exchange}	probability	Exchange-event rate
Rendering	Φ	renderer	Symbolic graph to natural-language story
Supervision	$x = (Q, \pi, a^\star)$	tuple	Task instance
Supervision	Q, π, $a^\star$	task fields	Question, gold trace, and gold answer
Supervision	$\pi^{\text{ded}}_{\text{dyn}}$, $\pi^{\text{abd}}_{\text{dyn}}$, $\pi^{\text{ind}}_{\text{stat}}$, $\pi^{\text{ana}}_{\text{stat}}$	traces	Task-specific gold trace formats
Supervision	$a^{\star,\text{ind}}$, $a^{\star,\text{ana}}$	answers	Task-specific gold answers for static tasks
Supervision	$\hat{\pi}$, $\hat{a}$	model outputs	Predicted trace and predicted answer
Supervision	$\mathrm{m_P}$, $\mathrm{m_A}$	predicates	Process and answer matching criteria
Splits	n_{cell}	count	Graphs per (D, T) cell

Table 2: Notation Used in the Data Generation Framework

At time t, the projection for object o is

$$s_t(o) = (\mathrm{owner}(o,t), \mathrm{possessor}(o,t), \mathrm{integrity}(o,t)).$$

Initial declarations set the initial state and are collected in $\mathcal{A}_{\text{init}} = \{\mathrm{own}, \mathrm{create}\}$. They are not counted as operation events for reasoning depth. Subsequent operation events are collected in $\mathcal{A}_{\text{op}}$ and update object states through a deterministic transition function

$$\delta : \mathcal{S} \times \mathcal{A}_{\text{op}} \to \mathcal{S},$$

so that $s_t = \delta(s_{t-1}, e_t)$ for $e_t \in \mathcal{A}_{\text{op}}$. Events whose preconditions are not satisfied are rejected during generation, which keeps the symbolic trace well defined.

Static Kinship Rules The static kinship component is

$$\mathcal{C}_{\text{stat}} = (\mathcal{E}_{\text{stat}}, \mathcal{W}, \mathcal{R}).$$

It describes fixed person-person relations. The relation set is

$$\mathcal{E}_{\text{stat}} \subseteq \mathcal{P} \times \mathcal{W} \times \mathcal{P},$$

where $\mathcal{W}$ is a basic kinship vocabulary. Each fact (p, w, p') is closed under inverse, so the generator also inserts (p', w^{-1}, p) when required. A composite kinship rule $r \in \mathcal{R}$ is a named relation realized by a path $\gamma(r) \in \mathcal{W}^{\mathrm{hop}(r)}$, with $\mathrm{hop}(r) \in \{2, 3\}$. The static component is sampled independently of the dynamic transition function.

A.2.3 Dynamic Event Semantics

Dynamic Event Classes The dynamic component groups events by which slots of $s_t(o)$ they change. Table 3 gives the resulting state-slot semantics. This presentation abstracts away from surface forms, while preserving the distinction between ownership transfer, possession-only transfer, and integrity change. Exchange events are included with ownership-possession transfers because, for

Event class	Included events	State update for object o
Initialization	own, create	$s_0(o) = (p, p, \text{intact})$. These declarations initialize an object state and are excluded from $\mathcal{A}_{\text{op}}$.
Ownership-possession transfer	gift, sale, exchange	If $s_{t-1}(o) = (p, p, g)$, then $s_t(o) = (p', p', g)$. For exchange, p' is the new holder of o.
Possession-only transfer	loan, return	If $s_{t-1}(o) = (u, v, g)$, then $s_t(o) = (u, v', g)$. The owner remains u.
Integrity update	break, repair	If $s_{t-1}(o) = (u, v, g)$, then $s_t(o) = (u, v, g')$, where $g, g' \in \{\text{intact}, \text{broken}\}$ and $g' \neq g$.

Table 3: State-Slot Semantics of Dynamic Event Classes

each exchanged object, both the owner and possessor slots move to the new holder.

Interleaved Object Chains A graph may contain several dynamic chains, one per object, and these chains are interleaved into a shared timeline. Entity crossover allows the same person to appear in multiple chains. Distractor-chain events are rendered in the story even when they do not affect the target object's answer. Thus the surface story may contain explicit but irrelevant events, while the gold trace only follows the target object's causal chain.

Dynamic Task Traces Let $e_1, \ldots, e_D$ be the D target-object operation events on object $o^\star$, in causal order. Let $s_k(o^\star)$ be the target-object state after the k-th such event. For a deductive task, the gold dynamic trace is

$$\pi^{\text{ded}}_{\text{dyn}} = \left\langle s_0(o^\star), (e_k, s_k(o^\star))_{k=1}^{D} \right\rangle,$$

and the gold answer is the queried slot of the relevant state. For an abductive task, one event or initial fact is hidden. Let m denote the mask mode, $\tilde{e}_k$ denote the visible target-object events, and $\tilde{s}_k^-$ and $\tilde{s}_k^+$ denote the states immediately before and after a visible event. The gold trace is

$$\pi^{\text{abd}}_{\text{dyn}} = \Big\langle m,\ s_0^{(m)}(o^\star),\ (\tilde{e}_k, \tilde{s}_k^-(o^\star), \tilde{s}_k^+(o^\star))_{k=1}^{\tilde{D}},\ h^\star \Big\rangle.$$

Here $s_0^{(m)}$ is the visible initial state after masking, $\tilde{D}$ is the number of visible target-object events, and $h^\star$ is the hidden event or initial fact. The prediction is correct only when both the reconstructed process and the final answer match the gold structure.

A.2.4 Static Component

Basic and Composite Kinship The static component uses the basic vocabulary

$$\mathcal{W} = \{\text{father}, \text{mother}, \text{son}, \text{daughter}, \text{husband}, \text{wife}, \text{brother}, \text{sister}, \text{friend}\}.$$

Hop	Rule	Path $\gamma(r)$
2	grandmother	(mother, father)
2	grandfather	(father, father)
2	aunt	(sister, father)
2	uncle	(brother, father)
2	mother-in-law	(mother, husband)
2	father-in-law	(father, husband)
2	brother-in-law	(brother, husband)
2	sister-in-law	(sister, husband)
2	grandson	(son, son)
2	granddaughter	(daughter, son)
2	nephew	(son, brother)
2	niece	(daughter, brother)
2	son-in-law	(husband, daughter)
2	daughter-in-law	(wife, son)
3	great-grandfather	(father, father, father)
3	great-grandmother	(mother, father, father)
3	great-grandson	(son, son, son)
3	great-granddaughter	(daughter, son, son)
3	great-uncle	(brother, father, father)
3	great-aunt	(sister, father, father)
3	grand-nephew	(son, son, brother)
3	grand-niece	(daughter, son, brother)
3	cousin	(son, brother, father)

Table 4: Composite Kinship Rules and Canonical Composition Paths

A composite rule r names a relation realized by a path $\gamma(r)$ of length two or three through $\mathcal{W}$. For example, $\text{father}(x, y)$ and $\text{father}(y, z)$ imply $\text{grandfather}(x, z)$, so $\gamma(\text{grandfather}) = (\text{father}, \text{father})$. Table 4 lists the 23 composite rules used by the generator.

Static Task Traces For an inductive task, the generator plants support chains that realize a selected rule r, adds noise kinship facts, and asks the model to recover the composite rule. The gold static trace records the support sentences, the path $\gamma(r)$, and the rule name:

$$\pi^{\text{ind}}_{\text{stat}} = \left\langle \{(\sigma_j, \rho_j)\}_{j=1}^{k}, \gamma(r), \hat{r} = r \right\rangle.$$

σ_j denotes the j-th support example, and ρ_j is the corresponding basic-relation path expressed in $\mathcal{W}$. The tuple records the support evidence, the shared composition path $\gamma(r)$, and the gold composite rule $\hat{r} = r$. For an analogy task, the generator selects

Complexity	Chain distribution	Persons	Crossover	Exchange
T1	$\{1:1.0\}$	5–8	0.0	0.0
T2	$\{1:0.5, 2:0.5\}$	8–12	0.2	0.0
T3	$\{2:1.0\}$	10–15	0.3	0.2
T4	$\{3:1.0\}$	12–18	0.5	0.5
T5	$\{4:1.0\}$	15–22	0.8	0.8
T6	$\{5:1.0\}$	18–25	1.0	1.0

Table 5: Structural Complexity Presets

Variant	Example surface form
Active	Alice gives the red ball to Bob.
Passive	The red ball is given to Bob by Alice.
Cleft / role-based	It is Alice who gives the red ball to Bob.
Epistemic-marked	Alice decides to give the red ball to Bob.

Table 6: Surface-Form Variants Used by the Renderer

two person pairs that instantiate the same path. The gold trace records the shared path and the matched target person:

$$\pi^{\text{ana}}_{\text{stat}} = \big\langle \gamma(r),\ (A \xrightarrow{\gamma(r)} B),\ (C \xrightarrow{\gamma(r)} \hat{P}),\ \hat{P} \big\rangle.$$

The first pair (A, B) provides the relation pattern, and the second pair $(C, \hat{P})$ applies the same path to a new source entity. The final element $\hat{P}$ is the gold target entity.

A.2.5 Reasoning Depth and Environment Complexity

Reasoning Depth Definition Difficulty is controlled by reasoning depth D and environment complexity T. Depth counts only operation events on the target object's causal chain:

$$D(\mathcal{G}) = |\{e_i \in \mathcal{E}_{\text{dyn}} : e_i \in \mathcal{A}_{\text{op}},\ o^\star \in \text{Obj}(e_i)\}|\,.$$

Here $\text{Obj}(e_i)$ is the set of objects affected by event e_i. For ordinary single-object events, this set has one element. For exchange events, it contains the two exchanged objects, so the event contributes to the depth of a target chain only when $o^\star$ is one of those objects. Initialization declarations in $\mathcal{A}_{\text{init}}$ set s_0 and are not counted in $D(\mathcal{G})$. Pretraining uses $D \in \{1, \ldots, 4\}$, while RL and evaluation use $D \in \{1, \ldots, 10\}$.

Environment Complexity Presets The environment complexity $T(\mathcal{G}) \in \{1, \ldots, 6\}$ indexes a structural parameter map

$$\Theta(T) = (k_{\text{chains}}, n_{\text{persons}}, \rho_{\text{crossover}}, \rho_{\text{exchange}}).$$

k_{chains} specifies the complexity-specific distribution over the number of parallel object chains. n_{persons} specifies the person-count range used when sampling entities. $\rho_{\text{crossover}}$ controls the probability of cross-chain person overlap, and ρ_{exchange} controls the probability of exchange events between chains. Table 5 lists the presets.

Depth-Complexity Cell Targeting Because D and T are set independently, the generator can target any cell in the 60-cell depth-by-complexity grid. This design allows the experiments to separate long-chain reasoning from reasoning under distractor structure.

A.2.6 Entity Sampling and Surface Rendering

Entity and Object Sampling After T fixes the chain distribution, person-count range, crossover rate, and exchange rate, person entities are sampled without replacement from synthetic full-name pools. Object entities are then sampled to instantiate the chain objects; any extra objects are rendered as distractors. Names are kept distinct across splits to reduce direct memorization.

Natural-Language Rendering A renderer Φ maps the symbolic graph and a surface grammar to a natural-language story:

$$\Phi : (\mathcal{G}, \text{surface_grammar}) \rightarrow \text{story}.$$

The renderer uses a small set of surface variants for each canonical operator. Table 6 illustrates variants for $\text{transfer_gift}(\text{Alice}, \text{Bob}, \text{red_ball})$.

Interleaving Story Streams Dynamic and static streams are rendered separately and then interleaved into one story. This creates surface-level interference between event-state facts and kinship facts while preserving their symbolic separation. The rendered dynamic events can be divided into explicit events, which appear in the story, and implicit events or states, which are used by the simulator or gold trace but are not always stated directly.

A.2.7 Split Construction

Split Size Construction The generator enumerates the requested (D, T) cells and produces a fixed number of graphs per cell:

$$|\text{Data}_{\text{split}}| = |D| \cdot |T| \cdot n_{\text{cell}}.$$

This prevents the empirical distribution from drifting toward medium-complexity regions. The pretrain split contains $D1$–$D4$ and $T1$–$T2$, giving 8

cells with 400,000 graphs per cell and 3.2M graphs in total. The shared RL pool contains all $D1$–$D10$ and $T1$–$T6$ cells, with 20,000 graphs per cell and 1.2M graphs in total. The evaluation split uses the same 60-cell grid with 80 graphs per cell, for 4,800 graphs.

Sampling Within Each Cell Within a cell, the generator samples one target object, a length-D sequence of target-chain operations, and additional operations on parallel chains. Cross-chain exchange events are inserted according to ρ_{exchange}. The static component is generated over the same person pool, closed under inverse relations, and augmented with noise kinship facts. RL variants are then sampled without replacement from the shared RL pool, bucket-wise within each (D, T) cell. Thus the RL recipes differ in data distribution while sharing the same underlying generator.

A.2.8 Task-Specific Supervision

Process-Verified Task Instance Each task instance is a triple

$$x = (Q, \pi, a^\star), \tag{1}$$

where Q is the question, π is the gold solution trace, and $a^\star$ is the gold answer. Under process-verified evaluation, a model prediction $(\hat{\pi}, \hat{a})$ is correct only if both the process and the answer match the gold structure:

$$\mathrm{m_P}(\hat{\pi}, \pi) \wedge \mathrm{m_A}(\hat{a}, a^\star).$$

The framework supplies two distinct trace formats, chosen by the task family.

Deductive and Abductive Supervision Deductive and abductive tasks use the dynamic gold solution traces defined in Section A.2.3. For deductive tasks, the generator instantiates up to three `deduction_full_info` queries per graph, one for each state slot `owner`, `possessor`, or `integrity`, queried at the final time T_{final}. It also instantiates up to three `deduction_hard` queries, where the same three slots are queried at an intermediate time $T_q < T_{\text{final}}$. The trace is $\pi = \pi^{\text{ded}}_{\text{dyn}}$, and the gold answer is the projection of $s_{k_q}(o^\star)$ onto the queried slot. For abductive tasks, the generator chooses `mask_one_event` with probability 0.75 and `mask_one_initial_fact` with probability 0.25. The chosen element is removed from the rendered story and recorded as the gold answer. The trace is $\pi = \pi^{\text{abd}}_{\text{dyn}}$, and the gold answer is the natural-language realization of the masked element. If a target object would end in the broken state in a way that makes the masked inference ambiguous, an automatic repair event is appended to preserve solvability.

Inductive and Analogy Supervision Inductive and analogy tasks use the static gold solution traces defined in Section A.2.4. They sample the planted composite rule r from a six-group frequency stratification of the 23 composite rules in $\mathcal{R}$. Pretraining uses the long-tailed distribution in Table 7; evaluation uses uniform coverage over the same groups; and RL uses uniform sampling over the non-holdout groups. Inductive tasks draw r from groups A–E, giving 19 rules in total. Analogy tasks draw r from groups A–C only, giving 12 rules. Group F is excluded from pretraining and RL to preserve a clean evaluation axis for unseen rule compositions.

Static Query Construction For inductive tasks, the generator builds two or three support sentences, each realizing the planted rule r on a distinct person triple drawn from $\mathcal{E}_{\text{stat}}$. It then adds one to three noise kinship facts and fabricates a fresh trio of synthetic persons for the query path. The trace is $\pi = \pi^{\text{ind}}_{\text{stat}}$, and the gold answer is $a^{\star,\text{ind}} = r$. For analogy tasks, the generator selects two paths in $\mathcal{E}_{\text{stat}}$ that share the same composition $\gamma(r)$ and rephrases the question as "A is to B as C is to ?". The trace is $\pi = \pi^{\text{ana}}_{\text{stat}}$, and the gold answer is the matched person $a^{\star,\text{ana}} = \hat{P}$.

Question-Form Inventory The four reasoning families differ not only in skill but also in the number of distinct question forms produced by the generator. Deductive supervision yields six forms, corresponding to the Cartesian product of three state slots (`owner`, `possessor`, `integrity`) and two temporal modes (`full_info` at the final time T_{final} versus `hard` at an intermediate time $T_q < T_{\text{final}}$). Abductive supervision yields two forms that share the same surface template but differ in the masking target: `mask_one_event` hides a single operation, while `mask_one_initial_fact` hides a single initial-state fact. Inductive and analogy supervision each yield one question form, drawn at sampling time from four interchangeable phrasing variants to reduce surface-level memorisation. Table 8 summarises the inventory.

Missing Task Families For a given graph, some task families may be empty if the structural sam-

Group	Weight	Relations
A	0.50	grandmother, grandfather, great-grandfather, great-grandmother
B	0.30	aunt, uncle, cousin, grand-nephew
C	0.15	mother-in-law, brother-in-law, grand-niece, great-uncle
D	0.04	grandson, granddaughter, great-grandson, great-granddaughter
E	0.01	nephew, niece, great-aunt
F	0.00	son-in-law, daughter-in-law, sister-in-law, father-in-law

Table 7: Kinship Rule Frequency Groups in Inductive Dataset.

Task	#	Form name	Canonical question template
Deductive	1	Final possessor	"After all events, who possesses {the object}?"
	2	Final owner	"After all events, who owns {the object}?"
	3	Final integrity	"Is {the object} broken or intact after all events?"
	4	Kth-step possessor	"After the kth operation involving {the object}, who possesses {the object}?"
	5	Kth-step owner	"After the kth operation involving {the object}, who owns {the object}?"
	6	Kth-step integrity	"After the kth operation involving {the object}, is {the object} broken or intact?"
Abductive	1	Mask one event	"Some information is missing from the story. Please infer the missing event or information involving {the object}." (a single event in the temporal sequence is hidden)
	2	Mask one initial fact	Same surface template, but a single initial-state fact is hidden instead
Inductive	1	Rule composition	"What is {subj} to {obj}?" — compose the planted kinship rule from support chains. Four interchangeable paraphrases.
Analogy	1	A:B::C:?	"{A} is to {B} as {C} is to ?" — complete the analogy with the missing entity. Four interchangeable paraphrases.

Table 8: Question-form inventory across the four reasoning task families. Deductive yields six forms from three query slots (`possess`, `own`, `integrity`) crossed with two temporal modes (`full_info` at T_{final} versus `hard` at $T_q < T_{\text{final}}$). Abductive yields two forms that share a single surface template but differ in the masking target. Inductive and analogy each use one question form realised through four interchangeable phrasing variants.

pler does not produce the required prerequisites; for example, the static component may be sampled with too few persons to support a two-hop rule. The worked example in Section A.2.9 shows one such graph, in which deductive and inductive tasks are populated by the generator while abductive and analogy tasks are constructed manually for pedagogy.

A.2.9 Worked Example

Concrete Example Instance Table 9 shows one concrete graph instance rendered as a full story and instantiated as four reasoning tasks. The example is chosen because its target object timeline contains diverse dynamic operations rather than only break/repair events. For pedagogy, the abductive and analogy rows are constructed from the same symbolic graph by masking one event and reusing the planted static relation path.

A.3 Training Setup

This appendix gives the implementation details for the training setup described in Section 2.4. The reward definition is given in the main text; here we summarize the model, pre-training configuration, GRPO implementation, parsing and matching protocol, and recipe coverage. All training runs were conducted on a single node with 8 NVIDIA H200 GPUs. One pre-training epoch took approximately 1.5 hours, and one GRPO post-training epoch took approximately 5 hours.

A.3.1 Model Architecture and Tokenizer

All experiments use the same 107M-parameter Qwen 2.5 decoder-only transformer. The model is initialized from scratch with the domain tokenizer and then trained on the KG pre-training corpus. Table 10 lists the architectural settings used by the model.

A.3.2 Pretraining Objective and Hyperparameters

Pre-training uses full-parameter causal language modeling from scratch rather than instruction tuning. The corpus covers the pre-training block $D \in \{1, \ldots, 4\}$ and $T \in \{1, 2\}$. The final pre-trained checkpoint is used as the shared initialization for all RL recipes.

A.3.3 GRPO Post-Training Implementation

RL post-training starts from the final pre-trained checkpoint and uses GRPO with a rule-based reward computed from parsed answers and traces. We do not train a separate learned reward model. Table 12 gives the main GRPO settings.

A.3.4 Generated-Content Parsing and Matching

The verifier converts each generated response into two objects: a predicted answer $\hat{a}$ and a predicted reasoning trace $\hat{\pi}$. It first extracts the solution and answer fields for the target task family. If the solution field is missing or cannot be parsed into the expected trace structure, the process indicator is set to $\mathrm{m_P} = 0$. The answer indicator $\mathrm{m_A}$ is computed by matching the extracted answer against the gold answer after light normalization; for abductive tasks, equivalent descriptions of the same missing event or initial fact are also accepted.

Process matching is task-specific. Dynamic tasks are parsed into object-state traces, where each comparable state records owner, possessor, and integrity. Static tasks are parsed into relation paths and composite-rule applications. Table 13 summarizes the protocol used to compute the binary process and answer indicators.

The examples below separate the generated text from the structured view produced by the same parser schema used during evaluation.

The binary indicators used in Section 2.4 are therefore $A = \mathrm{m_A}(\hat{a}, a^\star)$ and $P = \mathrm{m_P}(\hat{\pi}, \pi)$. Intermediate parser diagnostics are used only for analysis; reward and strict correctness use the binary indicators.

A.3.5 RL Recipes and Task Instantiation

Each recipe selects a subset of the (D, T) grid and allocates the same task budget reported in Section 2.4. The main sampler recipes are drawn by cell; uniform and diagonal recipes are produced by the corresponding curriculum or data-volume schedules. Table 16 summarizes the 15 recipe specifications.

The Baseline recipe is a pre-training-region control. The Cmplx recipes, Cmplx-Mid, Cmplx-High, Cmplx-Intersect, and Cmplx-Uniform, keep depth shallow and vary environment complexity. The Depth recipes, Depth-Mid, Depth-High, Depth-Intersect, and Depth-Uniform, keep environment complexity low and vary the depth axis. The Diag recipes, Diag-Mid and Diag-High, move both axes together. The Mix recipes, Deep-Mix, Shallow-Mix, Offbase-Mix, and Full-Coverage, compare broad-coverage distributions that include or exclude the pre-training block while expanding depth and complexity coverage.

For each recipe, tasks are distributed uniformly over its selected cells. The total task budget is fixed at 220K tasks, with 140K deductive, 60K abductive, 10K inductive, and 10K analogy tasks. This fixed mixture makes recipe comparisons mainly reflect grid coverage rather than changes in task-family proportion.

A.4 Experiment Evaluation Metrics

This appendix gives the formal definitions used by the experiment sections. The main text uses these metrics only at a high level, while the formulas below specify how the reported recipe ranks, heatmaps, in/out-of-recipe gains, and task-pair correlations are computed.

Strict pass@k, SG, and CG For recipe r, task family q, and evaluation subset S, let $p^{(k)}_{r,q}(S)$ denote strict process-verified $\mathrm{pass@}k$. The pre-trained model is denoted by pre. Single-sample gain (SG) and ceiling gain (CG) are

$$\begin{aligned} \mathrm{SG}_{r,q}(S) &= p^{(1)}_{r,q}(S) - p^{(1)}_{\mathrm{pre},q}(S), \\ \mathrm{CG}_{r,q}(S) &= p^{(128)}_{r,q}(S) - p^{(128)}_{\mathrm{pre},q}(S). \end{aligned}$$

SG measures greedy or single-sample improvement, whereas CG measures the change in the sampled reasoning ceiling under 128 attempts. As a worked example, the Shallow-Mix recipe on deductive at $(D{=}5, T{=}3)$ yields $p^{(1)}_{\text{Shallow-Mix,Ded}} = 0.58$ versus $p^{(1)}_{\mathrm{pre,Ded}} = 0.42$, giving $\mathrm{SG} = +0.16$; the same recipe-cell pair gives $p^{(128)}_{\text{Shallow-Mix,Ded}} = 0.79$ versus $p^{(128)}_{\mathrm{pre,Ded}} = 0.64$, so $\mathrm{CG} = +0.15$. When the two gains disagree, the (SG, CG) plane separates capability extension (both positive) from mode collapse (positive SG with near-zero or negative CG).

Global Mean Rank Let $C_{q,r}(d,t) = \mathrm{CG}_{r,q}(S_{d,t})$ be the cell-level ceiling gain of recipe r on task family q, where $S_{d,t}$ is the evaluation subset at depth d and complexity level t. Let Ω_q be the valid evaluated cells for task q. For a complexity level t, define $\Omega_q(t) = \{d : (d,t) \in \Omega_q\}$; for a depth d, define $\Omega_q(d) = \{t : (d,t) \in \Omega_q\}$. The complexity-axis

mean rank is

$$\bar{C}^{T}_{q,r}(t) = \frac{1}{|\Omega_q(t)|} \sum_{d:(d,t)\in\Omega_q} C_{q,r}(d,t),$$

$$\mathrm{MR}^{T}_{q,r} = \frac{1}{|\mathcal{T}_q|} \sum_{t\in\mathcal{T}_q} \mathrm{rank}^{\downarrow}_{r}\big(\bar{C}^{T}_{q,r}(t)\big)\,,$$

where $\mathcal{T}_q$ is the set of evaluated complexity levels. The depth-axis mean rank is defined symmetrically:

$$\bar{C}^{D}_{q,r}(d) = \frac{1}{|\Omega_q(d)|} \sum_{t:(d,t)\in\Omega_q} C_{q,r}(d,t),$$

$$\mathrm{MR}^{D}_{q,r} = \frac{1}{|\mathcal{D}_q|} \sum_{d\in\mathcal{D}_q} \mathrm{rank}^{\downarrow}_{r}\big(\bar{C}^{D}_{q,r}(d)\big)\,,$$

where $\mathcal{D}_q$ is the set of evaluated depths. The operator $\mathrm{rank}^{\downarrow}_{r}$ ranks recipes by mean CG in descending order within a slice, so rank 1 is best and lower mean rank is better.

Local Cell CG For each valid cell $(d,t) \in \Omega_q$, the local cell-level gain is

$$\mathrm{CG}_{q,r}(d,t) = p^{(128)}_{r,q}(S_{d,t}) - p^{(128)}_{\mathrm{pre},q}(S_{d,t}).$$

The marginal heatmaps in the main text and appendix visualize these values directly. Positive cells indicate sampled-ceiling expansion relative to the pre-trained model, whereas negative cells indicate suppression. The in-recipe / out-of-recipe split used in Section 4 is defined by partitioning Ω_q into the cells that overlap recipe r's post-training distribution and the cells that do not; the per-cell formula above is unchanged, only the aggregation set differs.

Task-Pair Pearson Correlation For task-pair analysis, we compare the CG patterns of two task families (q_a, q_b). Let $u = (r,d,t)$ denote one recipe-cell observation and let $\mathcal{U}_{q_a,q_b}$ be the set of observations that are valid for both tasks. In the full grid, $|\mathcal{U}_{q_a,q_b}| = 15 \times 60 = 900$. If a task pair has invalid cells, we use only the common valid observations. For compactness, write $C_{q,u} = C_{q,r}(d,t)$. The global mean for task q over this pooled set is

$$\bar{C}_q = \frac{1}{|\mathcal{U}_{q_a,q_b}|} \sum_{u\in\mathcal{U}_{q_a,q_b}} C_{q,u}.$$

Let $\mathcal{U} = \mathcal{U}_{q_a,q_b}$. The Pearson correlation is

$$\begin{aligned} N_{ab} &= \sum_{u\in\mathcal{U}} (C_{q_a,u} - \bar{C}_{q_a}) \\ &\quad \times (C_{q_b,u} - \bar{C}_{q_b}), \\ Z_a &= \sqrt{\sum_{u\in\mathcal{U}} (C_{q_a,u} - \bar{C}_{q_a})^2}, \\ Z_b &= \sqrt{\sum_{u\in\mathcal{U}} (C_{q_b,u} - \bar{C}_{q_b})^2}, \\ r(q_a,q_b) &= \frac{N_{ab}}{Z_a Z_b}. \end{aligned}$$

We compute this correlation once over the merged recipe-cell observations, rather than computing one correlation per recipe and averaging those values. The same formula is applied to subsets of $\mathcal{U}$ to obtain the family- and region-conditional correlations reported in Appendix A.6.2; for example, restricting $\mathcal{U}$ to the four Depth-family recipes ($|\mathcal{U}| = 4 \times 54 = 216$ valid observations) yields the within-family Pearson $r = 0.62$ cited for the deductive–abductive pair.

A.5 Ablation Study

This appendix collects ablations for implementation choices that are fixed in the main experiments. These studies are not separate research questions; they check whether the main recipe-level conclusions depend on the post-training budget, GRPO rollout group size, or deductive–abductive task composition.

A.5.1 Data Volume

This ablation supports the main experimental choice of using a 220k post-training budget. We vary the amount of post-training data while keeping the recipe family fixed. Figure 6 reports the per-task breakdown across the four recipe families: deductive, abductive, inductive, and analogy all reach higher endpoint performance at larger budgets, although individual curves are not strictly monotonic and the gain magnitudes differ by task. The 220k setting used in the main experiments captures most of the gain, while 440k still improves performance at a higher data cost. This supports using the 220k setting as a practical middle point rather than as an artificially saturated regime.

A.5.2 Number of Rollouts

This ablation checks whether the main GRPO rollout group size behaves like a capability-scaling

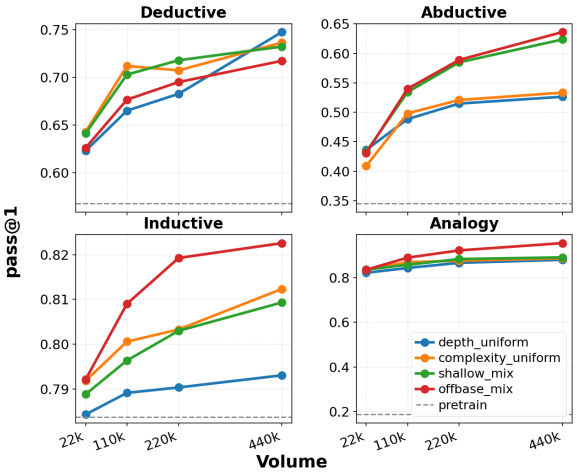


Figure 6: Per-task data-volume scaling. The four panels report greedy pass@1 for deductive, abductive, inductive, and analogy as the post-training data budget varies from 22k to 440k.

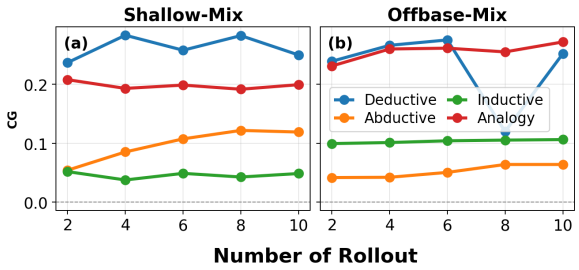


Figure 7: Rollout-count ablation. PR pass@128 changes relative to the pre-trained model as the GRPO rollout group size varies over $N \in \{2, 4, 6, 8, 10\}$ for Shallow-Mix and Offbase-Mix. Larger rollout groups do not yield monotonic gains.

knob. We vary the rollout group size while keeping the recipe family fixed. Figure 7 shows that larger rollout groups do not produce monotonic improvements. Shallow-Mix reaches its strongest overall PR pass@128 at $N = 8$, whereas Offbase-Mix is less stable under larger rollout groups and shows a sharp deductive drop at $N = 8$. The recipe-conditional optimum is consistent with the bias–variance trade-off of GRPO's group-wise advantage estimator: a larger N reduces the variance of the advantage at the cost of fewer optimisation updates per prompt under a fixed total-rollout budget. The sweet spot therefore depends on how dispersed each recipe's reward distribution is across prompts—Shallow-Mix, whose post-training cells include more high-variance regions, benefits from larger groups, while Offbase-Mix, whose trained cells are more reward-concentrated, appears closer to saturation by $N = 6$. Rollout count is therefore a recipe-dependent stability knob rather than a uniform capability-scaling lever, which is why we hold it fixed across the main-paper recipes.

A.5.3 Task Mix

This ablation checks whether the main deductive–abductive training mixture is important under a fixed post-training budget. We keep the total budget near 220k and vary the internal deductive–abductive composition. Figure 8 compares the three mixed-task settings across all four recipe families. Mixed compositions are more stable than pure single-task training: d140/a60 gives the strongest overall result, while increasing the abductive share improves abductive performance but can trade off against other task families.

Figure 9 provides the complementary pure-task comparison. Pure deductive and pure abductive post-training are less stable than mixed training across recipe families, which supports using a mixed dynamic-task budget in the main experiments.

A.5.4 Process-Reward Ratio

This ablation examines how the trade-off between process- and outcome-based reward signals affects RL post-training. We sweep `process_ratio` $\in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ on two representative RL recipes, Offbase-Mix and Shallow-Mix, keeping the total post-training budget near 220k. `process_ratio` is the step-level weight in the gate-conditioned reward, with the outcome (final-answer) weight fixed at $1 -$ `process_ratio`; `process_ratio` = 0.8 is the default used in the main experiments. Figure 10 reports skill gain (SG = pass@1(RL greedy) − pass@1(Pretrain-Ep3)) as a function of `process_ratio` for the four task families. Pure outcome reward (`process_ratio` = 0.0) degrades abductive performance below the pre-trained baseline in both recipes; introducing even a small process-reward component (`process_ratio` = 0.2) recovers and stabilises it. Inductive and analogy gains remain positive but fluctuate above `process_ratio` = 0.2, while deductive gains peak at the default `process_ratio` = 0.8, supporting the main-experiments choice.

A.6 Supplementary Experiments and Results

This appendix collects supplementary figures that extend the main experimental sections without changing their core claims. Each group below states which main-text result it supports, what the

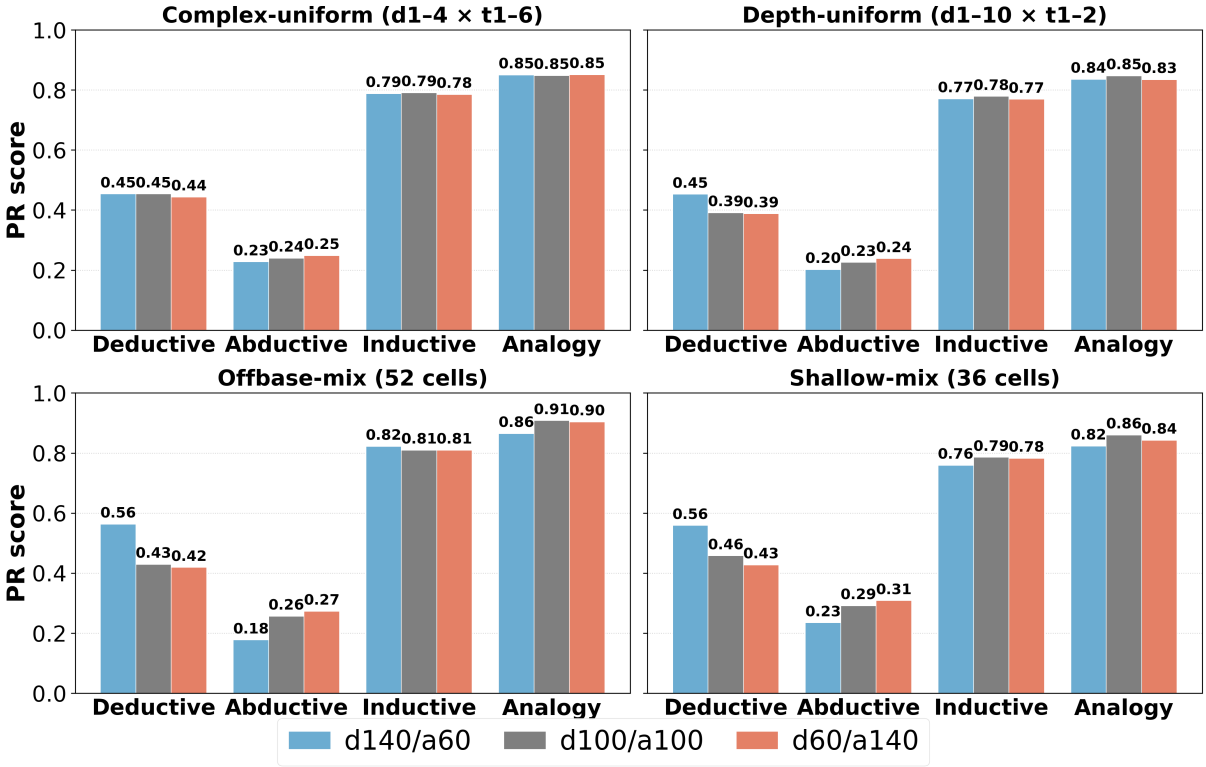


Figure 8: Task-mix ablation under a fixed 220k budget across all four recipe families (Complex-uniform, Depth-uniform, Offbase-mix, Shallow-mix). The three mixed-task settings (d140/a60, d100/a100, d60/a140) vary the deductive–abductive ratio while keeping the total budget approximately fixed.

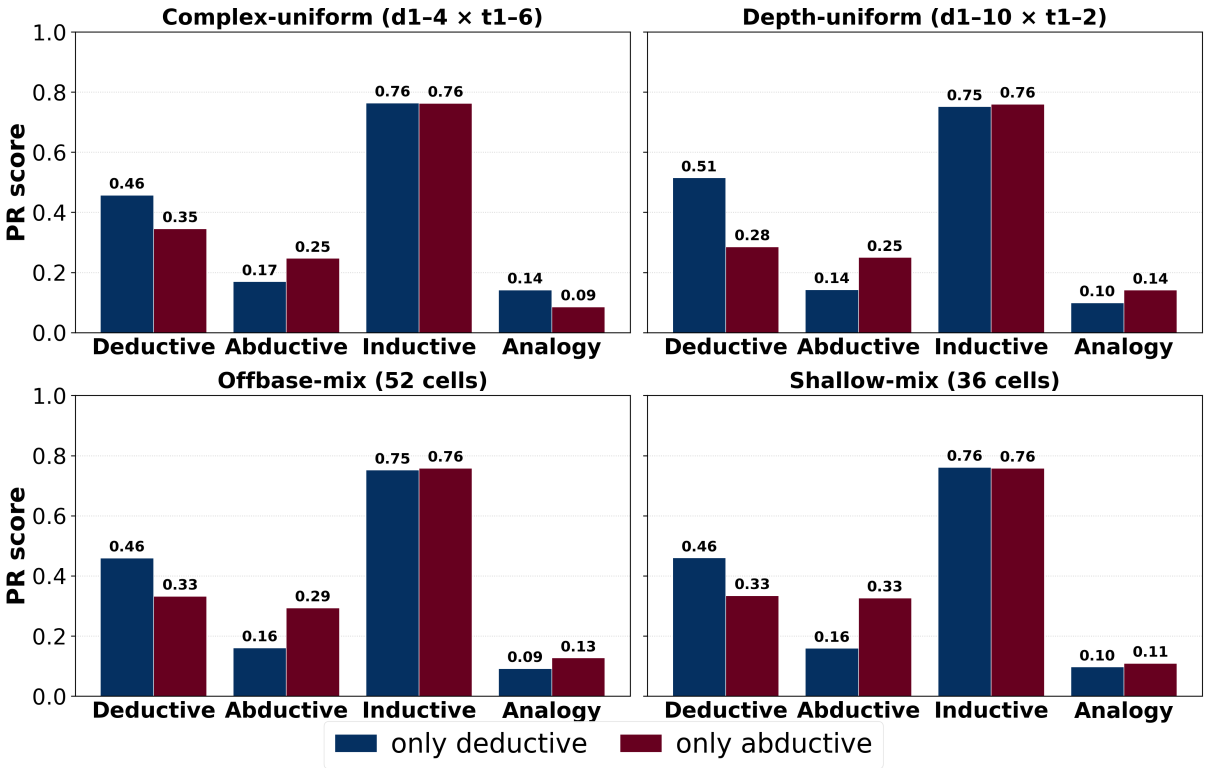
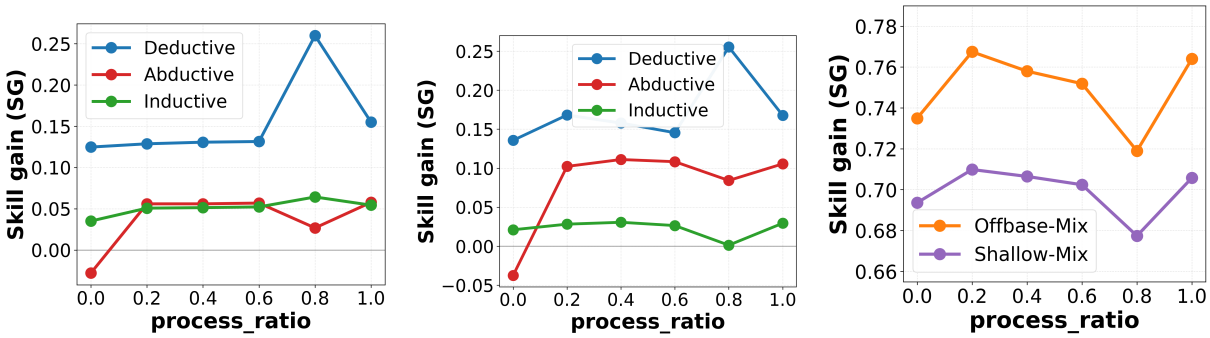


Figure 9: Pure-task task-mix ablation under a fixed 220k budget. The figure compares pure deductive and pure abductive post-training across the four recipe families.



(a) Offbase-Mix. (b) Shallow-Mix. (c) Analogy across both recipes.

Figure 10: Process-reward-ratio ablation. Skill gain (SG = pass@1(RL greedy) − pass@1(Pretrain-Ep3)) as a function of `process_ratio`. Panels (a) Offbase-Mix and (b) Shallow-Mix show the three core task families (deductive, abductive, inductive); panel (c) compares analogy across the two recipes on a separate axis because analogy SG is much larger than the other families. Pure outcome reward (`process_ratio` = 0.0) hurts abductive in both recipes.

figure measures, and what pattern should be read from the figure.

A.6.1 Full Marginal CG Heatmaps

These heatmaps complement the data-distribution and task-family analyses in Sections 3 and 4. The main text shows representative marginal CG heatmaps for deductive and abductive tasks; here we show the full 15-recipe grid for all four task families. The figures measure cell-level CG on the 60-cell (D, T) grid, so they reveal where each post-training recipe expands or suppresses the sampled reasoning ceiling relative to the pre-trained model. The main visible pattern is that joint-coverage recipes produce broader gains than single-axis recipes, while abductive heatmaps contain more negative regions when the post-training region does not support the relevant inverse-reasoning behavior.

These supplementary heatmaps also place our recipe design in context. Related RLVR work asks whether RL mainly reweights behaviors already supported by the base model or expands the model's reasoning boundary (Ni et al., 2025; Yue et al., 2025; Liu et al., 2025b). Our setting adds a distributional view of that question: coverage determines which parts of the depth–complexity grid give RL enough reward signal to use existing capability or extend it. Controlled reasoning benchmarks vary difficulty axes for evaluation stress-tests (Mirzadeh et al., 2025; Wang et al., 2026; Paqaleh et al., 2025); here the same kind of axis structure defines the post-training recipe itself, making the heatmaps a direct view of where each recipe transfers, saturates, or suppresses capability.

Reading the heatmaps row by row, three patterns are immediately visible. First, the in-recipe cells (black borders) are the most reliably positive for every family except abductive, confirming that RL gains are densest inside a recipe's own coverage and that out-of-recipe transfer is the loose end. Second, joint-coverage recipes (Shallow-Mix, Offbase-Mix, Full-Coverage) paint a much greener band across the depth axis for deductive and abductive than the single-axis recipes, matching the joint-coverage takeaway of Section 3. Third, the abductive block shows visible *negative* (red) cells outside the pre-training region $(D{\leq}4, T{\leq}2)$, and these red cells are concentrated in the same single-axis recipes that omit complexity training; this localises the abductive distribution sensitivity reported in Section 4 to specific axis-omission patterns.

A.6.2 Task-Family Supplementary Results

These figures complement the main task-family analysis in Section 4. The mean-rank plot in Figure 13 extends the recipe-ranking view from deductive and abductive tasks to inductive and analogy tasks. It shows that joint-coverage recipes remain at least competitive with the best single-axis recipes on both axes. The remaining correlation plots in Figure 14 complete the task-pair correlation analysis: deductive–analogy is moderately coupled, while deductive–inductive, abductive–inductive, and abductive–analogy are weakly coupled. This supports the main claim that reasoning task families do not form one uniform RL regime.

Task-pair correlation drilldowns by recipe family. The overall Pearson correlations reported in

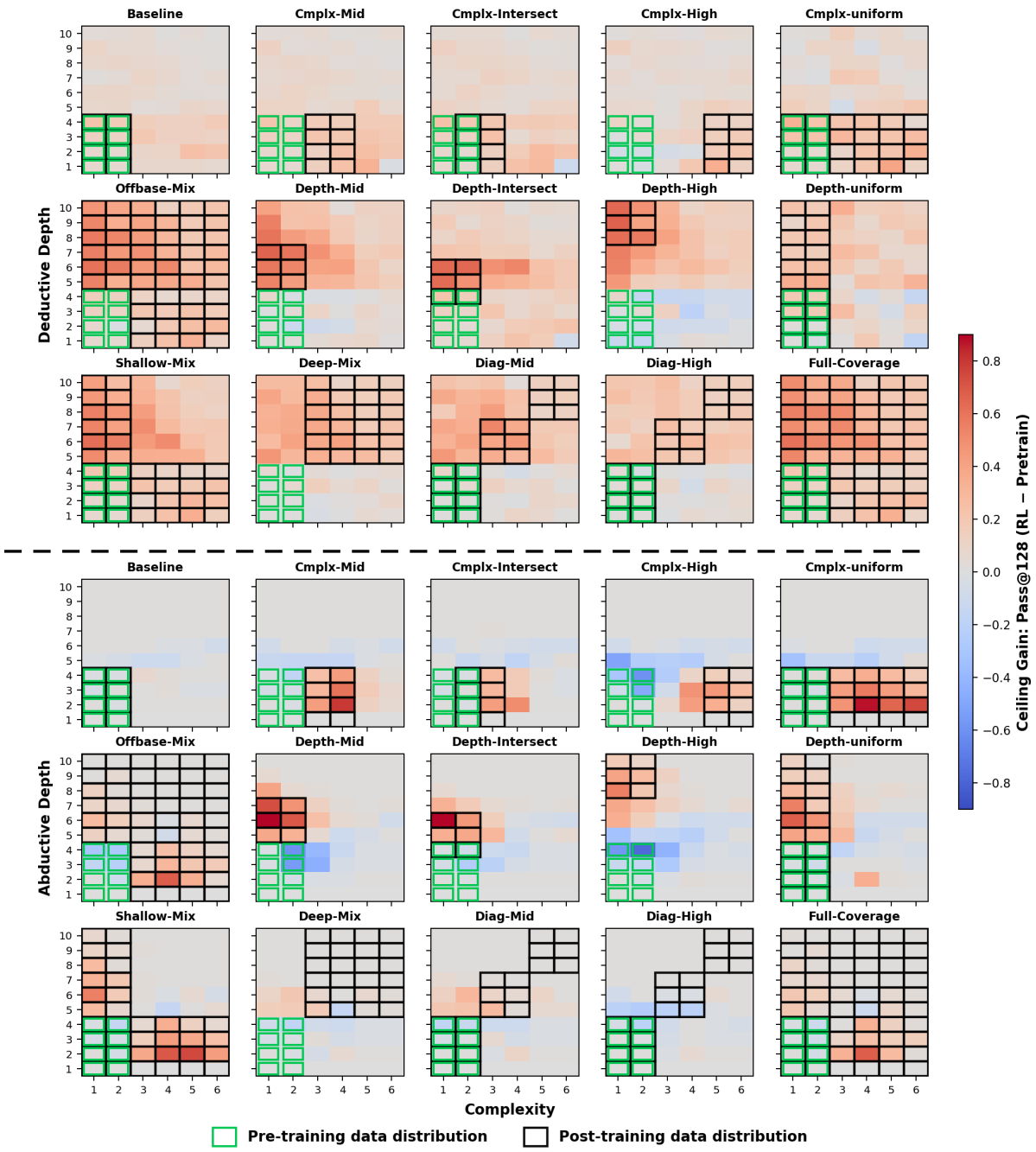


Figure 11: Full 15-recipe per-cell marginal CG heatmaps on the 60-cell (D, T) grid for deductive (top three rows) and abductive (bottom three rows). Each task block shares the same column order: top row is the baseline plus the four Cmplx single-axis variants, middle row is Offbase-Mix plus the four Depth single-axis variants, and bottom row groups the joint-coverage, diagonal and full-coverage recipes. Black borders mark each recipe's post-training cells, green borders mark the pre-training cells $(D{\leq}4, T{\leq}2)$, and grey cells were not evaluated. The colour scale is shared with the inductive / analogy figure below.

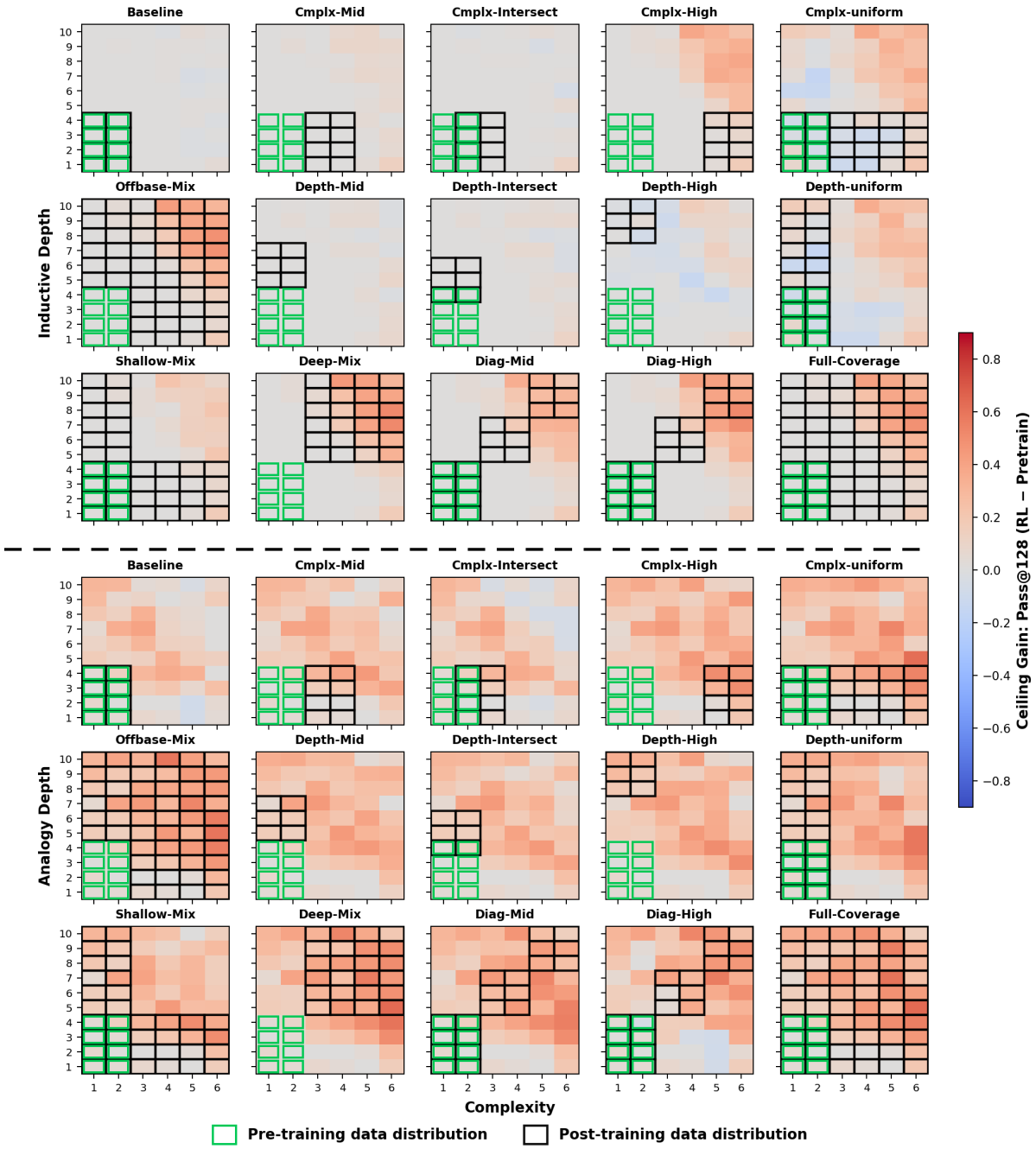


Figure 12: Full 15-recipe per-cell marginal CG heatmaps on the 60-cell (D, T) grid for inductive (top three rows) and analogy (bottom three rows). The layout and the colour scale match the deductive / abductive figure above, so cells can be compared across all four task families.

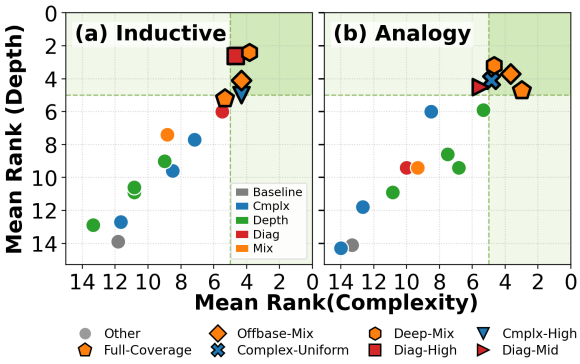


Figure 13: Mean depth-rank versus mean complexity-rank for inductive and analogy. Each point corresponds to one RL recipe. Lower values are better on both axes. Joint-coverage recipes remain at least competitive with the best single-axis recipes.

Figures 5 and 14 aggregate across all 15 recipes and 60 cells. To localise where the deductive–abductive and inductive–analogy couplings come from, we drill the two strong pairs down by recipe family (Baseline, Cmplx, Depth, Diag, Mix). Figures 15 and 16 show the family drilldowns. Within-family correlations are heterogeneous rather than uniformly stronger: for the deductive–abductive pair the overall Pearson $r = 0.40$ rises to $r = 0.62$ in the Depth family and $r = 0.47$ in the Cmplx family, but drops to $r = 0.17$ in the Mix family and $r = 0.16$ for the Baseline. For inductive–analogy the strongest within-family couplings are in Diag ($r = 0.61$) and Mix ($r = 0.56$), while the Depth family is weakest ($r = 0.19$). The overall positive correlations therefore reflect family-specific mechanisms rather than a uniform global property: depth-axis training keeps deductive and abductive coupled, while diagonal or balanced mixing keeps inductive and analogy coupled.

A.6.3 Pass@k Exploration Profile

These pass@k curves complement the SG/CG analysis in Section 4 by showing the full exploration profile behind a model's pass@1 (skill) and pass@128 (ceiling) summary statistics. Figure 17 plots pass@k for $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ on a log-k axis for the 15 RL recipes together with the Pretrain-Ep3 baseline, separately per task family.

The four task families show qualitatively different exploration profiles. For *deductive*, every RL recipe clearly dominates the baseline across the whole k range; joint-coverage Mix-family recipes occupy the top envelope at every k. For *abductive*, gains are smaller and recipe-dependent: several RL curves track the baseline closely, while the best recipes shift the curve upward without changing its overall shape. For *inductive*, the baseline's pass@k climbs steeply with k and approaches the RL recipes by $k = 128$; the RL advantage is concentrated at small k and shrinks with sampling. For *analogy*, the baseline shows the largest exploration headroom (pass@1 ≈ 0.10 climbing to pass@128 ≈ 0.65), while almost every RL recipe sits at pass@$k \approx 0.85$ and is essentially flat across k—the canonical signature of mode collapse, consistent with the analogy pass@128-vs-pass@1 collapse highlighted in Section 4 and the (SG, CG) plane of Figure 4. The single non-collapsed analogy exception (Diag-High, orange triangle) shows a normal upward sweep (pass@1 ≈ 0.55 to pass@128 ≈ 0.85) and visually confirms that mode collapse is recipe-conditional rather than intrinsic to the task.

A.6.4 Benchmark Supplementary Results

This appendix complements the benchmark evaluation in Section 6. Table 17 records the model metadata behind the leaderboard. Public models are cited from official pages, Hugging Face model cards, arXiv papers, or official documentation. Offbase-mix, Shallow-mix, Cmplx-uniform, and Depth-uniform are internal 107M Qwen2.5-style checkpoints from this work. For FP8 releases, the table reports the architecture scale rather than Hugging Face tensor-artifact totals; GPT-5.4 does not disclose its parameter count.

The main text reports the benchmark leaderboard, while Figure 18 shows how models rank across the depth and complexity axes within each reasoning family. The benchmark pattern supports the main task-family asymmetry: strong models can do well on deductive tasks while remaining much weaker on abductive tasks.

A.6.5 Best RL Recipe per Region

Complementing the recipe-design view in Section 3, we report *which* RL recipe achieves the highest pass@128 in each (task, region) cell, treating the recipe choice itself as the free axis. Figures 19, 20, and 21 show the per-cell winner for the depth axis ($D = 1, \ldots, 10$), the complexity axis ($T = 1, \ldots, 6$), and the nine depth-band by tier-band regions, respectively. Each cell is coloured by the winning RL recipe and annotated with that recipe's pass@128 score in the corresponding scope.

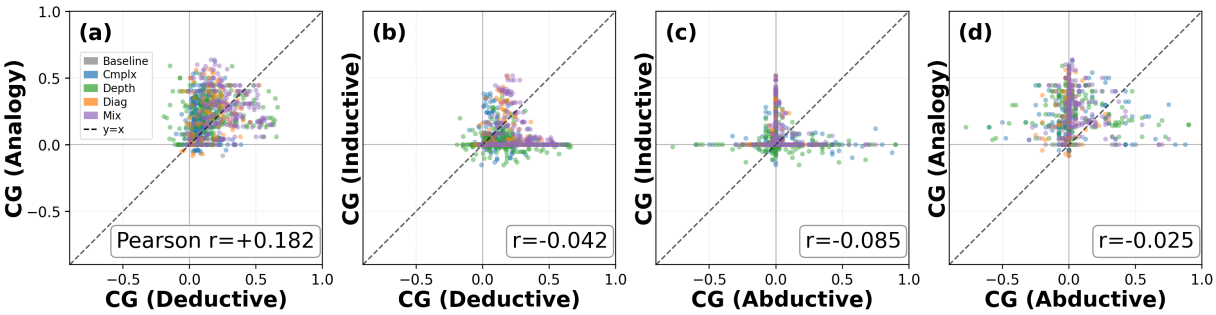


Figure 14: Remaining task-pair CG correlations across the 60-cell grid and 15 RL recipes. The deductive–analogy pair is moderately coupled, while the deductive–inductive, abductive–inductive, and abductive–analogy pairs are weakly coupled. Both axes report CG. The two strongest task pairs (deductive–abductive and inductive–analogy) are shown in the main text in Figure 5.

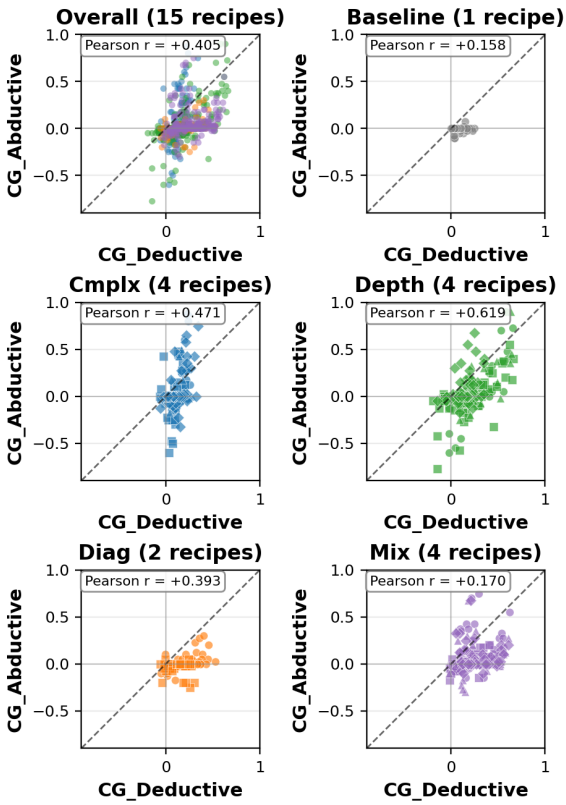


Figure 15: Deductive–abductive correlation drill-down by recipe family. "Overall" uses all 15 recipes; each subsequent panel restricts to one recipe family and recomputes the Pearson correlation within that family. The Depth family ($r = 0.62$) shows the strongest within-family coupling, substantially above the overall $r = 0.40$; the Mix family ($r = 0.17$) and Baseline ($r = 0.16$) are weaker.

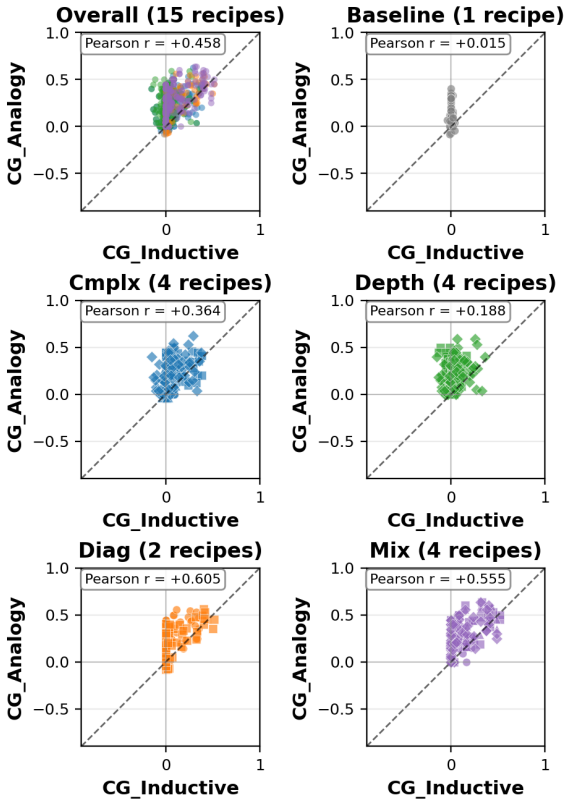


Figure 16: Inductive–analogy correlation drill-down by recipe family. Same layout and convention as Figure 15. The within-family pattern differs from deductive–abductive: Diag ($r = 0.61$) and Mix ($r = 0.56$) families show the strongest coupling, while the Depth family is comparatively weak ($r = 0.19$).

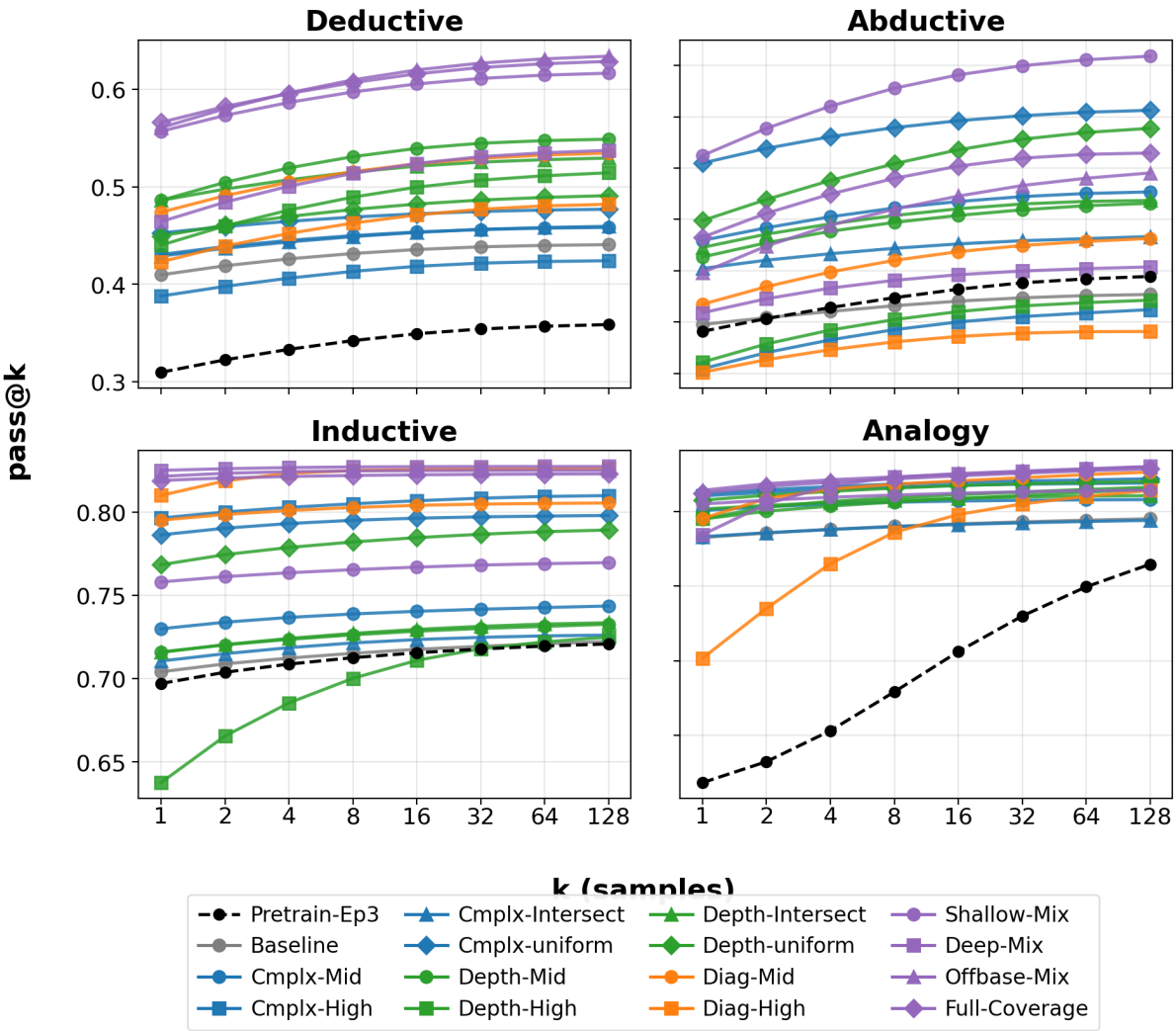


Figure 17: Pass@k curves per task family for the 15 RL recipes and the Pretrain-Ep3 baseline (black dashed). The x-axis is sample budget k on a log-2 scale. RL recipes are coloured by family (grey Baseline, blue Cmplx, green Depth, orange Diag, purple Mix), with marker variation within each family. Deductive shows consistent RL gains across k, while abductive gains are smaller and recipe-dependent; inductive shows shrinking RL advantage as k grows; analogy shows the clearest mode-collapse pattern, with most RL recipes flat near pass@$k \approx 0.85$ while the baseline's pass@k rises sharply.

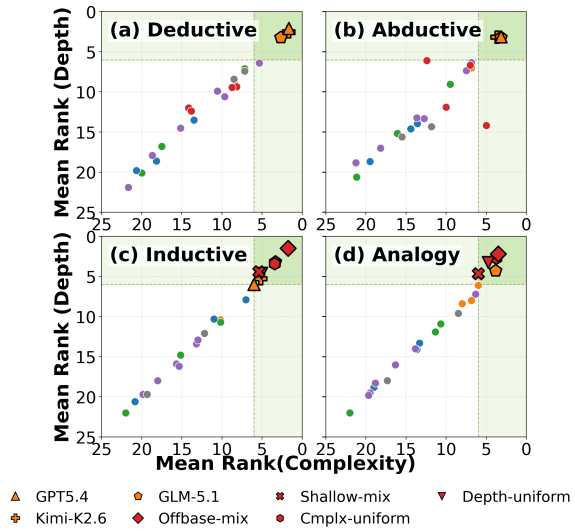


Figure 18: Cross-axis mean-rank benchmark over 22 models. Each panel ranks models by their mean performance across depth and complexity tiers for one reasoning family. Labels are shown only for models in the top-right dual-winner region.

Three patterns are visible across the three views; see also Appendix A.6.2 for the correlation-based view of the same task-family pattern. First, the winner is *task-dependent*: deductive and abductive favour Cmplx-$*$, Depth-$*$, or Mix recipes depending on the region, while inductive and analogy keep high pass@128 under several recipe families. Second, the winner shifts along each axis: for deductive and abductive, Cmplx-uniform is strongest in several shallow-depth or low-complexity regions, but Depth- and Mix-family recipes appear more often in harder cells; the absolute pass@128 also drops sharply with combined difficulty (deductive D10 $\approx$ 0.35; abductive D10 $\approx$ 0.05). Third, inductive and analogy values stay in a much narrower band across regions (typically 0.78–1.00), consistent with the smaller magnitudes already reported in Appendix A.6.2; for these two families, the winner often changes while the score changes only by a few percentage points. Together these views support recipe selection as a region- and task-conditional decision rather than a single best-overall recipe.

A.7 Curriculum Supplementary Experiments

This appendix collects the curriculum analyses that complement Section 5. The main text reports the fixed-budget granularity result; here we provide the curriculum-order and replay settings, the full deductive/abductive trajectories, the full-grid heatmaps, and the inductive/analogy extensions.

A.7.1 Task Setting

Task 1: Curriculum Order. On both axes, we run a 3-block hard-to-easy (H2E) control against the easy-to-hard (E2H) curriculum. E2H starts with the easiest cells (low D or low T) and ends with the hardest; H2E reverses the order.

Task 2: Historical-Data Replay. We add 20% historical-data replay to the 3-block curriculum and compare it with the matched 3-block no-replay baseline. The replay sample is drawn uniformly from already-seen blocks, so under an E2H schedule it concentrates on easier cells.

A.7.2 Observation

Figure 22 shows two additional patterns. First, curriculum order changes where gains remain: E2H favours hard regions and H2E favours easy regions, but neither matches uniform mixing overall. Second, replay most clearly helps earlier or easier regions, especially $D \leq 5$ on the depth axis or $T \leq 3$ on the complexity axis; in harder regions, replay and no-replay are closer and less consistently separated.

A.7.3 Discussion

Curriculum order changes retained regions but not global coverage. SGD-style optimisation exhibits a recency bias: the final gradient updates exert the strongest influence on the trained policy because no subsequent step overwrites them. Easy-to-hard (E2H) curricula end on the hardest block and preserve hard-region capability; hard-to-easy (H2E) curricula end on the easiest block and preserve easy-region capability. The directional pattern is a property of the optimiser rather than of the data, and curriculum order cannot substitute for coverage. Recent E2H curriculum work on LLM RL (Parashar et al., 2025) attributes the hard-task gain to scaffolding; our data instead support a recency-based attribution, since uniform mixing—which contains no scaffolding phase—outperforms E2H on the global grid. Reverse-curriculum studies (Florensa et al., 2017) report H2E advantages in sparse-reward robotic RL through near-goal-state exposure, a mechanism that does not transfer to dense-reward RLVR. Self-paced LLM RL (Do et al., 2025) adaptively selects training data based on difficulty estimates; in our framework this corresponds to a retention knob rather than a coverage substitute.

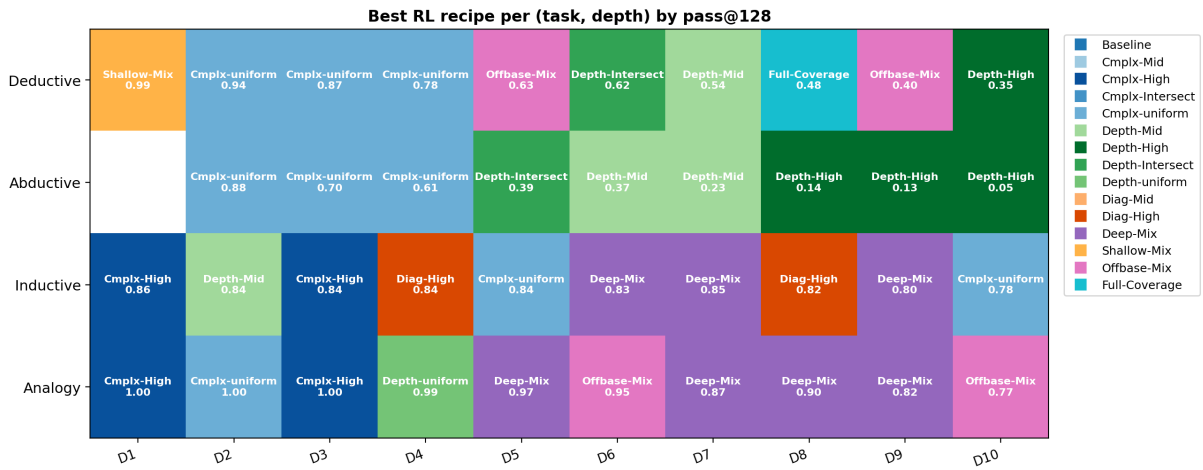


Figure 19: Best RL recipe per (task, depth) cell by pass@128. Rows are task families; columns are depth values $D = 1, \ldots, 10$. Each cell is coloured by the winning recipe and labelled with the recipe name plus its pass@128 score. Deductive and abductive winners shift from Cmplx-$*$ at shallow depth to Depth- or Mix-family recipes at larger depth, with absolute pass@128 dropping sharply; inductive and analogy stay in a narrow high band across depths despite changes in the winning recipe.

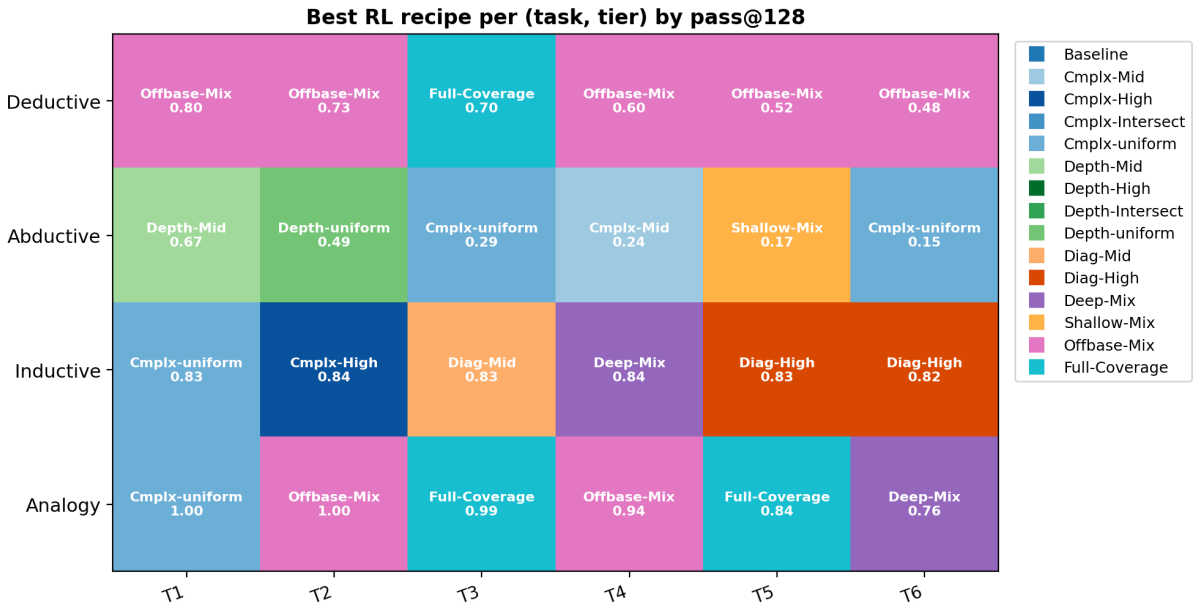


Figure 20: Best RL recipe per (task, complexity tier) cell by pass@128. Same convention as Figure 19 but the x-axis is the complexity tier $T = 1, \ldots, 6$. Cmplx-family recipes appear in several low-complexity columns, while higher-complexity columns mix Depth-, Mix-, and diagonal-family winners depending on the task.

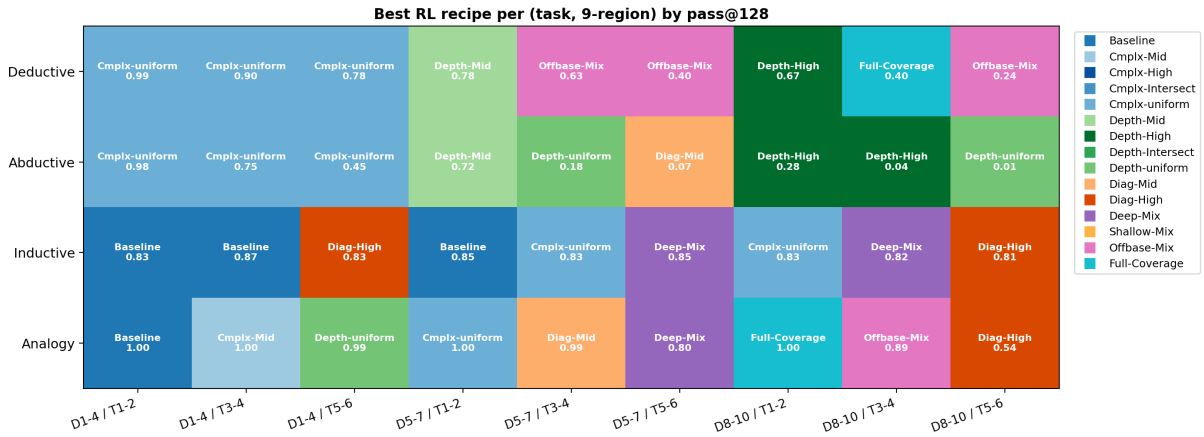


Figure 21: Best RL recipe per (task, region) cell by pass@128. The nine regions partition the 60-cell grid into depth bands ($D \in [1,4], [5,7], [8,10]$) crossed with complexity bands ($T \in [1,2], [3,4], [5,6]$). Same convention as Figures 19 and 20.

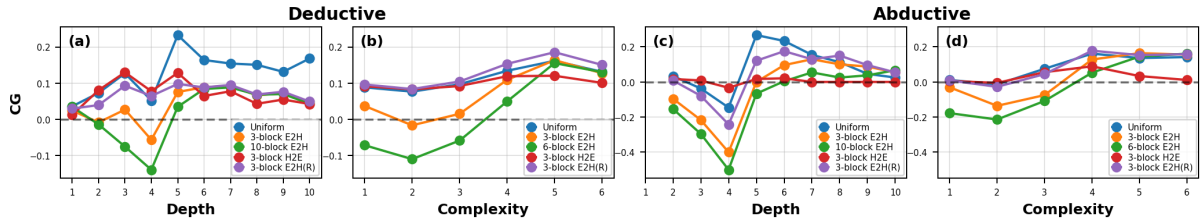


Figure 22: Curriculum CG trajectories for deductive and abductive on the depth and complexity axes. Within each task pair, the left panel shows depth schedules and the right panel shows complexity schedules. Higher curves indicate stronger CG.

Replay benefits easier regions more reliably than harder regions. A uniform-time replay buffer draws from already-seen blocks, which under an E2H curriculum correspond to easier cells. The replayed 20% therefore acts primarily as a retention signal on those cells, while harder current-block cells receive the same 80% update stream as the no-replay baseline. Replay therefore behaves more like a retention mechanism than a direct acquisition mechanism: easier regions improve when forgetting is the main bottleneck, whereas harder regions show smaller and less consistent changes when the bottleneck is acquiring new behavior. Recent LLM-RL replay methods—RLEP (Zhang et al., 2025b), Retrospective Replay (Dou et al., 2025), ExGRPO (Zhan et al., 2026), and EFRame (Wang et al., 2025a)—all report accuracy gains on mathematical reasoning, but each employs prioritised or filtered replay (verified successes, exploration-promising trajectories, or correctness-weighted experience). Our analysis predicts that such prioritisation should accelerate hard-region learning; our deliberately uniform-time design isolates the retention-only contribution. Classical experience replay (Lin, 1992; Mnih et al., 2015) is foundational to these methods and aligns with our retention-based reading.

A.7.4 Full-Grid and Task-Family Extensions

The full-grid curriculum heatmaps in Figure 23 show where each curriculum variant improves or suppresses CG across the full (D, T) grid for deductive and abductive tasks. They clarify that curriculum ordering changes retained regions, while uniform mixing remains the stronger global default. Figure 24 extends the trajectory analysis to inductive and analogy tasks. Together with Figure 22, these results show that the granularity ordering (uniform $\geq$ 3-block $\geq$ many-block) and the replay-helps-easier-regions tendency hold as global patterns across all four task families. The effect is less pronounced for inductive and analogy because their CG range is narrower, leaving less ceiling headroom for curriculum order to separate retained from non-retained cells.

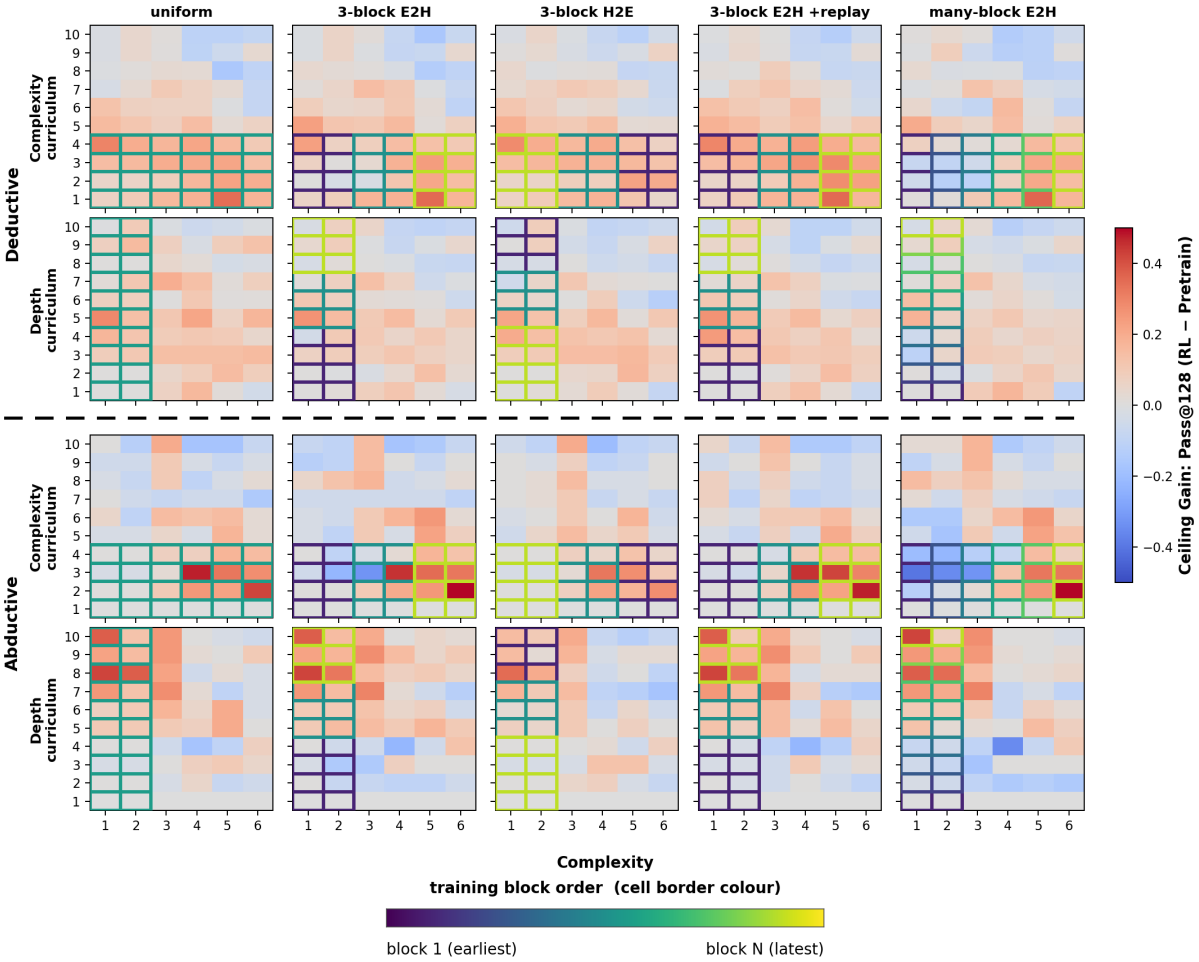


Figure 23: Full-grid CG heatmaps for the depth- and complexity-axis curricula on deductive (top two rows) and abductive (bottom two rows). Within each task block the top sub-row is the complexity-axis curriculum and the bottom sub-row is the depth-axis curriculum; the five columns are uniform, 3-block E2H, 3-block H2E, 3-block E2H +replay and many-block E2H. Cell borders are viridis-coloured by training block index (block 1 → block N, dark purple → yellow). The colour scale is shared across the two task families, and a bold dashed line separates the two task blocks.

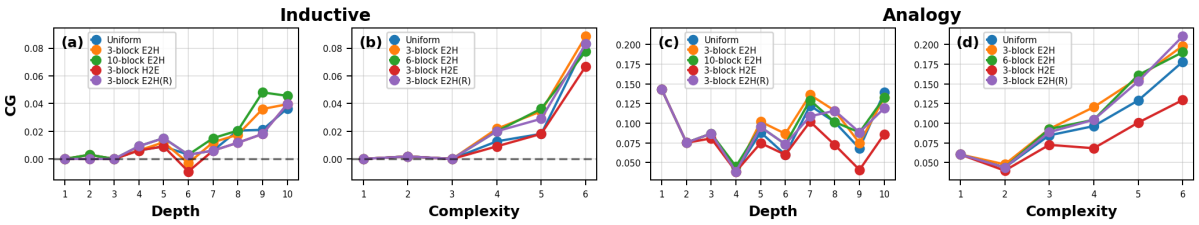


Figure 24: Curriculum CG trajectories for inductive (left two panels) and analogy (right two panels) along the depth and complexity axes. Same layout and conventions as the deductive/abductive trajectory figure (Figure 22). The inductive and analogy trajectories show the same qualitative curriculum and replay patterns as deductive and abductive, with smaller magnitudes.

Field	Content
Structural facts	Target-object reasoning depth $D = 10$, generated from two chains with per-chain depth 7, complexity level $T = 3$, entity crossover enabled, and exchange enabled. The target object is the ivory chess piece; the second object is the ebony cane. The planted static rule is `great-aunt`, with path (`sister`, `father`, `mother`).
Target timeline	The target-object timeline is `sale` $\rightarrow$ `loan` $\rightarrow$ `return` $\rightarrow$ `break` $\rightarrow$ `exchange` $\rightarrow$ `repair` $\rightarrow$ `break` $\rightarrow$ `gift` $\rightarrow$ `repair` $\rightarrow$ `sale`.
Rendered story	Karen Baylor Sawyer currently own the ivory chess piece. The ebony cane is owned by Esther Delvin Murdoch. Agnes Drummond Fielding is the sister of Charity Lark Amberley. Karen Baylor Sawyer sell Charity Lark Amberley the ivory chess piece. Charity Lark Amberley holds the relation of sister to Agnes Drummond Fielding. The ivory chess piece is lent to Dustin Keith Ravenscroft by Charity Lark Amberley. Charity Lark Amberley acts in the role of father to Jacqueline Coby Pettigrew. The ivory chess piece is returned to Charity Lark Amberley by Dustin Keith Ravenscroft. Jacqueline Coby Pettigrew holds the relation of son to Charity Lark Amberley. It is Esther Delvin Murdoch who break the ebony cane. Jacqueline Coby Pettigrew holds the relation of mother to Andrea Bryson McCaffrey. Charity Lark Amberley break the ivory chess piece. Andrea Bryson McCaffrey serves as Jacqueline Coby Pettigrew's daughter. Charity Lark Amberley exchange the ivory chess piece with Esther Delvin Murdoch for the ebony cane. Edward Delvin Clements acts in the role of sister to Cyril Amias Devonshire. Charity Lark Amberley repair the ebony cane. Cyril Amias Devonshire is the sister of Edward Delvin Clements. Charity Lark Amberley accidentally break the ebony cane. Cyril Amias Devonshire serves as Bella Aether Ainsworth's father. Esther Delvin Murdoch decide to repair the ivory chess piece. Bella Aether Ainsworth is the son of Cyril Amias Devonshire. It is Esther Delvin Murdoch who break the ivory chess piece. Bella Aether Ainsworth acts in the role of mother to Andrew Essex Northcott. Esther Delvin Murdoch give the ivory chess piece to Karen Baylor Sawyer. Andrew Essex Northcott holds the relation of daughter to Bella Aether Ainsworth. Karen Baylor Sawyer repair the ivory chess piece. Denise Hanley Glendale is the son of Dustin Keith Ravenscroft. Karen Baylor Sawyer decide to sell the ivory chess piece to Eric Aston Fairbanks. Dustin Keith Ravenscroft holds the relation of father to Denise Hanley Glendale. Andrea Bryson McCaffrey holds the relation of brother to Denise Hanley Glendale. Denise Hanley Glendale serves as Andrea Bryson McCaffrey's brother. Edward Delvin Clements acts in the role of daughter to Andrea Bryson McCaffrey. Andrea Bryson McCaffrey is the mother of Edward Delvin Clements.

Dynamic Reasoning Tasks

Task	Question	Solution	Answer
Deductive	After all events, who possess the ivory chess piece?	Initial state: Karen Baylor Sawyer own and possess the ivory chess piece (intact). Step 1: Karen Baylor Sawyer sells the ivory chess piece to Charity Lark Amberley. State: Charity Lark Amberley own and possess the ivory chess piece (intact). Step 2: Charity Lark Amberley lends the ivory chess piece to Dustin Keith Ravenscroft. State: Charity Lark Amberley own the ivory chess piece, Dustin Keith Ravenscroft possess the ivory chess piece (intact). Step 3: Dustin Keith Ravenscroft returns the ivory chess piece to Charity Lark Amberley. State: Charity Lark Amberley own and possess the ivory chess piece (intact). Step 4: Charity Lark Amberley breaks the ivory chess piece. State: Charity Lark Amberley own and possess the ivory chess piece (broken). Step 5: Charity Lark Amberley exchanges the ivory chess piece with Esther Delvin Murdoch for the ebony cane. State: Esther Delvin Murdoch own and possess the ivory chess piece (broken). Step 6: Esther Delvin Murdoch repairs the ivory chess piece. State: Esther Delvin Murdoch own and possess the ivory chess piece (intact). Step 7: Esther Delvin Murdoch breaks the ivory chess piece. State: Esther Delvin Murdoch own and possess the ivory chess piece (broken). Step 8: Esther Delvin Murdoch gives the ivory chess piece to Karen Baylor Sawyer. State: Karen Baylor Sawyer own and possess the ivory chess piece (broken). Step 9: Karen Baylor Sawyer repairs the ivory chess piece. State: Karen Baylor Sawyer own and possess the ivory chess piece (intact). Step 10: Karen Baylor Sawyer sells the ivory chess piece to Eric Aston Fairbanks. State: Eric Aston Fairbanks own and possess the ivory chess piece (intact). Therefore, the answer is Eric Aston Fairbanks.	Eric Aston Fairbanks
Abductive	Some information is missing from the story. Please infer the missing event or information involving the ivory chess piece.	Tracing backwards. Last step: Karen Baylor Sawyer sells the ivory chess piece to Eric Aston Fairbanks. Before: Karen Baylor Sawyer own and possess the ivory chess piece (intact). After: Eric Aston Fairbanks own and possess the ivory chess piece (intact). Previous step: Karen Baylor Sawyer repairs the ivory chess piece. Before: Karen Baylor Sawyer own and possess the ivory chess piece (broken). After: Karen Baylor Sawyer own and possess the ivory chess piece (intact). After matches before of the next step, no gap. Previous step: Esther Delvin Murdoch gives the ivory chess piece to Karen Baylor Sawyer. Before: Esther Delvin Murdoch own and possess the ivory chess piece (broken). After: Karen Baylor Sawyer own and possess the ivory chess piece (broken). After matches before of the next step, no gap. Previous step: Esther Delvin Murdoch breaks the ivory chess piece. Before: Esther Delvin Murdoch own and possess the ivory chess piece (intact). After: Esther Delvin Murdoch own and possess the ivory chess piece (broken). After matches before of the next step, no gap. Previous step: Esther Delvin Murdoch repairs the ivory chess piece. Before: Esther Delvin Murdoch own and possess the ivory chess piece (broken). After: Esther Delvin Murdoch own and possess the ivory chess piece (intact). After matches before of the next step, no gap. Previous step: Charity Lark Amberley breaks the ivory chess piece. Before: Charity Lark Amberley own and possess the ivory chess piece (intact). After: Charity Lark Amberley own and possess the ivory chess piece (broken). After does not match before of the next step, gap found. Before the missing step: Charity Lark Amberley own and possess the ivory chess piece (broken). After the missing step: Esther Delvin Murdoch own and possess the ivory chess piece (broken). Therefore, the missing information is: Charity Lark Amberley exchanges the ivory chess piece with Esther Delvin Murdoch for the ebony cane.	Charity Lark Amberley exchanges the ivory chess piece with Esther Delvin Murdoch for the ebony cane.

Static Reasoning Tasks

Task	Question	Solution	Answer
Inductive	Andrea Bryson McCaffrey serves as Edward Delvin Clements's mother. Agnes Drummond Fielding holds the great-aunt relation to Andrea Bryson McCaffrey. Dustin Keith Ravenscroft holds the father relation to Denise Hanley Glendale. Edward Delvin Clements acts as the great-aunt of Andrew Essex Northcott. Andrea Bryson McCaffrey holds the daughter relation to Jacqueline Coby Pettigrew. Bonnie Ansel Barncastle acts in the role of sister to Pauline Beckett Brookhaven, and Pauline Beckett Brookhaven serves as Phyllis Euan Ames's father, and Phyllis Euan Ames holds the relation of mother to Antonio Beck Harmon. What is the relationship between Bonnie Ansel Barncastle and Antonio Beck Harmon?	Example: Agnes Drummond Fielding is the sister of Charity Lark Amberley, and Charity Lark Amberley is the father of Jacqueline Coby Pettigrew, and Jacqueline Coby Pettigrew is the mother of Andrea Bryson McCaffrey, so Agnes Drummond Fielding is the great-aunt of Andrea Bryson McCaffrey. Example: Edward Delvin Clements is the sister of Cyril Amias Devonshire, and Cyril Amias Devonshire is the father of Bella Aether Ainsworth, and Bella Aether Ainsworth is the mother of Andrew Essex Northcott, so Edward Delvin Clements is the great-aunt of Andrew Essex Northcott. The pattern is sister then father then mother gives great-aunt. Applying to query: Bonnie Ansel Barncastle is the sister of Pauline Beckett Brookhaven, and Pauline Beckett Brookhaven is the father of Phyllis Euan Ames, and Phyllis Euan Ames is the mother of Antonio Beck Harmon. By the same pattern, Bonnie Ansel Barncastle is the great-aunt of Antonio Beck Harmon. Therefore, the answer is great-aunt.	great-aunt
Analogy	Agnes Drummond Fielding is to Andrea Bryson McCaffrey as Edward Delvin Clements is to ?.	From the story, Agnes Drummond Fielding is the sister of Charity Lark Amberley, and Charity Lark Amberley is the father of Jacqueline Coby Pettigrew, and Jacqueline Coby Pettigrew is the mother of Andrea Bryson McCaffrey. Through sister then father then mother, Agnes Drummond Fielding is the great-aunt of Andrea Bryson McCaffrey. Similarly, Edward Delvin Clements is the sister of Cyril Amias Devonshire, and Cyril Amias Devonshire is the father of Bella Aether Ainsworth, and Bella Aether Ainsworth is the mother of Andrew Essex Northcott. By the same pattern, Edward Delvin Clements is the great-aunt of Andrew Essex Northcott. Therefore, the answer is Andrew Essex Northcott.	Andrew Essex Northcott

Table 9: Worked Example. Deductive and inductive are generated task instances. Abductive is constructed by masking this operation: Charity Lark Amberley exchanges the ivory chess piece with Esther Delvin Murdoch for the ebony cane. Analogy reuses the planted `great-aunt` path.

Component	Value	Description
Architecture	Qwen 2.5 decoder	Causal decoder-only language model
Parameters	107M	Model scale used in all experiments
Hidden size	768	Transformer hidden dimension
Layers	12	Number of decoder blocks
Attention heads	12	Query attention heads
KV heads	2	Grouped-query key/value heads
FFN size	3072	Intermediate dimension in the feed-forward block
Vocabulary size	4096	KG-domain tokenizer vocabulary
Context length	2048	Maximum model context window
Weight tying	enabled	Input and output embeddings are tied
Precision	bfloat16	Training and rollout precision where supported

Table 10: Model Architecture and Tokenizer Settings

Setting	Value	Notes
Objective	causal LM	Full-parameter pre-training from scratch
Epochs	3	Final checkpoint used for RL initialization
Learning rate	2×10^{-4}	Peak learning rate
Scheduler	cosine with min LR	Learning-rate schedule
Warmup ratio	0.05	Fraction of steps used for LR warmup
Weight decay	0.1	AdamW regularization
Per-device batch	16	Training batch size per GPU
Gradient accumulation	2	Effective global batch uses 8 GPUs
Optimizer	AdamW	Betas $(0.9, 0.999)$, epsilon 10^{-8}
Seed	42	Pre-training seed

Table 11: Pretraining Hyperparameters

Setting	Value	Notes
Implementation	distributed GRPO	Post-training implementation
Initial checkpoint	final pre-trained checkpoint	Shared start point for all recipes
Rollouts per prompt	6	Group size for GRPO advantage estimation
Train batch size	1024	Prompt batch size
Prompt length	1024	Maximum prompt tokens
Response length	1024	Maximum generated tokens
Actor learning rate	10^{-6}	Actor optimizer LR
KL loss coefficient	0.001	Fixed KL regularization coefficient
GPUs	8	Single-node multi-GPU training
Reward model	none	Rule-based verifier only

Table 12: GRPO Hyperparameters

Task family	Parsed process	Process match $\mathrm{m_P}$	Answer match $\mathrm{m_A}$
Deductive	Initial object state and ordered post-operation states for the target object	$\mathrm{m_P} = 1$ only if the full owner, possessor, and integrity state sequence matches the gold trace with no missing or extra step	Normalized predicted answer equals the gold state value
Abductive	Backward visible steps, each step's before/after states, and the required initial state or missing-step gap	$\mathrm{m_P} = 1$ only if all visible states and the missing-condition states match the gold backward trace	Normalized predicted answer matches any valid description of the same missing event or initial fact
Inductive	Support-example relation path, inferred composite rule, and query application	$\mathrm{m_P} = 1$ only if the basic-relation path, inferred rule, and query-side rule application all match the gold structure	Normalized predicted rule name equals the gold composite rule
Analogy	Source-pair relation pattern and target-pair mapped answer	$\mathrm{m_P} = 1$ only if the source path/rule and the target mapping both match the gold relation pattern	Normalized predicted entity equals the gold mapped entity

Table 13: Parsing and Matching Protocol for Generated Solutions

Task	Generated solution	Parsed content from parser schema
Deductive	Initial state: Arthur Brolin Bellingham own and possess the copper coin (intact). Step 1: Arthur Brolin Bellingham gives the copper coin to Mason Kent Glendale. State: Mason Kent Glendale own and possess the copper coin (intact). Step 2: Mason Kent Glendale lends the copper coin to Helen Rufus Oaksworth. State: Mason Kent Glendale own the copper coin, Helen Rufus Oaksworth possess the copper coin (intact). Step 3: Helen Rufus Oaksworth returns the copper coin to Mason Kent Glendale. State: Mason Kent Glendale own and possess the copper coin (intact). Step 4: Mason Kent Glendale sells the copper coin to Amber Blair Houghton. State: Amber Blair Houghton own and possess the copper coin (intact). Therefore, the answer is Amber Blair Houghton.	`object_name`: the copper coin. `initial_state`: (Arthur Brolin Bellingham, Arthur Brolin Bellingham, intact). `operations`: 1. gives; before (Arthur, Arthur, intact), after (Mason, Mason, intact). 2. lends; before (Mason, Mason, intact), after (Mason, Helen, intact). 3. returns; before (Mason, Helen, intact), after (Mason, Mason, intact). 4. sells; before (Mason, Mason, intact), after (Amber, Amber, intact). `answer`: Amber Blair Houghton.
Abductive	Tracing backwards. Last step: Connor Arbor Wolcott repairs the leather bag. Before: Ariana Reef Nelson own the leather bag, Connor Arbor Wolcott possess the leather bag (broken). After: Ariana Reef Nelson own the leather bag, Connor Arbor Wolcott possess the leather bag (intact). Previous step: Ariana Reef Nelson lends the leather bag to Connor Arbor Wolcott. Before: Ariana Reef Nelson own and possess the leather bag (broken). After: Ariana Reef Nelson own the leather bag, Connor Arbor Wolcott possess the leather bag (broken). After matches before of the next step, no gap. Previous step: Connor Arbor Wolcott returns the leather bag to Ariana Reef Nelson. Before: Ariana Reef Nelson own the leather bag, Connor Arbor Wolcott possess the leather bag (broken). After: Ariana Reef Nelson own and possess the leather bag (broken). After matches before of the next step, no gap. Previous step: Ariana Reef Nelson lends the leather bag to Connor Arbor Wolcott. Before: Ariana Reef Nelson own and possess the leather bag (broken). After: Ariana Reef Nelson own the leather bag, Connor Arbor Wolcott possess the leather bag (broken). After matches before of the next step, no gap. Previous step: Ariana Reef Nelson breaks the leather bag. Before: Ariana Reef Nelson own and possess the leather bag (intact). After: Ariana Reef Nelson own and possess the leather bag (broken). After matches before of the next step, no gap. Required initial state: Ariana Reef Nelson own and possess the leather bag (intact). The story does not state this. Therefore, the missing information is: Ariana Reef Nelson own the leather bag.	`object_name`: the leather bag. `operations` are parsed in backward order: 1. repairs; before (Ariana, Connor, broken), after (Ariana, Connor, intact). 2. lends; before (Ariana, Ariana, broken), after (Ariana, Connor, broken). 3. returns; before (Ariana, Connor, broken), after (Ariana, Ariana, broken). 4. lends; before (Ariana, Ariana, broken), after (Ariana, Connor, broken). 5. breaks; before (Ariana, Ariana, intact), after (Ariana, Ariana, broken). `initial_state`: (Ariana Reef Nelson, Ariana Reef Nelson, intact). `missing_step_before`: null. `missing_step_after`: null. `answer`: Ariana Reef Nelson own the leather bag.

Table 14: Concrete Parsing Examples for Dynamic Reasoning Tasks. Generated solutions are copied verbatim from model outputs; parsed content is reconstructed with the same parser schema used during evaluation. Dataset identifiers are omitted. State tuples are $(owner, possessor, integrity)$, and both examples contain at least four target-object operations.

Task	Generated solution	Parsed content from parser schema
Inductive	Example: Hope Baden Paddington is the sister of Asher Fischer Bellamy, and Asher Fischer Bellamy is the mother of Anita Cole Clarkson, so Hope Baden Paddington is the aunt of Anita Cole Clarkson. Example: Oscar Rayner Frazier is the sister of Lena Ciel Stanford, and Lena Ciel Stanford is the mother of Liam Carlyle Merriweather, so Oscar Rayner Frazier is the aunt of Liam Carlyle Merriweather. The pattern is sister then mother gives aunt. Applying to query: Jean Digby Murdoch is the sister of Annika Ridge Overton, and Annika Ridge Overton is the mother of Oliver Ivar Grayson. By the same pattern, Jean Digby Murdoch is the aunt of Oliver Ivar Grayson. Therefore, the answer is aunt.	`pattern_path`: [sister, mother]. `pattern_rule`: aunt. `query_rule`: aunt. `answer`: aunt.
Analogy	From the story, Kimberly Chandos Fitzgerald is the brother of Anthony Bryden Emerson, and Anthony Bryden Emerson is the mother of Jared Remington Berwick. Through brother then mother, Kimberly Chandos Fitzgerald is the uncle of Jared Remington Berwick. Similarly, Emma Adair Cromwell is the brother of Dorothy Lester Blanchard, and Dorothy Lester Blanchard is the mother of Julia Thane Burleigh. By the same pattern, Emma Adair Cromwell is the uncle of Julia Thane Burleigh. Therefore, the answer is Julia Thane Burleigh.	`source_path`: [brother, mother]. `source_rule`: uncle. `target_rule`: uncle. `target_D`: Julia Thane Burleigh. `answer`: Julia Thane Burleigh.

Table 15: Concrete Parsing Examples for Static Reasoning Tasks. Generated solutions are copied verbatim from model outputs; parsed content is reconstructed with the same parser schema used during evaluation. Dataset identifiers are omitted.

Family	Recipe	Coverage region	Cells	Source
Baseline	Baseline	$D1$–$D4$, $T1$–$T2$	8	Main recipe sampler
Cmplx	Cmplx-Mid	$D1$–$D4$, $T3$–$T4$	8	Main recipe sampler
Cmplx	Cmplx-High	$D1$–$D4$, $T5$–$T6$	8	Main recipe sampler
Cmplx	Cmplx-Intersect	$D1$–$D4$, $T2$–$T3$	8	Main recipe sampler
Cmplx	Cmplx-Uniform	$D1$–$D4$, $T1$–$T6$	24	Complexity-uniform schedule
Depth	Depth-Mid	$D5$–$D7$, $T1$–$T2$	6	Main recipe sampler
Depth	Depth-High	$D8$–$D10$, $T1$–$T2$	6	Main recipe sampler
Depth	Depth-Intersect	$D4$–$D6$, $T1$–$T2$	6	Main recipe sampler
Depth	Depth-Uniform	$D1$–$D10$, $T1$–$T2$	20	Depth-uniform schedule
Diag	Diag-Mid	$D5$–$D7$, $T3$–$T4$	6	Diagonal middle block
Diag	Diag-High	$D8$–$D10$, $T5$–$T6$	6	Diagonal hard block
Mix	Deep-Mix	$D5$–$D10$, $T3$–$T6$	24	Main recipe sampler
Mix	Shallow-Mix	$D1$–$D4$, $T1$–$T6$; $D5$–$D10$, $T1$–$T2$	36	Main recipe sampler
Mix	Offbase-Mix	all cells except $D1$–$D4$, $T1$–$T2$	52	Main recipe sampler
Mix	Full-Coverage	$D1$–$D10$, $T1$–$T6$	60	Main recipe sampler

Table 16: RL Training Recipes over the Depth-by-Complexity Grid

Model	Release / Source	Scale	Notes
Kimi-K2.6 (Moonshot AI, 2026)	Apr. 2026, Moonshot / Hugging Face	MoE, about 1T total / 32B activated	Open-weight Kimi K2.6 model card.
GPT-5.4 (OpenAI, 2026a)	Mar. 2026, OpenAI	Undisclosed	Official OpenAI release.
GLM-5.1-FP8 (Z.ai, 2026)	Apr. 2026, Z.ai / Hugging Face	MoE, about 754B artifact scale, FP8	FP8 release of GLM-5.1; model family citation points to the GLM-5 technical report.
Offbase-mix	This work	107M Qwen2.5-style decoder	Internal RL-trained checkpoint.
Shallow-mix	This work	107M Qwen2.5-style decoder	Internal RL-trained checkpoint.
DeepSeek-R1 (DeepSeek-AI, 2025)	Jan. 2025, DeepSeek / arXiv	MoE, 671B total / 37B activated	Open reasoning model trained with large-scale RL.
Qwen3.6-35B-A3B (Qwen Team, 2026e)	Apr. 2026, Qwen / Hugging Face	MoE, 35B total / 3B activated	Open-weight Qwen3.6 model.
Mixtral-8x22B-Instruct-v0.1 (Mistral AI, 2024)	Apr. 2024, Mistral AI	MoE, 141B total / 39B active	Official instruct release of Mixtral 8x22B.
Kimi-K2-Thinking (Moonshot AI, 2025)	Nov. 2025, Moonshot / Hugging Face	MoE, 1T total / 32B activated	Thinking-agent version of Kimi K2 with long-context tool use.
Qwen3.5-397B-A17B-FP8 (Qwen Team, 2026c)	Feb. 2026, Qwen / Hugging Face	MoE, 397B total / 17B activated, FP8	FP8 quantized release; table reports architecture scale.
Cmplx-uniform	This work	107M Qwen2.5-style decoder	Internal RL-trained checkpoint.
Depth-uniform	This work	107M Qwen2.5-style decoder	Internal RL-trained checkpoint.
Qwen3.5-35B-A3B (Qwen Team, 2026b)	Feb. 2026, Qwen / Hugging Face	MoE, 35B total / 3B activated	Qwen3.5 open-weight model.
QwQ-32B-Preview (Qwen Team, 2024)	Nov. 2024, Qwen / Hugging Face	Dense, 32B	Preview reasoning model based on Qwen2.5-32B.
Qwen2.5-32B-Instruct (Qwen et al., 2025)	Sept. 2024, Qwen / Hugging Face	Dense, 32.8B	Instruction-tuned Qwen2.5 model.
Qwen3-8B (Qwen Team, 2025)	Apr. 2025, Qwen / Hugging Face	Dense, 8.2B	Qwen3 dense open-weight model.
DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025)	Jan. 2025, DeepSeek / Hugging Face	Dense, 32.8B	Distilled from DeepSeek-R1 using the Qwen2.5-32B base.
Qwen2.5-72B-Instruct (Qwen et al., 2025)	Sept. 2024, Qwen / Hugging Face	Dense, 72.7B	Instruction-tuned Qwen2.5 flagship model.
Qwen3.6-27B (Qwen Team, 2026d)	Apr. 2026, Qwen / Hugging Face	Dense, 27B	Dense Qwen3.6 open-weight model.
Qwen2.5-7B-Instruct (Qwen et al., 2025)	Sept. 2024, Qwen / Hugging Face	Dense, 7.6B	Instruction-tuned Qwen2.5 model.
Qwen3.5-122B-A10B-FP8 (Qwen Team, 2026a)	Feb. 2026, Qwen / Hugging Face	MoE, 122B total / 10B activated, FP8	FP8 quantized release; table reports architecture scale.
DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025)	Jan. 2025, DeepSeek / Hugging Face	Dense, 7.6B	Distilled from DeepSeek-R1 using the Qwen2.5-7B base.

Table 17: Metadata for the benchmark models in Table 1.