

Universiteit Leiden

Master Computer Science

Does Reinforcement Learning Teach Abstract Reasoning in Large Language Models?

Name:	Dawei Wang
Student ID:	s4176650
Date:	July 9, 2026
Specialisation:	Artificial Intelligence
1st supervisor:	Aske Plaat
2nd supervisor:	Michiel van der Meer

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Reinforcement learning with verifiable rewards (RLVR) raises large-language-model accuracy on reasoning benchmarks and is widely read as improving reasoning ability, yet which aspect of reasoning it improves remains unclear. We ask whether RLVR cultivates abstract reasoning: the ability to look past a problem's surface and reason over its relational structure. We design four process-level metrics grounded in Barsalou's cognitive decomposition of abstraction — type/role binding, structured relation construction, schematic sufficiency, and explicit schema extraction — each scored on a 0–3 ordinal scale by two correctness-blind LLM judges. We train and evaluate models from three families (Qwen3, Qwen2.5, Mistral) at scales from 4B to 24B parameters, using outcome RL on mathematics and code. All models are tested zero-shot on held-out analogy tasks disjoint from every model's training data.

We find that RL raises abstraction on letter-string and story analogies, consistently across three model families, two judges, and multiple training domains. The gain appears zero-shot on analogy tasks entirely absent from the RL training data, and takes a concrete form: RL-trained models more actively seek rules and relationships in their reasoning. An intervention experiment shows that merely adopting an abstract reasoning style does not improve accuracy; what matters is whether the model identifies the correct rule. RL amplifies exactly this behaviour: it reshapes chain-of-thought openings toward rule-finding declarations, and a causal test confirms that these openings shape subsequent reasoning. These results offer a rethinking of the recent view that RL merely concentrates probability mass without creating new capability. RL does concentrate the distribution, but it concentrates toward paths that reason more abstractly and seek rules more actively — and this quality shift is what drives the accuracy gain.

Contents

1 Introduction **1**

2 Background **3**
2.1 Abstraction . . . 3
2.2 Large language models . . . 4
2.3 Reinforcement learning, from MDPs to GRPO . . . 5
2.4 Chain-of-thought reasoning . . . 7
2.5 LLM-as-judge . . . 7

3 Related Work **7**

4 Method **8**
4.1 Abstraction metrics . . . 9
4.2 Evaluation procedure . . . 11
4.3 Sparse autoencoder and intervention analysis . . . 13

5 Experiments **14**
5.1 Tasks . . . 14
5.2 Models and training . . . 15
5.3 Does RL induce abstract reasoning? . . . 15
5.4 Abstraction and accuracy across analogy tasks . . . 18
5.5 Does abstraction cause accuracy? . . . 19
5.6 Abstraction and correctness share a coupled internal direction . . . 20
5.7 Rule-framing openings in chain-of-thought . . . 21
5.8 Abstraction and the pass@k pattern . . . 25
5.9 Scaling and RL as comparable routes to abstraction . . . 27
5.10 Supplementary analyses . . . 28

6 Discussion **29**
6.1 Finding 1: RL raises process-level abstraction . . . 29
6.2 Finding 2: abstraction and correctness share a coupled internal direction . . . 30
6.3 Finding 3: RL reshapes chain-of-thought text toward rule-framing . . . 30
6.4 Finding 4: RL changes how the model reasons, not just what it solves . . . 31
6.5 Synthesis . . . 32

7 Conclusions and Further Research **32**
7.1 Limitations . . . 33
7.2 Further Research . . . 33

References **38**

A Bare-prompt protocol controls **38**

B Scoring Rubrics **39**
B.1 Judge System Prompt . . . 39
B.2 Letter-String Analogy Rubrics . . . 39
B.2.1 Type/Role Binding . . . 40
B.2.2 Structured Relation Construction . . . 40

B.2.3 Schematic Sufficiency . . . 41
B.2.4 Explicit Schema Extraction . . . 41
B.3 Story Analogy Rubrics . . . 42
B.3.1 Type/Role Binding . . . 42
B.3.2 Structured Relation Construction . . . 43
B.3.3 Schematic Sufficiency . . . 44
B.3.4 Explicit Schema Extraction . . . 44
B.4 Digit-Matrix Analogy Rubrics . . . 45
B.4.1 Type/Role Binding . . . 45
B.4.2 Structured Relation Construction . . . 46
B.4.3 Schematic Sufficiency . . . 47
B.4.4 Explicit Schema Extraction . . . 47

C Case studies: base vs RL chain-of-thought traces **48**
C.1 Letter-string analogy . . . 48
C.2 Story analogy . . . 49
C.3 Digit-matrix analogy . . . 51

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) reliably raises large-language-model accuracy on mathematical and coding benchmarks. DeepSeekMath demonstrated that outcome reinforcement learning pushes open-weight models toward frontier-level mathematics [35], and DeepSeek-R1 showed that pure outcome RL yields emergent long chain-of-thought reasoning and self-verification without process supervision [8].

Whether these accuracy gains reflect genuine improvement in reasoning capability, or whether RL merely reshapes the output distribution, has attracted growing attention. Yue et al. examined this question through pass@k analysis [51]: as k increases, the accuracy advantage of RL-trained models over their base counterparts shrinks and eventually reverses, with no new problems becoming solvable at large k. Their conclusion is that RL concentrates probability mass on correct paths the base model already contains, rather than expanding the model's capability frontier. Shao et al. reinforced this view by showing that even random rewards produce comparable accuracy gains on certain model families [34]. Recent studies have begun to look beyond accuracy at the reasoning traces themselves, examining branching behaviour [50] and cross-task skill transfer [16], but they evaluate structural features of traces or downstream task accuracy, not the quality of the reasoning itself. Whether RL changes the reasoning process itself, rather than just its accuracy, remains an open question. Answering it requires moving beyond outcome accuracy toward metrics that characterise the reasoning process within a chain of thought — how the model reasons, not just whether it reaches the correct answer.

To construct such a metric, we turn to cognitive science, where abstraction — the ability to reason over a problem's relational structure rather than its concrete surface — has long served as a core diagnostic of reasoning quality [2, 12]. Recent work has begun to incorporate abstraction explicitly into reinforcement learning: RLAD rewards procedural abstraction generation [31], AR^2 uses adversarial problem rewriting to force structural generalisation [49], and AbstRaL replaces concrete values with symbolic variables and trains with an abstraction-alignment reward [11]. All three inject an abstraction signal into RL and report improved out-of-distribution generalisation. These results raise a natural question: does standard outcome RL, which provides no abstraction signal, also induce more abstract reasoning on its own? They also suggest that abstraction — a capacity central to human intelligence [2] — can serve as a process-level metric for evaluating the quality of reasoning within a chain of thought.

We answer this by training models with outcome RL on mathematics and code — tasks whose reward carries no abstraction signal — and evaluating them zero-shot on held-out analogy benchmarks from a different domain. We choose analogy tasks because cognitive science identifies analogy as a direct test of abstraction: solving an analogy requires mapping relational structure rather than matching surface features [12, 15]. The separation between training and evaluation domains makes this a clean test: because no analogy item appears in any model's training data, any change in how abstractly the model reasons must arise from the RL process itself. The analogy suite also includes conditions where the alphabet is permuted, removing any memorised letter orderings. Models that rely on surface patterns fail under permutation while humans do not [22, 14], making permuted items a test of whether the model reasons over structure or simply recalls familiar orderings. Analogy tasks offer a further advantage: because each item requires the solver to identify relationships among elements, the chain of thought exposes whether the model seeks a governing rule or processes the instance position by position. The trace itself becomes evidence of how abstractly the model reasons.

A growing body of work tests whether language models can solve abstraction and analogy tasks [43, 22, 28], but these evaluations score final-answer accuracy: they reveal whether the

model solved the problem, not whether it reasoned abstractly along the way. A model might arrive at the correct answer through pattern matching without engaging in genuine abstraction, or reason abstractly yet fail to execute the final step. How the model reasons — whether it identifies structure, formulates rules, and generalises across instances — is precisely what outcome-level benchmarks leave unobserved. No existing metric measures abstraction in the reasoning process itself.

To measure whether RL changes how abstractly models reason in their chains of thought, we need metrics that operate on the reasoning process rather than the final answer. Designing such metrics raises a definitional challenge: what counts as "abstract reasoning" in a chain of thought? The deliberate-abstraction approaches discussed above — RLAD, AR^2, and AbstRaL — each operationalise abstraction differently [31, 49, 11], and within cognitive science the boundaries remain debated [2, 12]. Individual judgments of whether a trace "looks abstract" are subjective and difficult to reproduce. We address this by decomposing abstraction into independently identifiable components grounded in cognitive science, each defined by observable behaviours that a scorer can verify in the text of a reasoning trace. Rather than asking whether a trace is globally "more abstract," we ask whether specific abstraction behaviours are present, making abstraction in the reasoning process quantifiable across models and conditions.

With this measurement framework in hand, we pursue three research questions:

1. **Does outcome RL induce more abstract reasoning in chain-of-thought traces, and how does this manifest?** If RL changes the reasoning process, there should be a measurable shift in how abstractly models reason on held-out tasks, and the shift should be traceable to specific changes in the chain of thought.

2. **What is the relationship between abstraction in the reasoning process and task accuracy?** If RL raises abstraction, the two gains may be causally linked, jointly produced by a common internal change, or entirely independent.

3. **How does the abstraction finding relate to the distributional-sharpening account of RL?** Yue et al.'s pass@k analysis concludes that RL concentrates probability mass without expanding capability [51]. If RL simultaneously raises how abstractly the model reasons, this characterisation is incomplete: sharpening may involve a qualitative change in the reasoning paths that RL favours, not merely a quantitative reweighting.

We design four process-level metrics grounded in Barsalou's cognitive decomposition [2] to score abstraction in chain-of-thought traces [44, 18]: type/role binding, structured relation construction, schematic sufficiency, and explicit schema extraction, each scored on a 0–3 ordinal scale by two correctness-blind LLM judges (Section 4). We train Qwen3 [47] models at two sizes (4B, 8B) with GRPO [35] on the DeepScaleR recipe (competition mathematics) and add two models (4B, 8B) trained on the DeepCoder recipe (code) as cross-domain controls; we also evaluate published SimpleRL-Zoo checkpoints [52] on Qwen2.5 (7B, 14B), and Mistral-Small-24B. We test every model zero-shot on held-out task families from the Lewis–Mitchell analogy suite [22, 21]: letter-string, story, and digit-matrix analogies. No analogy item appears in any model's RL training data; every training set consists entirely of mathematics or code problems, graded by rule-based answer checking. The complete separation between training and evaluation domains makes this a clean test of emergence: because no model encounters an analogy task during RL, any change in how abstractly it reasons on analogies cannot arise from fitting the training data. A gain in abstraction on held-out tasks would show that RL reshapes the reasoning process itself, producing a capability that transfers spontaneously beyond the

training domain. To test whether any observed abstraction gain causally drives accuracy, we conduct a five-arm intervention experiment on the base models, manipulating the reasoning style through prompt priming while holding all other factors fixed (Section 4.3).

Four findings emerge (discussed in full in Section 6). First, RL raises abstraction on letter-string and story analogies across model families and training domains; this gain manifests as a shift toward proactive rule identification in the chain-of-thought opening. Digit-matrix analogies stay flat, and our mechanism analysis shows this is consistent with the same account: the base model already opens in rule-framing mode on this task, leaving no margin for RL to shift. Second, a five-arm intervention shows that abstract reasoning form alone does not drive accuracy; rule identification is the operative variable, and sparse-autoencoder analysis reveals that abstraction and correctness share a coupled internal direction rather than being causally sequential. Third, RL reshapes chain-of-thought text toward rule-framing openings that declare intent to identify the governing pattern before working through the instance; a causal test confirms that these openings shape subsequent reasoning and accuracy. Fourth, Yue et al. [51] argue that RL merely concentrates probability mass on correct paths without creating new reasoning capability; we replicate this distributional convergence on our benchmark, but show that the paths RL favours are qualitatively different: more abstract and more rule-deriving.

Contributions.

1. A process-sensitive method for measuring abstraction in chain-of-thought reasoning: four cognitively grounded metrics scored by blinded LLM judges.

2. An empirical finding that outcome RL emergently induces more abstract reasoning on held-out tasks, consistently across model sizes, families, and training domains.

3. Evidence that RL changes how the model reasons, not just which items it solves: intervention, sparse-autoencoder, causal, and outcome-stratified analyses reveal a rule-seeking disposition that outcome-level evaluations miss [51].

Section 2 reviews abstraction in cognitive science, RLVR and GRPO, chain-of-thought, and LLM-as-judge methodology; Section 3 positions the work among related empirical studies; Section 4 presents the method; Section 5 describes the experimental setup and reports results; Section 6 discusses the findings; and Section 7 concludes.

2 Background

This section reviews abstraction (Section 2.1), large language models (Section 2.2), reinforcement learning from MDPs to the GRPO algorithm (Section 2.3), chain-of-thought reasoning (Section 2.4), and LLM-as-judge methodology (Section 2.5).

2.1 Abstraction

Abstraction, as this thesis uses the term, is the capacity to reason over a problem's structural properties rather than its concrete surface: to identify what roles the elements play, to construct the relations among them, and to extract a reusable pattern that applies beyond the particular instance [2, 12]. A solver who recognises "advance the last element to its successor" has abstracted; one who notes "c follows b" has not, because the latter representation is bound to specific letters and breaks the moment the surface changes.

Operationalising abstraction. Our measurement draws primarily on Barsalou [2], who analyses abstraction as a construct with at least six distinct senses and focuses on "summary representation." He identifies three properties of summary representations that recur as observable behaviours: type–token interpretation (a concrete token is read as an instance of a higher-level type or role), structured representation (the roles stand in specified relations, not a bag of features), and dynamic realisation (the representation is constructed on the fly for the task rather than retrieved as a fixed template). He separately characterises schematic representation: a sparse representation that preserves a problem's critical structure while discarding irrelevant detail. Several complementary accounts converge on the same core intuition. Gentner's structure-mapping theory [12] grounds the use of analogy tasks as tests of abstraction, since solving an analogy requires mapping relational structure rather than matching surfaces. Chater and Vitányi [4] cast abstraction as preferring the simplest pattern consistent with the data, and a separate line locates it in operating over re-bindable variables [26] and running rules rather than retrieving memorised mappings [19]. No single definition unifies these accounts. We therefore take a behavioural approach: we identify the convergent properties that an abstract reasoner exhibits across these frameworks, and score their presence in chain-of-thought traces. The four metrics derived from this synthesis are presented in Section 4.1.

2.2 Large language models

A large language model (LLM) is a neural network, built on the Transformer architecture [41], that generates text autoregressively. Text is segmented into discrete tokens drawn from a fixed vocabulary, and the model produces an output sequence $o = (o_1, \ldots, o_T)$ one token at a time, each conditioned on the prompt q and all tokens emitted so far. Writing the model's distribution as π_θ, where θ denotes its parameters, the likelihood of a complete sequence factorises as

$$\pi_\theta(o \mid q) \;=\; \prod_{t=1}^{T} \pi_\theta(o_t \mid q, o_{<t}).$$

The Transformer's self-attention layers let every position attend directly to every earlier position, modelling long-range dependencies without the sequential bottleneck of recurrence and parallelising computation across the sequence during training. Training optimises θ by maximum likelihood on large text corpora: the model predicts each next token from its left context, and the same factorisation that defines generation also defines the objective. Next-token prediction is the only learning signal during pretraining; there is no task label beyond the text itself. Yet at scale, this signal is enough to induce broad linguistic and factual competence. Scale (the number of parameters and the volume of pretraining tokens) is the dominant lever on capability. As both grow, models trained on the single next-token objective acquire in-context or few-shot ability, performing new tasks from instructions or a handful of examples placed in the prompt, with no gradient update [3].

A pretrained model predicts plausible text continuations but does not, on its own, reliably follow instructions or answer in the form a user intends. Post-training closes this gap in two stages. Supervised fine-tuning continues the same maximum-likelihood objective on curated prompt–response demonstrations, teaching the model the format and behaviour of an assistant. Reinforcement learning from human feedback then optimises the model against a reward derived from human preference comparisons, which sharpens instruction-following and aligns outputs with what raters judge helpful and harmless [30]. This second stage treats the model as a policy whose generations are reinforced or suppressed by a reward rather than imitated from a fixed target. The reinforcement-learning formulation is developed in Section 2.3.

2.3 Reinforcement learning, from MDPs to GRPO

MDPs and the RL objective. Reinforcement learning formalises sequential decision-making as a Markov decision process, the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$: a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition kernel $P(s' \mid s, a)$ giving the probability of moving to state s' after taking action a in state s, a reward function $r(s, a)$, and a discount factor $\gamma \in [0, 1]$ [38]. A policy $\pi(a \mid s)$ maps states to distributions over actions; rolling it out produces a trajectory whose discounted return from time t is $G_t = \sum_{k \geq 0} \gamma^k r_{t+k}$. Two quantities summarise a policy: the value $V^\pi(s) = \mathbb{E}_\pi[\, G_t \mid s_t = s \,]$, the expected return from a state, and the action-value $Q^\pi(s, a) = \mathbb{E}_\pi[\, G_t \mid s_t = s, a_t = a \,]$, the expected return from taking a and following π thereafter. Their difference is the advantage $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, which measures how much better an action is than the policy's average behaviour in that state. For a parameterised policy π_θ, the learning objective is to maximise expected discounted return,

$$\max_\theta \; \mathbb{E}_{\pi_\theta}\left[\sum_{t \geq 0} \gamma^t r_t\right].$$

Autoregressive text generation is an instance of this template. The prompt q is the initial state, an action is the emission of the next token, the state at each step is the prompt together with the tokens produced so far $(q, o_{<t})$, the transition is the deterministic concatenation of the chosen token, and an episode is one complete output sequence $o = (o_1, \ldots, o_T)$. The language model is the policy $\pi_\theta(o_t \mid q, o_{<t})$, and a reward is assigned to the finished sequence. Optimising the return then means adjusting θ so that completions earning high reward become more probable.

Policy-gradient methods. Policy-gradient methods optimise θ by ascending the gradient of expected return directly. The REINFORCE estimator [45] obtains an unbiased gradient from sampled trajectories, $\nabla_\theta \, \mathbb{E}_{\pi_\theta}[G] = \mathbb{E}_{\pi_\theta}\big[\sum_t G_t \, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\big]$: each action's log-probability is pushed up in proportion to the return that followed it. The policy-gradient theorem [38] generalises this result, expressing the gradient as an expectation over the policy's own action-values and so making it estimable from on-policy samples without differentiating the environment dynamics. Using the raw return as the multiplier yields a high-variance estimator, because returns vary widely across trajectories regardless of which actions were responsible. Subtracting a state-dependent baseline $b(s)$ leaves the gradient unbiased while reducing its variance; the value function $V^\pi(s)$ is the canonical choice, which replaces the return by the advantage $A^\pi(s, a)$ as the weighting signal. Estimating that advantage accurately, trading off bias against variance, is itself a design problem, addressed by generalised advantage estimation [32]. The recurring pattern is therefore to raise the probability of actions whose advantage is positive and lower it for those whose advantage is negative.

Trust regions and PPO. Plain policy-gradient updates are unstable: a large step can move the policy into a region where the on-policy samples no longer inform the gradient, collapsing performance. Trust-region methods constrain each update to stay close to the policy that generated the data. Proximal Policy Optimisation (PPO) [33] enforces this cheaply through a clipped surrogate objective. Writing the probability ratio between the updated and sampling policies as $\rho = \pi_\theta(a \mid s) / \pi_{\theta_{\text{old}}}(a \mid s)$, PPO maximises

$$\mathbb{E}\Big[\min\Big(\rho\, A,\; \text{clip}(\rho,\, 1 - \varepsilon,\, 1 + \varepsilon)\, A\Big)\Big],$$

where A is the estimated advantage and ε the clip width. The clip removes the incentive to move ρ beyond $[1 - \varepsilon, 1 + \varepsilon]$: once the ratio leaves the trust region the objective flattens, so

an advantageous action is promoted only up to a bounded factor. A penalty on the Kullback–Leibler divergence from a reference policy is often added to bound the drift further. PPO is an actor–critic method: the advantage A is supplied by a learned value network (the critic) trained alongside the policy, which carries a second model of comparable size.

Reinforcement learning with verifiable rewards (RLVR). When the reward is supplied by an automatic, rule-based verifier, the setup is reinforcement learning with verifiable rewards (RLVR). Tasks with a deterministic correct answer, such as competition mathematics or code generation, are natural fits: the verifier simply checks whether the model's final answer matches the gold solution. Generation is cast as the terminal-reward MDP above: the prompt is the initial state, tokens are the actions, the language model is the policy π_θ, and the verifier scores the completed sequence, so reward arrives only at the end of the episode. Because the verifier is cheap and objective, RLVR scales to large amounts of self-generated data and is central to recent reasoning systems [8, 35]. A KL-to-reference term regularises the update, keeping the optimised policy close to the pretrained model π_{ref} and preventing reward maximisation from degrading fluency. What the verifier scores defines the supervision. An outcome reward model (ORM) rewards only final-answer correctness, giving one scalar per episode, whereas a process reward model (PRM) scores intermediate steps and so supplies denser, step-level credit [23, 42].

GRPO. Group Relative Policy Optimisation (GRPO), introduced with DeepSeekMath [35], removes the critic. A learned value network is the costly component of PPO, and with a terminal scalar reward it must estimate the return of a full generation from intermediate token states; GRPO replaces it with a baseline computed from a group of sampled completions. For each prompt q it draws a group of G completions $\{o_1, \ldots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, scores each with the reward r_i, and uses the group's own statistics as the baseline: the advantage of completion i is the standardised reward

$$A_i \;=\; \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}.$$

A completion is reinforced to the extent that it beats the other completions of the same prompt, so the group average serves as a per-prompt baseline in place of V^π. The policy is then updated with a clipped PPO-style surrogate and a KL penalty toward the reference policy π_{ref}:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\Big(\rho_i A_i,\ \text{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)\, A_i\Big) - \beta\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)\right], \qquad \rho_i = \frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)}.$$

Here ρ_i is the sequence-level probability ratio for completion i, ε the clip width, and β the KL coefficient; this sequence-level form abstracts the per-token objective that is actually optimised, in which the ratio, clip, and KL penalty apply at each token and the per-completion sum is length-normalised. The objective retains PPO's clipped trust region while drawing its advantage entirely from the group, so the only models in play are the policy and a frozen reference. This work uses the outcome (ORM) reward, which scores final-answer correctness alone. Although ORM provides only a terminal signal, the group-relative baseline performs implicit Monte-Carlo credit assignment: standardising within a group distributes the outcome signal across the tokens of each completion, and this has been read as supplying process supervision in effect [37].

2.4 Chain-of-thought reasoning

Wei et al. [44] show that prompting a model with a few worked examples, each pairing a problem with its step-by-step solution, sharply raises accuracy on arithmetic, commonsense, and symbolic tasks. The gains are largest when the solution decomposes into several dependent sub-steps, because each generated step enters the context window and conditions the tokens that follow: a computation too complex for a single forward pass unfolds across the generated sequence. Kojima et al. [18] find that the zero-shot instruction "Let's think step by step" elicits the same staged reasoning without exemplars, establishing chain-of-thought (CoT) as a general model capability. Reinforcement learning from verifiable rewards amplifies the behaviour further. The reward signal depends only on the final answer, yet training drives models toward longer reasoning chains, making extended intermediate reasoning the default mode rather than a prompted add-on.

CoT traces do not necessarily record the computation that produced the answer. Turpin et al. [40] expose this gap with biased prompts: when a feature in the prompt steers the model toward a particular answer, the trace produces a fluent rationale for that answer without mentioning the steering feature. Lanham et al. [20] probe faithfulness by paraphrasing, truncating, or corrupting the trace and checking whether the final answer changes; often it does not. Stronger reasoning models exhibit the same pattern [5, 1]. Any property measured from a trace therefore characterises the reasoning a model externalises, not the hidden process behind its answer, and demonstrating that the externalised reasoning matters requires a separate intervention.

2.5 LLM-as-judge

Scoring tens of thousands of free-text traces against a multi-dimensional rubric by hand is infeasible, so a strong language model serves as the scorer. The judge receives a scoring instruction, the rubric, and the text to be assessed, and returns a verdict parsed into a score [54, 24, 17]. Judge verdicts agree with human preferences at rates approaching inter-annotator agreement [54], which licenses substituting the judge for a panel of human raters. Judging reasoning traces is harder than judging open-ended responses, because the assessment turns on intermediate steps rather than a final answer [53, 46]; competence on general preference data does not transfer automatically to reasoning items [39]. LLM judges also exhibit systematic biases: position bias (preference for a particular slot order), verbosity bias (preference for longer outputs), and self-preference (favouring the judge's own model family) [13, 36]. Standard controls counter these biases by randomising presentation order, testing against length-matched inputs, drawing judges from a different model family than the subject, and blinding the judge to correctness and model identity.

3 Related Work

This section positions the present study against four lines of work: evaluating abstraction in language models, reinforcement learning for reasoning, LLM-as-judge methodology, and the analysis of chain-of-thought.

Evaluating abstraction and analogy in LLMs. Webb et al. reported that GPT-3 matches or exceeds undergraduates on analogy tasks, interpreting this as evidence of an emergent analogy mechanism [43]. Subsequent work showed that this performance relies on surface cues: letter-string perturbations that humans solve easily cause GPT-3 to fail [14], and posing

analogies over permuted alphabets at graded difficulty reveals that human accuracy holds near 75% while every GPT model degrades sharply [22, 21]. Their counterfactual benchmark establishes the graded surface-robustness axis we adopt. The Abstraction and Reasoning Corpus (ARC) [6] is the field's most prominent benchmark for abstract reasoning, requiring solvers to induce transformation rules from a few input–output grid examples. ConceptARC decomposes ARC into 16 core-knowledge concept groups and reports that GPT-4 scores 0.33 against a human 0.91 [28, 27]. Work that varies surface and relational similarity independently shows LLMs perform partial structure-mapping but are more susceptible than humans to surface distractors [29]. All of this work evaluates frozen pretrained models and scores abstraction from final answers; we measure abstraction in the reasoning trace of RL-trained models.

Reinforcement learning for LLM reasoning. A key design choice in our study is that the RL reward contains no abstraction signal: models are trained with outcome GRPO on mathematics or code, and any change in abstraction emerges spontaneously from correctness-only training. This contrasts with process-supervision approaches, which inject step-level signals [23, 42, 7]. Process rewards can inflate trace length without raising accuracy [48], and existing process reward models score near chance on hard steps [53]. Sullivan et al. show that outcome GRPO already performs implicit per-step credit assignment through shared-prefix rollouts [37].

RLAD [31], AR^2 [49], and AbstRaL [11] take a different approach: they make abstraction the optimisation objective, whether through rewarded abstraction generation, adversarial problem rewriting, or symbolic-variable substitution with alignment rewards. Our study asks the complementary question: does abstraction emerge from outcome RL alone, without any abstraction signal in the reward?

LLM-as-judge. LLM judges are now widely adopted as scalable substitutes for human annotation. MT-Bench and Chatbot Arena [54] established the protocol at scale for open-ended response quality; G-Eval [24] and Prometheus 2 [17] extended it to fine-grained rubric scoring; and ReasonEval [46] demonstrated that a step-level judge can decouple answer accuracy from reasoning quality, scoring each step for validity independently of the final answer. This last point is the closest precedent for our design: we similarly need a judge that scores the reasoning process (abstraction) independently of the outcome (correctness). Our judge applies rubric-mode scoring with correctness-blind prompts and standard bias controls (position randomisation, cross-family judging, length checks) [36, 13].

Across the work reviewed above, we identify a gap. Existing studies either evaluate abstraction on frozen pretrained models [43, 22, 28], deliberately train models to abstract [31, 49, 11], judge a different property of the reasoning trace [9], or measure transfer to non-abstract skills [16]. None asks whether vanilla outcome RL, trained without any abstraction signal, changes how abstractly the model reasons. This thesis occupies that gap: we measure abstraction directly in chain-of-thought traces, ground the measurement in cognitive science, and test it across model families, training domains, and evaluation tasks.

4 Method

We pursue two complementary studies that share a common measurement instrument. The first is an abstraction evaluation: we fine-tune language models with GRPO on mathematics and code and evaluate them zero-shot on analogy benchmarks disjoint from the RL training

data, measuring both outcome accuracy and CoT abstraction from the same inference traces. Because no analogy item appears during RL training, every analogy result is transfer rather than training, and any base-to-RL change is evidence about a general ability rather than memorised task-specific behaviour. The second is an intervention experiment: we manipulate the abstraction exhibited in the chain-of-thought through prompt priming on the base models, testing whether the abstraction gain observed under RL causally drives accuracy or merely accompanies it. Section 4.1 defines the four abstraction metrics; Section 4.2 describes the evaluation procedure; Section 4.3 describes the SAE and intervention analysis; and Section 5 presents the experimental setup.

4.1 Abstraction metrics

We operationalise abstraction as four indicators scored from a reasoning trace. Each indicator is grounded in the cognitive-science framework of Section 2.1, scored independently on a 0–3 ordinal scale, and reported as a four-dimensional profile rather than collapsed into a scalar. Scores are assigned to the reasoning text alone; the judge never sees the final answer or knows whether it is correct. Rubric anchors at each ordinal level are derived from an LLM ensemble and reviewed by the author.

Metric 1: Type/role binding. Does the trace map concrete surface tokens to task-relevant abstract roles? This indicator captures Barsalou's type–token interpretation and Marcus's variable binding [2, 26]. In a letter-string analogy, a high-scoring trace identifies that the last element is the locus of change and treats the specific letters as fillers of positional roles, rather than reasoning only about the particular symbols. The binding must be used downstream; relabelling tokens without exploiting the roles does not count.
0 Only concrete detail repeated.
1 Entities or quantities identified without clear abstract roles.
2 Most tokens mapped to roles but one or more bindings missing or vague.
3 Concrete tokens mapped to task-relevant abstract roles and used later in reasoning.

Metric 2: Structured relation construction. Does the trace state the relation among the roles (a rule, formula, mapping, or dependency) rather than merely listing them? This indicator captures Barsalou's structured representation and Gentner's systematicity [2, 12]. "The last letter advances by one in the alphabet" is a relation; "the letters are i, j, k" is not. Verbal, symbolic, and worked forms are all accepted; what is scored is whether a relation is explicitly constructed.
0 No relation.
1 A relation implied but not clearly expressed.
2 A relation expressed but incomplete, overly concrete, or partly incorrect.
3 A complete, task-appropriate abstract relation explicitly constructed.

Metric 3: Schematic sufficiency. Does the trace form a sparse but sufficient representation, discarding irrelevant surface detail while preserving everything required to solve the problem? This indicator captures Barsalou's schematic representation, with Chater and Vitányi as soft support [2, 4]. It is not measured by trace length: a terse "it is just arithmetic" is insufficient because it drops needed structure, and a verbatim restatement of the prompt is over-concrete. The high score goes to the trace that keeps exactly the task-relevant structure.
0 Overly concrete or omits necessary information.

Task prompt

`[a b c] changes to [a b d]; [i j k] changes to ?`

High-abstraction trace (RL model, scores: Role=3, Relation=3, Sufficiency=3, Schema=3)

The **transformed slot** is the last position; the **operand** is its letter, fed to a successor operation. This is the successor-on-last-element schema: replace the final letter with its alphabetic successor, leaving the rest fixed. Apply to [i j k]: last slot is k, successor is l → **[i j l]**.

Low-abstraction trace (base model, scores: Role=0, Relation=0, Sufficiency=0, Schema=0)

The first set is [a b c] and the second is [a b d]. The answer for [i j k] is [i j l].

Figure 1: Contrasting chain-of-thought traces on a letter-string analogy item. The high-abstraction trace (green) binds abstract roles, states the relation as a general rule, drops irrelevant detail, and names a reusable schema. The low-abstraction trace (red) produces the same correct answer but through surface-level pattern matching with no visible abstraction.

1 Removes some detail but loses part of the necessary structure.
2 Preserves the necessary structure but retains unnecessary surface detail.
3 A sparse, sufficient, task-relevant representation.

Metric 4: Explicit schema extraction. Does the trace explicitly name the problem as an instance of a general, reusable schema or rule type? This indicator captures Barsalou's summary representation [2]. It is distinct from Metric 2: Metric 2 asks whether the specific relation is stated; Metric 4 asks whether the trace lifts that relation to a named general pattern ("this is a successor-of-last-element transformation"). A trace can construct the correct relation for the instance (high Metric 2) yet never generalise it to a schema (low Metric 4).
0 No general schema.
1 A vague problem type ("this is a math problem").
2 A useful schema partly stated but tied to surface detail, or a clearly stated rule that does not fit the task.
3 A clear, reusable schema, rule, or problem type explicitly extracted, capturing the task's actual structure.

A worked example. Consider the letter-string item "`abc` changes to `abd`; what does `ijk` change to?" A trace scoring high on all four metrics binds the last position as the locus of change (Metric 1), states the rule "advance the final element to its alphabetic successor" (Metric 2), notes that the identities of the non-final letters are irrelevant and need only be copied (Metric 3), and names the transformation as a general successor-on-last-element schema (Metric 4). The counterfactual (permuted-alphabet) variant makes the metrics bite: when the problem supplies a non-standard ordering, "successor" must be computed in that ordering, and a trace that abstracts the relation correctly applies it to the given ordering, whereas one that has merely memorised the ordinary alphabet silently falls back to the standard `a`–`z` successor. The four metrics make that difference visible in the reasoning, a distinction final-answer accuracy alone cannot capture.

The four metrics operationalise constructs from Barsalou's framework, with Gentner warranting Metric 2 and Chater–Vitányi warranting Metric 3.

4.2 Evaluation procedure

The evaluation has two stages: generating chain-of-thought traces from each model, and scoring those traces for abstraction.

Trace generation. Both base and RL conditions are evaluated with identical prompts. Each model receives a Kojima-style zero-shot chain-of-thought prompt ("think step by step") [18], ending in a parseable "`Answer:`" line. This yields both an automatically gradable answer and a free-form reasoning trace. Because the prompt imposes no structural scaffold, any abstraction in the trace reflects the model's spontaneous reasoning rather than prompt-induced structure. All evaluations use greedy decoding (T=0) with fixed, recorded seeds, and every generation is saved as a structured record containing the prompt, the full chain-of-thought, the final answer, the gold answer, and correctness. These records form the shared corpus for both the outcome analysis and the process-level scoring.

Abstraction scoring. The process-level analysis scores the logged traces with the four metrics defined in Section 4.1 and tracks how the metric profile evolves across the RL trajectory. Figure 2 summarises the pipeline.

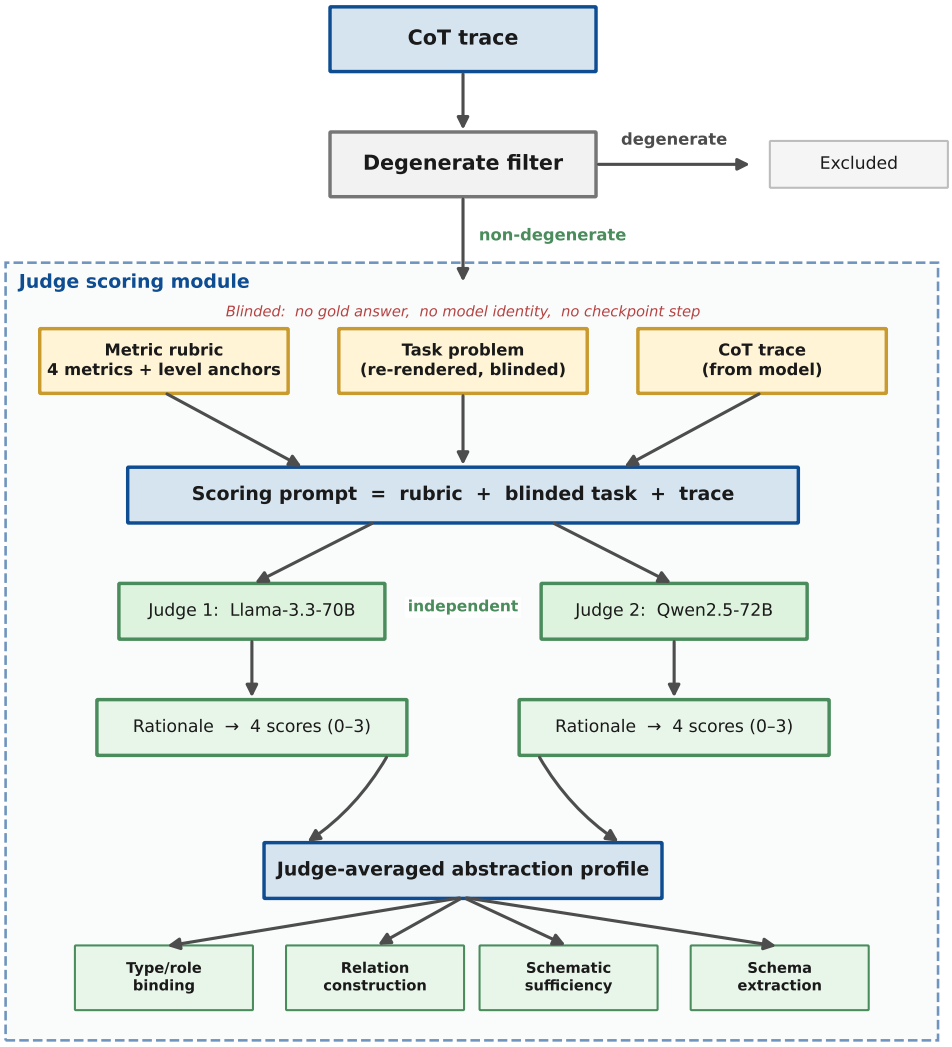


Figure 2: Scoring pipeline. Each model trace passes through a degenerate filter; non-degenerate traces are scored independently by two blinded judges on four abstraction metrics (0–3 ordinal scale). Scores are averaged across judges.

Degenerate-trace filter. Before judge scoring, a content-based filter identifies degenerate traces: those containing no genuine reasoning beyond echoed prompt boilerplate and a final-answer line. The detector strips boilerplate and the answer line, then tests whether reasoning content remains. Degenerate traces receive a score of 0 on all metrics and bypass the judge. The degenerate fraction varies sharply by task and prompt (about 2% on letter-string analogy, substantially higher on some downstream tasks) and is reported per condition, since RL can change not only reasoning quality but reasoning propensity.

Judge. For each non-degenerate trace, the judge produces a rationale followed by an ordinal score on each applicable metric. The judge receives a scoring prompt that contains the metric rubrics with level-anchored examples, the task problem (re-rendered without the gold answer), and the model's chain-of-thought trace; the full prompt template and rubrics are given in

Appendix B.1 and B. The judge is blinded to answer correctness, model identity, checkpoint step, and prompt variant. The scoring uses two independent 70B-class open-weight judges, Llama-3.3-70B-AWQ and Qwen2.5-72B-AWQ, which run the same prompt and rubrics independently, following the reliability requirements of Section 2.5. Every judgment is cached under a content-addressed key that includes the judge model and prompt version, so a rubric change invalidates the cache rather than mixing versions. Human–judge agreement is reported in Section 5.10.

4.3 Sparse autoencoder and intervention analysis

The abstraction metrics quantify the change in reasoning traces; to understand what drives this change internally, we apply sparse autoencoder (SAE) feature analysis to the model's internal activations. This analysis characterises both how RL reshapes chain-of-thought style (the shift toward rule-framing openings) and whether abstraction and correctness share a common internal basis.

Sparse autoencoders. A Transformer's hidden states are high-dimensional and distributed: no single neuron corresponds to a single interpretable concept. A sparse autoencoder (SAE) addresses this by learning a dictionary of directions in the model's activation space, each of which activates on a semantically coherent set of inputs [10]. For any given input, only a small number of these directions are active, making the representation sparse and interpretable. Concretely, given an activation vector $\mathbf{h}$, the SAE computes $\mathbf{f} = \text{TopK-ReLU}(\mathbf{h}\mathbf{W}_{\text{enc}}^T + \mathbf{b}_{\text{enc}})$, producing a sparse feature vector $\mathbf{f}$ with at most k non-zero entries. We use the Qwen-Scope SAE trained on Qwen3-8B-Base with 64,000 features (k=50). This decomposition lets us ask which learned directions are associated with abstraction and which with correctness, and whether those directions overlap.

Activation extraction. For each non-degenerate base-model trace in the letter-string corpus ($n \approx 2{,}450$), we extract the residual-stream activation at layer 18 at the execution-commit position: the last token before the "`Answer:`" line. This position captures the model's state after reasoning but before committing to the answer, and carries information about both the reasoning process and the impending output.

Feature–behaviour correlations. For each of the 64,000 SAE features, we compute two Pearson correlations over the labelled corpus: (a) with the judge-averaged abstraction composite (A-correlation) and (b) with binary correctness (E-correlation). The top-k features by each correlation define two feature sets: A-features (abstraction-predictive) and E-features (correctness-predictive). Whether these sets overlap or diverge determines whether abstraction and correctness are one direction or two in the model's representations. We test this by comparing the cosine similarity between the A-feature and E-feature directions against a null distribution of random k-feature pairs, across multiple values of k.

Rule-framing analysis. To connect the SAE features to observable behaviour, we characterise the CoT traces that maximally activate the top abstraction features, quantify the prevalence of rule-framing openings (provenance-verified rule/pattern keywords in the first sentence of the trace) across the RL trajectory, and conduct a three-arm causal test on 500 base-model items: we teacher-force a rule-framing, neutral, or concrete-framing first sentence (length-matched, no answer information leaked) and have the judge score only the model's continuation. The neutral

arm forces a sentence that commits to no reasoning strategy, providing a fair baseline against which the rule-framing arm can be evaluated.

Intervention experiment. To test whether abstract reasoning style causally drives accuracy, we conduct a five-arm intervention on the base models (Qwen3-4B-Base and Qwen3-8B-Base). Each arm uses the same 2,490 letter-string analogy items with greedy decoding; only the few-shot prefix varies, manipulating the reasoning style while holding model weights and test items fixed. The arm designs and contrasts are described alongside the results in Section 5.5.

5 Experiments

This section describes the tasks, models, and experimental design. The method (Section 4) defines the measurement instrument; this section describes what we measure and on what.

5.1 Tasks

We evaluate on three held-out task families from the Lewis–Mitchell counterfactual analogy suite [22, 21]. These tasks probe abstract reasoning because solving them requires extracting the governing rule and generalising it to new instances. The core ability each task demands, identifying structural patterns and summarising them as reusable rules, directly reflects whether the model exhibits the abstraction dimensions we defined in Section 4.1 within its chain-of-thought reasoning.

Letter-string analogies. Each item shows two letter strings related by a hidden transformation rule, then gives a third string and asks the model to produce the fourth by applying the same rule. For example, if `[a b c]` becomes `[a b d]` (the last letter advances by one), and the third string is `[i i j]`, the correct answer is `[i i k]`. The model is never told the rule; it must discover the rule from the example pair and apply it to the new input. The suite includes standard conditions (the familiar `a-b-c-...-z` alphabet) and permuted-alphabet conditions where the letter ordering is scrambled (e.g. `m r a z t c b p ...`). Permuted conditions come at four difficulty levels (permutation depth $2, 5, 10, 20$) and serve as a contamination-robust control: because the alphabet is unfamiliar, the model cannot rely on memorised letter orderings and must reason over the structure of the transformation. The full set contains 2,490 items per checkpoint.

Story analogies. Each item presents one reference story (Story 1) and two candidate stories (Story A and Story B). The model must decide which candidate shares the same relational structure as Story 1. The three stories differ in surface content — different characters, settings, and domains — so the model cannot match on surface features. It must abstract away the concrete details and compare the underlying structure: who plays what role, what causes what, and how events relate. For example, a story about a hawk offering feathers to a hunter and a story about a country selling weapons to an aggressor may share the structure "a weaker party appeases a stronger party to avoid conflict." The set contains 76 items.

Digit-matrix analogies. Each item presents a number matrix (typically 3×3) with one cell left blank. The numbers in each row follow a rule — for instance, the third cell might be the sum of the first two, or the difference between them. The model must discover the rule by

examining the completed rows, then apply it to fill the blank cell. Rules range from simple operations (constants, fixed-step progressions) to compositions of arithmetic operations. Unlike letter-string analogies, where the structure is sequential, digit-matrix items require the model to identify both the rule and the axis along which it operates (row-wise, column-wise, or both).

All three task families are entirely disjoint from every model's RL training data. No analogy item appears in any training set, so any base-to-RL change reflects transfer rather than task-specific learning. Appendix C shows example problems from all three task types.

5.2 Models and training

To establish that the abstraction effect is a general property of outcome RL rather than an artifact of a particular model, dataset, or scale, we evaluate across three dimensions: model family (Qwen3, Qwen2.5, Mistral), training domain (mathematics, code), and model size (4B to 24B).

We train Qwen3-4B-Base and Qwen3-8B-Base with GRPO [35] on the DeepScaleR recipe [25], an outcome reward on roughly forty thousand competition mathematics problems. The two resulting models differ only in base size, isolating RL effects from scale effects. Both base models are also trained on the DeepCoder recipe (competitive programming tasks), which tests whether the abstraction gain depends on the training domain. Throughout the results, we distinguish the two recipes with parenthetical labels: Qwen3-4B (DeepScaleR) and Qwen3-4B (DeepCoder) share the same base model but differ in RL training data. Checkpoints are saved every 100 steps across the run (steps $100, 200, \ldots, 1000$ for mathematics; up to step 500 for code), producing an RL trajectory for later analysis. Throughout the results, "RL" denotes the final checkpoint (step 1000 for mathematics, step 500 for code, and the single released checkpoint for SimpleRL-Zoo) and "base" denotes the base model.

We additionally evaluate published SimpleRL-Zoo checkpoints [52] on three base architectures outside the Qwen3 family: Qwen2.5-7B, Qwen2.5-14B, and Mistral-Small-24B, all trained with the same GRPO and outcome reward recipe. The Qwen2.5 pair provides a scaling ladder on an independent family; Mistral-Small-24B extends the test to a non-Qwen architecture at larger scale. As an in-distribution control, Qwen3-4B (DeepScaleR) raises MATH500 accuracy from 63.0% to 71.9%, confirming that the RL training learned mathematics.

5.3 Does RL induce abstract reasoning?

The central question of this thesis is whether outcome RL spontaneously induces more abstract reasoning on tasks the model was never trained on. We test this by scoring each model's chain-of-thought traces on the four abstraction metrics defined in Section 4.1: type/role binding, structured relation construction, schematic sufficiency, and explicit schema extraction. Each metric is scored on a 0–3 ordinal scale anchored by level-specific rubrics (Appendix B). Two independent LLM judges from different model families (Llama-3.3-70B-AWQ and Qwen2.5-72B-AWQ) score each non-degenerate trace independently, blinded to answer correctness, model identity, and checkpoint step (Section 4.2). All scores reported below are averaged across the two judges.

Table 1 presents the results: RL raises all four abstraction scores in the model's chain-of-thought reasoning on both letter-string and story analogies, across all tested model families and both judges.

Table 1: Abstraction scores in chain-of-thought reasoning: base and RL scores with change (Δ) on four metrics (0–3 ordinal scale), averaged across two independent judges (excluding degenerate traces). All RL training uses GRPO with outcome reward.

		Role			Relation			Sufficiency			Schema		
Task	Model	Base	RL	Δ	Base	RL	Δ	Base	RL	Δ	Base	RL	Δ
Letter-str.	Qwen3-4B	2.13	2.49	+.36	2.24	2.65	+.41	2.36	2.70	+.34	1.49	2.08	+.59
Letter-str.	Qwen3-8B	2.44	2.60	+.16	2.66	2.80	+.14	2.72	2.82	+.10	1.94	2.34	+.40
Letter-str.	Qwen3-4B (DeepCoder)	2.13	2.36	+.23	2.24	2.50	+.26	2.36	2.57	+.21	1.49	1.89	+.40
Letter-str.	Qwen3-8B (DeepCoder)	2.44	2.57	+.13	2.66	2.77	+.11	2.72	2.81	+.09	1.94	2.26	+.32
Letter-str.	Qwen2.5-7B	2.34	2.53	+.19	2.56	2.73	+.17	2.64	2.77	+.13	1.76	2.14	+.38
Letter-str.	Qwen2.5-14B	2.48	2.82	+.34	2.68	2.94	+.27	2.74	2.94	+.20	1.98	2.55	+.57
Letter-str.	Mistral-Small-24B	1.38	2.59	+1.21	1.44	2.87	+1.43	1.51	2.86	+1.35	0.71	2.23	+1.52
Story	Qwen3-4B	2.18	2.39	+.21	2.21	2.39	+.18	2.14	2.34	+.20	1.82	2.02	+.20
Story	Qwen3-8B	2.30	2.43	+.13	2.30	2.41	+.11	2.22	2.37	+.14	1.95	2.10	+.15
Story	Qwen3-4B (DeepCoder)	2.18	2.36	+.18	2.21	2.32	+.11	2.14	2.26	+.12	1.82	1.95	+.13
Story	Qwen3-8B (DeepCoder)	2.29	2.41	+.12	2.29	2.39	+.11	2.22	2.36	+.14	1.95	2.12	+.18
Story	Qwen2.5-7B	1.99	2.19	+.20	1.97	2.16	+.19	1.94	2.14	+.20	1.55	1.80	+.26
Story	Qwen2.5-14B	2.31	2.46	+.15	2.26	2.48	+.22	2.19	2.45	+.26	1.88	2.23	+.36
Story	Mistral-Small-24B	2.23	2.40	+.17	2.23	2.40	+.16	2.14	2.34	+.20	1.95	2.15	+.20

Letter-string analogies. On the primary letter-string task, all four metrics rise across all seven models and three model families (Table 1). The largest gains are on explicit schema extraction, which requires the model to state the governing rule as a general pattern rather than merely applying it to the instance at hand: +0.59 for Qwen3-4B (DeepScaleR), +0.57 for Qwen2.5-14B, and +0.38 for both Qwen3-4B (DeepCoder) and Qwen2.5-7B. The remaining three metrics rise together. The 8B model shows the largest gain on schema extraction (+0.40), while the remaining three metrics rise more modestly (+0.10 to +0.16). Mistral-Small-24B produces the largest gains (+1.21 to +1.52 across the four metrics): its base scores range from 0.71 to 1.51, but RL lifts them into the range of the Qwen3 models. We also evaluated Llama-3.1-8B (SimpleRL-Zoo), but its base performance on these analogy tasks is near zero (letter-string accuracy 0.0%), placing it below the capability threshold needed for the abstraction effect to manifest; details are in Section 5.10. Section 5.7 identifies the concrete chain-of-thought change that underlies the consistent score shift across the seven models in the main tables.

The trajectory across RL steps reveals two patterns (Figure 3). Qwen3-4B (DeepScaleR) rises monotonically from base to step 1000 on all four metrics. The 8B model jumps sharply at step 100 and then plateaus on the first three metrics; schema continues to rise throughout training. Both judges reproduce these patterns. The code-trained models (Qwen3-4B (DeepCoder) and Qwen3-8B (DeepCoder)) confirm the effect is not mathematics-specific: both show the same monotonic rise on letter-string analogies, with schema carrying the largest gain (Table 1; Figure 3, bottom rows).

Story analogies. All four metrics rise on stories as well, though more modestly (+0.11 to +0.36; Table 1, Figure 4). The pattern is consistent across all models and both judges. The sample size is small (n=76 items per checkpoint), so these estimates are noisier than the letter-string results, but the direction is positive for every model-metric pair.

Digit-matrix analogies. The two tasks above show a consistent abstraction gain. Digit-matrix analogies are an exception: RL leaves the mean abstraction score unchanged across all three models we evaluate. Section 5.7 analyses this contrast in detail and traces it to a different starting point: digit-matrix base traces already open in rule-framing mode, so the mode-entry

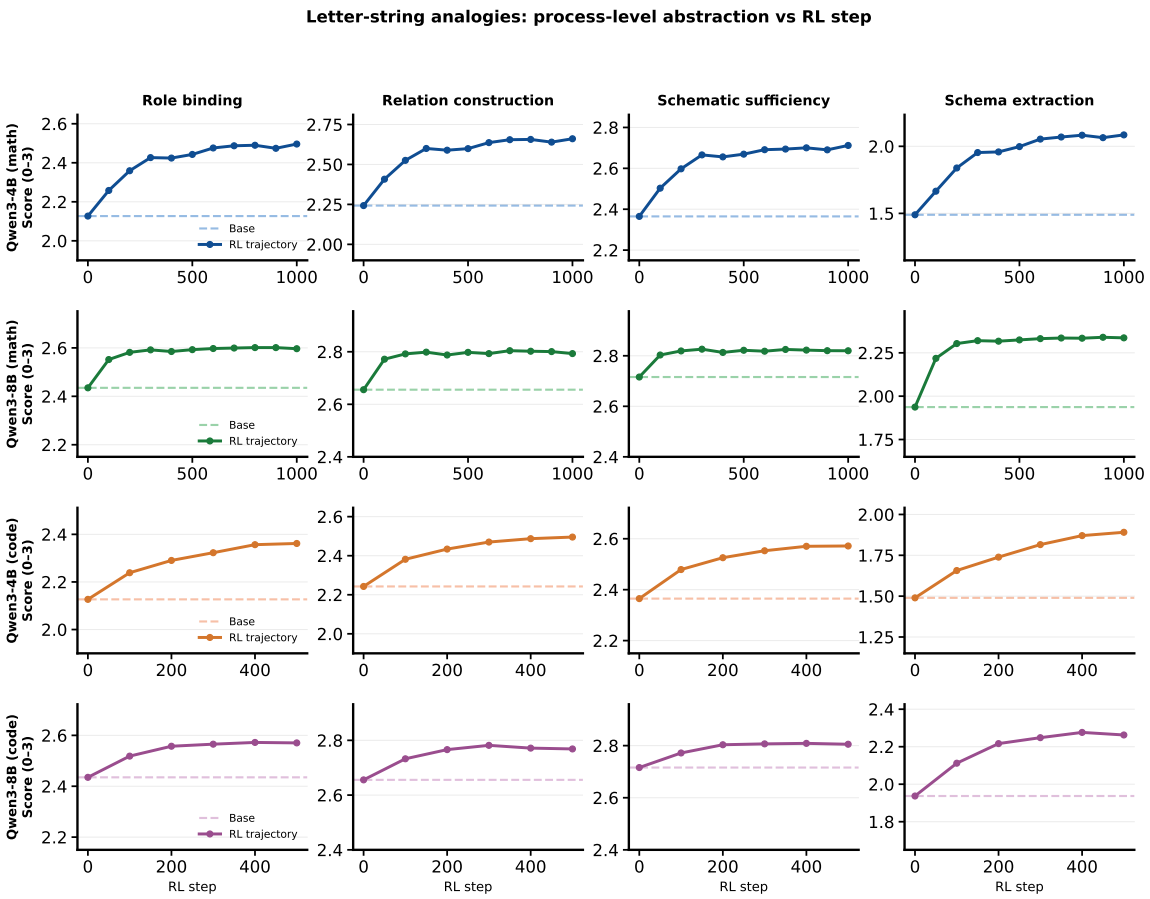


Figure 3: Letter-string analogy abstraction across RL steps. Each panel shows one of the four metrics (judge-averaged, non-degenerate traces). The 4B model rises monotonically; the 8B model jumps early and plateaus on the first three metrics, with schema carrying the later gain.

shift that drives the letter-string and story gains has no margin to operate. The flat digit-matrix result is consistent with the mechanism we identify, not a counterexample to it.

RL does increase how often the model produces genuine reasoning traces on digit-matrix items: P(reason) rises from 0.25 to 0.53 for the 4B, from 0.34 to 0.78 for Qwen2.5-7B, and from 0.89 to 0.92 for the 8B. The accuracy consequence varies with model capacity: the 4B loses 13.6 pp because its direct intuition was more accurate than its reasoning, the 8B is unchanged, and Qwen2.5-7B gains +10.3 pp because deliberation is its only viable strategy.



Figure 4: Story analogy abstraction across RL steps (same format as Figure 3). All four metrics rise modestly for both models. The smaller gains and tighter spacing reflect both the domain gap from the mathematics training task and the smaller sample size (n=76 items per checkpoint).

5.4 Abstraction and accuracy across analogy tasks

The preceding section establishes that RL raises abstraction on letter-string and story analogies while leaving digit-matrix scores unchanged. A crucial follow-up is whether the abstraction gain translates into better answers, or whether the two are independent effects.

Letter-string analogies: abstraction tracks accuracy. RL raises letter-string accuracy across all seven models (Table 2), from +2.7 pp (Qwen3-4B (DeepCoder)) to +36.8 pp (Mistral-Small-24B). At the item level, abstraction and accuracy are positively linked. A logistic regression predicting binary correctness from the judge-averaged abstraction composite, fitted separately for each model–judge pair (n=2,490 items each), is significant for every pair. Items on which the model reasons more abstractly are substantially more likely to be answered correctly, and the association is strongest in permuted-alphabet conditions, where surface recall is impossible and only genuine abstraction can help.

RL also narrows the accuracy gap between standard and permuted conditions. Section 5.5 tests whether this association is causal.

Story analogies: abstraction transfers across domains. All four abstraction metrics rise on stories across seven models and three families (Table 1): the Qwen3 models gain +0.11 to +0.21 per metric, the Qwen2.5 SimpleRL models gain +0.15 to +0.36, and Mistral-Small-24B

Table 2: Accuracy across all base–RL pairs (greedy, T=0). Letter-string: n=2,490 items per model. Story: n=76 items per model; story deltas are directional at this sample size.

		Letter-string			Story		
Model	Recipe	Base	RL	Δ pp	Base	RL	Δ pp
Qwen3-4B	DeepScaleR	.082	.144	+6.2	.592	.539	−5.3
Qwen3-8B	DeepScaleR	.254	.341	+8.7	.395	.474	+7.9
Qwen3-4B	DeepCoder	.077	.104	+2.7	.579	.579	0.0
Qwen3-8B	DeepCoder	.254	.310	+5.6	.395	.408	+1.3
Qwen2.5-7B	SimpleRL-Zoo	.088	.147	+5.9	.171	.263	+9.2
Qwen2.5-14B	SimpleRL-Zoo	.254	.330	+7.6	.421	.526	+10.5
Mistral-Small-24B	SimpleRL-Zoo	.084	.452	+36.8	.329	.447	+11.8

gains +0.16 to +0.20. Accuracy does not follow. Across the story pairs in Table 2, the accuracy deltas range from −5 to +12 percentage points with no consistent direction, and the same logistic regression finds no significant association between abstraction and correctness on stories.

5.5 Does abstraction cause accuracy?

The previous section shows that abstraction and accuracy rise together on letter-string analogies. A natural hypothesis is that RL raises abstraction and abstraction in turn raises accuracy. If this causal chain holds, then externally imposing abstract reasoning on base models should improve their accuracy. We test this by manipulating reasoning style on the base models through five prompt arms: **A0** (zero-shot baseline), **concrete** (three worked exemplars in a letter-by-letter style), **abstract** (the same exemplars in an abstract rule-stating style), **abstract-wrong** (abstract style but with deliberately incorrect answers, to separate style from answer leakage), and **rule-given** (the abstract prefix plus the correct transformation rule for each test item). Model weights, test items, and decoding are identical across arms; only the few-shot prefix varies.

Table 3 presents accuracy and judge-scored abstraction for each arm on both base models. The manipulation check confirms that the prompt successfully moved the mediator: abstract and abstract-wrong both raise chain-of-thought abstraction to approximately 2.9, well above the A0 baseline (1.89 for the 4B, 2.35 for the 8B) and the concrete arm (1.78, 1.75). The accuracy contrasts that follow are therefore interpretable: the abstract arms did induce abstract reasoning, as the judge confirms.

Abstract reasoning does not raise accuracy. The most direct test of the causal hypothesis is to compare abstract and concrete reasoning on the same problems: if abstraction drives accuracy, the abstract arm should outperform the concrete arm. The two arms use the same exemplar problems and the same few-shot format; only the reasoning style differs. The abstract arm successfully raises abstraction: both models score above 2.9 on the 0–3 scale, compared to 1.75–1.78 for the concrete arm. Yet the models that reason more abstractly answer fewer questions correctly. On the 4B, accuracy under the abstract arm is 12.5%, compared to 21.9% under the concrete arm; on the 8B, 24.0% versus 32.4%. If abstraction drove accuracy, the more abstract arm should perform better. The data show the opposite on both models.

Table 3: Intervention results on letter-string analogies (greedy T=0). Accuracy is computed over all 2,490 items per arm. Abstraction is the judge-averaged composite (0–3 scale) scored on a stratified subsample of 440 items per arm.

	Qwen3-4B-Base		Qwen3-8B-Base	
Arm	Accuracy	Abstraction	Accuracy	Abstraction
A0 (baseline)	0.090	1.89	0.261	2.35
Concrete	0.219	1.78	0.324	1.75
Abstract	0.125	2.92	0.240	2.93
Abstract-wrong	0.102	2.92	0.143	2.93
Rule-given	0.367	2.80	0.371	2.33

The abstraction metric is not contaminated by correctness. Before interpreting the accuracy results, we need to verify that the judge's abstraction score genuinely measures reasoning form rather than answer correctness. The abstract-wrong arm tests this: it uses the same abstract reasoning style as the abstract arm but deliberately provides incorrect answers in the exemplars. If the judge were scoring correctness rather than reasoning form, the two arms should receive different abstraction scores. They do not: both score 2.92 on the 4B and 2.93 on the 8B, while their accuracy diverges (12.5% versus 10.2% on the 4B, 24.0% versus 14.3% on the 8B). The judge assigns the same abstraction score whether the demonstrated answer is right or wrong, confirming that the abstraction metric captures how the model reasons, not whether it reaches the correct answer.

Rule identification is the operative variable. If merely adopting an abstract reasoning style does not help, what does? The rule-given arm tests whether identifying the correct rule is the missing ingredient. It adds one element on top of the abstract arm: the correct transformation rule for each test item. Everything else stays the same. This single addition raises accuracy dramatically, from 12.5% to 36.7% on the 4B and from 24.0% to 37.1% on the 8B, while abstraction stays comparable or even decreases (2.80 on the 4B, 2.33 on the 8B). On the 8B, the pattern is especially clear: rule-given scores lower on abstraction than the zero-shot baseline (2.33 versus 2.35) yet achieves substantially higher accuracy (37.1% versus 26.1%). What drives accuracy is not abstract style per se but the discovery of the underlying rule. This raises the question of whether RL's abstraction gain reflects a deeper shift: not just a stylistic change, but a disposition to seek rules — a possibility the next two sections investigate through internal representations and chain-of-thought openings.

5.6 Abstraction and correctness share a coupled internal direction

The previous section identifies rule identification, not abstract style, as the operative variable for accuracy. Yet abstraction and accuracy are strongly correlated at the item level (Section 5.4). If these two properties are not causally sequential but arise from a shared internal basis, the model's representations should reflect this coupling. We test this with sparse-autoencoder feature analysis (Section 4.3).

We extract residual-stream activations from layer 18 of Qwen3-8B-Base at the last token before the answer line (the execution-commit position; Section 4.3) and decompose them with a 64,000-feature SAE. For every feature, we compute two correlations across the letter-string corpus ($n \approx 2{,}450$ non-degenerate base traces): one with the judge-averaged abstraction composite

(producing an "A-correlation") and one with binary correctness (producing an "E-correlation"). The top-k features by each correlation define two feature sets: A-features (the directions in the model's representations most predictive of abstraction) and E-features (those most predictive of correctness). If abstraction and correctness are independent internal signals, these two feature sets should point in different directions. If they are coupled, the sets should overlap.

We measure the overlap by computing the cosine similarity between the A-feature and E-feature directions. To calibrate whether this similarity is meaningful, we draw 50 random k-feature pairs from the same SAE and compute their cosine as a baseline. At every scale tested (k=5 through k=50), the observed cosine between the abstraction and correctness directions far exceeds the random baseline: 0.25 vs 0.03 at k=5, 0.68 vs 0.09 at k=20, and 0.88 vs 0.21 at k=50. The two feature sets point in the same direction to a degree that random feature subsets never approach. The feature sets are stable: re-running the analysis on 50 resampled versions of the corpus recovers 75–91% of the same features each time. Inside the model, the features that predict abstraction and the features that predict correctness point in the same direction.

5.7 Rule-framing openings in chain-of-thought

The intervention (Section 5.5) identifies rule identification as the operative variable for accuracy, and the SAE analysis confirms that abstraction and correctness are coupled internally. A natural next question is what concrete change RL produces in the chain-of-thought text. By examining the traces that maximally activate the top abstraction features, we find that RL amplifies rule-framing openings, in which the model declares intent to identify the rule or pattern before working through the specific instance.

Identifying rule-framing openings. High-abstraction traces open with language such as "to solve this, we need to **determine** the **pattern** or **rule** that connects the given letter strings," while low-abstraction traces begin with the concrete letters—"the first pair of brackets contains the letters w, t, u, v"—and proceed position by position (Figure 5). We quantify this with a frozen instrument: the first sentence of each trace is scored for seven provenance-verified rule keywords (**pattern**, **rule**, **determine**, **identify**, **relationship**, **alphabet order**, **analyze**). These seven were selected by verifying which candidate words actually appear in the openings of the base model's highest-activating traces. Across the 2,490 base-model traces, traces whose first sentence contains more rule keywords correlate positively with judge-scored abstraction (Pearson $r = 0.24$) and with the top-50 SAE abstraction features at the execution layer (Pearson $r = 0.43$), where r ranges from -1 (inverse) through 0 (unrelated) to $+1$ (perfectly co-occurring). Appendix C shows a concrete example: a base trace that echoes the brackets letter by letter, with no rule keyword anywhere in the trace, scores 0 on all four metrics, while the RL trace on the same item opens by declaring intent ("we need to understand the given alphabet order and apply it"), extracts the correct rule, and scores 3 on all four.

RL amplifies rule-framing across training. The contrast between high- and low-abstraction openings suggests that RL's abstraction gain may be linked to the prevalence of these rule-framing openings. To test this, we track rule-framing systematically across RL training steps with two complementary instruments applied to the first sentence of each trace. A declaration detector checks whether the sentence contains a rule-finding declaration: an intent phrase ("we need to", "let's", "to solve") co-occurring with a target ("identify the pattern", "determine the rule") or a direct rule statement ("the pattern is..."). A keyword counter scores the density of seven provenance-verified rule words; from it we derive entry P($\geq$1 keyword), measuring

Base model opening (step 0, same item, 0 rule keywords in first sentence)

"The first pair of brackets contains the letters w, t, u, v. The second pair of brackets contains the letters r, t, u, v. The third pair of brackets contains the l. . . "

RL model opening (step 1000, same item, 2 rule keywords in first sentence)

"To solve this problem, we need to **identify** the **pattern** in the given letter-string analogies and apply it to find the missing letters."

Forced-opening causal test templates (length-matched, no answer leak)

RULE: "First, I will identify the general rule that transforms each input string into its output, and then apply that same rule."

NEUTRAL: "First, I will look carefully at the given examples and think about the problem before writing down my answer."

CONCRETE: "First, I will write out the strings and compare them one position at a time, checking what happens to each letter."

Figure 5: Contrasting chain-of-thought openings on the same letter-string item. The base model opens by enumerating letters; after RL, the model opens by declaring that it will identify the pattern. Rule keywords from the provenance-verified list are **bolded**. The three causal test templates (blue) are teacher-forced on the base model; judges score only the model's continuation.

whether the model opens in rule-framing mode at all, and depth P(≥3 keywords), measuring how densely it commits to rule vocabulary.

Table 4a and Figure 6 track these metrics across RL training on Qwen3-8B. The declaration rate rises from 91.1% at base to 99.4% at step 100. Entry saturates in the same window: 91.8% to 100%. Depth rises more gradually, from 36.9% to 44.7% at step 100 and plateaus near 48%. Judge-scored abstraction (2.44 → 2.62) and accuracy (25.4% → 32.4%) co-acquire in the same window (Figure 6). The rapid entry saturation reflects a structural shift: RL eliminates the concrete instance-echo openings ("the first two groups of letters are [z v i t c]. . . ") that account for the base model's 8% non-declaring residue.

Table 4b confirms that the amplification holds across models and recipes. Entry rises toward 100% for every pair; depth shows the sharpest gains on models with the lowest base starting points — Qwen2.5-7B jumps from 8.6% to 76.3%, and Qwen2.5-14B from 0.7% to 63.1%. Code-RL models (Qwen3-4B and Qwen3-8B, both trained with DeepCoder) show the same pattern on held-out letter-string analogies, confirming that the rule-framing disposition transfers across training domains.

Rule-framing shapes subsequent reasoning. Does this opening merely co-occur with abstract reasoning, or does it shape what follows? We answer with a three-arm causal test: on 500 base-model items, we teacher-force a rule-framing, neutral, or concrete-framing first sentence (length-matched at ∼20 words, no answer information leaked; see Figure 5) and have the judges score only the model's own continuation, with the forced opening excluded. The neutral arm commits to no reasoning strategy, providing a fair baseline. Paired by item, continuation

(a) Training trajectory (Qwen3-8B, DeepScaleR, letter-string task)

	Base	**Step 100**	**Step 200**	**Step 1000**
Entry P(≥1 keyword)	91.8%	100.0%	100.0%	100.0%
Depth P(≥3 keywords)	36.9%	44.7%	47.1%	48.0%
Mean keywords	1.96	2.25	2.30	2.28
Abstraction (0–3)	2.44	2.59	2.62	2.64
Accuracy	25.4%	29.7%	32.4%	34.1%

(b) Cross-model confirmation (letter-string task, base → RL)

		Entry P(≥1)		**Depth P(≥3)**	
Model	**Recipe**	Base	RL	Base	RL
Qwen3-8B	DeepScaleR	91.8%	100.0%	36.9%	48.0%
Qwen3-8B	DeepCoder	91.8%	100.0%	36.9%	48.8%
Qwen3-4B	DeepCoder	69.4%	81.9%	24.9%	63.1%
Qwen2.5-7B	SimpleRL-Zoo	39.8%	99.7%	8.6%	76.3%
Qwen2.5-14B	SimpleRL-Zoo	81.5%	100.0%	0.7%	63.1%

Table 4: Rule-framing amplification in chain-of-thought openings (letter-string task). Entry = P(≥1 rule keyword in first sentence), e.g. "First, we need to understand the given alphabet **order**..." scores 1. Depth = P(≥3), e.g. "...we need to **determine** the **pattern** or **rule**..." scores 3. (a) On Qwen3-8B, entry saturates at step 100; depth and abstraction co-acquire in the same window. (b) Every model–recipe pair shows the same direction; depth rises most sharply on Qwen2.5-7B (8.6% → 76.3%).

abstraction is 2.09 for rule-framing, 1.76 for neutral, and 1.18 for concrete-framing. The rule-framing arm raises abstraction by +0.33 above the neutral baseline (n=500, paired by item), confirming that the opening genuinely steers the reasoning rather than merely outperforming an artificially suppressive concrete pole. Accuracy follows the same ordering: rule 0.246, neutral 0.196, concrete 0.063. This connects to the intervention result in Section 5.5: rule identification drives accuracy, and declaring intent to find the rule at the start of reasoning facilitates that identification.

The same mechanism appears on story analogies. On letter-string tasks, RL makes the model identify and articulate the governing rule before working through the instance. A natural question is whether the same tendency appears on a structurally different task, or whether it is specific to letter-string analogies.

We apply the same SAE analysis to Qwen3-8B story traces (n=76). The coupling between abstraction and correctness features is present at large feature counts (k=50 and above) but weaker at small k, consistent with the smaller sample. Story analogies do not exhibit rule-framing openings in the same form—there is no transformation rule to declare in a single sentence. To measure the analogous behaviour, we exploit a feature of the story task prompt, which includes the directive "Reason about the relational structure of each story." When a model's trace contains the phrase "relational structure," it co-occurs with actual cross-story relational analysis, so its presence marks genuine structural reasoning. When we remove the directive from the prompt (bare-prompt control), the phrase drops to 0% across four models and both arms, confirming that the model does not produce it spontaneously. The directive therefore

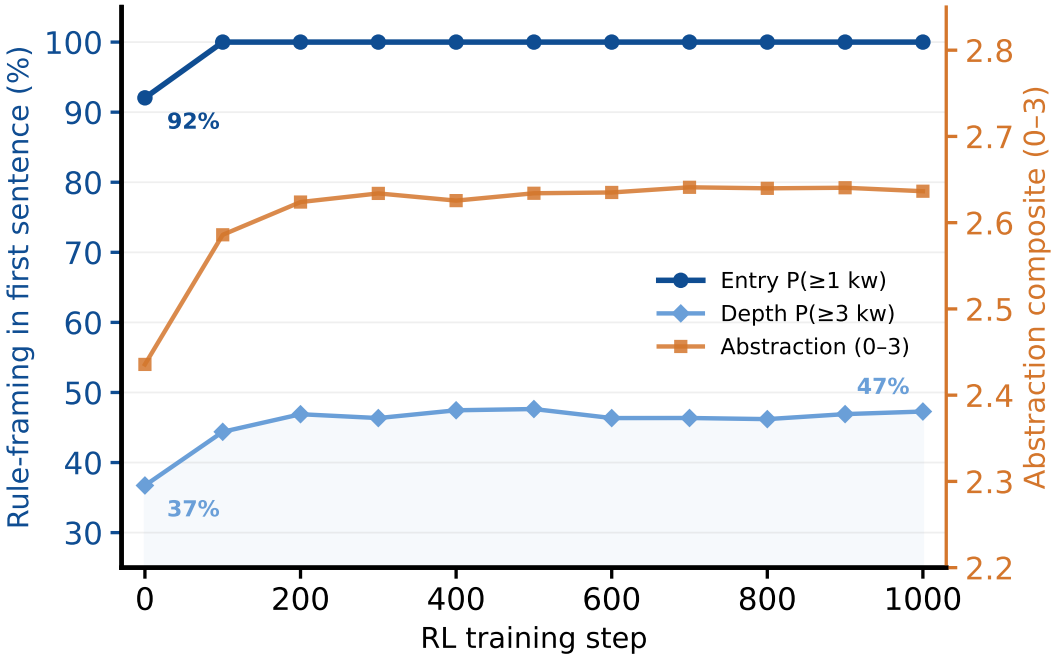


Figure 6: Co-acquisition of rule-framing and abstraction during RL training (Qwen3-8B, DeepScaleR, letter-string task). Entry P(≥1 keyword; dark blue) saturates at step 100; depth P(≥3; light blue) and judge-scored abstraction (orange, right axis) rise together and plateau after step 200.

acts as a probe: each occurrence in the trace indicates that the model executes the instructed structural analysis, and tracking how often this occurs across base and RL reveals whether RL amplifies the same rule-seeking disposition observed on letter-string tasks.

RL amplifies this behaviour consistently. On Qwen3-8B (DeepScaleR, math-RL), the probe rate rises from 58% to 61%; on Qwen3-8B (DeepCoder, code-RL), trained on a different domain with the same base model, it rises from 58% to 64%. On Qwen2.5-7B (SimpleRL-Zoo, a different model family), the rate doubles from 32% to 64%, while its traces grow shorter (255 → 235 words), ruling out a simple length account. Under the bare prompt, the judge-measured abstraction gain still replicates on all four models (Appendix A), confirming that the increased probe rate reflects deeper structural analysis, not generic instruction compliance. Inspecting individual traces confirms that when the model utters "relational structure," it follows through with genuine cross-story relational analysis: the case study in Appendix C shows an RL trace that names the shared relational structure and then constructs a full causal schema, scoring higher on all four abstraction metrics than the base trace on the same item, which partially abstracts but does not elaborate the causal structure or use it to discriminate between candidates. Unlike letter-string tasks, where the prompt is short and the model can open directly with a rule-finding declaration, story prompts are dominated by the stories themselves, which may explain why rule-framing declarations do not emerge spontaneously on this task and require the directive as a probe.

Why digit-matrix abstraction stays flat. If RL amplifies rule-framing openings, and rule-framing openings predict abstraction scores, why does digit-matrix abstraction not rise? The entry/depth instrument answers this question. Applying the same frozen detector to Qwen3-8B under a bare prompt (directive sentence removed, letter-string-mirrored template),

digit-matrix base traces already open with a rule-finding declaration at 99.7% (entry), and their first contentful sentence carries a mean of 3.07 rule keywords with 100% scoring P($\geq$3) (depth). Both margins exceed the RL endpoint on letter-string tasks (entry 99.4%, depth mean 2.27, P($\geq$3) 47%). RL moves whatever entry and depth margins exist: on letter-string analogies both are open, so opening statistics and judge scores rise; on digit-matrix analogies both are born saturated, and the residual score variance reflects rule-search success rather than mode entry. The disposition instead expresses on the engagement margin: P(reason) rises from 0.25 to 0.53 (4B), from 0.34 to 0.78 (Qwen2.5-7B), and from 0.89 to 0.92 (8B). A case study in Appendix C makes this visible: both base and RL traces open with the identical declaration ("let's analyze the relationships between the cells"), but the base cycles through wrong formulas within single rows (scoring 0.50), while the RL trace compares across rows and extracts a reusable pattern (scoring 3.00).

5.8 Abstraction and the pass@k pattern

The pass@k analysis of Yue et al. [51] is the central claim in the RLVR literature that RL produces only distributional sharpening rather than new capability. To position our abstraction findings within this debate, we replicate their experiment on Qwen3-4B (DeepScaleR) using the letter-string benchmark at T=0.6 with up to k=512 samples per item. Yue et al. showed that the accuracy advantage of RL-trained models shrinks and vanishes as the number of sampled attempts k grows, concluding that RL concentrates probability mass on correct solution paths the base model already contains.

Table 5 confirms the pattern. At k=1, RL leads on standard (+2.3 pp) and permuted (+2.3 pp) conditions. At k=128, the standard gap closes (+1.2 pp). At k=512, the permuted gap is dead even ($-$0.2 pp) and symbolic is similarly flat (+2.5 pp). The Yue pattern replicates on a held-out analogy benchmark outside the mathematics domain: RL improves first attempts without expanding the set of solvable problems.

Figure 7 shows the full pass@k curves. The convergence rate differs across conditions: on standard items, the base model catches up at $k\approx$32 and the two curves overlap by k=128; on permuted items, RL maintains a visible lead through k=128 and the curves converge only at k=512; on symbolic items, RL leads through k=256. Because permuted and symbolic conditions remove memorised letter orderings, the slower convergence indicates that RL's advantage persists longer when the model must rely on structural reasoning rather than surface recall.

Table 5: Pass@k on letter-string analogies (Qwen3-4B (DeepScaleR), T=0.6). Gap = base − RL; negative values indicate RL leads. Permuted pools perm-$\{2, 5, 10, 20\}$; n=128 per item for standard, up to n=512 for permuted, and n=512 for symbolic. At k=1, RL leads; by extended k the gap closes on every condition.

Condition	k	Base	RL	Gap
Standard	1	0.115	0.138	−0.023
Standard	128	0.931	0.918	+0.012
Permuted	1	0.064	0.087	−0.023
Permuted	512	0.847	0.850	−0.002
Symbolic	1	0.005	0.007	−0.002
Symbolic	512	0.650	0.625	+0.025

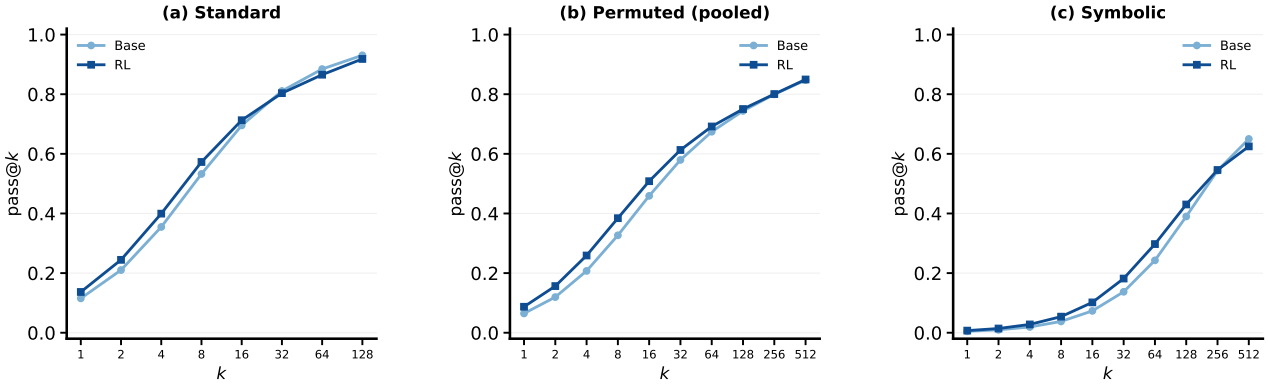


Figure 7: Pass@k curves for base and RL models (Qwen3-4B, DeepScaleR, T=0.6). (a) Standard items (n=128 samples per item): the base model catches up by k≈32. (b) Permuted items pooled (n=256–512): RL leads through k=128; the gap closes only at k=512. (c) Symbolic items (n=512): RL leads through k=256. The slower convergence on permuted and symbolic conditions suggests that RL's advantage reflects structural reasoning, not surface recall.

The abstraction gain is not a byproduct of accuracy sharpening. The pass@k result, combined with the item-level association between abstraction and accuracy (Section 5.4), raises a natural concern: if RL sharpens toward correct paths, and correct traces tend to be more abstract, the observed abstraction gain might be entirely explained by a compositional shift toward more correct traces rather than a genuine change in reasoning quality. We test this by stratifying items according to whether base and RL produce the same correctness outcome on each item (greedy, T=0), and measuring the abstraction difference within each stratum (Table 6).

On items where both base and RL answer correctly, RL traces are still more abstract in all seven models (+0.075 to +0.430; Table 6, left). On items where both answer incorrectly, the gain is larger (+0.150 to +1.534). Correctness is held constant in both strata, so the abstraction difference cannot arise from RL solving more items. RL genuinely changes how the model reasons, not just which items it gets right.

Table 6: Outcome-stratified abstraction control (letter-string analogies, greedy T=0, judge-averaged composite). Items are paired by problem; Δ = RL − base abstraction within each correctness stratum. RL traces are more abstract even when correctness is held constant.

	Both correct			**Both wrong**		
Model	n	Base abs	Δ	n	Base abs	Δ
Qwen3-4B (DeepScaleR)	67	2.54	+0.244	1793	2.01	+0.437
Qwen3-8B (DeepScaleR)	436	2.76	+0.075	1322	2.34	+0.191
Qwen3-4B (DeepCoder)	52	2.49	+0.149	1885	2.02	+0.251
Qwen3-8B (DeepCoder)	421	2.76	+0.093	1396	2.33	+0.150
Qwen2.5-7B	90	2.59	+0.138	1908	2.30	+0.210
Qwen2.5-14B	379	2.70	+0.167	1364	2.42	+0.362
Mistral-Small-24B	178	2.33	+0.430	575	1.01	+1.534

5.9 Scaling and RL as comparable routes to abstraction

Is the abstraction gain specific to RL, or does scaling model size produce the same effect? Comparing untrained bases across the 4B–8B–14B parameter ladder shows that abstraction rises with scale independently of RL. On letter-string analogies, the 8B base matches the 4B after full RL on role binding, relation construction, and schematic sufficiency (approximately 2.6 on a 0–3 scale). Schema, with more headroom (base scores 1.4–1.9), is where both routes continue to improve. RL on the 4B raises abstraction to approximately the level of the 8B base; RL on the 8B approaches the 14B base.

The two routes differ in what else they bring. Scaling from 4B to 8B raises accuracy by +17.2 pp while raising abstraction by +0.39; RL on the 4B raises abstraction by a comparable +0.43 but accuracy by only +6.2 pp. Scaling lifts accuracy and abstraction together as part of a broad capability gain; RL produces a disproportionate abstraction gain relative to its accuracy improvement, suggesting that RL's effect on abstraction is more specific than scaling's.

The SimpleRL-Zoo Qwen2.5 models confirm the pattern on an independent model family. On the 7B–14B ladder, RL raises the composite abstraction score by +0.22 (7B) and +0.34 (14B), and RL on the 7B (2.54) exceeds the untrained 14B base (2.47). Figure 8 visualises the result: on both ladders, RL at a given size lifts abstraction to approximately the level of the next-larger untrained base.

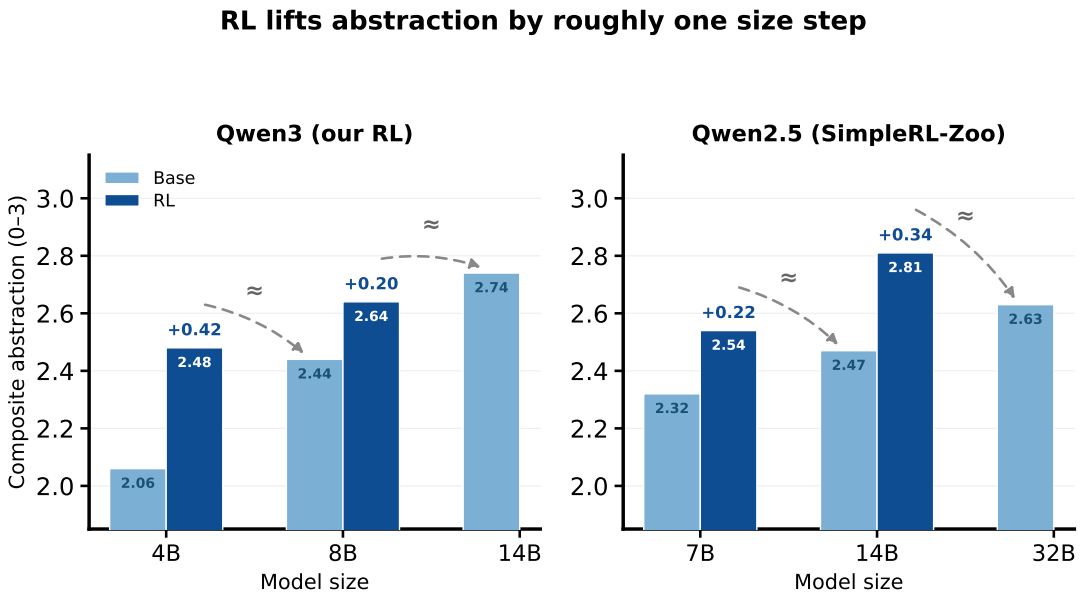


Figure 8: RL and scaling as comparable routes to abstraction. Light bars show untrained bases; dark bars show RL-trained models. Dashed arrows marked "≈" highlight the key pattern: on both the Qwen3 ladder (left, our RL) and the Qwen2.5 ladder (right, SimpleRL-Zoo), RL at size X raises abstraction to approximately the level of the untrained base at size $X{+}1$.

5.10 Supplementary analyses

Llama-3.1-8B: a capability-limited case. The Llama-3.1-8B SimpleRL-Zoo model is excluded from the main tables because its base performance on these analogy tasks is near zero (letter-string accuracy 0.0%, with abstraction scores in the 0.3–1.6 range). RL raises abstraction modestly (+0.05 to +0.30 on letter-string), but the absolute scores remain far below the other model families, and letter-string accuracy reaches only 1.2%. These results do not contradict the main finding but reflect a capability floor: the model lacks sufficient base performance on analogy tasks for the abstraction effect to manifest.

Human–judge agreement on letter-string analogies. To validate the LLM judges against human judgment, the author independently scored a random subset of 30 letter-string traces on all four abstraction metrics using the same 0–3 rubric. Table 7 reports the quadratic weighted kappa (QWK) between the human scores and each judge. Agreement is moderate to substantial on relation construction and schema extraction (QWK 0.62–0.68), and lower on role binding and schematic sufficiency (0.18–0.49). Using two judges gives a consistency check on the rubric, but this validation is limited: it involves a single annotator (the author) rather than independent raters, so inter-annotator reliability cannot be assessed. The agreement levels are sufficient to support the directional claims in this thesis but would benefit from a multi-annotator study in future work.

Table 7: Human–judge agreement on letter-string analogies (n=30 items). QWK = quadratic weighted kappa on the 0–3 ordinal scale.

Metric	Human–Llama	Human–Qwen
Type/role binding	0.35	0.49
Relation construction	0.62	0.67
Schematic sufficiency	0.19	0.18
Schema extraction	0.68	0.62

6 Discussion

RL raises abstraction in chain-of-thought reasoning, and this gain takes a concrete form: models shift toward proactive rule identification. The experiments establish this change (Finding 1), reveal its internal structure (Finding 2), identify its behavioural mechanism (Finding 3), and place it within the broader debate about what outcome RL achieves (Finding 4). Specifically: outcome RL raises all four abstraction metrics on held-out analogy tasks across model families, training domains, and task types, and the abstraction gain manifests as rule-framing openings in the chain-of-thought text. Sparse-autoencoder analysis reveals that abstraction and correctness share a coupled internal direction; the operative variable for accuracy is rule identification, not abstract reasoning form. RL concretely reshapes chain-of-thought text toward rule-framing openings, and a causal test confirms that these openings shape subsequent reasoning. Pass@k replication confirms distributional sharpening, but the process-level evidence shows that RL sharpens toward qualitatively different reasoning, not merely toward the same paths at higher probability. Together, these findings place RL between the optimistic view (RL expands what the model can do) and the pessimistic view (RL merely concentrates probability mass). We discuss each in turn, synthesise them, and draw out their implications.

6.1 Finding 1: RL raises process-level abstraction

Across all seven base–RL pairs, three model families, and two training domains, RL raises all four abstraction metrics on letter-string analogies (Table 1). The gains range from +0.09 (Qwen3-8B schematic sufficiency) to +1.52 (Mistral-Small-24B schema extraction), with schema extraction consistently showing the largest improvement. This gain takes a concrete form: RL-trained models more consistently open their reasoning by declaring intent to identify the governing rule or relationship before working through the instance, a shift we quantify and causally test in Finding 3. The same direction holds on story analogies, though more modestly (+0.11 to +0.36). RL trained on mathematics and RL trained on code both produce the effect; it transfers zero-shot from the training domain to structurally different analogy tasks. The consistency across these conditions establishes the gain as a general property of outcome RL, not a side effect of fitting a particular training distribution.

Digit-matrix analogies remain flat, and the mechanism analysis (Section 5.7) explains why: digit-matrix base traces already open in rule-framing mode (entry 99.7%, depth mean 3.07), so the mode-entry shift that drives the letter-string and story gains has no margin to operate.

Two further patterns reinforce this interpretation. First, on both the Qwen3 4B–8B ladder and the Qwen2.5 7B–14B ladder, RL at a given size lifts abstraction to approximately the level of the next-larger untrained base (Figure 8). RL on the Qwen2.5-7B reaches a composite of 2.54, exceeding the untrained 14B base (2.47). That two independent interventions — scaling and reinforcement learning — both raise abstraction and converge on a shared ceiling suggests

that abstraction is a stable model property rather than an artifact of a particular training recipe. The two routes differ in what else they bring, however. Scaling from 4B to 8B raises accuracy by +17.2 pp while raising abstraction by +0.39; RL on the 4B raises abstraction by a comparable +0.43 but accuracy by only +6.2 pp. Scaling lifts accuracy and abstraction together as part of a broad capability gain; RL produces a disproportionate abstraction gain relative to its accuracy improvement. One possible reading is that scaling raises abstraction as a byproduct of general capability growth, while RL more specifically reshapes chain-of-thought reasoning toward abstract patterns, with accuracy improvement as a secondary consequence rather than the primary change.

Second, story analogies show that the abstraction gain does not guarantee accuracy improvement on every task. RL raises all four metrics on stories across seven models (Table 1), yet accuracy shows no consistent improvement (−5 to +12 pp; Table 2). The abstract reasoning that RL cultivates through mathematics and code training transfers to story analogies at the process level but does not reliably convert into correct answers on a task that demands deeper narrative comprehension.

6.2 Finding 2: abstraction and correctness share a coupled internal direction

The intervention experiment (Table 3) creates an apparent paradox. On letter-string analogies, abstraction and accuracy are strongly associated at the item level: more abstract traces are substantially more likely to be correct. Yet forcing abstract reasoning through prompt priming does not raise accuracy; it lowers it. The abstract arm scores 2.92–2.93 on abstraction but achieves only 12.5% accuracy (4B) and 24.0% (8B), compared to 21.9% and 32.4% under the concrete arm. The abstract-wrong arm confirms that the judge is not contaminated by correctness: both abstract arms receive identical abstraction scores regardless of whether the demonstrated answers are correct.

The rule-given arm identifies what does drive accuracy: rule identification. Supplying the correct transformation rule raises accuracy from 12.5% to 36.7% (4B) and from 24.0% to 37.1% (8B), while abstraction stays comparable or even decreases. What drives accuracy is knowing the rule, not reasoning abstractly.

SAE analysis adds an internal perspective. Inside the model, the features most predictive of abstraction and those most predictive of correctness point in the same direction, far above chance (Section 5.6). Abstraction and correctness are not two independent internal signals; they are one coupled direction that RL up-regulates. This coupling is consistent with the intervention result: if abstraction and correctness are joint effects of a shared internal direction, then moving the surface text toward abstract-looking language without engaging that direction would raise abstraction scores without improving accuracy.

6.3 Finding 3: RL reshapes chain-of-thought text toward rule-framing

The coupled direction identified in Finding 2 predicts that RL should produce observable changes in chain-of-thought text. Examining the traces that maximally activate the top SAE abstraction features reveals what this change looks like: RL-trained models open their reasoning by declaring intent to identify the governing pattern before working through the instance — a behaviour we call rule-framing (Section 5.7). We operationalise this by counting provenance-verified rule keywords in the first sentence of each trace; their presence correlates positively with both

judge-scored abstraction (Pearson $r = 0.24$) and SAE abstraction-feature activation (Pearson $r = 0.43$). The textual opening is not merely stylistic but connects to the model's internal abstraction representation. RL amplifies rule-framing openings: on Qwen3-8B, entry P($\geq$1 keyword) rises from 92% to 100% by step 100, while depth P($\geq$3) rises from 37% to 48% (Table 4). Abstraction and accuracy co-acquire in the same window. The change is visible across model families: Qwen2.5-7B entry rises from 40% to 100% and depth from 9% to 76%.

This shift is not merely stylistic. A three-arm causal test on 500 base-model items confirms that rule-framing openings shape the subsequent reasoning: teacher-forcing a rule-framing first sentence raises continuation abstraction by +0.33 above a neutral baseline, and accuracy follows the same ordering (rule 0.246, neutral 0.196, concrete 0.063). That the judge-scored abstraction gains correspond to a measurable linguistic shift in the traces, and that this shift causally affects the model's subsequent reasoning, provides independent validation that the process-level metrics capture a real change in how the model reasons.

The same disposition surfaces on story analogies in a different form. Story traces do not exhibit rule-framing openings, but RL amplifies relational-structure naming: on Qwen2.5-7B, the rate doubles from 32% to 64%. The underlying behaviour, articulating the governing pattern before applying it, generalises across task types even though its surface expression differs.

6.4 Finding 4: RL changes how the model reasons, not just what it solves

A prominent account of RLVR holds that RL merely concentrates probability mass on correct solution paths the base model already contains, without creating new reasoning capability [51]. Our pass@k replication (Table 5) confirms the distributional convergence on a held-out benchmark: at k=1, RL leads; at extended k, the gap closes across all conditions. RL improves first attempts without expanding the set of solvable problems. A purely distributional reading treats this as blind probability concentration, re-weighting existing paths without altering their character.

The process-level evidence challenges this reading. The paths RL concentrates mass on are qualitatively different from those it down-weights. At k=1, the RL model commits to paths that are more abstract and more rule-deriving (Finding 1); the base model reaches equivalent accuracy only by sampling broadly across a distribution that weights those paths less heavily. The intervention (Finding 2) identifies what makes these paths more effective: rule identification, not abstract form. Rule-framing openings make rule identification more likely (Finding 3), and the SAE evidence shows that the internal features driving this behaviour are coupled with those driving correct execution.

The outcome-stratified control (Table 6) strengthens this reading. On items where both base and RL answer correctly, RL traces are still more abstract; on items where both answer incorrectly, the gap is larger still. If the abstraction gain were merely a compositional artifact of RL solving more items, it would vanish when correctness is held constant. It does not.

These two accounts are not in conflict. RL does concentrate probability mass, but it concentrates toward paths that reason more abstractly and seek rules. Because rule-seeking paths are more likely to find the correct answer, this concentration is what drives RL's accuracy gain at k=1.

This interpretation also explains why the base model eventually catches up at large k (Figure 7). RL commits the model to rule-seeking paths, which means it relies less on alternative strategies that the base model still uses freely — pattern matching, memorised orderings, and surface shortcuts. At small k this is an advantage, because rule-seeking paths find the correct

answer more often. At large k it becomes a cost: the base model's wider variety of strategies lets it stumble upon correct answers that RL's narrower distribution misses.

The convergence rate across conditions fits this picture. Standard items use the familiar alphabet, so the base model can exploit memorised letter orderings as a shortcut and catches up quickly ($k \approx 32$; Figure 7a). Permuted and symbolic items scramble the alphabet, removing this shortcut and forcing the model to discover the governing rule from the examples alone — so RL's rule-seeking advantage persists through $k=128$ and the curves converge only at $k=512$ (Figure 7b,c). The more a problem depends on genuine rule discovery rather than prior knowledge, the longer RL stays ahead.

6.5 Synthesis

The four findings compose a single account. RL raises abstraction on held-out tasks, and this gain manifests as a shift toward proactive rule identification — on letter-string tasks through rule-framing openings, on story tasks through more frequent relational-structure analysis (Finding 1). On digit-matrix analogies, where the base model already reasons abstractly, the gain is absent. Inside the model, abstraction and correctness share a coupled representational direction; the operative variable for accuracy is rule identification, not abstract form (Finding 2). The abstraction gain manifests concretely as rule-framing openings that shape subsequent reasoning and generalise across task types (Finding 3). The distributional convergence reported by Yue et al. [51] is real, but it is not blind probability concentration: RL sharpens toward this more abstract, rule-deriving reasoning mode (Finding 4).

Yue et al. conclude that RL merely concentrates probability mass on correct paths the base model already contains. Our findings qualify this view. RL does not expand the solution space — the distributional evidence confirms this. But the paths RL concentrates toward are qualitatively different: more abstract, more rule-deriving, and rooted in a coupled internal direction that co-produces abstraction and correct execution. RL does not create new solutions, but it commits the model to a reasoning mode that the base model accesses only at low probability.

This suggests that RL's accuracy improvement at $k=1$ is itself a consequence of this quality shift. RL-trained models reason more abstractly, and more abstract reasoning facilitates rule identification, which drives correct answers. The accuracy gain is not a separate effect that happens alongside the abstraction gain; it flows from it.

7 Conclusions and Further Research

This thesis asked whether outcome RL, trained only for correct final answers, cultivates more abstract reasoning in the model's chain of thought. We measured abstraction directly in chain-of-thought traces using four metrics grounded in Barsalou's cognitive decomposition [2], scored by correctness-blind LLM judges, and applied them to models trained with GRPO on mathematics and code, evaluated zero-shot on held-out analogy benchmarks.

The answer is yes, with a qualification. RL raises process-level abstraction consistently across three model families, two training domains, and two of three analogy tasks. The gain takes a concrete form: RL-trained models more actively seek rules and relationships in their reasoning, and all four abstraction metrics rise accordingly. Digit-matrix analogies stay flat because the base model already reasons abstractly on this task. SAE analysis reveals that abstraction and correctness share a coupled internal direction, and a five-arm intervention shows that rule identification, not abstract form, drives accuracy. A causal test confirms that rule-framing openings shape subsequent reasoning. Pass@k replication confirms that RL concentrates

probability mass without expanding the solution space, but the paths it concentrates toward are qualitatively different: more abstract, more rule-deriving, and rooted in an internal direction the base model accesses only at low probability.

Methodological contribution. This work contributes a reusable measurement instrument. The four process-level metrics apply to any chain-of-thought reasoning model. The intervention protocol offers a template for testing whether a process-level metric is causally operative or merely a marker.

7.1 Limitations

The process-level metrics rely on LLM-as-judge scoring. We anchor each metric with concrete behavioural definitions on a 0–3 ordinal scale, use two independent judges from different model families, and verify that the abstract-wrong intervention arm rules out simple correctness leakage. These measures increase reliability, but they do not guarantee it: LLM judges may share systematic biases that human annotators would not, and human validation would strengthen the measurement. Furthermore, both judges are 70B-class open-weight models (Llama-3.3-70B and Qwen2.5-72B). For the models under evaluation (up to 24B parameters), the capacity gap between judge and examinee is sufficient to support reliable scoring, but these judges still fall short of frontier closed-source models. With greater computational budget, frontier judges could provide higher-quality scoring and reduce the risk of systematic blind spots in trace evaluation.

The intervention experiment manipulates reasoning style through prompt priming. Although we control for answer leakage (the abstract-wrong arm), length-match the forced openings, and exclude the forced text from judge scoring, each arm's prompt inevitably introduces stylistic and framing differences beyond the target variable. We minimise these confounds but cannot eliminate them entirely. The intervention addresses the causal question at the prompt level and only on base models; whether the same dissociation holds at the representation level, or for RL-trained models, remains open.

The SAE analysis shows that abstraction and correctness features are coupled and identifies rule-framing openings as the behavioural mechanism, but it does not go deeper. The strongest single feature explains only modest variance ($r \approx 0.36$), and steering the top features does not raise abstraction more than steering random features of the same count. Techniques like activation patching or circuit-level analysis could test whether the coupled direction is necessary for accuracy or merely co-activated. We identify the coupling but not a causal handle on it.

Our analogy benchmarks test relational reasoning but are simpler than tasks like ARC-AGI [6], which requires compositional induction over grids. The models evaluated here (up to 24B) would likely struggle with ARC-AGI, so it remains open whether the abstraction gain we document extends to harder benchmarks or is specific to the pattern-completion tasks studied here.

7.2 Further Research

The SAE analysis provides initial evidence for a common-cause structure: abstraction and correctness are coupled in the model's representations, and RL amplifies rule-framing openings that causally shape abstraction. Steering the identified features does not cleanly move abstraction beyond what random features achieve, so the analysis locates the coupling without isolating a causal handle. Activation patching or causal tracing could test whether the coupled direction is causally necessary for the accuracy gain, or merely co-activated.

This study examines only outcome-based RL (GRPO with outcome reward). Process-reward models provide step-level supervision and optimise for reasoning quality directly. PRM training could produce the same abstraction gain through a different internal route, or a larger gain because the process is directly rewarded. It could also produce no gain, if step-level feedback constrains the exploratory reasoning from which abstraction emerges. Comparing outcome and process-supervised training on the same base would clarify whether the abstraction effect is specific to outcome RL or general to reinforcement-based fine-tuning.

Our metrics measure abstraction after training; a natural next step is to use them during training. Incorporating the abstraction scores as an explicit reward signal — rewarding the model not only for correct answers but also for reasoning abstractly — could amplify the abstraction gain that outcome RL produces spontaneously. Recent work in this direction, such as RLAD [31] and AbstRaL [11], rewards procedural abstraction and reports improved generalisation. Our four-metric framework offers a more fine-grained reward, but this approach places stringent demands on judge quality: reward noise from unreliable scoring would propagate directly into policy gradients.

The analogy benchmarks used here, though diverse in surface form, are pattern-completion tasks from a single suite. The models evaluated here would likely struggle with harder benchmarks like ARC-AGI [6], so it remains unclear whether the abstraction gain generalises to more compositional reasoning. Testing on ARC-AGI with frontier-scale models, or on other widely used reasoning benchmarks such as causal reasoning and multi-step mathematical proof, would clarify whether the findings reported here reflect a general property of RL-induced reasoning or one specific to relational analogy.

We opened with a question: does outcome RL, trained only for correct answers, cultivate more abstract reasoning? The answer is yes, but with a qualification. RL does not expand what the model can solve, but it changes how the model reasons. After training, models seek rules and relationships before working through instances, and this shift shows up across model families, training domains, and evaluation tasks. The shift is not just stylistic: rule-seeking paths find the correct answer more often, and the accuracy gain at k=1 follows from this. What RL produces is not new capability in the distributional sense, but a commitment to a reasoning mode that the base model uses only occasionally. How this commitment arises during training remains open.

References

[1] Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful, 2025. arXiv:2503.08679.

[2] Lawrence W. Barsalou. Abstraction in perceptual symbol systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1435):1177–1187, 2003.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[4] Nick Chater and Paul Vitányi. Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences*, 7(1):19–22, 2003.

[5] Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think, 2025. arXiv:2505.05410.

[6] François Chollet. On the measure of intelligence, 2019. arXiv:1911.01547.

[7] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards (PRIME), 2025. arXiv:2502.01456.

[8] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025.

[9] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs, 2025. arXiv:2503.01307.

[10] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024. arXiv:2406.04093.

[11] Silin Gao, Antoine Bosselut, Samy Bengio, and Emmanuel Abbe. AbstRaL: Augmenting LLMs' reasoning by reinforcing abstract thinking, 2025. arXiv:2506.07751.

[12] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[13] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on LLM-as-a-Judge, 2025. arXiv:2411.15594.

[14] Damian Hodel and Jevin West. Response: Emergent analogical reasoning in large language models, 2023. arXiv:2308.16118.

[15] Douglas R. Hofstadter. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, 1995.

[16] Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general LLM capabilities? understanding transferability of LLM reasoning, 2025. arXiv:2507.00432.

[17] Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[19] Brenden M. Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623:115–121, 2023.

[20] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, et al. Measuring faithfulness in chain-of-thought reasoning, 2023. arXiv:2307.13702.

[21] Martha Lewis and Melanie Mitchell. Evaluating the robustness of analogical reasoning in large language models, 2024. arXiv:2411.14215.

[22] Martha Lewis and Melanie Mitchell. Using counterfactual tasks to evaluate the generality of analogical reasoning in large language models, 2024. arXiv:2402.08955.

[23] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

[24] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[25] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL, 2025. Agentica project; technical report / HuggingFace release.

[26] Gary F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, 2001.

[27] Melanie Mitchell, Alessandro B. Palmarini, and Arseny Moskvichev. Comparing humans, GPT-4, and GPT-4V on abstraction and reasoning tasks, 2023. arXiv:2311.09247.

[28] Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research*, 2023.

[29] Sam Musker, Alex Duchnowski, Raphaël Millière, and Ellie Pavlick. LLMs as models for analogical reasoning, 2024. arXiv:2406.13803.

[30] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 27730–27744, 2022.

[31] Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. RLAD: Training LLMs to discover abstractions for solving reasoning problems, 2025. arXiv:2510.02263.

[32] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. arXiv:1707.06347.

[34] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in RLVR, 2025. arXiv:2506.10947.

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

[36] Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in LLM-as-a-Judge, 2024. arXiv:2406.07791.

[37] Michael Sullivan and Alexander Koller. GRPO is secretly a process reward model, 2025. arXiv:2509.21154.

[38] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[39] Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations (ICLR)*, 2025.

[40] Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[42] Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[43] Taylor W. Webb, Keith J. Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7:1526–1541, 2023.

[44] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[45] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

[46] Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[47] An Yang and others (Qwen Team). Qwen3 technical report, 2025. arXiv:2505.09388.

[48] Chenlu Ye, Zhou Yu, Ziji Zhang, Hao Chen, Narayanan Sadagopan, Jing Huang, Tong Zhang, and Anurag Beniwal. Beyond correctness: Harmonizing process and outcome rewards through RL training (PROF), 2025. arXiv:2509.03403.

[49] Cheng-Kai Yeh, Hsing-Wang Lee, Chung-Hung Kuo, and Hen-Hsen Huang. AR^2: Adversarial reinforcement learning for abstract reasoning in large language models. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2025.

[50] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in LLMs, 2025. arXiv:2502.03373.

[51] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[52] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. arXiv:2503.18892.

[53] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

[54] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

A Bare-prompt protocol controls

The story and digit-matrix tasks carry directive sentences inherited from their original benchmarks ("Reason about the relational structure of each story" and "Think step by step about the relationship between cells"), while the letter-string prompt is bare by default. To verify that the abstraction results are not artifacts of these directives, we re-run the evaluation with the directives removed and measure the same judge-scored composite.

Story analogies. Under the bare prompt, the RL-induced abstraction gain replicates on all four models: Qwen3-8B +0.142 (instructed: +0.146), Qwen2.5-7B +0.051 (+0.21), Qwen2.5-14B +0.341 (+0.25), and Mistral-Small-24B +0.100 (+0.18). The "relational structure" phrase occurs in 0% of bare-prompt traces across all four models and both arms, confirming the zero-background condition for the probe analysis in Section 5.7.

Digit-matrix analogies. Under the bare prompt (directive and system line removed, letter-string-mirrored template), judge-scored abstraction remains flat: Qwen3-8B −0.016, Qwen3-4B +0.054, Qwen2.5-7B +0.031. The bare prompt releases headroom: P(= 3) drops by 14 pp on Qwen3-8B (from 0.71 to 0.57), yet RL still does not exploit this headroom. Accuracy roughly halves under bare (e.g. Qwen3-8B 0.35 → 0.15), indicating that the directive helps solving but not mode entry. Digit-matrix flatness survives all three protocol controls (reference on/off, instructed/bare prompt, headroom release).

B Scoring Rubrics

This appendix documents the complete scoring rubrics used by the two LLM judges (Llama-3.3-70B-AWQ and Qwen2.5-72B-AWQ) to score abstraction in chain-of-thought traces. Each judge receives (a) the system prompt below, (b) the task prompt shown to the model, (c) the model's reasoning trace, and (d) the metric rubric with task-specific anchor examples. Most anchor examples are real chain-of-thought traces from the scored corpus, selected by a multi-judge LLM ensemble; a small number are idealised exemplars. The judge is never told the correct answer, the model's identity, or any training details.

B.1 Judge System Prompt

The following system prompt is shared across all tasks and metrics.

> You are an expert evaluator of reasoning traces. Your job is to score whether a model's chain-of-thought (CoT) exhibits specific abstraction-like behaviours. You are NOT judging whether the final answer is correct, and you are NOT told the answer, the model's identity, or any training details.
>
> **Principles:**
>
> 1. Score observable behaviour in the trace, not hidden reasoning and not the outcome. A trace can be abstract yet wrong, or correct yet unabstract. Do not raise any score because the trace reaches the correct final answer, and do not lower any score because the answer is wrong.
> 2. Independence. Score each metric on its own 0–3 scale, independently. Do not let a high score on one metric inflate another.
> 3. Length is not abstraction. A long, verbose, or repetitive trace is not more abstract; a short trace can be highly abstract. Never reward verbosity, restatement, or schema-flavoured boilerplate that is not doing real work.
> 4. Use the task prompt (and the task reference, if provided) to know what the correct abstract structure would be, but judge the trace, not your own solution.
> 5. Ground every score. For each metric, write a 1–3 sentence rationale citing what in the trace supports the score, then give the integer score (0–3).
> 6. Output strict JSON matching the requested schema. No text outside the JSON.

B.2 Letter-String Analogy Rubrics

Each metric below is scored 0–3. The task reference tells the judge that the correct abstract structure is the transformation rule (successor, predecessor, sort, add-letter, remove-redundant, or fix-alphabet) applied to a letter string under a given ordering.

B.2.1 Type/Role Binding

Measures: whether the trace maps concrete surface tokens to task-relevant abstract roles and uses those roles in subsequent reasoning (Barsalou's type-token interpretation).

0 No role abstraction. The trace only repeats concrete details or restates the problem.

1 Some relevant entities or quantities are identified, but without clear abstract roles.

2 The load-bearing roles are bound but one is vague or implicit, or they are bound but only lightly used.

3 The task-relevant abstract roles are bound and used downstream. For letter-string analogy the load-bearing roles are the transformed slot (which position changes) and the operand of the transformation (e.g. "the last letter" feeding a successor). A trace that binds and reuses these earns 3.

Anchor examples.

- **Level 0:** "The first set is [e f g h] and the second is [e f g h k], so the answer is [f g h k]." — pure surface restatement; no token is mapped to a role.
- **Level 1:** "The second string is the first with 'y' and 'w' swapped — 'y' moves to 'w's position and 'w' to 'y's." — names the changed letters but binds them as concrete tokens, not abstract roles.
- **Level 2:** "The second string is the same as the first except the last letter, replaced with 'c'; apply the same to the second pair." — binds the transformed slot (last letter) and operand, but the successor operation stays implicit.
- **Level 3:** "Roles: the transformed slot is the final position; the operand is its last letter, fed to a successor op. Apply that role mapping to [h i j k]: last slot, operand 'k', successor 'l' → [h i j l]." — binds transformed-slot and operand as abstract roles and reuses both.

B.2.2 Structured Relation Construction

Measures: whether the trace organises the abstract roles into a relation (a schema, formula, or dependency structure) rather than merely listing the variables (Barsalou structured representation + Gentner structure-mapping).

0 No relation is constructed.

1 A relation is implied but not clearly expressed.

2 A relation is expressed but incomplete, overly concrete, or partially incorrect.

3 A complete and task-appropriate abstract relation is explicitly constructed.

Anchor examples.

- **Level 0:** Restates the two strings and writes "Step 2 — Apply the rule ... Answer: ?" without constructing any relation.
- **Level 1:** "The second string is a rearrangement of the first, with the letters in a different order." — a relation is implied (rearrangement) but never expressed (which or how).
- **Level 2:** "The rule is that the second string is the first string with an additional letter at the end." — a genuine relation, but incomplete (never says which letter).
- **Level 3:** "second = first + an additional letter at the end; to find which, use the given ordering — the next after the last letter 'k' is 'l'; so second = first + successor(last) → [h i j k l]." — complete, task-appropriate append-successor relation.

B.2.3 Schematic Sufficiency

Measures: whether the trace forms a sparse but sufficient representation, discarding irrelevant surface detail while preserving everything needed to solve the task (Barsalou schematic representation). This is not token length: short ≠ sufficient; long ≠ insufficient.

0 The trace is either overly concrete (keeps irrelevant detail) or omits necessary information.

1 Removes some detail but loses part of the necessary structure.

2 Preserves the necessary structure but still carries unnecessary surface detail.

3 Sparse, sufficient, task-relevant representation.

Anchor examples.

- **Level 0:** Re-enumerates both strings and ends with "Answer: ?" — keeps surface detail and omits the transformation entirely.
- **Level 1:** "The second string is the reverse of the first; reversing [l n m o p] gives [p o m n l]." — sparse, but loses the necessary structure (mischaracterises the transformation).
- **Level 2:** "List them out: first pair * @ = &, second * @ = ¡. The last letter of the first (&) is replaced by the last of the second (¡); apply to [= & ¡ +]." — preserves the necessary structure but still carries surface restatement.
- **Level 3:** "Only the last slot changes: new last letter = successor of the old last (orderings agree, so k → l); apply to the target's last slot → [h i j l]." — sparse, every necessary element present, no padding.

B.2.4 Explicit Schema Extraction

Measures: whether the trace explicitly summarises the specific problem into a reusable abstract schema, rule, pattern, or problem type, naming or framing it as an instance of something more general (Barsalou summary representation).

0 No schema extracted: the trace only restates the genre ("this is a letter-string analogy", "find the pattern and apply it"), or applies an operation purely inline with no general framing.

1 A vague problem type or partial rule is mentioned but not framed as reusable.

2 A reusable rule is stated but stays tied to this instance's specific letters, is only partially stated, or names the wrong transformation. A clearly named but wrong rule caps here.

3 A fully articulated, reusable transformation is named or framed as a general schema that captures the task's actual structure.

Anchor examples.

- **Level 0:** "The first set is [e f g h] and the second [e f g h k]; missing 'k', so the answer is [f g h k]." — no rule or problem-type named.
- **Level 1:** "The second string adds one letter to the end; the added letter is 'u', the next after 't'. Now apply the same pattern." — describes the operation for this instance but does not frame it as a reusable schema.
- **Level 2:** "The rule is that the second string is the first string with an additional letter at the end." — a reusable-ish rule, but partial (omits the successor mechanism).
- **Level 3:** "This is the successor rule: replace the last letter with its alphabet successor, leaving the rest fixed; apply to [h i j k] → k advances to l → h i j l." — names a fully articulated reusable schema capturing the task's true structure.

Correctness coupling. Answer or rule correctness is not scored at levels 0–1 (an absent or vague schema scores low regardless of the final answer). Rule correctness gates only the 2-versus-3 boundary: a reusable rule naming the wrong transformation caps at 2; only a rule capturing the task's true structure reaches 3.

B.3 Story Analogy Rubrics

Story analogies require the model to identify which candidate story (A, B, or both) shares Story 1's relational structure rather than its surface content. The task reference tells the judge that the correct abstract structure is Story 1's relational and causal schema: the load-bearing roles (the acting agent, the counterpart, and the pivotal object or outcome) together with the causal or contingency relations among them. Irrelevant elements are the specific characters, settings, and objects.

B.3.1 Type/Role Binding

Measures: whether the trace identifies Story 1's relational roles and binds them to the corresponding roles in the candidate stories, then reuses those bindings to drive the comparison.

0 The trace only restates surface events (names, settings, objects, plot beats) without identifying relational roles; no cross-story correspondence is drawn.

1 Relevant participants or events are named, but as concrete story tokens rather than abstract roles, and any cross-story link is by surface similarity rather than explicit role-correspondence.

2 The load-bearing relational roles of Story 1 are bound across stories, but one mapping is vague or implicit, or the binding is stated yet only lightly used.

3 The load-bearing roles are bound as explicit cross-story correspondences and those bindings are reused to drive the relational comparison and the decision.

Anchor examples.

- **Level 0:** "In Story 1, Mark is the chicken farmer who is attacked by his chickens, and later, a thief steals the eggs." — retells each story's surface plot in isolation; no cross-story role mapping.
- **Level 1:** "The recipient (squirrel, mother) is not willing to reciprocate the effort or gift." — gestures at a role and parenthesises the two fillers, but binds them as a loose surface grouping rather than an explicit mapping.
- **Level 2:** "The hunter wants the hawk's feathers, while in Story A, Gagrach wants Zerdia's computers." — binds two load-bearing roles (the demanding party and the valued object) but leaves the counterpart and reciprocal-promise roles implicit.
- **Level 3:** "The cobra (Pierre) is analogous to the young boy (Gene), and the falcon is analogous to the policeman." — explicit cross-story role binding; reasoning then runs over these bound roles.

B.3.2 Structured Relation Construction

Measures: whether the trace constructs Story 1's relational or causal schema (the roles and the relations among them) rather than listing surface events.

0 No schema for Story 1. The trace only lists surface elements or events or makes a vague "similar structure" claim; no relation among Story 1's roles is expressed.

1 A causal or relational structure is implied but not expressed. The trace hints at an event sequence or role mapping but never states what causes, or is exchanged for, what.

2 A real relation among roles is articulated, but it stays bound to concrete events, omits a key element, or is partly inaccurate.

3 A complete abstract schema is explicitly built: roles plus the relations among them, abstracted from surface, and mapped or compared across candidates.

Anchor examples.

- **Level 0:** "A character drops an item, loses something else, and ends up without the item." — flattened to disconnected events, no relation constructed.
- **Level 1:** "The police officer's reaction to Gene's action is similar to the falcon's reaction to the cobra's action." — a relation is implied via reaction-to-action, but Story 1's schema is never stated.
- **Level 2:** "Motivated by an incentive to change their behaviour, but ultimately fails to achieve the desired outcome due to time constraints." — a real abstract relation, but omits the reward-contingency that defines Story 1.
- **Level 3:** "The incident (cutting his neck) led to the realisation that a safety feature was necessary, and Mr. Newton acted on this realisation." — a full abstract causal schema (incident → realise a measure is needed → the agent acts), then mapped across candidates.

B.3.3 Schematic Sufficiency

Measures: whether the trace forms a sparse but sufficient relational schema of Story 1 (all load-bearing roles and relations present, irrelevant surface detail dropped).

0 Fails to form a usable relational schema: either stays at the surface (retelling events) or asserts an empty or tautological "structure" that does no discriminating work.

1 Abstracts away from surface but the schema is lossy in a load-bearing way: a relation needed to decide the analogy is dropped or mischaracterised.

2 Captures the load-bearing relational schema correctly and uses it to compare candidates, but still carries unnecessary surface detail.

3 Extracts a sparse, sufficient relational schema: abstracts Story 1 to its load-bearing roles and relations, explicitly drops surface content, maps roles across stories, and compares on that schema with no missing relation and no surface padding.

Anchor examples.

- **Level 0:** "Story A is more analogous to Story 1 because it involves a character (Jenny) who initially follows a rule but then disregards it, resulting in a negative outcome. In contrast, Story B involves a character (Jenny) who initially follows a rule but then disregards it, leading to a negative outcome." — the "structure" stated for A is reproduced almost verbatim for B, so the schema is tautological and does no discriminating work.
- **Level 1:** "Both stories have a similar structure: a character drops an item, loses something else, and ends up without the item." — abstracts away from surface and is sparse, but the compression is lossy: it reduces both candidates to the identical skeleton, collapsing the causal ordering that actually differentiates A from B.
- **Level 2:** "Story A is a better analogy to Story 1, as it maintains the relational structure of the original story, where a third party intervenes to divide the valuable resource and leads to a positive outcome." — captures the load-bearing relational schema correctly, but still carries surface restatement of the specific characters and objects beyond the role-level schema.
- **Level 3:** "In Story 1, Peter's decision to tan himself in the sun without understanding the risks leads to severe consequences, causing him to leave Florida and return to Iceland." — extracts the load-bearing causal schema (a risky decision made without understanding the risks, a severe consequence, a reversal) and re-expresses each candidate at that same role level, with surface particulars collapsed into schema terms.

B.3.4 Explicit Schema Extraction

Measures: whether the trace explicitly states a reusable, character-independent relational schema that captures Story 1's actual structure.

0 No relational schema extracted. The trace only restates the genre ("analyse the relational structure and compare") or narrates surface events and characters.

1 A vague shared gist or partial relation is mentioned but not framed as a reusable, character-independent schema.

2 A reusable relational schema is stated, but it stays tied to surface tokens, is only partially stated, or names the wrong or over-broad structure for Story 1.

3 A fully articulated, character-independent relational schema is stated and it captures Story 1's actual structure: generalised roles and the causal order among them as a transferable template.

Anchor examples.

- **Level 0:** "Story 1: Mr. Newton, razor, metal, safety feature, incident, alarm, production." — lists surface nouns per story with no generalised template.
- **Level 1:** "Both involve a person hiring someone else to take care of their needs, but not getting what they expected." — a vague one-line shared gist, not framed as a reusable role-mapped schema.
- **Level 2:** "Both involve losing an item, trying to take another person's item, and facing a consequence." — abstracts a token-independent action-chain but is over-broad and partial.
- **Level 3:** "Because it follows the same relational structure: a trigger event leads to a reaction, which results in an action." — a fully articulated, character-independent schema with named generalised roles and a correct causal sequence.

B.4 Digit-Matrix Analogy Rubrics

Digit-matrix analogies require the model to induce a rule R that maps operand cells to the result cell across each row of a number matrix, then apply R to fill the blank. The task reference tells the judge that the correct abstract structure is the operative relation R (e.g. constant, progression by a fixed step, set-union/concatenation, logical AND/XOR, or a composition of these) together with the operand-versus-result assignment and the axis the rule runs along.

B.4.1 Type/Role Binding

Measures: whether the trace binds the structural roles (which cells are operands versus the to-fill result) and the axis the rule runs along.

0 No structural-role binding. The trace re-lists the matrix cells or restates each row without assigning any cell to an abstract role. The matrix axis is not identified.

1 Operand or result cells are identified, but without clear structural roles. The trace points to the right cells but treats them as concrete numbers rather than binding the axis and the operand-versus-result structure.

2 The load-bearing roles are bound and reused, but one binding is vague or implicit: the axis is fixed and cells are mapped to operand versus result, but the operand order, permutation, or per-component assignment is stated imprecisely.

3 The structural roles are bound and used downstream. The trace fixes the axis, cleanly binds which cells are operands versus the to-fill result, and reuses those bindings to fill the blank.

Anchor examples.

- **Level 0:** "The first number in the first set (8) is not directly related to the numbers in the second and third sets." — walks through every cell declaring it "not directly related"; no cell is assigned a structural role.
- **Level 1:** "The relationship could be that the third number is the difference between the first two numbers ($7 - 2 = 5$)." — gestures at an operand→result relation but never commits to a consistent axis.
- **Level 2:** "The third set is formed by combining the first and second numbers in reverse order." — roles bound and reused, but the order or permutation sub-binding is vague and self-contradicted across rows.
- **Level 3:** "The first number in each pair increases by 1: 6." — both load-bearing roles bound and reused downstream: the axis and the per-component operand-roles each drive the final to-fill cell.

B.4.2 Structured Relation Construction

Measures: whether the trace builds a generalisable relation (the rule R) rather than a list of surface values.

0 No rule is constructed. The trace restates cells, may assert "the same rule" or "a pattern exists", but never says what the rule does: no operation, mapping, or dependency between operand and result cells.

1 A rule is implied but its content is left unspecified. The trace names a relational mode ("rearrangement", "mirroring", "a pattern of change") but never states which operation or how operand cells produce the result.

2 A rule is expressed but it is the wrong relation, only partially specified, or stated as concrete per-row arithmetic rather than a generalisable schema.

3 A complete, task-appropriate rule R is explicitly constructed: result = R(operands), shown to hold across example rows, and applied to the blank.

Anchor examples.

- **Level 0:** "The pattern suggests that the third line should follow the same rule as the first two lines." — asserts a rule exists but never states what it does.
- **Level 1:** "One possible approach is to assume that the third number could be the one that completes the sequence in a way that mirrors the rearrangement seen in the first two sets." — a relational mode (rearrangement) is named, but the which and how are never specified.
- **Level 2:** "We can combine the numbers from both sets in a specific way. One possible approach is to add the corresponding numbers from each set." — a genuine operation is expressed (componentwise addition), but it is the wrong relation for this concatenation task.
- **Level 3:** "The third set is simply the concatenation of the first two sets: [8] + [1] = [8 1]." — a complete rule stated as result = concatenation(first, second), verified on example rows.

B.4.3 Schematic Sufficiency

Measures: whether the trace infers a rule that captures the necessary structure and is sufficient to regenerate the matrix, dropping irrelevant surface detail.

0 The trace restates matrix cells verbatim without ever abstracting the relation R, or omits a necessary part of the schema. No usable rule is isolated.

1 The trace drops surface detail and commits to a single compact rule R, but that R mischaracterises or only partially captures the operative relation. The retained schema is too lossy to regenerate the correct result cell.

2 The trace correctly identifies the operative relation R and which cells are operands versus result (sufficient to regenerate the result cell), but still carries unnecessary surface detail such as per-row cell enumeration.

3 The trace states the operative relation R sparsely and sufficiently: result = R(operands) is named once with exactly the elements R needs, irrelevant digit identities and verbatim restatement are dropped, and nothing load-bearing is omitted.

Anchor examples.

- **Level 0:** "The first two lines are identical, with each number repeated three times." — states the answer first, then re-enumerates surface facts about the cells without ever isolating the operative relation as a reusable schema.
- **Level 1:** "The first number in each pair decreases by 4 (8 to 4, then 4 to 0), and the second number remains the same." — compact, but mischaracterises the load-bearing relation (a permuted set operation, not per-position arithmetic) and even uses a different decrement per row, so the retained schema cannot regenerate the result cell.
- **Level 2:** "First row: [8 9] [6 7] [4 5]; the numbers are decreasing by 2 in each subsequent pair: 8 to 6, 6 to 4." — identifies the operative progression relation, sufficient to regenerate the result pair, but restates every row's full cell contents alongside the abstraction.
- **Level 3:** "The third element is the concatenation of the first two elements." — names the operative relation once, assigns operand-versus-result roles, drops every irrelevant digit identity, and applies it directly to the blank.

B.4.4 Explicit Schema Extraction

Measures: whether the trace explicitly states the general rule in words.

0 No matrix rule R is extracted. The trace only restates the genre ("this is a pattern problem, identify the relationship") or floats disconnected per-cell observations without committing to any general R.

1 A vague or partial rule is mentioned but not framed as a reusable R. The trace says something like "the pattern seems to be that the numbers in each row are the same", hedged to this specific grid.

2 A reusable rule is stated but stays tied to specific digits, is only partially specified, or clearly names the wrong relation for this rule type. A clearly named but wrong R caps here.

3 A fully articulated, reusable R is named or framed as a general schema capturing the task's true structure (e.g. "the third cell is the concatenation of the first two cells" for set-union, or "the first component decreases by 1 and the second increases by 1 across the row" for a progression composition).

Anchor examples.

- **Level 0:** "The first number increases from 1 to 2, which is an increase of 1. The second number increases from 2 to 6, which is an increase of 4. Given these observations, let's hypothesize the next pair." — floats disconnected per-cell deltas that differ row to row and only hypothesises a next pair inline; no token-independent rule is ever committed to.
- **Level 1:** "The pattern seems to be that the numbers in each row are identical, and the third number in the third row is missing." — a partial rule, hedged ("seems to be") and bound to this specific grid rather than framed as a reusable relation.
- **Level 2:** "Here, the first two elements of each triplet are increasing by 1, while the third element remains the same." — clearly articulated and reusable in form, but it names the wrong transformation (the true relation is set intersection), so it caps at 2.
- **Level 3:** "The first number decreases by 1 each time (5, 4, 3). The second number increases by 1 each time (7, 8, 9)." — a fully articulated, token-independent rule, the actual operative composition for this matrix, applied to the blank by the same mapping.

Correctness coupling. As with the letter-string rubrics, rule correctness gates only the 2-versus-3 boundary: a clearly articulated but wrong rule R caps at 2; only a rule capturing the task's true structure reaches 3.

C Case studies: base vs RL chain-of-thought traces

Each example below shows a base-model trace and an RL-trained trace on the same item from Qwen3-8B (DeepScaleR). Per-metric scores are from both judges (0–3).

C.1 Letter-string analogy

This item comes from a permuted-alphabet condition at permutation depth 2: the prompt instructs the model to use the ordering below, in which `c` and `f` have swapped places.

```
Alphabet:  a b f d e c g h i j k l m n o p q r s t u v w x y z
[d b f a e] [a b f d e] / [j i h k l] [ ?  ]
```

Base model

The first pair of brackets is [d b f a e]. The second pair of brackets is [a b f d e]. The third pair of brackets is [j i h k l]. The fourth pair of brackets is [?]. The first letter in the fourth pair of brackets is j. The second letter in the fourth pair of brackets is i. The third letter in the fourth pair of brackets is h. The fourth letter in the fourth pair of brackets is k. The fifth letter in the fourth pair of brackets is l. The answer is: Answer: j i h k l.

RL model

To solve this problem, we need to understand the given alphabet order and apply it to the analogy provided. The alphabet order is: a b f d e c g h i j k l m n o p q r s t u v w x y z
Step 1: Analyze the first analogy. The first analogy is [d b f a e] [a b f d e]. Here, the first set of letters is [d b f a e], and the second set is [a b f d e]. We need to determine the relationship between these two sets. In the first set, the letters are in the order: d, b, f, a, e. In the second set, the letters are in the order: a, b, f, d, e. The relationship here is that the second set is the first set with the letters rearranged in a specific order. Specifically, the letters are sorted in ascending order based on the given alphabet.
Step 2: Apply the same logic to the second analogy. The second analogy is [j i h k l] [?]. Here, the first set of letters is [j i h k l], and we need to find the second set by sorting the letters in ascending order based on the given alphabet. In the first set, the letters are in the order: j, i, h, k, l. Sorting these letters in ascending order based on the given alphabet gives us: h, i, j, k, l.
Step 3: Write the final answer. The second set of letters is [h i j k l].
Answer: h i j k l

	Role	Relation	Sufficiency	Schema	Composite
Base (Llama / Qwen)	0 / 0	0 / 0	0 / 0	0 / 0	0.00
RL (Llama / Qwen)	3 / 3	3 / 3	3 / 3	3 / 3	3.00

The base trace never leaves the concrete instance. It reads out each bracket, then reads out the third bracket letter by letter, and copies it unchanged as its answer, which is wrong. None of the seven rule keywords appears anywhere in the trace, so under the mode-entry instrument of Section 5.7 this trace never enters rule mode. The RL trace enters rule mode in its first sentence, declaring intent to understand the given alphabet order and apply it. It then extracts the correct rule (the second set is the first set sorted in ascending order of the given alphabet) and applies it to reach the right answer. The sorted output also shows that the model genuinely used the permuted ordering: under the standard alphabet, sorting d, b, f, a, e would give a b d e f, while the trace produces a b f d e, which is ascending only in the given alphabet, where `f` precedes `d`. Both judges score the base trace 0 and the RL trace 3 on all four metrics.

C.2 Story analogy

The full prompt for this item is reproduced below; the model sees all three stories and must choose which candidate shares the same relational structure as Story 1.

> **Story 1:** Once there was a teacher named Mrs. Jackson who wanted a salary increase. One day, the principal said that he was increasing his own salary by 20 percent. However, he said, there was not enough money to give the teachers a salary increase. When Mrs. Jackson heard this she became so angry that she decided to take revenge. The next day, Mrs. Jackson used gasoline to set fire to the principal's office. Then she went to a bar and got drunk.
>
> **Story A:** McGhee was a sailor who wanted a few days of vacation on land. One day, the captain announced that he would be taking a vacation in the mountains. However, he said, everyone else would have to remain on the ship. After McGhee heard this he became so upset that he decided to get revenge. Within an hour McGhee blew up the captain's cabin with dynamite.
>
> **Story B:** McGhee was a sailor who wanted a few days of vacation on land. One day McGhee became so impatient that he tried to blow up the captain's cabin using dynamite.

After this incident, the captain announced that he would be taking a vacation in the mountains. However, he said, everyone else would have to remain on board to repair the ship.

Which of Story A and Story B is a better analogy to Story 1?

Base model

In Story 1, Mrs. Jackson becomes angry and takes revenge by setting fire to the principal's office. In Story A, McGhee becomes upset and takes revenge by blowing up the captain's cabin. Both stories involve a person taking revenge after being upset by a decision made by someone in authority. In Story B, McGhee's action of blowing up the cabin is not a response to a decision made by someone in authority, but rather an impulsive act. Therefore, Story A is a better analogy to Story 1.
Answer: A

RL model

In Story 1, Mrs. Jackson is a teacher who wants a salary increase but is denied it. She becomes angry and takes revenge by setting fire to the principal's office. In Story A, McGhee is a crew member who becomes upset when the captain announces a vacation and everyone else has to stay on the ship. He takes revenge by blowing up the captain's cabin with dynamite. In Story B, McGhee tries to blow up the captain's cabin using dynamite, and then the captain announces a vacation and everyone else has to stay on board to repair the ship.
Both Story A and Story B share a similar relational structure with Story 1. In all three stories, a person becomes upset due to a decision made by a higher authority and takes revenge by causing damage to the authority's property. However, Story A is a better analogy to Story 1 because the sequence of events is more similar. In Story 1, Mrs. Jackson becomes angry and then takes revenge, while in Story A, McGhee becomes upset and then takes revenge. In Story B, the order of events is reversed, with McGhee taking revenge first and then the captain announcing the vacation.
Answer: A

	Role	Relation	Sufficiency	Schema	Composite
Base (Llama / Qwen)	2 / 2	2 / 2	2 / 2	1 / 2	1.88
RL (Llama / Qwen)	2 / 3	2 / 3	2 / 3	1 / 3	2.38

On role binding, both traces reach a similar level of abstraction — the base writes "a person taking revenge after being upset by someone in authority," which already recasts the characters into abstract roles (both judges score 2). The remaining three metrics show clearer differences. On relation construction, the base notes that both stories "involve a person taking revenge" but leaves the causal structure implicit; the RL trace spells out the full chain: authority decision → upset → revenge on authority's property. On schematic sufficiency, the base carries story-specific details ("setting fire," "blowing up") through its comparison; the RL trace strips these away and operates on the abstract schema alone. On schema extraction, the base never names a reusable pattern; the RL trace articulates one ("a person becomes upset due to a decision made by a higher authority and takes revenge by causing damage to the authority's property") and uses the temporal ordering of events to discriminate between candidates. The Qwen judge treats this elaboration as the 2-to-3 boundary on all four metrics; the Llama judge scores both traces at 2, treating the base's partial structure as sufficient. The RL trace's use of the phrase "relational structure" matches the probe finding in Section 5.7.

C.3 Digit-matrix analogy

```
[3 3 6] [3 4 5] [3 5 7] / [4 4 5] [4 5 7] [4 6 6] / [8 5 7] [8 6 6] [ ?  ]
```

Both traces open with the same rule-finding declaration ("let's analyze the relationships between the cells step by step"), illustrating mode-entry saturation: the opening is identical, so the entry and depth metrics cannot register a difference.

Base model

To solve this analogy problem, let's analyze the relationships between the cells step by step.
Step 1: Observe the first row. [3 3 6]: the first two numbers are the same (3), and the third is 6. [3 4 5]: the first is 3, second is 4, third is 5. [3 5 7]: the first is 3, second is 5, third is 7.
Pattern in the first row: the third number seems to be the sum of the first two minus 1. $3+3-1=5$ (but the third is 6, doesn't fit). Alternatively, $3+3+0=6$ (fits). $3+4+0=7$ (but the third is 5, doesn't fit). The pattern is not consistent.
Step 2: Observe the second row. [4 4 5]: $4+4-1=7$ (but third is 5, doesn't fit). $4+4+0=8$ (doesn't fit). The pattern is not consistent in the second row either.
Step 3: Observe the third row. [8 5 7]: ...

RL model

To solve this analogy problem, let's analyze the relationships between the cells step by step.
Step 1: Observe the first two cells in each row. First row: [3 3 6] and [3 4 5] — the third number decreases by 1 (6 to 5). Second row: [4 4 5] and [4 5 7] — the third number increases by 2 (5 to 7). Third row: [8 5 7] and [8 6 6] — the third number decreases by 1 (7 to 6).
Step 2: Identify the pattern. The changes alternate: decrease by 1, increase by 2, decrease by 1. The next change should be an increase by 2.
Step 3: Apply to the third row. The third number in [8 6 6] is 6. Following the pattern: $6+2=8$.
Answer: [8 6 8]

	Role	Relation	Sufficiency	Schema	Composite
Base (Llama / Qwen)	1 / 1	1 / 1	0 / 0	0 / 0	0.50
RL (Llama / Qwen)	3 / 3	3 / 3	3 / 3	3 / 3	3.00

Despite the identical opening, the two traces differ sharply in what follows. The base trace describes each cell individually and tries arithmetic formulas within single rows ("3+3−1=5, but the third is 6, doesn't fit"), cycling through candidates without settling on a rule. It scores 1 on role binding and relation construction (partial role assignment) but 0 on sufficiency and schema extraction (no reusable rule extracted). The RL trace compares consecutive cells within each row, identifies a cross-row pattern (alternating decrease and increase), and applies it, scoring 3 on every metric. Both traces declare intent to find relationships, but only the RL trace succeeds in constructing a reusable schema. This is the mode-entry saturation described in Section 5.7: on digit-matrix tasks the base model already opens in rule-framing mode, so RL cannot shift the opening; the residual score variance comes from whether the rule search succeeds, not from whether it is attempted.