Beyond 'Aha!': Toward Systematic Meta-Abilities Alignment in Large Reasoning Models

Zhiyuan Hu[1]* **Yibo Wang**[2] **Hanze Dong**[3] **Yuhui Xu**[3]
Amrita Saha[3] **Caiming Xiong**[3] **Bryan Hooi**[1,†] **Junnan Li**[3,†]
[1] National University of Singapore [2] Tsinghua University
[3] Salesforce AI Research

Abstract

Large reasoning models (LRMs) already possess a latent capacity for long chain-of-thought reasoning. Prior work has shown that outcome-based reinforcement learning (RL) can incidentally elicit advanced reasoning behaviors such as self-correction, backtracking, and verification. These phenomena often referred to as the model's "aha moment". However, the timing and consistency of these emergent behaviors remain unpredictable and uncontrollable, limiting the scalability and reliability of LRMs' reasoning capabilities. To address these limitations, we move beyond reliance on prompts and unpredictable "aha moments". Instead, we explicitly align models with three meta-abilities: **deduction, induction, and abduction**, using automatically generated, self-verifiable tasks. Our three-stage pipeline (individual alignment, parameter-space merging, domain-specific reinforcement learning) boosts performance by over 10% relative to instruction-tuned baselines. Furthermore, domain-specific RL from the aligned checkpoint yields an additional gain in performance ceiling for both 7B and 32B models across math, coding, and science benchmarks, showing that explicit meta-ability alignment offers a scalable and dependable foundation for reasoning. Code and data are here[2].

1 Introduction

Large reasoning models, including OpenAI-o1 [11], o3 [17], DeepSeek-R1 [8], Grok 3.5 [29], and Gemini 2.5 Pro [3], have demonstrated remarkable capabilities. These models excel at generating long Chain-of-Thought (CoT) [25] responses when tackling complex tasks and exhibit advanced, reflection-like reasoning behaviors. Recently, DeepSeek-R1 has shown that, starting from pretrained base or instruction-tuned models, pure reinforcement learning (RL) with rule-based rewards can spontaneously lead to the emergence of long CoT reasoning, self-correction, self-reflection, and other advanced behaviors, collectively referred to as the "aha moment". An "aha moment" is usually defined as after a few steps where the model exhibits a sudden performance surge and emergency of reasoning behaviors. Other open-source works, such as SimpleRL-Zoo [33], tinyzero [18], and Logic-RL [30], which attempt to reproduce R1's performance and technical details, have also observed similar aha moments. These behaviors, such as self-correction, self-verification, and backtracking, signal the model's internal experience of strong reasoning ability.

However, relying solely on emergent behaviors is inherently unreliable and difficult to control. Models may fail to consistently manifest these advanced reasoning schemes, which limits both the predictability and scalability of LLM-based reasoning. To overcome this, we propose to explicitly align LLMs with three domain-general reasoning meta-abilities: deduction, induction, and abduction, drawn from Peirce's classical inference triad [19].

*Zhiyuan Hu. Email: zhiyuan$_hu@u.nus.edu$. Corresponding authors (†): Bryan Hooi, Junnan Li.
[2]https://github.com/zhiyuanhubj/Meta-Ability-Alignment

Deduction infers specific outcomes from general rules and hypotheses ($H + R \to O$), enabling rigorous prediction and verification. Induction abstracts rules from repeated co-occurrences ($H + O \to R$), facilitating pattern discovery and generalization. Abduction infers the most plausible explanation for surprising observations ($O + R \to H$), promoting creative and backward reasoning. Together, they form a closed inferential loop for hypothesis generation, testing, and revision, mirroring scientific method and supporting robust and interpretable reasoning.

Together, they form a closed inferential loop for hypothesis generation, testing, and revision, mirroring the scientific method and supporting robust and interpretable reasoning.

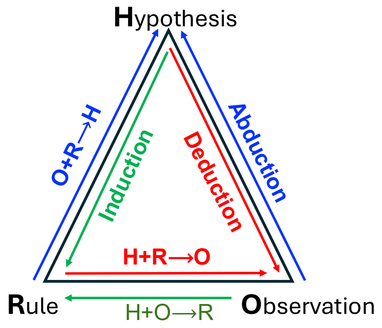


Figure 1: These meta-abilities form a unified reasoning framework.

To operationalize these meta-abilities, we construct a task suite with programmatically generated instances and automatic verifiability. Each task targets one core reasoning mode: Deduction: Propositional satisfiability tasks use rule sets R and candidate hypotheses H to test if all premises entail the observation O. Induction: Masked-sequence completion requires models to infer latent rules R from partial inputs H, O. Abduction: Inverse rule-graph search backchains from observed consequences O through a rule graph R to infer the minimal explanatory H. These tasks are constructed from synthetic distributions that lie out-of-distribution relative to common pretraining corpora, ensuring that gains reflect genuine meta-ability acquisition rather than memorization or shortcut exploitation.

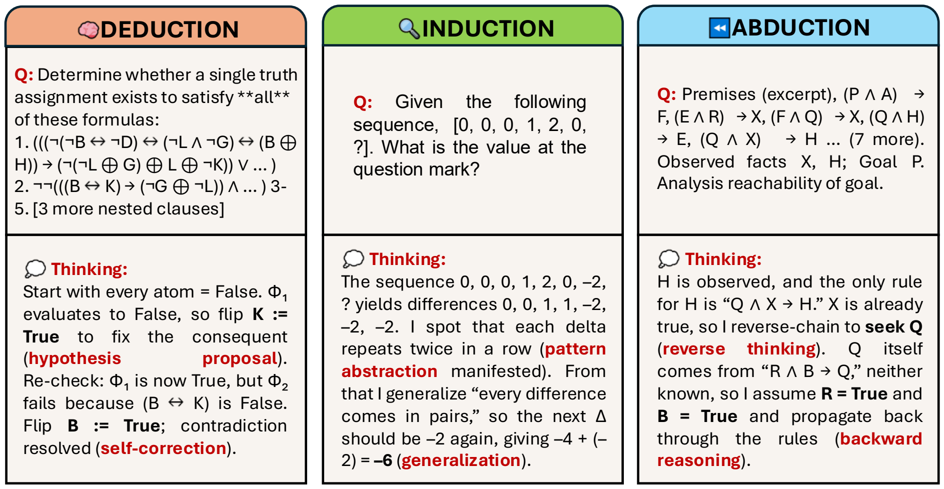
We observe that models aligned to individual meta-abilities make complementary errors. Aggregating their predictions raises overall accuracy by more than 10% relative to a vanilla instruction-tuned baseline. To incorporate the three competencies into a single network, Rather than training the model on a mixed task corpus, we utilize parameter-space model merging to integrate these meta-abilities. Parameter-space merging lifts average accuracy by around 2% at 7B and about 4% at 32B over the instruct baseline, and by 16.3% over the 7B Base model, showing strong generalization of merged meta-abilities. Additionally, Our approach triggers a performance surge and emergent reasoning in around 25 iterations (400 examples), offering more control and efficiency than other methods. More details will be elaborated in Section 5.1 and 5.3.

Furthermore, to evaluate whether meta-ability alignment offers a stronger foundation for subsequent learning, we resumed domain-specific RL training from a checkpoint that had already been aligned and compared it with the same procedure applied to an instruction-tuned model. Starting from the meta-ability checkpoint raises the attainable performance ceiling: after identical continual domain-specific RL training, the model achieves an average gain of about 2% over its instruction-only counterpart. Our key contributions are as follows:

- **Task suite for meta-abilities.** We introduce a novel RL task suite aligned with three classical meta-abilities: deduction, induction, and abduction, each constructed to train and validate domain-general reasoning skills in LLMs.
- **Recipe for reasoning mastery.** We propose a three-stage recipe: (1) independently align models to each meta-ability; (2) merge them via parameter-space integration; (3) fine-tune with domain-specific RL. This leads to improved generalization and downstream task accuracy.
- **Upper-bound boost and scalability.** We empirically demonstrate that meta-ability alignment raises the performance ceiling: our 7B and 32B models show consistent gains over instruction-tuned and base model baselines, across math, coding, and science benchmarks.

2 Related Work

RL-Driven Emergence of Reasoning Abilities Recent studies show that direct RL post-training can unlock long chain-of-thought reasoning beyond what supervised fine-tuning achieves [33, 31, 8]. *SimpleRL-Zoo* [33] proposes a zero-RL recipe using rule-based rewards, boosting math reasoning accuracy and inducing cognitive behaviors like self-verification across diverse base models. *DeepSeek-R1* [8] extends this idea to large-scale training. Its public replications, *Light-R1* [26], *Open-R1* [4],



DEDUCTION

Q: Determine whether a single truth assignment exists to satisfy **all** of these formulas:
1. $(((\neg(\neg B \leftrightarrow \neg D) \leftrightarrow (\neg L \land \neg G) \leftrightarrow (B \oplus H)) \rightarrow (\neg(\neg L \oplus G) \oplus L \oplus \neg K)) \lor \dots)$
2. $\neg\neg(((B \leftrightarrow K) \rightarrow (\neg G \oplus \neg L)) \land \dots)$ 3-5. [3 more nested clauses]

Thinking:
Start with every atom = False. Φ_1 evaluates to False, so flip **K := True** to fix the consequent (**hypothesis proposal**). Re-check: Φ_1 is now True, but Φ_2 fails because ($B \leftrightarrow K$) is False. Flip **B := True**; contradiction resolved (**self-correction**).

INDUCTION

Q: Given the following sequence, [0, 0, 0, 1, 2, 0, ?]. What is the value at the question mark?

Thinking:
The sequence 0, 0, 0, 1, 2, 0, –2, ? yields differences 0, 0, 1, 1, –2, –2, –2. I spot that each delta repeats twice in a row (**pattern abstraction** manifested). From that I generalize "every difference comes in pairs," so the next Δ should be –2 again, giving –4 + (–2) = **–6** (**generalization**).

ABDUCTION

Q: Premises (excerpt), $(P \land A) \rightarrow F$, $(E \land R) \rightarrow X$, $(F \land Q) \rightarrow X$, $(Q \land H) \rightarrow E$, $(Q \land X) \rightarrow H$ ... (7 more). Observed facts X, H; Goal P. Analysis reachability of goal.

Thinking:
H is observed, and the only rule for H is "$Q \land X \rightarrow H$." X is already true, so I reverse-chain to **seek Q** (**reverse thinking**). Q itself comes from "$R \land B \rightarrow Q$," neither known, so I assume **R = True** and **B = True** and propagate back through the rules (**backward reasoning**).

Figure 2: Meta-Ability Alignment Task Example and Corresponding Thought Process

and the minimal-cost *TinyZero* [18], confirm that curriculum schedules, DPO warm-up, and carefully shaped length rewards together yield stronger logical accuracy while keeping compute affordable. Complementary to these general pipelines, *Logic-RL* [30] applies rule-conditioned reinforcement learning to synthetic Knights-and-Knaves puzzles, Enabling transferable logical reasoning for math tasks. Together, these works establish RL as a viable path to large reasoning models.

Advanced reasoning ability In addition to enhancing long-chain reasoning through RL, recent work investigates specific reasoning skills such as self-correction, counterfactual inference, self-verification and others. Chen et al. [2] enhances overall reasoning performance by training models to reason both forward and backward, demonstrating that reverse-thinking objectives can improve forward reasoning as well. Kumar et al. [13] proposes SCoRe, an online RL method where a model iteratively critiques and improves its own answers, bridging the offline self-correction gap. Several recent studies also focus on equipping LLMs with self-verification and self-correction abilities, including ProCo [28], S^2R [14], and SETS [1].

Investigation of Aha-moment RL pipelines often show sudden accuracy jumps. Some concurrent analyses aim to uncover the internal "aha" moments that precede them. Gandhi et al. [5] introduces four reasoning behaviors as diagnostic tools to explain and engineer models' capacity for self-improvement under reinforcement learning. Yang et al. [32] shows that "aha moments" emerge through anthropomorphic language, uncertainty adjustment, and latent-space shifts, helping models avoid reasoning collapse and adapt to problem difficulty. Additionally, Zhou et al. [35] shows that similar emergence occurs in a 2B vision–language model without supervised warm-up.

3 Methodology

3.1 Task Design for Meta-Abilities Alignment

We design three reasoning tasks, each of which involves inferring one element: hypothesis (H), rules (R), or observation (O), given the other two. In *deduction* ($H + R \Rightarrow O$), the model is given a set of logical rules R and a candidate truth assignment H as hypothesis, and must verify whether the overall observation O (i.e., all formulas being true) follows, formulated as a **propositional satisfiability** task. In *induction* ($H + O \Rightarrow R$), the model is provided with observable items O and incomplete inputs H (e.g., masked tokens or implied guesses), and must abstract the underlying generative rule R to correctly complete the sequence, framed as a **masked-sequence completion** task. In *abduction* ($O + R \Rightarrow H$), the model is given observations O and a rule graph R, and must trace backward to recover the minimal set of hidden assumptions H that can logically explain the conclusion, posed as a **reverse rule-graph search** task. This design follows a strict two-known-one-infer schema, clearly ensuring a clean separation of reasoning types, and reformulates all tasks into a unified (H,R,O) triplet format. This enables consistent, comparable, and complementary training signals, systematically

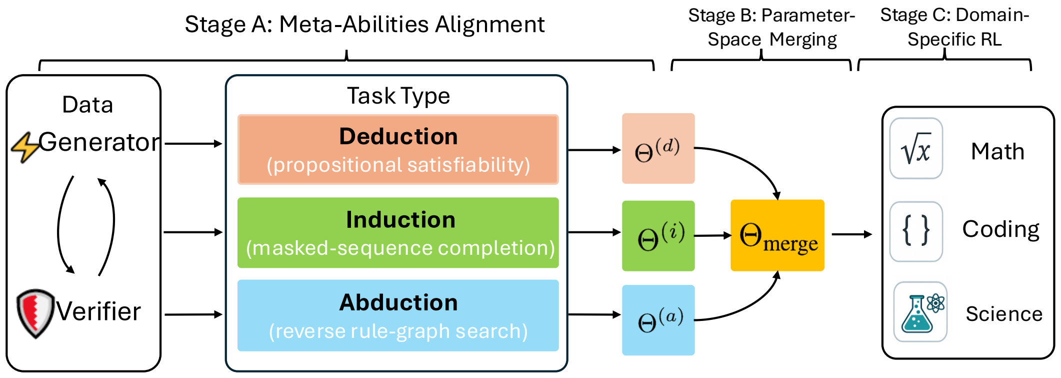

Stage A: Meta-Abilities Alignment
Stage B: Parameter-Space Merging
Stage C: Domain-Specific RL
Data Generator
Verifier
Task Type
Deduction
(propositional satisfiability)
Induction
(masked-sequence completion)
Abduction
(reverse rule-graph search)
$\Theta^{(d)}$
$\Theta^{(i)}$
$\Theta^{(a)}$
Θ_{merge}
Math
Coding
Science

Figure 3: Overview of the three-stage pipeline: align deduction, induction, and abduction specialists, merge them in parameter space, and continually RL-adapt the unified model to downstream domains.

equipping the model with a full range of meta-reasoning capabilities. As illustrated in Figure 3, each instance is produced by an automated *Generator* and screened by a *Verifier*, yielding large-scale, self-checked training data entirely free of manual annotation. The pseudo code for data synthesis and additional task examples are provided in Appendix A.

Deduction As the example shown in Figure 2, we pose task that present a concise cluster of nested propositional clauses involving the standard Boolean operators NOT, AND, OR, IMPLIES, IFF, and XOR; the model must either return a satisfying truth assignment or report that the clauses are unsatisfiable. The task poses a combinatorial explosion of possible variable assignments, with tightly coupled logical formulas such that the value of one variable may indirectly constrain or determine the values of many others through chains of logical dependencies. This interdependence renders purely enumerative or heuristically guessed assignments overwhelmingly unlikely to satisfy all constraints. The only tractable approach is to begin with a provisional assignment, treat each formula as a logical premise, systematically derive its consequences, identify any resulting contradictions, and revise the assignment accordingly. This iterative loop of hypothesis generation, logical consequence propagation, empirical inconsistency detection, and corrective refinement directly instantiates the core structure of deductive reasoning.

Induction To develop inductive reasoning capabilities in models, we design a task for automatically-generated sequence with hidden terms. Each instance presents a series of elements following an undisclosed pattern (including numeric reasoning, symbolic patterns, and multi-step operation cycles) and requires identification of a missing element. This methodology specifically targets induction as the model must extract the underlying regularity governing the visible sequence and apply it to predict unseen values. Inductive learning through such structured sequences enhances the model's fundamental capabilities in abstraction and generalization, which are essential components for robust reasoning across domains. We also provide the example in Figure 2 about induction task and ability behavior for better explanation.

Abduction To cultivate backward reasoning ability, we introduce a reverse rule-graph search task in which forward inference is deliberately obstructed while backward inference remains efficient. Each instance is formulated as a directed rule graph, with atoms as nodes and implications encoded as hyperedges from premise sets to conclusions. Observed facts activate source nodes, while target hypotheses correspond to sink nodes with unknown truth values. By inflating the branching factor in forward chaining, exhaustive exploration becomes computationally infeasible. In contrast, a backward strategy starts from a goal, hypothesizes minimal supporting premises, and verifies them against known facts. This approach can efficiently isolate relevant subgraphs. The design induces repeated cycles of goal-directed hypothesis formation, verification, and revision, thereby fostering the core mechanism of abductive reasoning.

3.2 Training Recipe for Reasoning

Figure 3 sketches how we transform the emergent "aha" moment into *controllable, composable* meta-abilities: we first carry out **Meta-Abilities Alignment**, independently training deduction, induction, and abduction specialists on synthetic diagnostics; we then fuse these specialists through **Parameter-Space Merging** to obtain a single checkpoint that retains their complementary strengths. Finally, **Domain-Specific Reinforcement Learning Training** refines the merged model on domain-specific data such as math, coding, and social dialogue.

3.2.1 Stage A: Meta-Abilities Alignment

We curate three synthetic but diagnostic datasets: *Deduction* (propositional satisfiability), *Induction* (masked-sequence completion), and *Abduction* (reverse rule-graph search). For a policy π_θ, we adopt the Group Relative Policy Optimization (GRPO) objective [21]:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{\sum_g |o_g|} \sum_{i=1}^{G} \sum_{t=1}^{|o_i|} \quad \min\Big\{ r_{i,t}\, \hat{A}_{i,t},\ \text{clip}\big(r_{i,t};\ 1-\epsilon,\ 1+\epsilon\big)\hat{A}_{i,t} \Big\} - \beta\, D_{\text{KL}}\big(\pi_\theta \| \pi_{\text{ref}}\big)$$

where $r_{i,t}$ is the per-token importance weight and π_{ref} is a fixed reference model used to regularize deviations.

Each reward r_i is computed via a rule-based scheme combining **Format Reward** and **Answer Reward**. The Format Reward checks structural compliance using regex-based rules: the model must place reasoning in `<think>` tags and the final answer in `<answer>` tags. A correct format yields $+1$, while any deviation gives -1. The Answer Reward evaluates correctness relative to the task-specific ground truth: a fully correct answer receives $+2$, and an unparseable or missing answer -2. Task-specific criteria guide this evaluation. A deduction (Propositional satisfiability) output is correct only if it satisfies all Boolean formulas; an induction (masked-sequence completion) prediction is valid if the predicted term fits the sequence pattern; and an abduction (inverse rule-graph search) answer is accepted when its premises form the minimal consistent causal path from evidence to target. The total reward is then normalized across the group to produce $\hat{A}_i$.

3.2.2 Stage B: Parameter-Space Merging for Meta-Ability Integration

To unify the strengths of models specialized in distinct meta-abilities, we adopt *parameter-space merging*, which enables: (i) a cost-efficient combination of complementary competencies without additional training, and (ii) a high-quality initialization for domain-specific fine-tuning in Stage C.

We denote the parameters of the deduction-, induction-, and abduction-aligned specialists as $\Theta^{(d)}$, $\Theta^{(i)}$, and $\Theta^{(a)}$, respectively. These models, trained separately on their respective meta-abilities, demonstrate highly complementary behaviors, aggregating their predictions. We construct the merged model Θ_{merge} by linearly interpolating the weights of the three specialists:

$$\Theta_{\text{merge}} = \lambda_d \Theta^{(d)} + \lambda_i \Theta^{(i)} + \lambda_a \Theta^{(a)} \tag{1}$$

We control the contribution of each specialist model via scalar weights λ_d, λ_i, and λ_a. These coefficients determine the relative influence of each meta-ability in the merged model. Notably, uniform weighting is not assumed. Unequal allocation may better reflect the asymmetry in task difficulty or generalization benefit across reasoning modes. Optimal weights are selected empirically based on the performance. In Section 5.2, we compare with nonlinear interpolation methods and assess the robustness of linear interpolation with respect to weight selection.

3.2.3 Stage C: Domain-Specific Reinforcement Learning Training

To evaluate whether meta-ability alignment provides a stronger foundation for downstream learning, we apply reinforcement learning to the aligned checkpoints using domain-specific data, specifically math tasks. For a fair and controlled comparison with instruction-tuned baselines, we follow the experimental settings of SimpleRL-Zoo [33]. Specifically, we adopt a rule-based reward function that assigns +1 to correct completions and 0 otherwise, and also use GRPO. These choices match SimpleRL-Zoo and help isolate the impact of initialization, ensuring performance gains arise from meta-ability alignment rather than optimization differences.

4 Experimental Performance

4.1 Experimental Setup

Dataset and Models For each meta-ability task we introduce a specific difficulty-controlling parameter. Thus, we generate multiple difficulty levels for every task and adopt the curriculum learning strategy that trains the model level by level from easy to hard (More details about difficulty control are in Appendix B.). With this schedule, the 7B model converges by Level 2, and its reward does not improve further at higher levels, so we restrict training to Levels 1–2. The 32B model occasionally benefits from Level 3 but shows unstable reward curves. Therefore, we also use only the first two levels for it. We sample 200 instances per task per level for the 7B model and 2000 instances per task per level for the 32B model. For further domain-specific RL training, we adopt the same dataset as SimpleRL-Zoo [33]. The base model and instruct model we used in this work are from Qwen-2.5 series [23].

Evaluation Setup To validate the generalization of these meta-ability, we select 7 benchmarks from math, coding and science domain. In math tasks, we utilize MATH-500 [9], the full AIME 1983–2024 corpus [24], the recent AMC 2023 [15] and AIME 2024 [16] sets and the Olympiad-level OmniMath subset [6] as the evaluation benchmark. LiveCodeBench (LCB) is designed for code generation [12], and GPQA [20] is aimed for graduate-level science QA. For most benchmarks, we report pass@1 results using temperature 0.0 and top-p 1.0. While, for AIME 2024 and AMC 2023, containing fewer problems, we use average accuracy (avg@32), computed 32 samples per problem with temperature 1.0 and top-p 0.95.

Hyperparameter for Three Stage Training We utilize VERL [22] for meta-abilities alignment and continual reinforcement learning, and adopt MERGEKIT [7] for parameter-space merging to integrate distinct meta-abilities. The optimal weighting coefficients are set to $\lambda_d = 1.0$, $\lambda_i = 0.2$, and $\lambda_a = 0.1$. We provide the comprehensive training setup in Appendix C.

Table 1: Performance of meta-ability–aligned models, merged ensembles, and oracle upper bounds on 7 benchmarks at both 7B and 32B parameter scales, illustrating consistent gains from scaling

Size	Model	Math					Coding	Science	Average	
		Math500	AIME	AIME24 (Avg@32)	AMC23 (Avg@32)	Olympic	LCB	GPQA	Math	Overall
7B	Qwen2.5-7B-Instruct	73.0	22.4	10.7	50.8	37.3	25.7	27.2	38.8	35.3
	Deduction-Aligned	75.8	22.6	10.2	51.4	39.3	25.8	31.5	39.9(+1.1)	36.7(+1.4)
	Induction-Aligned	75.0	22.5	**11.8**	52.3	37.5	**27.0**	33.0	39.8(+1.0)	37.0(+1.7)
	Abduction-Aligned	75.2	22.7	11.4	49.1	38.4	26.8	31.9	39.4(+0.8)	36.5(+1.2)
	Merged Model	**77.8**	**22.9**	11.5	**52.3**	**40.4**	26.0	**33.5**	**41.0(+2.2)**	**37.8(+2.5)**
	Oracle Ensemble	85.5	32.1	18.0	67.1	46.7	–	–	49.9(+11.1)	–
32B	Qwen2.5-32B-Instruct	79.8	31.2	15.3	62.7	45.6	39.5	38.0	46.9	44.6
	Deduction-Aligned	83.8	**36.9**	19.4	68.5	**47.4**	**42.1**	**38.6**	51.2(+4.3)	**48.1(+3.5)**
	Induction-Aligned	82.6	34.8	18.6	66.2	45.8	41.7	38.4	49.6(+2.7)	46.9(+2.3)
	Abduction-Aligned	83.8	33.3	17.5	65.9	46.2	41.1	38.6	49.3(+2.4)	46.6(+2.0)
	Merged Model	**84.2**	36.0	**19.7**	**69.5**	46.9	41.8	38.4	**51.3(+4.4)**	**48.1(+3.5)**
	Oracle Ensemble	88.1	43.2	27.3	76.8	53.0	–	–	57.7 (+10.8)	–

4.2 Out-of-Domain Generalization of Meta-Abilities

Table 1 shows that meta-ability alignment, trained solely on synthetic diagnostic tasks, already transfers to seven unseen benchmarks (covering five math evaluations: Math500, AIME, AIME24, AMC23, and Olympic, as well as the LiveCodeBench coding test and the GPQA science QA set). At the 7B scale, the induction-aligned model provides the largest average improvement, lifting the mean score by 1.7% (from 38.8% to 39.8%), whereas the deduction-aligned model yields the largest single math task gain with a 2.8% increase on Math500 (from 73.0% to 75.8%). Integrating the three meta-abilities in the Merged model further raises the overall average by 2.5% (to 37.8%), confirming that the abilities combine constructively, e.g., boosting LCB from 25.7% to 26.0% and GPQA from 27.2% to 33.5%. An Oracle Ensemble (correct if any aligned model succeeds) lifts Math average

Table 2: Comparison of 7B- and 32B-scale baseline instruction models and our domain-specific RL variants across math, code, and science benchmarks. *Domain-RL-Ins* denotes continual domain-specific RL starting from instruction model; *Domain-RL-Meta* applies the same RL schedule but from a meta-ability–merged initialization, yielding a higher attainable performance ceiling.

Size	Model	Math					Coding	Science	Average	
		Math500	AIME	AIME24 (Avg@32)	AMC23 (Avg@32)	Olympic	LCB	GPQA	Math	Overall
7B	Qwen2.5-7B-Instruct	73.0	22.4	10.7	50.8	37.3	**25.7**	27.2	38.8	35.3
	Domain-RL-Ins	78.2	23.6	11.9	53.2	39.3	25.1	33.0	41.2(+2.4)	37.8(+2.5)
	Domain-RL-Meta	**78.8**	**27.7**	**12.6**	**54.7**	**41.0**	25.4	**33.1**	**43.0(+4.2)**	**39.0(+3.7)**
32B	Qwen2.5-32B-Instruct	79.8	31.2	15.3	62.7	45.6	37.5	38.0	46.9	44.6
	Domain-RL-Ins	83.0	36.5	18.6	67.5	46.1	**41.8**	38.2	50.3(+3.4)	47.4(+2.8)
	Domain-RL-Meta	**84.6**	**38.2**	**19.8**	**70.4**	**48.7**	41.6	**38.6**	**52.3(+5.4)**	**48.8(+4.2)**

by 11.1% to 49.9%, highlighting untapped complementarity for better fusion. Additionally, our merged model on 7B size, trained without domain data, tops concurrent AbsoluteZero Reasoner [34] by 2.7% over AZR (Base) and 2.9% over AZR (Coder). While AbsoluteZero scales 768 seeds to 16,384 triplets via 500×384 self-play, our method reaches higher accuracy with only 1,200 examples, underscoring the effectiveness of factorization-plus-interpolation.

Scaling to the 32B model amplifies the pattern: each aligned model surpasses the Qwen2.5-32B-Instruct baseline, yielding a mean gain of 3.1% on the Math-overall metric and 2.6% on the overall average. For reference, the baseline already scores 46.9% on Math-overall and 44.6% overall. Deduction alignment contributes a +4.3% jump on Math-overall (to 51.2%), with peaks on AIME (+5.7%) and AMC23 (+5.8%), plus gains of +2.6% on LiveCodeBench and +0.6% on GPQA. Induction- and abduction-aligned models follow with respective Math-overall gains of +2.7% and +2.4%, and overall average lifts of +2.3% and +2.0%. Additionally, the Merged checkpoint improves the overall average by 3.5%, with a standout 4.4% gain on the Math-average (from 46.9% to 51.3%) and strong single-task performance such as +6.8% on AMC23. A full Oracle Ensemble run shows an additional 10.8% lift in the math average (to 57.7%), indicating that the three reasoning modes remain highly complementary even at larger scale.

4.3 Scalable Gains from Meta-Abilities Alignment

Table 2 shows that embedding meta-ability alignment into the domain-specific RL pipeline lifts performance at both 7B and 32B. For 7B, the instruction baseline scores 38.8 in math and 35.3 overall. Domain-RL-Ins reaches 41.2 and 37.8. Starting RL from the meta-merged checkpoint, Domain-RL-Meta reaches 43.0 and 39.0. Gains are most visible on AIME and Olympic, while code (LCB) and science (GPQA) stay stable or improve slightly.

At 32B, the instruction baseline is 46.9 in math and 44.6 overall. Domain-RL-Ins rises to 50.3 and 47.4, and Domain-RL-Meta to 52.3 and 48.8. Math improves by about seven percent with Domain-RL-Ins and about twelve percent with Domain-RL-Meta, indicating that teaching core reasoning before domain adaptation both raises starting point and increases the return from task-specific fine-tuning.

4.4 Robust gains from base initialization

To verify that our pipeline is not contingent on instruction-tuned checkpoints, we start from a *base* model and train meta-ability specialists, merge them, and optionally apply domain RL. As shown in Table 3, the merged model already delivers large gains over the base, and a light round of domain RL further lifts the ceiling, exceeding a strong R1-style baseline (SimpleRL-Zoo) on most metrics.

5 Analysis

5.1 Dose LRM really learn meta-abilities?

To further validate whether meta-ability alignment genuinely enhances the expected advanced reasoning abilities, we adopt the cognitive behavior framework proposed by Gandhi et al. [5] and utilize

Table 3: Effectiveness from *base* initialization (7B). Only the average columns report absolute improvements over the base model in parentheses.

Model	**Math**					**Coding**	**Science**	**Average**	
	Math500	**AIME**	**AIME'24** (Avg@32)	**AMC'23** (Avg@32)	**Olympiad**	**LCB** (Coding)	**GPQA** (Science)	**Math Avg.**	**Overall Avg.**
Qwen2.5-7B-Base	44.2	8.8	4.3	26.8	27.7	17.5	23.7	**22.4**	**21.9**
Base-Deduction-Aligned	70.0	18.0	9.8	46.3	36.5	22.8	30.4	36.1(+13.7)	33.4(+11.5)
Base-Induction-Aligned	68.5	17.0	9.5	45.5	35.0	21.8	29.5	35.1(+12.7)	32.4(+10.5)
Base-Abduction-Aligned	67.8	16.8	9.3	45.0	34.6	21.5	29.2	34.7(+12.3)	32.0(+10.1)
Base-Merged Model	73.5	19.2	10.1	47.8	38.0	23.0	31.0	37.7(+15.3)	34.7(+12.8)
SimpleRL-Zoo	75.6	**26.5**	11.7	53.2	38.4	23.3	29.8	41.1(+18.7)	36.9(+15.0)
Base Domain-RL-Meta	**77.4**	26.8	**11.9**	**54.2**	**40.0**	**23.6**	**33.2**	**42.1**(+19.7)	**38.2**(+16.3)

OpenAI o4-mini [10] to identify reasoning-related behaviors. A detailed correspondence between the cognitive behaviors associated with the three meta-abilities, their representative statements and evaluation prompts are provided in Appendix D. We report the proportion of model responses that exhibit these cognitive behaviors across the three meta-abilities.

Relative to the instruction model, the deduction behavior ratio rises from 24.3% to 29.4% in the Deduction-Aligned specialist and remains high at 28.3% in the merged model. Induction increases from 10.2% to 13.1% in the Induction-Aligned specialist, and further to 14.0% in the merged model. Because abductive behaviors are sparse, we report behaviour counts rather than percentages. from 9 in instruct model to 26 in Abduction-Aligned specialist and 18 in the merged model. These trends show that meta-ability alignment reliably amplifies targeted behaviors, while merged model preserves (and for induction, slightly improves) these gains, consistent with our deduction-heavy merge choice.

5.2 Which merging strategy is optimal for meta-ability integration?

Empirical comparison of merge methods. Beyond linear interpolation, we also compare three nonlinear merge families [27], TIES (task-informed sparsified delta add-back), DARE-TIES (density-adaptive, sign-consistent TIES), and Segmented SLERP (layer-wise spherical interpolation with a V-shaped mix). Across Math500, AIME, AIME24, and AMC23, linear interpolation consistently achieves the best scores while being the simplest and most compute-efficient, outperforming the more complex alternatives we tested.

Table 4: Merge method comparison. Linear interpolation (LI below) outperforms nonlinear methods while requiring no extra computing. SS is Segmented SLERP.

Setting	**Math500**	**AIME**	**AIME24**	**AMC23**
TIES	67.0	19.1	9.9	47.9
DARE-TIES	73.0	20.7	10.5	50.2
SS	72.0	20.5	10.1	49.8
LI (ours)	**77.8**	**22.9**	**11.5**	**52.3**

Sensitivity of linear interpolation and choice of best weights. To probe the robustness of linear interpolation, we sweep interpolation weights $(\lambda_d, \lambda_i, \lambda_a)$ over deduction, induction, and abduction specialists. Starting from a neutral average $(0.33, 0.33, 0.33)$, we observe that tilting toward a deduction-heavy mixture reliably improves all metrics; rotating dominance to induction or abduction degrades accuracy. Fixing the deduction expert at full weight ($\lambda_d = 1.0$) and lightly "sprinkling" induction/abduction yielded a broad, stable optimum around $(1.0, 0.20, 0.10)$. This setting matches our qualitative analysis in which deduction contributes the highest-frequency reasoning micro-skills, with induction/abduction adding complementary but lower-frequency behaviors. Importantly, performance varies smoothly around this point, showing stability over knife-edge sensitivity.

Table 5: Linear-merge weight sweep. A deduction-heavy mixture is best. $(1.0, 0.20, 0.10)$ is a simple, robust choice. Qwen2.5-7B-Ins is the Instruct model.

$(\lambda_d, \lambda_i, \lambda_a)$	**Math500**	**AIME**	**AIME24**	**AMC23**
Qwen2.5-7B-Ins	73.0	22.4	10.7	50.8
(0.33, 0.33, 0.33)	74.0	20.8	10.7	50.1
(0.70, 0.15, 0.15)	76.2	22.5	11.4	52.0
(0.15, 0.70, 0.15)	73.8	21.4	11.3	49.1
(0.15, 0.15, 0.70)	73.8	22.9	10.5	49.4
(1.00, 0.10, 0.10)	76.8	23.7	11.2	52.7
(1.00, 0.20, 0.10)	**77.8**	**22.9**	**11.5**	**52.3**

5.3 Empirical Validation of "Aha Moment"

An "aha moment" is typically characterized by a sharp surge in accuracy (such as greater than 5–10%) over just a few training steps on benchmarks, coinciding with the emergence of advanced behaviors such as long chain-of-thought reasoning, self-correction, and backtracking. This phenomenon is also noted by previous work [5], as well as in prior observations from DeepSeek R1, SimpleRL-Zoo, and Logic-RL. Empirically, compared with SimpleRL-Zoo's 40–80 iterations and 10k–20k examples (batch = 256) to hit its accuracy knee, we elicit the inflection and those behaviors in arond 25 iterations with batch = 16 (400 examples), yet still achieve robust gains 5.7% on a 7B model and 9.4% on a 32B model. This demonstrates a faster, data-efficient, and more predictable route to the "aha moment".

6 Conclusion

This work shows LLMs don't have to depend on unpredictable "aha moments." We explicitly align deduction, induction, and abduction via auto-generated, self-verifiable tasks, train specialist models, and merge them into a single checkpoint that beats an instruction baseline by greater 10% on diagnostics and up to 2% across seven math/code/science benchmarks. Starting domain-specific RL from this meta-ability-aligned model adds around 4% and the gains widen from 7B to 32B, indicating scalable benefits. This pipeline also work robustly from base initialization, lifts average accuracy by by 16.3% over the 7B Base model, showing strong generalization of merged meta-abilities. Our method needs no human labels, supports rapid iteration and transparent quality control, and enables finer behavior analysis for interpretability by isolating meta-abilities. Shifting from emergent "aha" to modular, self-verifying training yields reasoning systems that are more predictable and reliable.

7 Limitations

Our work still leaves several open questions. First, the training tasks we use are intentionally symbolic—Boolean formulae, masked sequences, and rule graphs—to guarantee automatic verification, but this controlled design inevitably omits the richness and noise of natural-language arguments, diagrams, or multimodal evidence chains. Extending the suite in those directions would test whether the same alignment signals transfer. Second, although we evaluate on seven established math, coding, and science benchmarks, we have not probed areas such as commonsense dialogue, legal reasoning, or creative writing, which demand different blends of deduction, induction, and abduction; adding such tasks would give a more rounded view of generalization. Finally, we observe that deduction currently contributes the largest share of the overall gain, while abduction lags behind, suggesting either that our backward-reasoning task could be refined or that its reward needs stronger shaping. Addressing these points will not overturn the main findings, but they offer concrete avenues for making explicit meta-ability alignment more robust, broadly applicable, and theoretically grounded.

References

[1] Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and SercanÖ. Arık. Sets: Leveraging self-verification and self-correction for improved test-time scaling. *arXiv preprint arXiv:2501.19306*, 2025. URL `https://arxiv.org/abs/2501.19306`.

[2] Justin Chih-Yao Chen, Zifeng Wang, Hamid Palangi, Rujun Han, Sayna Ebrahimi, Long Le, Vincent Perot, Swaroop Mishra, Mohit Bansal, Chen-Yu Lee, et al. Reverse thinking makes llms stronger reasoners. *arXiv preprint arXiv:2411.19865*, 2024.

[3] Google DeepMind. Gemini 2.5 pro: The latest gemini multimodal model. `https://deepmind.google/technologies/gemini/`, May 2024. Accessed: 2025-05-15.

[4] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

[5] 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. *arXiv preprint arXiv:2503.01307*, 2025.

[6] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omnimath: A universal olympiad level mathematical benchmark for large language models. In *Proc. of ICLR*, 2025.

[7] Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A toolkit for merging large language models. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.36. URL `https://aclanthology.org/2024.emnlp-industry.36`.

[8] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

[10] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[11] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[12] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[13] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[14] Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S^2r: Teaching llms to self-verify and self-correct via reinforcement learning. *arXiv preprint arXiv:2502.12853*, 2025. URL `https://arxiv.org/abs/2502.12853`.

[15] Mathematical Association of America. American Mathematics Competitions (AMC) 2023, 2023.

[16] Mathematical Association of America. American Invitational Mathematics Examination (AIME) 2024, 2024.

[17] OpenAI. Openai o3 and o4-mini system card. 2025.

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

[19] Charles Sanders Peirce. Collected papers of charles sanders peirce, volume 2: Elements of logic. pages Chapter 7, especially paragraphs 619–645. Harvard University Press, 1931. Peirce's formulation of deduction, induction, and abduction as distinct forms of inference.

[20] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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

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

[23] Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2:3, 2024.

[24] Hemish Veeraboina. Aime problem set 1983-2024, 2023. URL `https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024`.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[26] Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025.

[27] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[28] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification. *arXiv preprint arXiv:2405.14092*, 2024. URL `https://arxiv.org/abs/2405.14092`. EMNLP 2024.

[29] xAI. Grok 3.5: Advanced reasoning ai model by xai. `https://grok.x.ai/`, February 2025. Accessed: 2025-05-15.

[30] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[31] Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

[32] Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F Wong, and Di Wang. Understanding aha moments: from external observations to internal mechanisms. *arXiv preprint arXiv:2504.02956*, 2025.

[33] 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. *arXiv preprint arXiv:2503.18892*, 2025.

[34] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

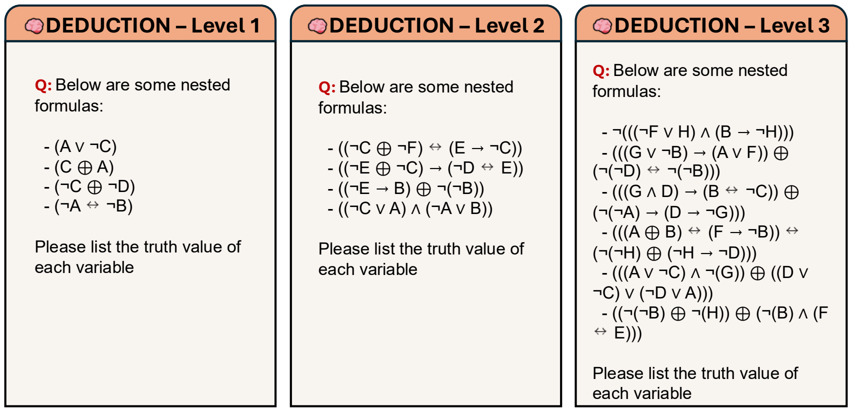
[35] Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*, 2025.

A Data-Synthesis Pseudocode and Additional Task Examples

A.1 Deduction

Algorithm 1: Propositional Satisfiability

```
Input: Clause set $\Phi$ drawn with hyper-parameters $(n_\ell, f_\ell, d_\ell)$ as defined in Appendix B.
Output: A satisfying assignment $\sigma$ or UNSAT.
Function DPLL($\Phi$, $\sigma$):
    if $\forall C \in \Phi$ : $C$ satisfied by $\sigma$ then return $\sigma$
    if $\exists C \in \Phi$ : $C$ falsified by $\sigma$ then return UNSAT
    while $\exists$ unit clause $l$ do
        $\sigma \leftarrow \sigma \cup \{l\}$;
        $\Phi \leftarrow \textsc{Simplify}(\Phi, l)$
    foreach literal $p$ that appears with only one polarity do
        $\sigma \leftarrow \sigma \cup \{p\}$;
        $\Phi \leftarrow \textsc{Simplify}(\Phi, p)$
    pick the first unassigned variable $x$
    for $b \in \{\textsc{True}, \textsc{False}\}$ do
        $\sigma' \leftarrow \sigma \cup \{x{=}b\}$
        $\Phi' \leftarrow \textsc{Simplify}(\Phi, x{=}b)$
        result $\leftarrow$ DPLL($\Phi'$, $\sigma'$)
        if result $\neq$ UNSAT then return result
    return UNSAT
Main:
return DPLL($\Phi$, $\emptyset$)
```


DEDUCTION – Level 1

Q: Below are some nested formulas:

- (A ∨ ¬C)
- (C ⊕ A)
- (¬C ⊕ ¬D)
- (¬A ↔ ¬B)

Please list the truth value of each variable

DEDUCTION – Level 2

Q: Below are some nested formulas:

- ((¬C ⊕ ¬F) ↔ (E → ¬C))
- ((¬E ⊕ ¬C) → (¬D ↔ E))
- ((¬E → B) ⊕ ¬(¬B))
- ((¬C ∨ A) ∧ (¬A ∨ B))

Please list the truth value of each variable

DEDUCTION – Level 3

Q: Below are some nested formulas:

- ¬(((¬F ∨ H) ∧ (B → ¬H)))
- (((G ∨ ¬B) → (A ∨ F)) ⊕ (¬(¬D) ↔ ¬(¬B)))
- (((G ∧ D) → (B ↔ ¬C)) ⊕ (¬(¬A) → (D → ¬G)))
- (((A ⊕ B) ↔ (F → ¬B)) ↔ (¬(¬H) ⊕ (¬H → ¬D)))
- (((A ∨ ¬C) ∧ ¬(G)) ⊕ ((D ∨ ¬C) ∨ (¬D ∨ A)))
- ((¬(¬B) ⊕ ¬(H)) ⊕ (¬(B) ∧ (F ↔ E)))

Please list the truth value of each variable

Figure 4: Examples of propositional satisfiability problems with difficulty levels ranging from 1 to 3.

Figure 4 presents examples of propositional satisfiability tasks at difficulty levels 1 through 3, illustrating how clause structures become increasingly complex. While exhaustive truth-assignment trials quickly become intractable, Algorithm 1 begins with unit propagation and pure-literal elimination to assign all forced literals. It then branches on unassigned variables, backtracking when conflicts arise. This process instantiates the hypothesis–consequence–contradiction–refinement cycle characteristic of deductive reasoning, while preserving an exact satisfaction oracle.

A.2 Induction

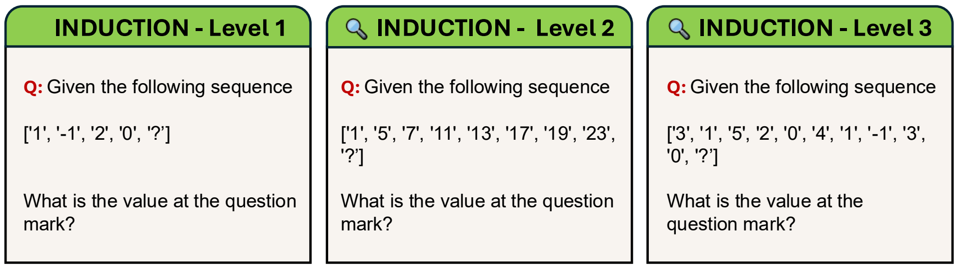
Algorithm 2: Masked-Sequence Completion

```
Input: Observed prefix $S = [s_1, \dots, s_{n-1},\ ?]$; cycle length $k \le 7$; operator alphabet
       $\Sigma \subseteq \{+, -, \times\} \times \{1, \dots, 4\}$.
Output: Predicted missing value
$\mathcal{B} \leftarrow \{(\langle\rangle, \textsc{score} = 0)\}$
for $t \leftarrow 1$ to $k$ do
    if $\mathcal{B} = \varnothing$ then
        return Fail
    end
    $\mathcal{B}' \leftarrow \varnothing$
    foreach candidate $(\vec{o}, \textsc{score}) \in \mathcal{B}$ do
        foreach $op \in \Sigma$ do
            $\vec{o}' \leftarrow \vec{o} \,\|\, op$
            if Consistent$(\vec{o}', S)$ then
                $\textsc{score}' \leftarrow \mathrm{MDL}(\vec{o}')$
                add $(\vec{o}', \textsc{score}')$ to $\mathcal{B}'$
            end
        end
    end
    keep top $B$ elements of $\mathcal{B}'$ by score
    $\mathcal{B} \leftarrow \mathcal{B}'$
end
return value produced by best $\vec{o}$ when applied to $S$
```

Figure 5 shows examples of masked-sequence completion tasks at difficulty levels 1 through 3, each featuring a hidden cycle of arithmetic operations of length k that induces a super-exponential hypothesis space. Algorithm 2 grows candidate operation sequences up to k, prunes those inconsistent with the observed prefix, scores survivors by minimal-description-length, and iteratively refines its beam until the optimal pattern is found, thereby realising the hypothesise–test–refine loop central to inductive reasoning.

A.3 Abduction

Figure 6 shows examples of reverse rule-graph structures at difficulty levels 1 through 3; Algorithm 3 begins at the goal g and recursively traverses candidate edges in reverse, memoizing results and pruning cycles until it either reaches a fact in F or exceeds the depth limit. The task therefore asks: given facts F and a goal g, determine reachability and, if successful, return the minimal explanation tree $\mathcal{T}$ comprising only the utilized hyper-edges. This goal-first, premise-verification loop captures abductive reasoning, while graph depth, branching factor, and distractor rules enable us to scale difficulty yet preserve an exact oracle and dense edge-level supervision for RL agents.


INDUCTION - Level 1

Q: Given the following sequence

['1', '-1', '2', '0', '?']

What is the value at the question mark?

INDUCTION - Level 2

Q: Given the following sequence

['1', '5', '7', '11', '13', '17', '19', '23', '?']

What is the value at the question mark?

INDUCTION - Level 3

Q: Given the following sequence

['3', '1', '5', '2', '0', '4', '1', '-1', '3', '0', '?']

What is the value at the question mark?

Figure 5: Examples of masked-sequence completion with difficulty levels ranging from 1 to 3.

Algorithm 3: Reverse Rule-Graph Search

```
Input: Rule-graph $G = (V, E)$; facts $F \subseteq V$; goal $g \in V$;
chain_depth $d$; distractor_count $h$; cycle_prob $\gamma$ (values sampled as in Appendix B).
Output: TRUE/FALSE and (optional) explanation tree $\mathcal{T}$.
1  Caches: memo Cache, recursion stack Path, edge set $\mathcal{E}$.
2  Function Prove($q$, depth):
3      if $q \in F$ then
4          return TRUE
5      if depth $> d \vee q \in$ Path then
6          return FALSE
7      if Cache[$q$] $\neq$ UNKNOWN then
8          return Cache[$q$]
9      Path $\leftarrow$ Path $\cup \{q\}$
10     foreach $e : (P \to q) \in E$ do                    // candidate hyper-edges
11         if $\forall p \in P$ : Prove($p$, depth$+1$) then
12             $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$
13             Cache[$q$] $\leftarrow$ TRUE; Path $\leftarrow$ Path $\setminus \{q\}$; return TRUE
14     Cache[$q$] $\leftarrow$ FALSE; Path $\leftarrow$ Path $\setminus \{q\}$; return FALSE
15 Main:
16 initialise caches;
17 success $\leftarrow$ Prove($g, 0$);
18 if success then
19     prune $\mathcal{E}$ to minimal tree $\mathcal{T}$
20 else
21     $\mathcal{T} \leftarrow \varnothing$
22 return success, $\mathcal{T}$
```

B Controlling Task Difficulty in Synthesis

We expose *one to four orthogonal hyper-parameters per task* and let the generator sample from progressively wider intervals as the curriculum advances. Each hyper-parameter has a clear algorithmic interpretation, so the reader can verify that higher levels provably enlarge the effective search space.

- **Deduction – Propositional Satisfiability.**
  Only the nested-formula mode is active in the released code. A level ℓ tuple $\langle n_\ell, f_\ell, d_\ell \rangle$ controls:
  - n_ℓ: number of propositional variables (`n_vars` in the script).
  - f_ℓ: number of independent formulas generated per instance.
  - d_ℓ: maximum parenthesis nesting depth (`max_depth`).

  The Boolean assignment space grows as 2^{n_ℓ}, while deeper nesting couples far-apart variables via implications, making local heuristics ineffective. We empirically observe that moving from $(n, d, f) = (6, 2, 3)$ to $(10, 4, 6)$ increases the search time of a basic backtracking solver, which systematically tries possible truth assignments, by two orders of magnitude.
- **Induction – Masked-Sequence Completion.**
  For each level we sample *cycle_length (pattern length, k (1–7))* $k \in \{1, \ldots, 7\}$, which is the number of distinct operations that repeat, the operator alphabet $\Sigma \subseteq \{+, -, \times\} \times \{1, \ldots, 4\}$, and the number of masked positions m (1–2).
  - A longer cycle k means the model must retain a larger latent state to describe the full rule, e.g. $\langle +2, \times 2, -3, +4, \times 2, -5 \rangle$.
  - Mixing symbolic with numeric operators further enlarges the hypothesis set to $|\Sigma|^k$; at $k = 7$ this already exceeds one million templates.

  - Multiple blanks m break the symmetry that a single missing suffix would have, requiring the agent to interpolate rather than extrapolate.

  Hence the difficulty rises super-exponentially in k and m, emphasizing abstraction and generalization over brute-force enumeration.

- **Abduction – Reverse Rule-Graph Search.**

  The generator draws `chain_depth` d, `num_goals` g, `distractor_count` h and `cycle_prob` γ from level-specific ranges, e.g. `d=3-4`, `g=2-3`, `h=5-7`, γ`=0.10-0.25` for level 3.

  - Larger d lengthens the *minimal backward proof*, forcing deeper recursion.
  - Each extra goal g multiplies the number of sink nodes the agent must explain in one episode.
  - Distractors h are additional hyper-edges whose premises share symbols with real rules; they explode the forward branching factor, so blind forward chaining becomes exponentially slower while a goal-directed agent is unaffected.
  - A non-zero cycle probability γ rewires some edges to form backward loops. Proof search must therefore keep a visited set and handle NOT-YETS—a hallmark of abductive reasoning.

  Jointly increasing $\langle d, g, h, \gamma \rangle$ produces a compounded state space whose size we approximate as $\mathcal{O}((b+h)^d)$ for forward search, where b is the intrinsic out-degree of the knowledge graph.

- **Abduction – Reverse Rule-Graph Search.**

  The generator draws `chain_depth` d (maximal backward-proof depth), `num_goals` g (number of sink goals), `distractor_count` h (spurious hyper-edges), and `cycle_prob` γ (edge-rewiring probability) from level-specific ranges, e.g. `d=3-4`, `g=2-3`, `h=5-7`, γ`=0.10-0.25` for level 3.

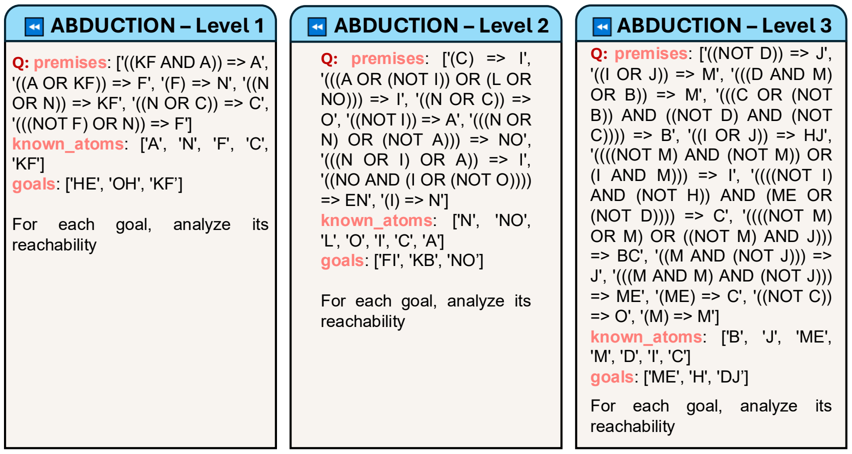
  - Larger d lengthens the *minimal backward proof*, forcing deeper recursion.
  - Each extra goal g multiplies the number of sink nodes the agent must explain in one episode.
  - Distractors h share symbols with true rules, exploding the forward branching factor—blind forward chaining slows exponentially, while a goal-directed agent is unaffected.
  - A non-zero cycle probability γ rewires some edges to form backward loops; proof search must therefore maintain a visited set and handle NOT-YETS, a hallmark of abductive reasoning.

  Jointly increasing $\langle d, g, h, \gamma \rangle$ produces a compounded state space whose size we approximate as $\mathcal{O}((b+h)^d)$ for forward search, where b is the intrinsic out-degree of the knowledge graph.

During curriculum training we feed the model instances in ascending level order (e.g. $d = 2 \rightarrow 7$ for abduction, $n = 6 \rightarrow 10$ for deduction, $k = 1 \rightarrow 7$ for induction), so it first acquires stable heuristics on low-entropy regimes before encountering the full combinatorial explosion.

C Comprehensive Multi-Stage Training Setup

Stage A: Meta-Abilities Alignment We warm-start from a frozen instruction model and train on three synthetic diagnostic corpora—Deduction (propositional SAT), Induction (masked-sequence completion) and Abduction (inverse rule-graph search), using the GRPO. Rollouts are generated with vLLM (temperature 0.7, $n = 4$, tensor-parallel size 2, max tokens $\approx$ 9 000), and optimized with AdamW (LR 5×10^{-7}) over 20 epochs. We use a training batch size of 32, PPO mini-batch size of 256, and PPO micro-batch size of 32. Rewards are normalized per group to produce $\hat{A}_i$, and we regularize via a KL penalty ($\beta = 0.001$, low-variance estimator) to the frozen reference model.



ABDUCTION – Level 1

Q: **premises**: ['((KF AND A)) => A', '((A OR KF)) => F', '(F) => N', '((N OR N)) => KF', '((N OR C)) => C', '(((NOT F) OR N)) => F']
known_atoms: ['A', 'N', 'F', 'C', 'KF']
goals: ['HE', 'OH', 'KF']

For each goal, analyze its reachability

ABDUCTION – Level 2

Q: **premises**: ['(C) => I', '(((A OR (NOT I)) OR (L OR NO))) => I', '((N OR C)) => O', '((NOT I)) => A', '(((N OR N) OR (NOT A))) => NO', '(((N OR I) OR A)) => I', '((NO AND (I OR (NOT O)))) => EN', '(I) => N']
known_atoms: ['N', 'NO', 'L', 'O', 'I', 'C', 'A']
goals: ['FI', 'KB', 'NO']

For each goal, analyze its reachability

ABDUCTION – Level 3

Q: **premises**: ['((NOT D)) => J', '((I OR J)) => M', '(((D AND M) OR B)) => M', '(((C OR (NOT B)) AND ((NOT D) AND (NOT C)))) => B', '((I OR J)) => HJ', '((((NOT M) AND (NOT M)) OR (I AND M))) => I', '((((NOT I) AND (NOT H)) AND (ME OR (NOT D)))) => C', '((((NOT M) OR M) OR ((NOT M) AND J))) => BC', '((M AND (NOT J))) => J', '(((M AND M) AND (NOT J))) => ME', '(ME) => C', '((NOT C)) => O', '(M) => M']
known_atoms: ['B', 'J', 'ME', 'M', 'D', 'I', 'C']
goals: ['ME', 'H', 'DJ']

For each goal, analyze its reachability

Figure 6: Examples of reverse rule-graph search with difficulty levels ranging from 1 to 3.

Stage C: Domain-Specific Reinforcement Learning on Math Starting from the Stage A checkpoint (after MergeKit parameter merging), we fine-tune on the dataset from SimpleRL Zoo using GRPO objective to match SimpleRL-Zoo settings. Inputs are capped at 1 024 + 3 072 tokens; train/val batch size 16; PPO mini-batch 128, micro per-GPU 2; rollouts with $n = 8$, temperature 1.0, TP size 2. Rewards are rule-based (+1 for correct, 0 otherwise) without format terms to isolate transfer effects. We use AdamW (LR 5×10^{-7}), clip ratio 0.2, entropy bonus 0.001, and KL to reference ($\beta = 0.001$).

D Quantifying Meta-Ability Behaviors through Targeted Prompts

Below we present the three carefully crafted sentence-extraction prompts. One each for Deduction, Induction and Abduction, that we feed to the model to capture its core reasoning behaviors. Each prompt defines the target micro-skills, constrains the extraction to only the relevant sentences, and enforces a strict JSON output format so that we can automatically compute the frequency of each reasoning mode. These prompts form the basis for quantifying how often the model engages in each type of advanced inference.

Deduction Prompt

```
Here is a thought record:

<thinking text>

We define ``Deduction'' to consist of exactly two sub-skills:
  1. HypothesisProposing: sentences that explicitly introduce a
     testable hypothesis or scenario, such as:
     * ``Assume <\dots>, then <\dots>.''
     * ``Case <number>: <\dots>.''
     * ``Suppose <\dots>, then <\dots>.''

  2. SelfVerification/SelfCorrection: sentences that explicitly
     validate or correct a prior result, such as:
     * ``After checking, <\dots>, so <\dots>.''
     * ``We see that <\dots> was wrong, so <\dots>.''
     * ``This shows an error; we must <\dots>.''
     * ``Correcting that, <\dots>.''

Extraction requirements:
```

```
- Extract **all and only** the sentences that clearly match one
    of the above patterns or equivalent wording that directly
    reflects the two sub-skills.
- Do **not** extract any explanatory, descriptive, or purely
    procedural sentences.
- Return a JSON array; each element must be exactly:
  {
    "category": "HypothesisProposing"|"SelfVerification/
        SelfCorrection",
    "sentence": "<the exact extracted sentence>"
  }
- After the array, output one more field:
  "total_count": <the total number of extracted sentences>

Please **strictly** follow this format without any additional
    commentary.
```

Induction Prompt

```
You are an expert reasoning analyst.

TASK: Induction Sentence Extraction (Wide-Net, English Only)
-----------------------------------
1. Input text is wrapped between THINKING: ... END THINKING.
2. Scan every sentence (and its immediately preceding sentence)
    and extract only those that
   state or imply a general rule, trend, or pattern derived from
        multiple cases or examples.
   Key English triggers include words or phrases such as:
     * therefore, thus, hence, it follows that
     * we observe that, observations show, we see that
     * typically, usually, in general, most, majority
     * suggests that, implies that, indicates that
     * pattern, trend, rule, conjecture, generalize
     * for all n, for any k, if ... then ...

   Extraction is OK for probabilistic statements (e.g.\ ‘‘Most A
        are B’’) and recursive claims
   (e.g.\ ‘‘If P(k) holds then P(k+1) holds, so P(n) for all n
      ’’).
3. Do NOT extract purely numerical computations, single-step
    deductions, or abduction sentences.

OUTPUT FORMAT (nothing else):
-----------------------------------
First, a JSON array of the exact extracted sentences as strings.
     Then, on the next line:
"total_count": <number_of_sentences>

Example:
[
  "For any $n\\equiv 0\\pmod{5}$, the position is losing.",
  "Thus the sequence converges to zero for all starting values."
]
"total_count": 2

THINKING:
<thinking text>
END THINKING
```

```
Remember: output valid JSON and the total_count line, and
    nothing else.
```

Abduction Prompt

```
You are an expert reasoning analyst.

**Task**
1. I will give you a block of text labelled
   THINKING: ... END THINKING.
2. Scan every sentence and extract **only** those that exhibit
    **abduction (backward reasoning)**.
3. Exclude any sentences that use forward reasoning (deduction
    or induction) or are purely descriptive/mathematical.
4. Return a JSON array; each element must be exactly:
   {
     "sentence": "<the exact extracted sentence>",
     "match_type": "template" | "close_paraphrase" | "
         clear_abduction_non_template",
     "matched_pattern": "<template code or empty>"
   }
5. After the array, output a final field on a **new line**:
   "total_count": <the total number of extracted sentences>

**Abduction vs.\ Forward Reasoning**
- **Abduction (backward reasoning)** starts from an observation
    or surprising fact and proposes or tests a hypothesis/
    explanation.
- **Forward reasoning** (deduction / induction) starts from
    premises or rules and derives consequences --- **do not
    extract** those.

**Abduction Templates (code: exact pattern)**
T1 Given-that -> ``Given that <observed fact>, it must be that <
    hypothesis>.''
T2 Since-result -> ``Since <result> holds, one explanation is <
    hypothesis>.''
T3 To-account-for -> ``To account for <outcome>, suppose <
    hypothesis>.''
T4 Observing-suggests -> ``Observing <phenomenon> suggests <
    hypothesis>.''
T5 Because-infer -> ``Because <result>, we infer <hypothesis>.''
T6 Hypothesis-test-revise -> ``Testing that hypothesis, we find
    <contradiction>, so <revision>.''
T7 Bad-consequence-fails -> ``However, this would imply <bad
    consequence>, thus the explanation fails.''
T8 Closer-inspection -> ``On closer inspection, <hypothesis>
    does not hold, so <revision>.''

**Close Paraphrase Examples**
- ``One possible cause of X is Y.''
- ``A plausible explanation for X could be Y.''
- ``If X happened, then Y might be responsible.''
- ``X seems best explained by Y.''
- ``Assuming Y, we can explain X.''
- ``Y predicts X; therefore, Y is likely.''
- ``Y would predict Z, yet we observe not-Z; hence Y is unlikely
    .''

**Strict Extraction Rules**
```

```
* Extract **all and only** sentences performing abduction as
    defined above.
* Exclude any forward-reasoning or purely procedural/
    mathematical sentences.
* If a sentence matches a template **exactly**, set `"match_type
    ":"template"` and `"matched_pattern"` to its code (e.g.\ `"T1
    "`).
* If it is a tight paraphrase (<= 3 word changes per clause),
    set `"match_type":"close_paraphrase"`.
* Otherwise, if it clearly does abduction but is not a template
    or close paraphrase, set `"match_type":"
    clear_abduction_non_template"` and leave `"matched_pattern"`
    blank.
* Output must be valid JSON with **no** extra keys or
    explanatory text.

THINKING:
<thinking text>
END THINKING

Please **strictly** follow the specified output format and
    include nothing else.
```