LATENT ROLE SEPARATION FOR CONTINUOUS COT VIA HIERARCHICAL LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous chain-of-thought (CoT) compresses a verbal trace into a short sequence of vectors. Those vectors must both discover a decomposition and carry out the computation, yet existing methods leave their computational roles unspecified. We introduce *Hierarchical Latent Reasoning* (HLR). A small student first emits one plan vector and then K detail vectors. Every detail is generated from a prefix that already contains the plan, so the plan-to-detail edge is present in the student forward pass and remains active at inference. The projected hierarchy is read by a 7B teacher. Training replaces gold plan and detail text from left to right with the corresponding projected student states, aligns each latent to its role, and scores one answer path per example. We compare HLR with six latent-reasoning baselines on two 7B backbones and five reasoning benchmarks, under a shared five-slot budget for every continuous method. Under this protocol, HLR reaches state-of-the-art accuracy. The gains are largest where one strategy is reused across successive state updates, so the plan stays available while the details execute. In the hierarchy ablation, removing the plan-to-detail edge accounts for the larger drop, and replacing the exclusive answer path accounts for a smaller one. CODI and SemCoT lead the remaining Mistral cells.

1 INTRODUCTION

Chain-of-thought (CoT) improves language-model reasoning by externalizing intermediate computation as text (Wei et al., 2022; Kojima et al., 2022). Its cost grows with trace length, and every intermediate state must pass through a discrete vocabulary. Continuous CoT removes that bottleneck: a model reasons through vectors that are consumed as pseudo-tokens rather than decoded as words (Hao et al., 2024; Shen et al., 2025; Xu et al., 2025; **?**; Wei et al., 2025). This creates a compact internal channel without specifying how a short latent sequence should divide planning from execution.

Current methods leave this division implicit. Coconut, CODI, and SoftCoT use different latent generators, but none assigns persistent roles to latent positions. With only a handful of vectors, the model must learn both *how to organize the trace* and *what computation to perform*. The burden is especially clear in problems that reuse one strategy across several updates. Solving a CoinFlip instance, for example, requires a stable rule—initialize the coin state and process each action in order—followed by a sequence of state transitions. A flat latent bank can represent this decomposition, but nothing in its computation graph requires it to do so.

Hierarchical Latent Reasoning (HLR) makes the decomposition explicit. A small student first produces a plan latent $\mathbf{h}^{\text{plan}}$, then generates K detail latents $\{\mathbf{h}_i^{\text{det}}\}_{i=1}^{K}$. Every detail is conditioned on the plan and all preceding details in the student's hidden space. The result is an ordered hierarchy with a direct strategy-to-execution edge at every step, and the hierarchy is retained at inference. The projected hierarchy is consumed by a 7B teacher whose base weights remain frozen; LoRA learns to read the continuous inputs. Figure 1 contrasts this graph with an equally sized flat bank.

The training procedure gives these roles semantic content. Starting from gold plan and detail text, we replace hierarchy positions from left to right with the corresponding projected student states, following the text-to-latent curriculum of Coconut (Hao et al., 2024). Positional, pooled, and relational objectives align the latent hierarchy with frozen-teacher encodings of the same gold structure. An exclusive-path objective scores either the curriculum input or the full latent bank on each example,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

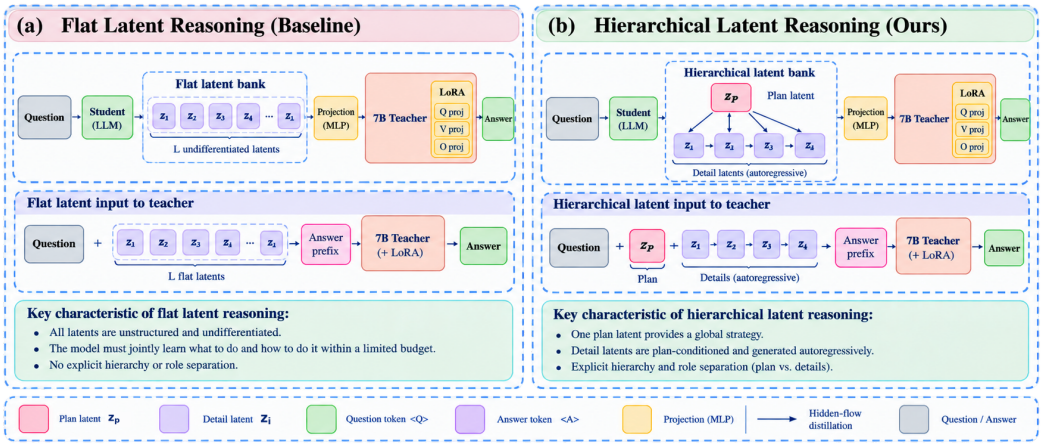

Figure 1: Flat versus hierarchical latent reasoning under the same student–teacher interface. (a) A flat latent bank emits L undifferentiated vectors; the model must jointly learn *what* to do and *how* to do it inside that budget. (b) HLR writes the split into the graph: one plan latent $\mathbf{h}^{\text{plan}}$ carries a global strategy, then K detail latents are generated autoregressively and always conditioned on the plan. Both layouts are projected into a frozen 7B teacher adapted with LoRA. The hierarchy is still present at inference.

avoiding duplicate answer supervision. A counterfactual utilization hinge penalizes a bank that fails to improve on an empty-prefix input.

We compare HLR with six implicit-reasoning baselines, including discrete iCoT-SI, on CoinFlip, CommonsenseQA, GSM8K, MultiArith, and SVAMP, using Qwen2.5 and Mistral teacher–student pairs. Data splits, random seeds, the 30-token answer budget, and the decoding-temperature grid are shared; every continuous method receives five latent slots. The largest margins are 12.0 points on Qwen CoinFlip and 20.3 points on Mistral MultiArith. On Qwen CoinFlip, the hierarchy ablation lowers accuracy from 87.3 to 78.5, and removing exclusive-path training lowers it to 83.3.

Contributions. (i) We introduce a role-separated continuous-CoT architecture in which one plan latent is a direct ancestor of every autoregressive detail latent; the hierarchy remains active at inference. (ii) We develop a text-to-latent curriculum that combines role-aligned representation targets, exclusive-path answer supervision, and an explicit utilization objective. (iii) We evaluate the complete systems of six baselines across two backbones and five benchmarks under shared data, seeds, latent slots, and answer budgets, and ablate the hierarchy and exclusive-path training within HLR.

2 Related Work

Explicit CoT. Prompting or fine-tuning a model to emit intermediate tokens (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023) exposes a verbal trace that can be supervised and inspected. Its inference cost scales with trace length, and every reasoning state must be represented as text.

Discrete extra compute. Pause tokens (Goyal et al., 2024) insert a fixed number of extra symbols before the answer, increasing computation without assigning semantics to the inserted positions. iCoT (Deng et al., 2023) distills teacher hidden states into layer-wise computation, whereas iCoT-SI (Deng et al., 2024) progressively removes gold CoT tokens from the input. We include iCoT-SI as a discrete internalization baseline; Pause provides related motivation but does not learn a continuous latent trace.

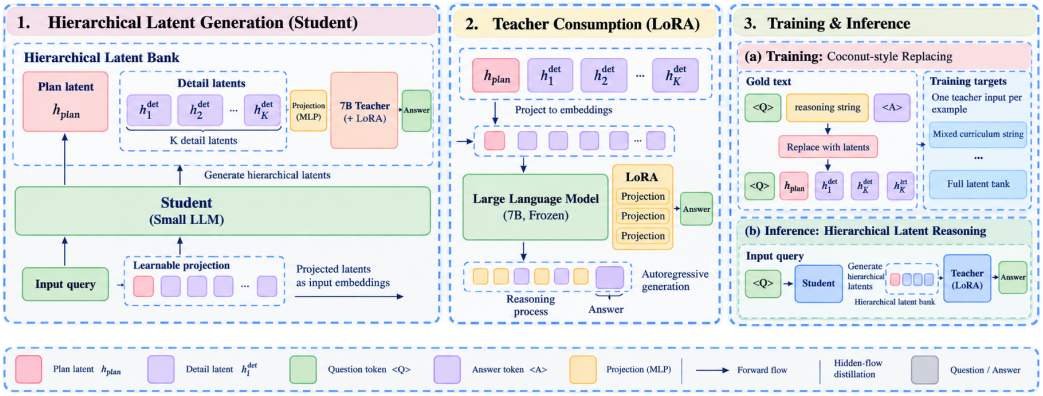

Figure 2: Implementation of the hierarchical bank in Figure 1(b). (1) A small student emits one plan latent and K plan-conditioned detail latents. (2) The bank is projected into the frozen 7B teacher, which is adapted with LoRA. (3) Training replaces gold hierarchy text left to right; inference keeps the learned $1+K$ bank and decodes the answer with no plan–detail text.

Continuous latent CoT. Coconut (Hao et al., 2024) recycles the last hidden state as the next latent while progressively replacing verbal CoT with continuous states. CODI (Shen et al., 2025) distills a teacher trace into a compact code. SoftCoT (Xu et al., 2025) trains a small model to produce soft prompts for a frozen LLM, and SemCoT (**?**) aligns contemplation tokens with a sentence encoder. SIM-CoT (Wei et al., 2025) adds a step decoder to Coconut during training; the decoder is removed at inference. These methods introduce distinct generators and supervision signals, but none assigns persistent strategy and execution roles to latent positions. HLR retains the student–teacher interface of soft-prompt methods and the progressive replacement principle of Coconut, while changing the latent computation graph. In our comparison, Coconut is capped at one continuous state per internalized step and five states in total, the same inference-time latent budget as HLR.

Hierarchy outside the latent path. Tree-of-thought prompting (Yao et al., 2023) imposes structure in text, not in the vectors a model uses as internal compute. Representational arguments separate transformer reasoning tasks by the depth and width required to solve them (Sanford et al., 2024). HLR places hierarchy in the latent path: the plan is computed first and remains an ancestor of every subsequent detail state.

3 Method

3.1 Architecture

HLR consists of a latent generator and an answer model connected by a one-way continuous interface. A student language model π_ϕ (Qwen2.5-0.5B-Instruct or Mistral-1.1B) constructs the latent hierarchy in width d_s. A projection maps the hierarchy to the embedding space of a 7B teacher θ (Qwen2.5-7B-Instruct or Mistral-7B-Instruct-v0.2). The teacher's base weights remain frozen, and LoRA learns to consume the latent inputs. The student–teacher channel transmits vectors only: the student never decodes text for the teacher (Hu et al., 2022). During progressive replacement, positions not yet supplied by the student remain gold text embeddings. Figure 2 summarizes generation, projection, and the train–test transition.

Plan-conditioned generation. We add plan and detail tokens to the student vocabulary. Let $E(q) \in \mathbb{R}^{T\times d_s}$ denote the embedded question. The hidden state at the plan token defines the global plan:

$$\mathbf{h}^{\text{plan}} = \text{LastHidden}\big(\pi_\phi([E(q); \mathbf{e}_{\langle\text{plan}\rangle}])\big) \in \mathbb{R}^{d_s}. \tag{1}$$

The student then extends a prefix containing the plan and all earlier details. Detail generation operates entirely in student space:

$$\mathbf{h}_i^{\text{det}} = \text{LastHidden}\big(\pi_\phi([E(q); \mathbf{h}^{\text{plan}}; \mathbf{h}_1^{\text{det}}; \cdots; \mathbf{h}_{i-1}^{\text{det}}; \mathbf{e}_{\langle\text{detail}\rangle}])\big). \tag{2}$$

Equations 1–2 induce

$$\text{pa}(\mathbf{h}^{\text{plan}}) = \{q\}, \qquad \text{pa}(\mathbf{h}_i^{\text{det}}) = \{q, \mathbf{h}^{\text{plan}}, \mathbf{h}_{<i}^{\text{det}}\}. \tag{3}$$

$\mathcal{H}_\phi(q) = (\mathbf{h}^{\text{plan}}, \mathbf{h}_{1:K}^{\text{det}})$ is an ordered hierarchy in which every detail depends directly on the plan as well as on the preceding detail trajectory. We unroll $K{=}4$ detail steps, yielding the same five-position budget used for all continuous baselines. A linear head on $\mathbf{h}^{\text{plan}}$ predicts an index in $\{0, \ldots, K{-}1\}$, corresponding to 1 through K details. Under the fixed-budget protocol, this head acts as a regularizer trained toward the K-detail class; all five positions remain active at test time.

Projection into the teacher. A float32 two-layer MLP $g_\phi : \mathbb{R}^{d_s} \to \mathbb{R}^{d_t}$, with GELU and output LayerNorm, maps each student state to the teacher width:

$$\mathbf{Z}_\phi(q) = \begin{bmatrix} g_\phi(\mathbf{h}^{\text{plan}}) \\ g_\phi(\mathbf{h}_1^{\text{det}}) \\ \vdots \\ g_\phi(\mathbf{h}_K^{\text{det}}) \end{bmatrix} \in \mathbb{R}^{(K+1)\times d_t}. \tag{4}$$

We write its rows as $\tilde{\mathbf{h}}^{\text{plan}}, \tilde{\mathbf{h}}_{1:K}^{\text{det}}$. The teacher then sees

$$\mathbf{X}_{\text{full}} = [E_\theta(q);\ \mathbf{Z}_\phi(q);\ E_\theta(p);\ E_\theta(a)] \tag{5}$$

where p is a model-specific answer prefix—for Qwen, an assistant turn that begins with "Answer:"; for Mistral, the instruction closer after "Answer:"—and a is the gold answer during training. The prefix forms are in Appendix B. At inference, a is replaced by decoding of at most 30 tokens. The end-to-end answer model factorizes as

$$p_{\theta,\phi}(a \mid q) = \prod_{t=1}^{|a|} p_\theta(a_t \mid E_\theta(q), \mathbf{Z}_\phi(q), E_\theta(p), a_{<t})\,. \tag{6}$$

Coconut, CODI, SIM-CoT, iCoT-SI, and HLR adapt the answer model with LoRA. SemCoT trains its contemplation generator and projection against a frozen 7B teacher; SoftCoT trains its projection. iCoT-SI has no latent slots and uses the same answer-token cap.

3.2 Curriculum and Answer Paths

Training proceeds from an explicit hierarchy to a fully continuous one. Following Coconut's progressive replacement principle (Hao et al., 2024), we begin with gold hierarchy text and substitute latent positions from left to right. The answer loss uses one sampled input path per example: either the mixed curriculum block or the full latent bank. Table 1 gives the schedule shared by all datasets and both backbones. These stage lengths are the training procedure of HLR. Baseline epoch budgets are likewise the training procedures of those methods (Appendix B), not a budget assigned so that HLR trains longer.

Warmup ($c{=}0$). Gold text occupies all hierarchy positions while g_ϕ, the depth head, and teacher LoRA are trained. The student backbone remains frozen, and the full-latent path is disabled.

Progressive replacement ($c{=}1 \ldots K$). At stage c, the first c positions of the hierarchy block (plan first, then details in order) are projected latent embeddings $\tilde{\mathbf{h}}^*$ in teacher space; the remaining $K{+}1{-}c$ positions stay gold text. The question, the answer prefix, and the answer in Equation 5 are left as text. If an annotation has more than K details, only the first K are kept. If it has fewer, the last kept detail is repeated so the block still has K detail positions. Let $y_0 = P$, let y_j ($j \geq 1$) be that truncated or repeated detail string, and let $\mathbf{z}_j$ be row j of Equation 4. The stage-c hierarchy block is

$$\mathbf{R}_c = [\mathbf{r}_0^{(c)}; \ldots; \mathbf{r}_K^{(c)}], \qquad \mathbf{r}_j^{(c)} = \begin{cases} \mathbf{z}_j, & j < c, \\ E_\theta(y_j), & j \geq c. \end{cases} \tag{7}$$

Hence $c = 0$ is fully textual and $c = K + 1$ is $\mathbf{Z}_\phi(q)$.

Table 1: HLR training schedule (K=4). "Latent slots c" is how many of the $1+K$ hierarchy positions are continuous; the rest are gold strings "Plan: ..." and "Detail i: ...". The all-latent entry is the mean epoch count over the reported HLR runs. Head LR 10^{-4}, backbone LR 10^{-6}, teacher LoRA LR 10^{-5}; AdamW; gradient clip 1.0; effective batch size 4 by accumulating single-example steps.

Stage	Latent slots c	Epochs	Student backbone
Explicit-hierarchy warmup	0 (all text)	3	frozen
Progressive replacement	$c = 1, 2, 3, 4$	2 each	unfrozen
All-latent fine-tuning	$c = 5$	8 (mean)	unfrozen

Exclusive-path answer CE. At each replacement-stage update, the answer loss is computed on one of two inputs: the current mixed curriculum block or the complete latent bank. The complete bank is selected with probability p=0.4; only this path receives the utilization loss. The mixed block is selected otherwise. Warmup uses the all-text input, and the final stage always uses the complete bank. We fix p across datasets and backbones. With $\mathbf{X}_c = [E_\theta(q); \mathbf{R}_c; E_\theta(p); E_\theta(a)]$, the sampled teacher input is

$$\mathbf{X}_{\text{train}} \sim \begin{cases} \mathbf{X}_0, & c = 0, \\ \mathbf{X}_{K+1} \text{ with prob. } p, \quad \mathbf{X}_c \text{ with prob. } 1-p, & 1 \le c \le K, \\ \mathbf{X}_{K+1}, & c = K+1. \end{cases} \tag{8}$$

All-latent fine-tuning (c=K+1). The final stage trains the full stack using only the latent bank. Across the reported HLR runs, this stage lasts 8 epochs on average. Hierarchy-matching and geometry weights are scaled by s=0.85; answer CE, utilization, and depth prediction retain their original weights. ReduceLROnPlateau (factor 0.5, patience 1) monitors a held-out 10% slice of the training pool using $\mathcal{L}_{\text{ans}}+\lambda_u\mathcal{L}_{\text{util}}+0.05\,\overline{\mathcal{L}}_{\text{sem}}$. Checkpoints follow one accuracy rule. A checkpoint is kept when greedy accuracy improves. One within 0.5 points of the best is also kept when answer CE falls by at least 0.05, and equal accuracy keeps the lower answer CE. Training stops after five epochs without a new checkpoint.

3.3 Training Objectives

The objective combines answer cross-entropy with role-specific alignment to frozen-teacher targets. Next-token cross-entropy trains the teacher on a; frozen-teacher representations provide targets for projected student states, which have no direct lexical identity. For a sampled path,

$$\mathcal{L} = \mathcal{L}_{\text{ans}} + s\,\mathcal{L}_{\text{slot}} + \lambda_u\,\mathbf{1}_{\{\mathbf{X}_{\text{train}}=\mathbf{X}_{K+1}\}}\,\mathcal{L}_{\text{util}} + \mathcal{L}_{\text{reg}}, \tag{9}$$

with s=0.85 in the all-latent stage and s=1 otherwise. The indicator activates only for the complete latent bank. $\mathcal{L}_{\text{ans}}$ is next-token CE under the sampled input in Equation 8. $\mathcal{L}_{\text{slot}}$ aligns plan-to-plan and detail-to-detail positions, the pooled hierarchy, and plan–detail residuals with their frozen-teacher targets (Appendix A). The utilization hinge $\mathcal{L}_{\text{util}}$ requires the latent bank to improve on an empty-bank counterfactual (λ_u=0.35). $\mathcal{L}_{\text{reg}}$ combines a 0.05-weight pairwise separation term with a 0.05-weight depth-classification term. All weights are shared across datasets and backbones.

3.4 Data Annotation

Because the target benchmarks provide at most one unstructured rationale, we derive plan–detail annotations from the question, gold answer, and available rationale. The annotator returns a one-sentence strategy without arithmetic and an ordered execution trace whose final state agrees with the answer. Validation rejects malformed records and, for numerical tasks, traces that omit the gold answer. We collapse near-duplicate consecutive details and reduce the plan-alignment weight when a plan enumerates its details or overlaps them lexically (the w_P term in Appendix A). Two annotation passes yield valid records for more than 99% of each split. The complete prompt and task-specific schemas appear in Appendix C. Annotations supervise training only; evaluation supplies the question and answer prefix, never gold plan or detail text.

4 EXPERIMENTS

4.1 SETUP

Backbones. We evaluate two independently parameterized student–teacher pairs: Qwen2.5-0.5B-Instruct with Qwen2.5-7B-Instruct (Qwen Team, 2024), and Mistral-1.1B with a Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) teacher. Coconut, CODI, SIM-CoT, iCoT-SI, and HLR adapt the answer model with LoRA. SemCoT trains its contemplation generator and projection against a frozen teacher; SoftCoT trains its projection. We evaluate each method as the complete system defined by its architecture and training recipe, under shared data splits, seeds, latent slots, and answer budgets.

Data. We use SemCoT-style processed splits of GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), MultiArith (Roy & Roth, 2015), CommonsenseQA (Talmor et al., 2019), and CoinFlip (Wei et al., 2022; Kojima et al., 2022). Every method uses the same seed-42 split, and the evaluation portion is the shared test set: $800/200$ examples for CoinFlip and CSQA, $780/200$ for GSM8K, $420/180$ for MultiArith, and $700/200$ for SVAMP. Every result aggregates training seeds $\{42, 43, 44\}$. For each seed, we take the highest accuracy over decoding temperatures $\{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$, then report the three-seed mean and sample standard deviation. Appendix B shows the temperature response.

Baselines. We compare with Coconut, CODI, SoftCoT, SemCoT, SIM-CoT, and iCoT-SI. Each baseline uses one fixed training configuration across all datasets and both backbones (Appendix B). We evaluate all 70 method–backbone–dataset cells with three training seeds each and the full temperature grid.

Evaluation controls. Every continuous method receives five latent positions at training and evaluation: one plan plus four details for HLR, and five undifferentiated positions otherwise. Coconut uses one continuous state per internalized step. All methods, including iCoT-SI, decode at most 30 answer tokens. Sequence length is 512 and effective batch size is 4; HLR and CODI use gradient accumulation from single examples. HLR uses AdamW with learning rates 10^{-4} for new heads, 10^{-6} for the student backbone, and 10^{-5} for teacher LoRA, with gradient clipping at 1.0. The shared controls are the data splits, seeds, latent-slot budget, answer cap, and temperature grid.

4.2 MAIN RESULTS

Table 2 summarizes accuracy under the shared evaluation protocol. On Qwen, HLR leads CoinFlip, GSM8K, MultiArith, and SVAMP and ties SemCoT on CSQA. The largest gains occur on tasks with repeated state transitions: CoinFlip improves from 75.3 ± 5.9 for the strongest baseline to 87.3 ± 4.0, while MultiArith improves from 70.0 ± 1.0 to 80.4 ± 1.7. GSM8K reaches 32.2 ± 0.8, 4.4 points above SIM-CoT, and SVAMP reaches 72.7 ± 1.5, one point above iCoT-SI.

On Mistral, HLR leads CSQA, MultiArith, and SVAMP, while backbone choice materially affects CoinFlip and GSM8K (Section 5.3). Its MultiArith accuracy is 50.9 ± 7.9, compared with 30.6 ± 6.9 for the strongest baseline, and its SVAMP accuracy is 57.3 ± 2.0, five points above SemCoT. CODI leads Mistral CoinFlip, and SemCoT leads Mistral GSM8K. Figure 3 reports the corresponding accuracy and variance profiles for all ten cells.

5 ANALYSIS

5.1 TASK STRUCTURE AND ANNOTATION COVERAGE

Annotated hierarchy depth varies by dataset. Figure 4 compares this annotation structure with the observed margin over the strongest baseline. Table 3 lists the same seed-42 splits. Coverage is complete after validation, but hierarchy depth varies substantially. CoinFlip concentrates on four-step state updates, whereas 546 of 700 SVAMP examples contain a single detail. The accuracy margin is not monotonic in raw depth: the same dataset can produce different margins across backbones, and Mistral MultiArith shows a large gain at moderate depth. Accuracy margins vary with backbone as

Table 2: Accuracy (%): for each seed the highest score on the decoding-temperature grid, then mean ± std over seeds $\{42, 43, 44\}$ on the seed-42 data split. The body is generated directly from experiment JSONL files. **Amber boldface** / teal underlining: best / second-best within each dataset column of a backbone block.

Qwen					
Method	CoinFlip	CSQA	GSM8K	MultiArith	SVAMP
Coconut	42.5±36.9	23.5±2.0	18.8±14.4	48.7±39.7	62.8±9.5
CODI	44.3±17.8	96.0±3.6	6.0±4.1	19.3±9.4	8.3±9.4
SoftCoT	19.3±16.6	93.7±0.6	8.0±1.0	7.4±0.3	11.5±0.9
SemCoT	56.0±22.3	**99.8±0.3**	20.7±17.5	70.0±1.0	63.8±3.5
SIM-CoT	60.0±14.1	27.3±7.6	27.8±1.3	50.9±38.4	24.8±38.7
iCoT-SI	75.3±5.9	96.7±1.5	10.8±9.9	3.1±0.3	71.7±2.9
HLR	**87.3±4.0**	**99.8±0.3**	**32.2±0.8**	**80.4±1.7**	**72.7±1.5**
Mistral					
Method	CoinFlip	CSQA	GSM8K	MultiArith	SVAMP
Coconut	55.7±6.6	72.3±3.8	8.3±1.6	23.9±2.0	40.7±2.8
CODI	**71.7±17.1**	96.2±1.3	4.2±2.5	22.4±12.1	31.5±8.5
SoftCoT	1.3±1.5	94.3±1.3	7.0±1.0	7.0±0.3	10.8±0.3
SemCoT	63.8±14.6	97.0±2.8	**12.3±6.2**	30.6±6.9	52.3±1.9
SIM-CoT	60.2±3.6	78.7±1.8	7.0±1.3	20.7±4.2	45.8±1.0
iCoT-SI	66.8±7.1	94.0±1.3	11.2±1.0	3.9±1.1	50.7±5.8
HLR	58.5±15.8	**98.0±0.5**	8.5±5.2	**50.9±7.9**	**57.3±2.0**

Table 3: Hierarchy-field coverage in the actual seed-42 training splits. A record is covered when it contains a nonempty plan and nonempty detail list; the descriptive lengths are computed before fixed-budget padding/truncation.

Dataset	Train	Eval	Coverage	Plan words	Details
CoinFlip	800	200	100.0%	11.90	4.00
CSQA	800	200	100.0%	14.43	5.82
GSM8K	780	200	100.0%	11.55	3.28
MultiArith	420	180	100.0%	10.51	2.06
SVAMP	700	200	100.0%	10.04	1.22

well as annotation depth; detail count alone does not explain the result. The fixed $K{=}4$ interface standardizes compute by truncating longer annotations and repeating the final detail in shorter ones; the reported depth statistics precede this transformation.

5.2 Ablations

Table 4 removes two components of HLR on Qwen CoinFlip. The hierarchy ablation drops the plan from the parent set of every detail, so later latents no longer read the plan. The five-slot budget is

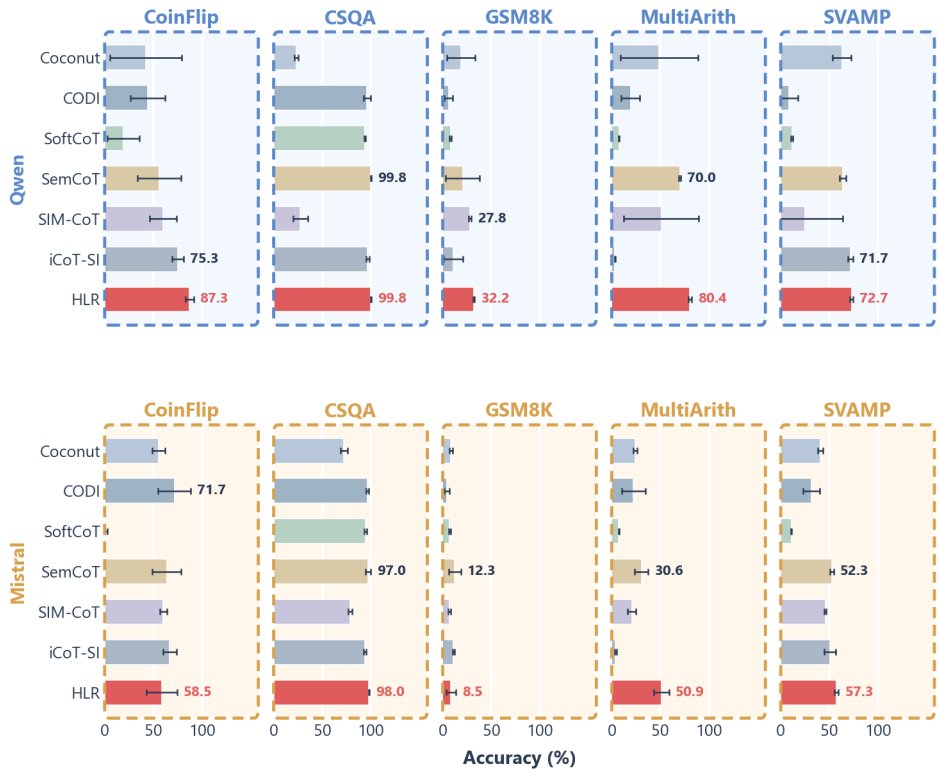

Figure 3: Main results on all ten backbone–dataset cells: for each seed we take the best decoding temperature, then plot the three-seed mean. Rows are backbones and columns are datasets. Method names sit on the left of each row. Numeric labels mark HLR and the strongest baseline in that cell, and error bars are ± 1 sample standard deviation. HLR is red. The same numbers appear in Table 2.

Table 4: Hierarchy and exclusive-path ablations on Qwen CoinFlip, under the same slots, answer budget, split, seeds, and best-temperature rule as Table 2. Accuracy is mean $\pm$ std (%). **Amber bold** marks full HLR.

Variant	Accuracy
HLR	**87.3±4.0**
Without hierarchy	78.5±13.9
Without exclusive path	83.3±9.7

unchanged. Accuracy falls from 87.3 ± 4.0 to 78.5 ± 13.9. The exclusive-path ablation scores the curriculum input and the full latent bank together on every example, and accuracy falls to 83.3 ± 9.7. The hierarchy accounts for the larger drop.

5.3 Backbone-Specific Behavior

The two cells not led by HLR are both on Mistral: CODI leads CoinFlip and SemCoT leads GSM8K. CoinFlip exhibits substantial optimization variance across latent methods. HLR spans 47.0–76.5 across seeds, while CODI reports 71.7 ± 17.1 and SemCoT 63.8 ± 14.6. In contrast, the Qwen version of the same task is both stronger and more stable. GSM8K shows a related backbone reversal: HLR leads on Qwen but SemCoT leads on Mistral, where HLR obtains 8.5 ± 5.2 and SemCoT obtains 12.3 ± 6.2; one HLR seed falls to 2.5. Thus the same hierarchical interface can produce different mean accuracy and seed variance across backbones, especially on CoinFlip and GSM8K.

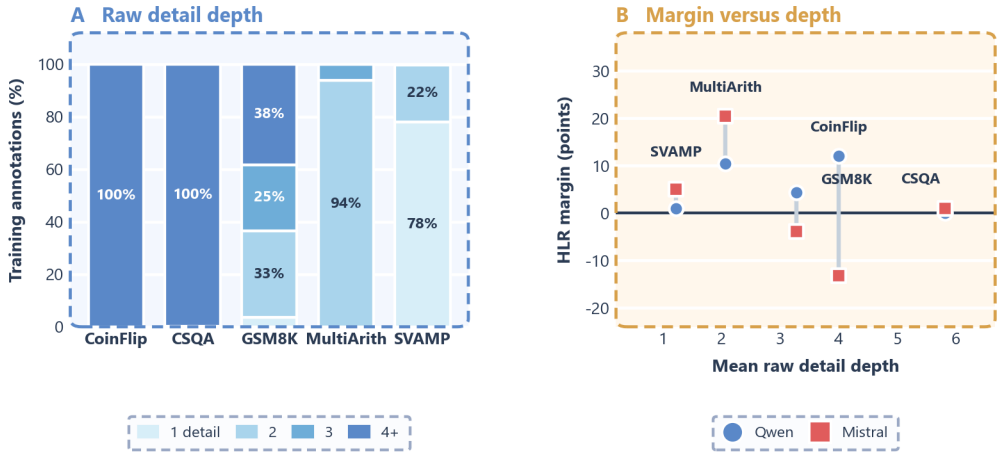

Figure 4: Available hierarchy and the observed accuracy margin. (a) Raw detail-count distribution in the seed-42 training split. (b) HLR margin over the strongest baseline against mean raw detail depth; connectors join the two backbones for each dataset. The points are descriptive; they do not establish annotation depth as the cause of the accuracy gap.

6 Conclusion

HLR separates planning from execution in the inference graph: one latent encodes the plan, and each subsequent detail latent is generated under direct plan conditioning. A role-aligned curriculum transfers this structure from text to vectors while scoring one answer path per example.

The largest gains are 12.0 points on Qwen CoinFlip and 20.3 points on Mistral MultiArith, two tasks that repeatedly apply a strategy across state updates. On Qwen CoinFlip, removing the hierarchy reduces accuracy by 8.8 points, and removing exclusive-path training reduces it by 4.0 points. The Mistral reversals on CoinFlip and GSM8K further indicate that the benefit of latent organization depends on the student–teacher pair as well as the task.

AI use statement

We used generative AI to edit prose, produce the plan–detail annotations described in Section 3.4, and draft evaluation-logging scaffolding. We did not use generative AI to produce experimental measurements or select reported values. The authors reviewed all assisted prose, schema-validated the annotations, and executed and tested the generated code. We take responsibility for the final text, claims, code, and artifacts.

Ethics statement

This work uses public reasoning benchmarks and involves no human subjects or personal data. Model-generated plan–detail annotations may inherit biases or errors from the annotation model; they are used only as intermediate supervision and are validated against the benchmark schema and gold answer. All evaluated models are open-weight.

Reproducibility statement

Sections 4.1, 3.2, and 3.3 specify the protocol, hyperparameters, data split, training seeds, loss weights, and curriculum. Appendix A defines hierarchy matching, and Appendix C gives the complete annotation prompt. Every method reads byte-identical splits. Tables and figures are generated from JSONL records whose manifest tracks source paths and seed coverage. We will release the annotation, training, evaluation, and figure-generation code, together with LoRA adapters and student checkpoints, upon acceptance.

REFERENCES

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

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

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*, 2022.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. In *Advances in Neural Information Processing Systems*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

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

Xilin Wei, Xiaoran Chen, Mingjun Ye, Jiaqi Wang, Yuhang Duan, Wenwei Zhang, Dahua Li, Yuhang Zang, and Kai Chen. SIM-CoT: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. *arXiv preprint arXiv:2502.12134*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

A Hierarchy Matching

The annotation and latent bank share an ordered plan–detail structure. $\mathcal{L}_{\text{slot}}$ transfers that structure at three resolutions: individual positions, the pooled hierarchy, and plan–detail relations. Every term uses cosine dissimilarity, with weights shared across datasets and backbones.

Matching the gold hierarchy. Gold plan P, details $\{D_i\}_{i=1}^{k}$, and rationale R are encoded as frozen-teacher layer-16 mean-pools $\bar{\mathbf{t}}^{\text{plan}}$, $\bar{\mathbf{t}}_i^{\text{det}}$, $\bar{\mathbf{t}}^{\text{reason}}$. Write

$$\mathcal{L}_{\text{slot}} = \sum_{(\mathbf{u},\mathbf{v})\in\mathcal{A}} w(\mathbf{u},\mathbf{v})\,\big(1-\cos(\mathbf{u},\mathbf{v})\big). \tag{10}$$

$\mathcal{A}$ is induced by that hierarchy. *Positions:* the plan slot against $\bar{\mathbf{t}}^{\text{plan}}$ (weight $w_P\lambda_p$) and each detail slot against $\bar{\mathbf{t}}_i^{\text{det}}$ (mean weight λ_d). *Readout:* the mean of all slots against $\bar{\mathbf{t}}^{\text{reason}}$ (weight λ_r). *Plan–detail relations:* the residual $\tilde{\mathbf{h}}_i^{\text{det}} - \tilde{\mathbf{h}}^{\text{plan}}$ against the same residual of the teacher targets (mean weight $w_P\lambda_c$). Coefficients: $\lambda_p{=}\lambda_d{=}0.3$, $\lambda_r{=}\lambda_c{=}0.2$. $w_P{=}0.5$ if the gold plan looks enumerative or its token Jaccard with any detail exceeds 0.65, else $w_P{=}1$.

Using the hierarchy. A no-gradient counterfactual drops the hierarchy but keeps the same question, answer prefix, and target a. With $m{=}0.05$,

$$\mathcal{L}_{\text{util}} = \max\big(0,\ \mathcal{L}_{\text{ans}} - \mathcal{L}_{\text{ans}}^{(\varnothing)} + m\big). \tag{11}$$

The teacher is in eval mode (LoRA dropout off); the hinge contributes no adapter gradient.

Separation and depth. Let $\hat{\mathbf{h}}$ be ℓ_2-normalized active latents. The depth head uses class indices $\{0,\dots,K{-}1\}$ for 1 through K details, and $k^*{=}K{-}1$ under the fixed budget is the class for K details ($\min(k,K){-}1$ if adaptive depth is on). Then

$$\begin{aligned}\mathcal{L}_{\text{reg}} = s\lambda_g \tfrac{2}{n(n-1)} \sum_{i<j} \mathrm{ReLU}\big(0.5 - \|\hat{\mathbf{h}}^{(i)} - \hat{\mathbf{h}}^{(j)}\|_2\big) \\ +\lambda_n\,\mathrm{CE}(\text{depth},\,k^*),\end{aligned} \tag{12}$$

with $\lambda_g{=}\lambda_n{=}0.05$ and n the number of active slots.

B Additional Experimental Details

The data split remains fixed at seed 42 while training varies over seeds $\{42, 43, 44\}$. Table 5 lists each HLR run at its best decoding temperature. Figures 5–6 report the temperature response and seed dispersion for all ten cells.

Answer prefixes. Shared by HLR and the baselines. For Qwen, the query is placed in the chat user turn after "Question:", and p is the assistant turn that starts with "Answer:". For Mistral, the query is wrapped as "[INST] Question:" and p is "Answer: [/INST]". Gold answers are concatenated after p during training; inference decodes from p.

Table 5: HLR per-seed accuracy (%): each seed uses its best score on the decoding-temperature grid. Seeds are $\{42, 43, 44\}$ and the data split seed is 42. CSQA is CommonsenseQA.

Backbone	Seed	GSM8K	CoinFlip	SVAMP	MultiArith	CSQA
Qwen	42	32.0	85.0	74.0	82.2	100.0
	43	33.0	92.0	73.0	80.0	100.0
	44	31.5	85.0	71.0	78.9	99.5
	Mean ± std	32.2±0.8	87.3±4.0	72.7±1.5	80.4±1.7	99.8±0.3
Mistral	42	11.5	76.5	55.0	59.4	98.5
	43	2.5	52.0	58.5	43.9	97.5
	44	11.5	47.0	58.5	49.4	98.0
	Mean ± std	8.5±5.2	58.5±15.8	57.3±2.0	50.9±7.9	98.0±0.5

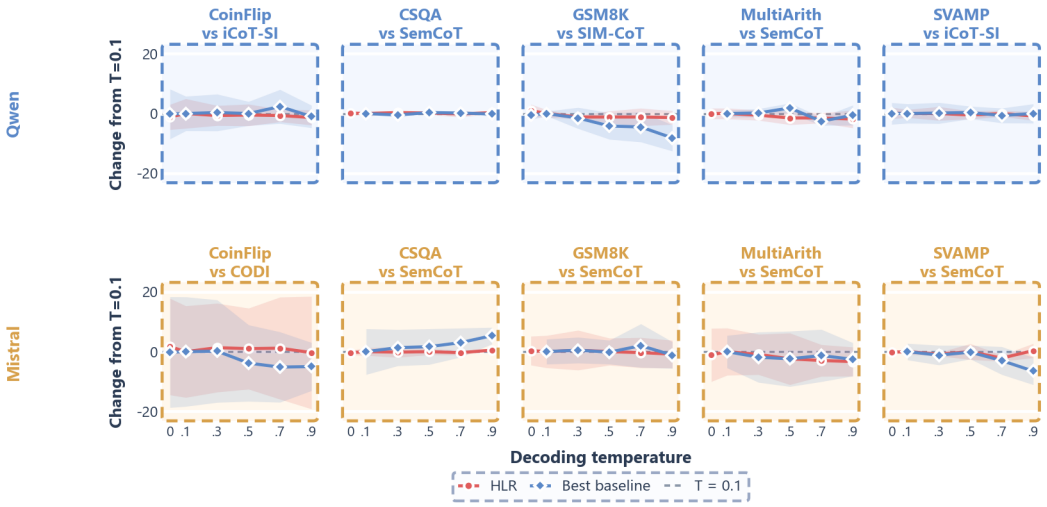

Figure 5: Decoding-temperature sensitivity on all ten backbone–dataset cells, shown as the change from the T=0.1 mean. Lines are three-seed means and shaded bands are ±1 sample standard deviation at that temperature. Each panel names its strongest complete baseline. The horizontal dashed line is the T=0.1 reference. The main table uses the best temperature per seed.

HLR hyperparameters (defaults used in the main tables). Unless noted, K=4 details; target layer 16; exclusive full-latent sample probability 0.4; additive aux weight 0; utilization weight 0.35; utilization margin 0.05; geometry margin 0.5 on ℓ_2-normalized latents; fixed latent budget on. The 7B base weights stay frozen. The default run adapts the teacher with LoRA (learning rate 10^{-5}). Main-table HLR uses these shared defaults on both backbones and all five datasets. Optimizer groups and epoch counts are in Table 1. Hierarchy matching and coefficients are in Appendix A.

Baseline defaults. Each baseline uses one configuration across all datasets and backbones. Batch size 4, sequence length 512, and multiple A100 GPUs are shared. Coconut, CODI, SIM-CoT, and iCoT-SI adapt the model with LoRA. SemCoT trains its generator and projection against a frozen teacher; SoftCoT trains only its projection. Coconut and SIM-CoT divide ten epochs into 7+3 curriculum stages. CODI, SoftCoT, and iCoT-SI use a single ten-epoch budget. SemCoT trains its two modules sequentially, each with a linear stage followed by a language-model stage.

C Data Annotation Prompt

Annotations are produced by DeepSeek-V4-Flash at temperature 0.1. The system prompt is:

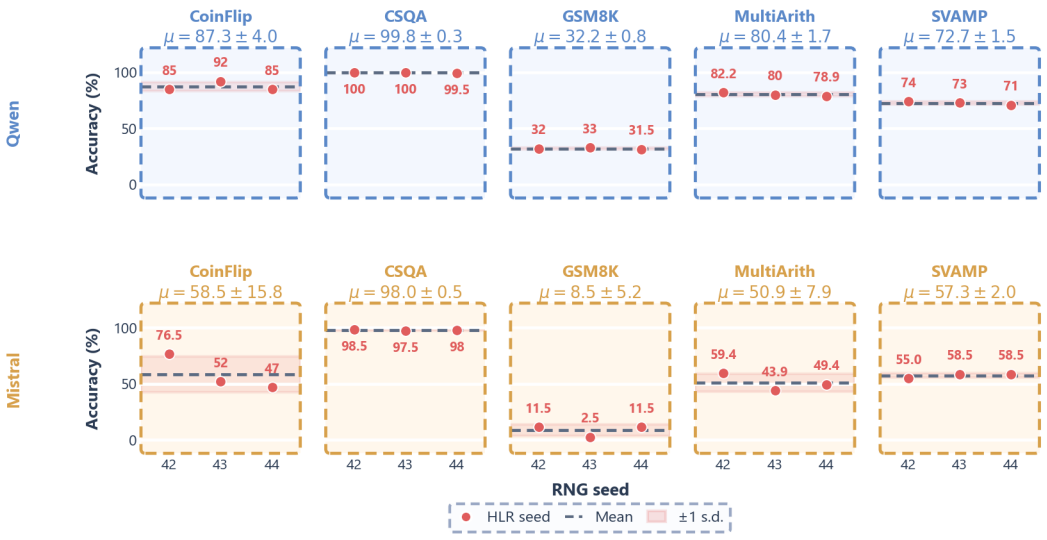

Figure 6: HLR seed stability on all ten backbone–dataset cells under the per-seed best-temperature rule. Points are seeds $\{42, 43, 44\}$ and are not connected. The dashed line is the three-seed mean, the shaded band is ± 1 sample standard deviation, and the panel title reports $\mu \pm$s.d.

Table 6: Default learning rate and epoch budget for each baseline, including CODI. The same row is used on both backbones and all five datasets. Shared compute: batch size 4, sequence length 512, multiple A100 GPUs.

Method	Learning rate	Epochs
Coconut	1×10^{-4}	10, split across stages
CODI	1×10^{-4}	10
SoftCoT	2×10^{-5}	10
SemCoT	1×10^{-4}, then 1×10^{-7}	7, then 3, per module
SIM-CoT	1×10^{-4}	10, split across stages
iCoT-SI	5×10^{-5}	10

> You are a senior annotation expert for Hierarchical Latent Chain-of-Thought. Convert one problem and its ground-truth reasoning into (1) `plan_text`, strategy only, and (2) `detail_texts`, execution steps only. Stay faithful to the provided ground-truth reasoning and do not invent another solution path. Keep the final-answer semantics unchanged. Strictly separate strategy from execution. Return one JSON object only, with no markdown fences and no commentary.

The user message gives the dataset name, the problem, the ground-truth reasoning, and the ground-truth answer, then one of the following English schemas.

> **GSM8K, SVAMP, MultiArith.** Return JSON with `plan_text` and `detail_texts`. `plan_text`: one sentence, 8–22 words (at most 30). Describe the operation order or method only. Do not include intermediate results or the final answer. Avoid equation chains. Prefer verbs such as convert, halve, subtract, scale, and aggregate. Referring to roles or quantities abstractly is allowed ("April sales"); raw result numbers are not. `detail_texts`: ordered atomic calculation steps that produce the ground-truth final answer. Prefer compact equations with full operands, such as "48/2=24", "290.0/2.0=145.0", and "30.0-11.0=19". Never drop leading digits. Broken forms such as

"0/2.0=145" and "0-11.0=19" are forbidden. One independent calculation per item. No prose paragraphs. Use the provided reasoning when it is consistent with the ground-truth answer. If the reasoning contradicts the ground-truth answer, ignore that reasoning and solve from the problem and the answer. The last computed value must equal the ground-truth answer. Typical length 2–6. Style example only: `plan_text`: "Halve April sales to obtain May sales, then sum both months." `detail_texts`: "48/2=24", "48+24=72".

CommonsenseQA. Return JSON with `plan_text` and `detail_texts`. `plan_text`: one sentence, 8–25 words (at most 35). State the decision strategy only: a constraint check, a causal fit, a definition match, or elimination among options. Do not reveal the final letter A–E. `detail_texts`: ordered checks of constraints or options. Each item evaluates one claim and briefly eliminates or supports it. Stay faithful to the provided reasoning. The final item must be "Choose X: *short justification*," where X is the ground-truth letter. Typical length 3–7. Style example only: `plan_text`: "Interpret the verb needed by context, then eliminate options that fail grammar or meaning." `detail_texts`: "Sentence needs a verb after 'seemed to'."; "Eliminate C: adjective, not a verb."; "Eliminate B: sanctions would not support school efforts."; "Choose A: 'ignore' fits punitive sanctions overlooking reforms."

CoinFlip. Return JSON with `plan_text` and `detail_texts`. `plan_text`: one sentence, 8–20 words (at most 28). The strategy tracks the binary coin state across flip and no-flip actions. Do not include the final yes or no. `detail_texts`: one state update per item. Use the style "Init: heads", "Name flips $\rightarrow$ tails", "Name no-flip $\rightarrow$ tails", and "Answer: no". The final item must match the ground-truth yes or no. Typical length 3–8. Style example only: `plan_text`: "Initialize coin state, update after each flip or non-flip, then report heads status." `detail_texts`: "Init: heads"; "sager no-flip $\rightarrow$ heads"; "zyheir flips $\rightarrow$ tails"; "Answer: no".