Why is A+B Better Than B? A Simple Graph Perspective on Task Transfer

Dang Nguyen[*1] **Jianhao Huang**[*1] **Ali Payani**[2] **Baharan Mirzasoleiman**[1]

[1]Department of Computer Science, University of California, Los Angeles [2]Cisco Systems Inc.

[*]Equal contribution.

Abstract

Joint training on an auxiliary task A and a target task B routinely outperforms training on B alone, but the conditions and mechanism behind this gap are poorly understood. We model the two tasks as goal-conditioned navigation on a shared directed graph and identify a *joint-dominance regime*, where neither task alone covers the full target trajectory but their union does; in this regime joint training preserves teacher-level accuracy along the entire target path while single-task training incurs an exponential penalty in the length of its uncovered region. We verify the theory in a controlled GPT-2 study on two algorithmic tasks sharing a depth-first-search backbone, a Qwen2.5 GRPO study on related math tasks, and a cross-representation procedural-reasoning setting. The findings translate into three checkable criteria for selecting auxiliary tasks: shared sub-skills, joint horizon coverage, and base-policy solvability.

1. Introduction

Modern post-training pipelines for large language models rarely optimize a target task in isolation. Instruction-tuning mixtures combine dozens of tasks (Longpre et al., 2023; Wang et al., 2023), math-specialized models train on blends of math and code (Shao et al., 2024), and RL fine-tuning typically draws rollouts from multiple task families (Lambert et al., 2024; Guo et al., 2025). A consistent empirical pattern emerges: jointly training on an auxiliary task A and a target task B ("train on $A{+}B$") often outperforms training on B alone. Yet selecting effective auxiliaries remains largely heuristic. Some auxiliaries yield substantial gains, others have negligible impact, and we lack a quantitative theory that predicts which regime a given pair falls into.

Existing explanations fall into two broad categories. Empirical studies on LLMs demonstrate *that* auxiliary tasks help, but rarely characterize *when* they fail. In contrast, theoretical work on multi-task learning and representational transfer in RL (Agarwal et al., 2023) formalizes the benefits of shared features, but assumes generative access to auxiliary-task MDPs and overlooks a fundamental constraint in supervised pretraining: the *trajectory horizon*. In practice, pretraining data consist of length-limited rollouts. If target-only supervision leaves part of the relevant trajectory distribution uncovered, the learner receives no signal on those regions regardless of how well it models the observed prefix. This motivates a theory whose predictions explicitly depend on trajectory coverage under finite training horizons.

Our abstraction. We model tasks as goal-conditioned navigation on a shared finite directed graph G, where each task corresponds to a distinct start vertex and both share a common goal vertex v_G. Training data consist of length-d_A and length-d_B trajectories sampled from a shared goal-directed transition kernel $P^\star$, which biases transitions toward the goal with margin η. The learner parameterizes a softmax transition kernel over neighbors and is trained via next-token prediction.

Within this abstraction, we identify the *joint-dominance regime*: neither task alone covers the full target trajectory, but the union of their covered regions does. Concretely, target-only supervision leaves an uncovered suffix near the goal, while auxiliary-only supervision leaves an uncovered prefix near the target start state. Joint training combines the two partial coverages and exposes the learner to the entire target path. Intuitively, each task supplies the portion of the trajectory that the other cannot reach.

Accepted to FoGen 2026: Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning, an ICML 2026 workshop (non-archival).

Theoretical results. Our analysis yields two main results. Theorem 3.1 shows that gradient descent on the joint next-token prediction loss decouples per column: each covered vertex is learned at rate $\widetilde{O}(1/\sqrt{T})$. Consequently, auxiliary data does not dilute estimation on shared regions and can improve coverage of regions the target task alone never observes.

Corollary 3.2 then characterizes downstream trajectory accuracy. In the joint-dominance regime, joint training covers the entire target rollout and therefore achieves teacher-level per-step accuracy along the full path. In contrast, each single-task baseline leaves a contiguous uncovered region of the target trajectory and defaults to near-uniform behavior there. As a result, success probability degrades exponentially in the number of uncovered steps when the action space is large. The mechanism is simple: the two tasks complete each other's coverage gaps, while either task alone leaves part of the target trajectory effectively unlearned.

From theory to practice. We empirically validate these predictions in three complementary settings (Section 4).

In a *controlled* setup, we train a small GPT-2 model on two algorithmic tasks that share a depth-first search (DFS) backbone. Adding auxiliary DFS supervision in pretraining yields modest zero-shot improvements and substantially larger gains after RL fine-tuning on the target, indicating that RL refines structural priors established during pretraining. We further show that a similar prior can be induced at the RL stage: a forward curriculum (auxiliary $\rightarrow$ target) outperforms both single-task training and the reverse order, with performance gains concentrated on the task trained last.

In a *practical* setting, we run GRPO with Qwen2.5-7B-Instruct on pairs of math tasks from the Omega dataset (Sun et al., 2025), where the target builds on the auxiliary as a sub-skill. Consistent with our theory, auxiliary mixing improves performance when the auxiliary is solvable by the base policy, and yields no benefit otherwise.

Finally, on AsyncHow procedural reasoning (Lin et al., 2026), we demonstrate that the regime extends to *cross-representation* transfer. The same partially ordered task graph can be presented either as a typed graph or as a natural-language step sequence. While mixed-batch training achieves the highest accuracy, a graph$\rightarrow$NL curriculum recovers most of this gain at significantly lower compute, whereas the reverse ordering does not. This asymmetry suggests that the more abstract representation more effectively installs the shared procedural backbone, while the surface form primarily serves to read it out—mirroring the curriculum effect observed in the controlled setting.

Contributions.

- We introduce a minimal graph-theoretic abstraction of auxiliary-task transfer and characterize a *joint-dominance regime* in which auxiliary and target tasks complement each other along a shared trajectory. In this regime, joint training covers the full target path, whereas single-task training leaves a contiguous uncovered segment whose effect on success probability degrades exponentially with its length.
- We validate the theory across three settings: controlled GPT-2 experiments on algorithmic tasks (including both pretraining mixtures and RL curricula), large-scale GRPO experiments on math tasks, and cross-representation procedural reasoning. Across these settings, we observe consistent patterns of positive transfer, predicted failure modes, and an asymmetric curriculum effect across both training stages and representations.
- We distill three practical criteria for selecting auxiliary tasks: shared sub-skills, joint horizon coverage, and base-policy solvability.

2. Preliminaries

Y-shape graph. We model the environment as a finite graph G with a vertex set V (acting as the "vocabulary"). Instead of a disjoint structure, the forward backbone of the graph comprises three distinct paths that converge on a shared goal vertex v_G: task A is a forward path from v_A to a junction v_s with length $\Delta_A - r_s$; task B is a forward path from v_B to v_s with length $\Delta_B - r_s$; and the trunk is a forward path from v_s to v_G with length r_s.

Remark 2.1. Synthetic graph path-finding tasks are widely used as a standard abstraction for studying reasoning in language models (Ye et al., 2025; Kim et al., 2025; Huang et al., 2026a;b). The setting captures the essential structure of auto-regressive planning while remaining simple enough to admit clean theoretical analysis. Within this abstraction, the Y-shape is the smallest geometry in which two tasks can have complementary coverage, making it the minimal setting for studying joint task transfer.

Teacher Kernel and Learner Model. At every non-goal vertex v of two paths, the teacher places probability $\alpha \in (0,1)$ on the correct successor $f(v)$ and distributes the remaining $1-\alpha$ uniformly over the other $|V|-1$ vertices:

$$P^\star(u \mid v) \;=\; \begin{cases} \alpha & u = f(v), \\ (1-\alpha)/(|V|-1) & u \neq f(v). \end{cases}$$

Following Kim et al. (2025), the model is parameterized as a softmax transition kernel over V with one-hot embeddings, defined as $p_W(\cdot|v) := \mathrm{softmax}(W_{\cdot,v})$ for weight matrix $W \in \mathbb{R}^{|V|\times|V|}$.

Training and estimators. The two tasks generate deterministic forward trajectories from their start vertices:

$$v_0^\diamond := v_\diamond, \quad v_{t+1}^\diamond := f(v_t^\diamond) \text{ for } t = 0, \ldots, d_\diamond - 1 \text{ and } \diamond \in \{A, B\}$$

The coverage sets are $C_A := \{v_t^A : 0 \le t < d_A\}$ and $C_B := \{v_t^B : 0 \le t < d_B\}$, the vertices visited during each task's training, and $C_J := C_A \cup C_B$. At every visited vertex, the loss is the cross-entropy from the model to the teacher

$$\mathcal{L}_\diamond(W) \;:=\; \sum_{t=0}^{d_\diamond - 1} D_{\mathrm{KL}}\big(P^\star(\cdot \mid v_t^\diamond) \,\big\|\, p_W(\cdot \mid v_t^\diamond)\big)$$

and $\mathcal{L}_J := \mathcal{L}_A + \mathcal{L}_B$. Each task's estimator $W^\diamond$ is the population minimizer with the convention that columns unconstrained by the loss are set to zero:

$$W^\diamond \;:=\; \arg\min_W \mathcal{L}_\diamond(W) \quad \text{subject to } W_{\cdot,v} = 0 \text{ for } v \notin C_\diamond.$$

While the objective is invariant to an additive shift in the input of $\mathrm{softmax}$, the minimum becomes unique upon imposing the zero-sum convention $\mathbf{1}^\top W_{\cdot,v} = 0$.

Success probability. Under auto-regressive rollout from $\hat{v}_0 = v_B$ with $\hat{v}_{t+1} \sim p_W(\cdot \mid \hat{v}_t)$, reaching v_G in exactly Δ_B steps requires every step to take the correct forward action. The success probability is therefore

$$\mathrm{Succ}(W) \;:=\; \prod_{t=0}^{\Delta_B - 1} p_W\big(v_{t+1} \mid \hat{v}_t\big).$$

3. Theoretical Explanations for Joint Task Transfer

In this section, we present two main theorems that together explain why joint training succeeds where either single-task baseline fails. Theorem 3.1 establishes that pretraining is feasible: gradient descent on the NTP loss converges to the global minimizer at rate of $\widetilde{O}(1/\sqrt{T})$ at every vertex visited during training. Corollary 3.2 then computes each estimator's downstream success probability in closed form. Throughout, $\widetilde{O}$ hides polylogarithmic factors in $|V|$.

Theorem 3.1 (Training convergence)**.** *Run gradient descent on the population NTP loss $\mathcal{L}_\bullet(W)$ from zero initialization with step size 1; let $\bar{W}^{(T)}$ denote the averaged iterate after T steps. Under the Y-shape setup, for every $v \in C_\bullet$,*

$$\sup_{u \in V} \big|\mathrm{softmax}(\bar{W}^{(T)}_{\cdot,v})_u - P^\star(u \mid v)\big| \;=\; \widetilde{O}\left(\sqrt{\tfrac{1}{T}}\right).$$

For $v \notin C_\bullet$, GD from $W^{(0)} = 0$ stays at $W^{(T)}_{\cdot,v} = 0$ for all T, yielding $\mathrm{softmax}(W^{(T)}_{\cdot,v}) = U_V$.

We refer to Appendix B for the proof. Theorem 3.1 gives a learnability guarantee. Standard gradient descent on the NTP loss recovers the kernel at rate $\widetilde{O}(1/\sqrt{T})$ at every covered vertex. With this guarantee, we compare the performance across different task settings.

Corollary 3.2 (Path-accuracy comparison)**.** *Take $W^\bullet := \arg\min_W \mathcal{L}_\bullet(W)$ for $\bullet \in \{A, B, J\}$. If*

$$d_B + r_\mathrm{s} \ge \Delta_B,$$

Countdown DFS trace

```
[GOAL] 12
[BUDGET] 19
[STATE] 6 13 19
[STEP] 19 - 6 [EVAL] 13 13
[STATE] 13 13
[STEP] 13 + 13 [EVAL] 26 [RES] INVALID
[BACKTRACK]
[BUDGET] 18
[BACKTRACK]
[BUDGET] 17
[GOAL] 12
[STATE] 6 13 19
[STEP] 19 - 13 [EVAL] 6 6
[STATE] 6 6
[STEP] 6 + 6 [EVAL] 12 [RES] VALID
[END]
```

3-SAT DFS trace

```
[GOAL] T
[BUDGET] 12
[STATE] !v2|!v3|!v4 & !v2|!v3|v4 & v1|!v3|v4
[STEP] v2 T [EVAL] !v3|!v4 & !v3|v4 & v1|!v3|v4
[STATE] !v3|!v4 & !v3|v4 & v1|!v3|v4
[STEP] v3 T [EVAL] !v4 & v1|v4 & v4
[STATE] !v4 & v1|v4 & v4
[STEP] v4 T [EVAL] F [RES] INVALID
[BACKTRACK]
[BUDGET] 11
[GOAL] T
[STATE] !v4 & v1|v4 & v4
[STEP] v4 F [EVAL] F [RES] INVALID
[BACKTRACK]
[BUDGET] 9
[STEP] v3 F [EVAL] T [RES] VALID
[END]
```

Figure 1. Shared DFS trace format across Countdown and 3-SAT. Both tasks alternate between state expansion (`[STATE]`), step execution (`[STEP]`), and verification (`[EVAL]`/`[RES]`), with backtracking on invalid steps. Despite identical syntax, Countdown performs arithmetic search while 3-SAT performs logical propagation.

Table 1. Pretraining mixture composition. Only Countdown DFS traces differ across conditions.

Component	with-DFS	no-DFS
Countdown DFS traces	72K	0
Countdown snippets	44K	44K
3-SAT snippets	66K	66K

then the success probabilities are

$$\begin{aligned}\mathrm{Succ}(W^{\mathrm{J}}) &= \alpha^{\Delta_B},\\ \mathrm{Succ}(W^{\mathrm{B}}) &= \alpha^{d_B}\cdot\left(\tfrac{1}{|V|}\right)^{\Delta_B-d_B},\\ \mathrm{Succ}(W^{A}) &= \alpha^{r_{\mathrm{s}}}\cdot\left(\tfrac{1}{|V|}\right)^{\Delta_B-r_{\mathrm{s}}}.\end{aligned}$$

We refer to Appendix C for the proof. *When $|V|$ is large, joint training is exponentially better and neither single task suffices.* The realistic regime in language modeling is $|V| \gg 1/\alpha$, in which case $\alpha|V| \gg 1$ and the ratios $(\alpha|V|)^{\Delta_B-d_B}$ and $(\alpha|V|)^{\Delta_B-r_{\mathrm{s}}}$ are both very large. Each single-task baseline leaves a contiguous uncovered stretch of target's rollout path: while target task (B) alone leaves the trunk near v_G uncovered, auxiliary task (A) alone leaves the prefix near V_B uncovered and pays exponentially. Joint training is the only configuration that covers the entire path, providing an intuitive theoretical explanation for the empirical superiority of multi-task training: *neither task suffices independently, and the two tasks complete each other's coverage gap*.

4. Experimental verification

We study three settings to characterize when auxiliary tasks help RL fine-tuning and when they do not: a controlled pretraining study, a large-model RL study, and a cross-representation procedural reasoning benchmark. Together, these experiments address the following research questions:

- **RQ1:** When does adding an auxiliary task improve downstream performance?
- **RQ2:** When does the auxiliary task become redundant or harmful?
- **RQ3:** How does transfer interact with representation choice and curriculum order?

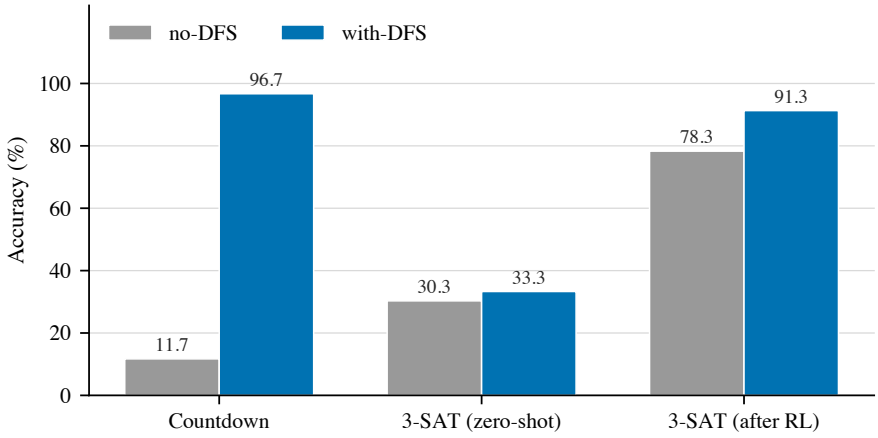

Figure 2. Controlled GPT-2 pretraining study on Countdown (auxiliary) and 3-SAT (target). The *with-DFS* condition includes full Countdown DFS trajectories during pretraining, while the *no-DFS* condition omits them. Both models receive only partial structural exposure to 3-SAT, making pre-RL evaluation on 3-SAT zero-shot. RL fine-tuning is performed only on 3-SAT. Auxiliary DFS supervision yields modest zero-shot transfer but substantially larger gains after RL.

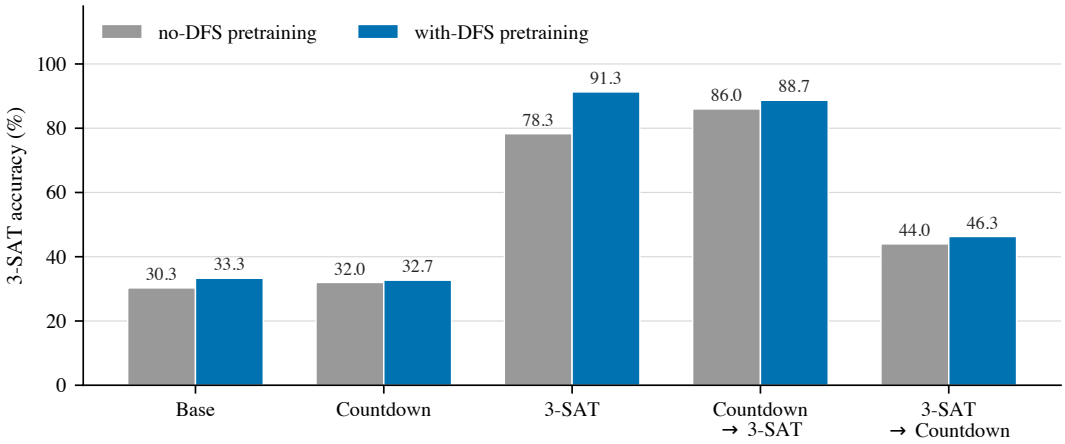

Figure 3. Effect of RL curricula on 3-SAT (target task) performance under two pretraining bases. The *no-DFS* base is pretrained without Countdown DFS trajectories, while the *with-DFS* base already contains the shared DFS structure from pretraining. The figure compares direct RL on Countdown or 3-SAT against two-stage curricula that vary task order.

4.1. Controlled GPT-2 study: DFS as shared structure

We study skill transfer in a controlled setting using a small transformer trained from scratch. We consider two algorithmic tasks: *Countdown* (Gandhi et al., 2024), a search-based arithmetic reasoning task, and *3-SAT* (Cook, 1971), a canonical satisfiability problem. Although superficially different, both admit a shared depth-first search (DFS) solution structure (Figure 1). Countdown serves as the auxiliary task and 3-SAT as the target task unless otherwise specified. We study (i) whether auxiliary DFS supervision in pretraining transfers to 3-SAT, and (ii) whether similar effects can be induced via RL curricula.

We design two interventions: (i) a pretraining mixture that varies whether DFS traces from Countdown are included, and (ii) an RL-stage curriculum that introduces Countdown supervision before optimizing on 3-SAT.

4.1.1. Pretraining mixture

We train a 3-layer GPT-2 (1.7M parameters) from scratch with identical architecture and optimization settings. The only difference between conditions is whether full DFS trajectories from Countdown are included in pretraining (Table 1). The *with-DFS* condition includes full DFS traces, while the *no-DFS* condition omits them. Both conditions include partial structural exposure to 3-SAT but no full solution traces, so 3-SAT evaluation is strictly zero-shot.

GCD (auxiliary) and Comp (target), easy variants

GCD: What is the greatest common factor of 249 and 166?
Answer: 83

Comp: For the polynomial $x^3 - 3x^2 - 9x + 27$, find the largest root in simplified form n/m (where n and m are coprime integers). Compute $\gcd(97 \cdot |n|,\ 970 \cdot |m|)$.
Answer: 97

Figure 4. Example prompts for the auxiliary (GCD) and target (Comp) tasks. Comp embeds GCD as its final step, making GCD a strict sub-skill.

Auxiliary DFS is necessary to learn the shared algorithm. As shown in Figure 2, removing DFS supervision collapses Countdown performance from 96.7% to 11.7%, confirming that full auxiliary traces are required to learn the shared algorithm.

Weak zero-shot transfer but strong RL amplification. Despite this large gap on Countdown, zero-shot performance on 3-SAT remains similar (33.3% vs. 30.3%, Figure 2). However, after RL fine-tuning on 3-SAT, the gap widens substantially (91.3% vs. 78.3%, Figure 2), showing that auxiliary supervision provides a foundation that is amplified by task-specific RL.

Takeaways:

Auxiliary supervision primarily improves representation quality, while RL determines how effectively that representation is exploited.

4.1.2. RL CURRICULUM ON THE AUXILIARY TASK

We next study whether the same effect can be induced at the RL stage. Starting from the no-DFS pretrained model, we compare single-task RL (Countdown or 3-SAT only) against RL curricula that order the two tasks.

Auxiliary RL induces structure when pretraining is weak. As shown in Figure 3, a forward curriculum (Countdown $\rightarrow$ 3-SAT) improves 3-SAT performance over direct RL (86.0% vs. 78.3%), indicating that auxiliary RL can bootstrap missing skills.

Diminishing returns when structure is already present. On the with-DFS model, the same curriculum provides no benefit and slightly degrades performance (88.7% vs. 91.3%, Figure 3), consistent with redundancy once DFS is already learned.

Strong directionality. The reverse curriculum consistently underperforms the forward one across both bases (Figure 3), showing that whichever task is trained last dominates final performance.

Takeaways:

Auxiliary RL helps only when it introduces missing skills the model lacks; otherwise it is redundant or harmful. Whichever task is trained last dominates final performance, so curricula should end on the target.

4.2. Qwen study: RL transfer in practice

We next evaluate whether the same phenomena persist in a practical RL-only setting using Qwen2.5-7B-Instruct, where pretraining is fixed and only the RL mixture is controlled.

Tasks. We use two tasks from the Omega dataset (Sun et al., 2025): `GCD` and `Comp`. The latter is a compositional task that explicitly contains GCD as a sub-skill (Figure 4). In this setting, GCD serves as the auxiliary task and Comp as the target task. We further stratify GCD into easy and hard variants to control whether the auxiliary task is within the base model's capability. The base model achieves 65.0% pass@1 on GCD-easy but only 36.8% on GCD-hard, indicating a clear regime shift from reliably solvable to partially out-of-reach.

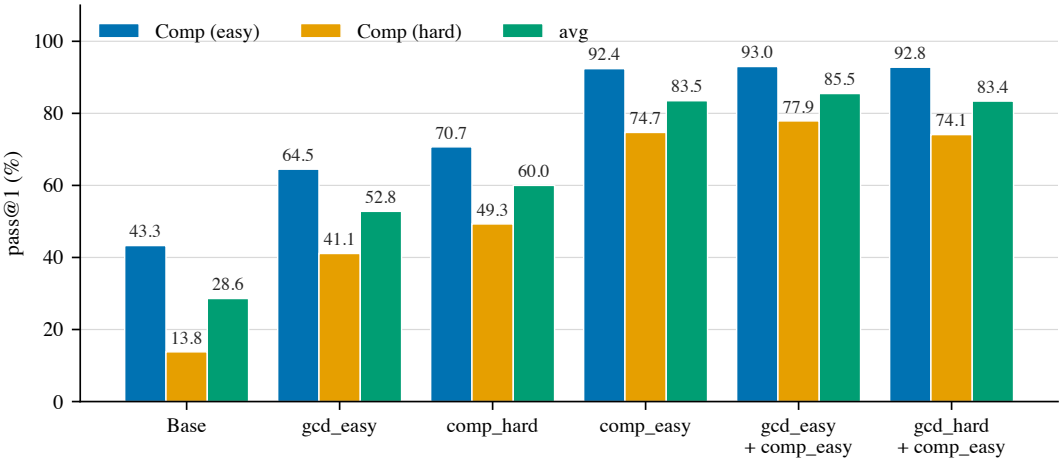

Figure 5. Effect of RL training mixtures on Qwen2.5-7B-Instruct (pass@1 performance). Two math tasks are considered: GCD as the auxiliary task (sub-skill) and Comp as the target compositional task that builds on GCD. Each bar corresponds to a different RL mixture controlling the proportion and difficulty of GCD and Comp data, while keeping model, optimizer, and GRPO settings fixed.

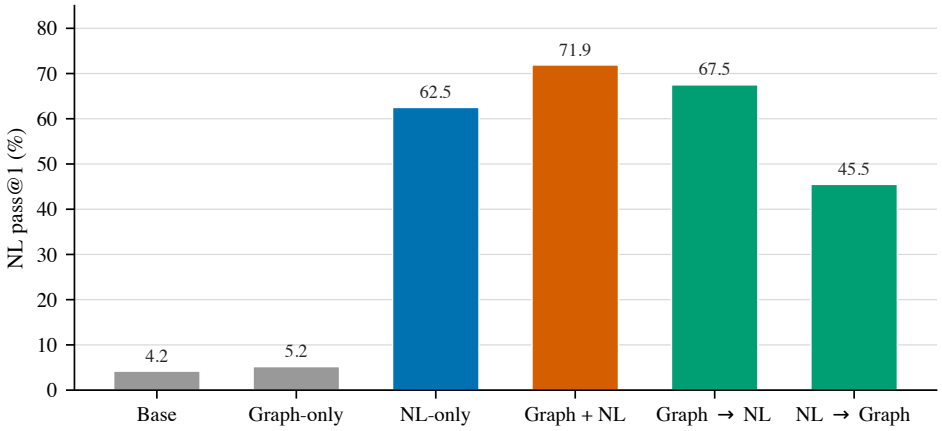

Figure 6. Cross-representation RL transfer on AsyncHow. Procedural reasoning is evaluated under two equivalent representations of the same underlying task graph: a structured graph format (auxiliary view) and a natural-language (NL) step sequence (target view). Evaluation is conducted on NL performance (pass@1), measuring how effectively procedural structure learned in one representation transfers to the other. The study compares single-representation training, mixed-batch training, and RL curricula that vary the ordering of graph and NL supervision.

Easy target data transfers better than hard data. As shown in Figure 5, training on easy instances of Comp yields substantially better generalization to hard instances than training directly on hard instances, despite never observing hard examples during training.

Auxiliary helps only when aligned with base capability. As shown in Figure 5, adding the auxiliary task GCD improves Comp performance only when the auxiliary lies within the base model's effective capability range. When GCD is easy (65.0% pass@1), it produces meaningful gradients and yields consistent gains on Comp. In contrast, when GCD is hard (36.8% pass@1), its signal becomes unreliable, and mixing it provides negligible or no benefit. This indicates that auxiliary tasks are useful only when the base model can already solve them with reasonable success rate, so that RL receives reliable gradient signal.

Takeaways:

Auxiliary RL helps only when the auxiliary task is within the base model's reach. If the model rarely solves the auxiliary, mixing it in adds noise without signal.

4.3. Cross-representation transfer: AsyncHow

Finally, we study cross-representation transfer on AsyncHow (Lin et al., 2026), where the same procedural graph can be expressed either as a typed graph or as a natural-language step sequence. Following Lin et al. (2026), we train Qwen2.5-1.5B with GRPO using identical hyperparameters across all conditions (Appendix D.3 for full training details). We fix Graph as the auxiliary task and NL as the target task, and compare the forward curriculum (Graph $\rightarrow$ NL) against the reverse curriculum (NL $\rightarrow$ Graph) as an ablation on ordering.

Mixed training is strongest, but curriculum is more efficient. As shown in Figure 6, full mixed training achieves the highest performance, but graph $\rightarrow$ NL curriculum recovers most of this gain at significantly lower compute, while the reverse direction performs worse.

Asymmetric transfer across representations. Training first on the graph representation consistently produces stronger downstream NL performance than the reverse (Figure 6), indicating that more abstract representations better support transfer.

Reward density is critical for transfer. A key finding (Table 2 in Appendix) is that performance is highly sensitive to the difficulty of stage-2 data: replacing easy NL with harder instances significantly degrades performance due to sparse reward signal. This shows that auxiliary usefulness depends not only on structural overlap but also on whether learning signals are dense enough at the current model capability.

Takeaways:

Representation choice and curriculum direction together determine transfer. More abstract representations make better auxiliaries, but only when paired with stage-2 data dense enough to produce reliable reward signal.

4.4. Unified interpretation.

Across all three settings, a consistent pattern emerges: auxiliary tasks help when they fill a coverage gap that lies within the base model's current reach, and are redundant or harmful otherwise. In GPT-2, DFS supervision installs a shared algorithmic skeleton that is later amplified by RL. In Qwen, auxiliary benefits appear only when the auxiliary is solvable by the base model, ensuring usable gradient signal. In AsyncHow, the same pattern extends to representation: more abstract representations provide stronger scaffolds, but only when paired with sufficiently dense reward signals.

Taken together, these results align with the joint-dominance regime predicted by our theory: effective auxiliaries are those that complete missing portions of a shared procedural structure that the base model can already partially execute, rather than those that are merely semantically related or data-rich.

5. Related work

Theoretical analyses of multi-task and representation transfer. The closest line of theoretical work is Agarwal et al. (2023), which shows that pretraining on a collection of task MDPs enables a learner to recover shared low-dimensional features that accelerate downstream RL. Their guarantees, however, rely on generative access to auxiliary-task MDPs and attribute gains to representation learning. Complementary information-theoretic perspectives on compositional generalization (Yao et al., 2026) decompose in-distribution and out-of-distribution gaps and show how chain-of-thought training induces multi-stage circuits that bridge compositional OOD generalization.

Our setting departs from both lines in a key respect: we study finite-horizon trajectory supervision without generative access, where the primary bottleneck is trajectory *coverage* rather than representation dimensionality alone. Even when two tasks share identical underlying structure, target-only supervision may leave portions of the target trajectory effectively unobserved. Our analysis characterizes this regime explicitly: single-task training leaves uncovered regions on the target rollout, while auxiliary supervision can complete those coverage gaps and restore teacher-level behavior on the full trajectory. This perspective is complementary to representation-learning analyses, focusing not on the dimensionality of shared features but on whether supervision reaches the relevant procedural states.

Length generalization and short-to-long curricula. A second line of work addresses long-horizon reasoning via curricula that progressively increase task difficulty within a single task. The h1 framework (Motwani et al., 2025) reports

substantial gains on long-horizon reasoning benchmarks by training with outcome-based rewards over increasingly deep compositions. In a controlled study on Countdown, Park et al. (2025) show that RL post-training improves generalization to larger inputs and novel expression structures, following a hierarchy of compositional difficulty. On the theoretical side, Wang et al. (2025) prove that shallow transformers can learn k-fold compositional structures with polynomial sample complexity under easy-to-hard curricula or mixtures of difficulties, while non-curriculum learners require exponentially many samples.

These works focus on *within-task* curricula, where progressively harder instances of the same task enable longer-horizon generalization. In contrast, we study *between-task* transfer: a distinct auxiliary task provides supervision over procedural regions that the target task alone does not sufficiently cover. From this perspective, auxiliary-task transfer acts as a cross-task analogue of easy-to-hard curricula, where compositional coverage is achieved by combining supervision across related tasks rather than scaling difficulty within a single task.

RL post-training dynamics and pre-/post-training synergy. Recent work has begun to characterize how supervised pretraining and RL fine-tuning interact. Shenfeld et al. (2026) show that on-policy RL implicitly minimizes the KL divergence to the base policy under the new-task distribution, favoring solutions close to the pretrained model. Empirically, Lai et al. (2025) find that reinforcement fine-tuning better preserves prior capabilities than SFT in continual settings, with reward variance acting as an implicit regularizer. From a theoretical perspective, Javanmard et al. (2026) analyze a tractable transformer–ICL pipeline and show that balanced pretraining can install latent capabilities that are later activated during post-training, with SFT and RL favoring different data regimes.

Our work complements these results by focusing on *positive transfer*: not just how pretrained skills are preserved, but how auxiliary-task pretraining can be amplified by subsequent RL. The mechanisms are naturally aligned. Auxiliary-task supervision installs a procedural prior over shared parts of the target trajectory, which RL then preserves (as in Shenfeld et al., 2026) and sharpens under task-specific rewards. This interaction is borne out in our controlled GPT-2 experiments (Section 4), where a modest pre-RL transfer gap ($+3$ pp) expands substantially after RL ($+13$ pp), consistent with the "activation of latent capability" perspective of Javanmard et al. (2026).

6. Conclusion

We studied when, and by how much, an auxiliary task A improves learning on a target task B. Within a minimal graph-theoretic abstraction, two key insights emerge. First, joint training does not dilute learning on shared components: auxiliary data can improve coverage of shared procedural structure without harming estimation where the two tasks overlap. Second, there exists a distinct *joint-dominance regime* in which neither task alone sufficiently covers the target trajectory, but their combination does. In this regime, each single-task baseline leaves part of the target rollout effectively unlearned, while joint training completes the missing coverage and restores teacher-level behavior along the full trajectory. When the action space is large, these uncovered regions induce an exponential degradation in success probability, yielding a sharp separation between joint and single-task training. Together, these results provide a concrete explanation for why training on $A{+}B$ can outperform training on B alone.

These predictions carry over to practice. Across three complementary settings—a controlled GPT-2 study on algorithmic tasks, a Qwen2.5-7B GRPO study on related math tasks, and a cross-representation procedural-reasoning benchmark—we observe consistent patterns of positive transfer, predicted failure modes, and asymmetric curriculum effects. Across all three settings, auxiliary tasks help precisely when they contribute missing portions of a shared procedural structure that the target task alone does not adequately cover. The experiments further show that this shared structure can appear both as an explicit sub-skill (e.g., GCD inside Comp) and as a latent procedural backbone expressed through multiple representations (graph versus natural language).

Limitations and outlook. Our analysis focuses on simplified graph structures and point start distributions; extending it to richer topologies, broader start distributions, and more complex auxiliary–target relationships remains an open direction. The current framework also treats supervised pretraining in isolation. A fuller account of the *pretraining* $\rightarrow$ *RL* pipeline—particularly why auxiliary-induced structure is amplified rather than overwritten during RL—requires modeling policy-gradient dynamics within the same abstraction. More broadly, our results suggest a practical principle for auxiliary-task selection: effective auxiliaries are not merely semantically related tasks, but tasks that expose procedural regions the target alone leaves under-covered. We hope this perspective helps guide auxiliary-data design in modern post-training pipelines, especially in settings where both supervision and reward are limited.

Acknowledgements

This research was supported by Cisco Systems, the NSF CAREER Award 2146492, NSF AI Institute for Foundations of Machine Learning (IFML) and NSF-Simons AI Institute for Cosmic Origins (CosmicAI).

We thank Allan Zhang for his contributions to the early codebase and tooling that this work built upon.

References

Agarwal, A., Song, Y., Sun, W., Wang, K., Wang, M., and Zhang, X. Provable benefits of representational transfer in reinforcement learning. In *Proceedings of the 36th Annual Conference on Learning Theory (COLT)*, volume 195 of *Proceedings of Machine Learning Research*, 2023. URL https://proceedings.mlr.press/v195/agarwal23b/agarwal23b.pdf.

Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and trends in Machine Learning*, 8(3-4): 231–357, 2015.

Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pp. 151–158. ACM, 1971. doi: 10.1145/3588287.3588297.

Gandhi, K., Lee, D. H. J., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. Stream of search (SoS): Learning to search in language. In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Huang, J., Zhou, Z., Xia, R., Mirzasoleiman, B., Su, W., and Huang, W. How transformers learn to plan via multi-token prediction. *arXiv preprint arXiv:2604.11912*, 2026a.

Huang, Y., Wen, Z., Singh, A., Chi, Y., and Chen, Y. Transformers provably learn chain-of-thought reasoning with length generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=aE0bCvXXBt.

Javanmard, A., Mirzasoleiman, B., and Mirrokni, V. Theoretical perspectives on data quality and synergistic effects in pre- and post-training reasoning models. *arXiv preprint arXiv:2603.01293*, 2026.

Kim, J., Wu, D., Lee, J. D., and Suzuki, T. Metastable dynamics of chain-of-thought reasoning: Provable benefits of search, RL and distillation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2HJcVtuovs.

Lai, S., Zhao, H., Feng, R., Ma, C., Liu, W., Zhao, H., Lin, X., Yi, D., Zhang, Q., Liu, H., et al. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, 2025.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Lin, F., Hofmann, V., Wan, X., Wang, W., Ding, Z., Cohn, A. G., and Pierrehumbert, J. B. Can large language models generalize procedures across representations? In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., et al. The flan collection: Designing data and methods for effective instruction tuning. In *International conference on machine learning*, pp. 22631–22648. PMLR, 2023.

Motwani, S. R., Ivanova, A., Cai, Z., Torr, P., Islam, R., Shah, S., de Witt, C. S., and London, C. h1: Bootstrapping llms to reason over longer horizons via reinforcement learning. *arXiv preprint arXiv:2510.07312*, 2025.

Park, S., Kaur, S., and Arora, S. How does rl induce skill composition? a case study using countdown. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shenfeld, I., Pari, J., and Agrawal, P. RL's razor: Why online reinforcement learning forgets less. In *The Fourteenth International Conference on Learning Representations*, 2026.

Sun, Y., Hu, S., Zhou, G., Zheng, K., Hajishirzi, H., Dziri, N., and Song, D. Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., and Hajishirzi, H. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*, 2023.

Wang, Z., Nichani, E., Bietti, A., Damian, A., Hsu, D., Lee, J. D., and Wu, D. Learning compositional functions with transformers from easy-to-hard data. In Haghtalab, N. and Moitra, A. (eds.), *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pp. 5632–5711. PMLR, 30 Jun–04 Jul 2025. URL https://proceedings.mlr.press/v291/wang25a.html.

Yao, X., Ren, R., Liao, Y., Ding, L., and Liu, Y. Compositional generalization from learned skills via cot training: A theoretical and structural analysis for reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Ye, J., Gao, J., Gong, S., Zheng, L., Jiang, X., Li, Z., and Kong, L. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=NRYgUzSPZz.

A. Per-column identification

This appendix proves the per-column decoupling and identification of the population estimators.

Lemma A.1 (Column decoupling)**.** *For $\diamond \in \{A, B, \text{joint}\}$,*

$$\mathcal{L}_\diamond(W) = \sum_{v \in V} \mathbb{I}\left[v \in C_\diamond\right] \cdot D_{\mathrm{KL}}\big(P^\star(\cdot \mid v) \,\big\|\, \mathrm{softmax}(W_{\cdot,v})\big),$$

Hence $\mathcal{L}_\diamond$ decomposes additively across columns and the optimization decouples per-column.

Proof. $p_W(\cdot \mid v) = \mathrm{softmax}(W_{\cdot,v})$ depends only on column $W_{\cdot,v}$ via the one-hot embedding. Substituting into the NTP definition gives the decomposition. □

Lemma A.2 (Per-column convexity)**.** *For $w \geq 0$, the function $F_v(z) := w \cdot D_{\mathrm{KL}}\big(P^\star(\cdot \mid v) \,\big\|\, \mathrm{softmax}_V(z)\big)$ on $\mathbb{R}^V$ is convex. It is shift-invariant: $F_v(z + c\mathbf{1}) = F_v(z)$ for all $c \in \mathbb{R}$. The Hessian has $\mathbf{1}$ in its null space, but is strictly positive definite on $\mathbf{1}^\perp = \{u \in \mathbb{R}^V : \mathbf{1}^\top u = 0\}$ when $w > 0$.*

Proof. Drop the z-independent term $\sum_u P^\star(u \mid v) \log P^\star(u \mid v)$:

$$F_v(z) = w \cdot \mathrm{LSE}(z) - w \sum_{u \in V} P^\star(u \mid v) z_u + \text{const},$$

where $\mathrm{LSE}(z) := \log \sum_{u \in V} e^{z_u}$. The Hessian of LSE is

$$\nabla^2 \mathrm{LSE}(z) = \mathrm{diag}(q(z)) - q(z)q(z)^\top, \qquad q(z) := \mathrm{softmax}_V(z),$$

the covariance matrix of the categorical distribution $q(z)$, which is positive semi-definite. So F_v is convex on $\mathbb{R}^V$.

Shift-invariance: $\mathrm{LSE}(z + c\mathbf{1}) = \mathrm{LSE}(z) + c$, and the linear term shifts by $c \cdot w \sum_u P^\star(u|v) = cw$. The two contributions cancel, giving $F_v(z + c\mathbf{1}) = F_v(z)$.

For positive definiteness on $\mathbf{1}^\perp$: the null space of $\mathrm{diag}(q) - qq^\top$ is exactly $\mathrm{span}(\mathbf{1})$ (since $\mathrm{softmax}_V(z)$ has full support, $\mathrm{Var}_q(u) = 0$ forces u constant). So $\nabla^2 F_v(z)$ restricted to $\mathbf{1}^\perp$ is strictly positive definite when $w > 0$. □

Lemma A.3 (Population identification)**.** *For $\diamond \in \{A, B, J\}$, the minimizer $W^\diamond := \arg\min_W \mathcal{L}_\diamond(W)$ subject to $W_{\cdot,v} = 0$ for $v \notin C_\diamond$ is unique on $\mathbf{1}^\perp$ per column, with*

$$p_{W_\diamond}(\cdot \mid v) = \begin{cases} P^\star(\cdot \mid v) & \text{if } v \in C_\diamond, \\ U_V & \text{if } v \notin C_\diamond, \end{cases}$$

where U_V is the uniform distribution on the entire vertex set V, with $U_V(u) = 1/|V|$ for every u.

Proof. By Lemma A.1, $\mathcal{L}_\diamond$ decouples across columns: the column $W_{\cdot,v}$ enters only through the term $\mathbb{I}\left[v \in C_\diamond\right] \cdot D_{\mathrm{KL}}(P^\star(\cdot \mid v) \,\|\, \mathrm{softmax}(W_{\cdot,v}))$, so we can minimize each column independently.

Uncovered ($v \notin C_\diamond$). The loss does not constrain $W_{\cdot,v}$. The convention sets $W_{\cdot,v} = 0$, giving $\mathrm{softmax}(0) = U_V$.

Covered ($v \in C_\diamond$). The per-column term $D_{\mathrm{KL}}(P^\star(\cdot \mid v) \,\|\, \mathrm{softmax}(z))$ is minimized when $\mathrm{softmax}(z) = P^\star(\cdot \mid v)$. Since $P^\star(\cdot \mid v)$ has full support on V (because $\alpha \in (0, 1)$ gives $P^\star(u \mid v) > 0$ for every u), this is achievable at the finite logit vector

$$z_u = \log P^\star(u \mid v) - \tfrac{1}{|V|} \sum_{u'} \log P^\star(u' \mid v).$$

The KL is strictly convex on the simplex, so $\mathrm{softmax}(W^\diamond_{\cdot,v}) = P^\star(\cdot \mid v)$ is the unique minimum, and the zero-sum convention makes $W^\diamond_{\cdot,v}$ unique on $\mathbf{1}^\perp$.

□

B. Pretraining convergence proof

We focus on the covered case $v \in C_\diamond$; abbreviate $p^\star := P^\star(\cdot \mid v)$ and $q(z) := \mathrm{softmax}(z)$, so $F_v(z) = D_{\mathrm{KL}}(p^\star \,\|q(z))$.

Lemma B.1 (Smoothness). *$\nabla F_v(z) = q(z) - p^\star$ and $\nabla^2 F_v(z) = \mathrm{diag}(q(z)) - q(z)q(z)^\top \preceq I$. Hence F_v is convex and 1-smooth.*

Proof. The first variation of $\log \sum_u e^{z_u}$ is q and its Hessian is $\mathrm{diag}(q) - qq^\top$ (covariance of the categorical q, hence PSD). Bound: $\mathrm{diag}(q) - qq^\top \preceq \mathrm{diag}(q) \preceq I$ since $q_u \in [0,1]$. □

Lemma B.2 (Bounded optimum). *Because $p^\star$ has full support on V ($p^\star_u \in \{\alpha, (1-\alpha)/(|V|-1)\} > 0$), F_v has a unique minimizer $z^* \in \mathbf{1}^\perp$ at which $q(z^*) = p^\star$, with $\|z^*\|_2^2 \le \log^2(|V|/(1-\alpha)) = O(\log^2 |V|)$.*

Proof. The unique minimizer of KL on the simplex is at $q = p^\star$, achieved by any z with $z_u = \log p^\star_u + c$; the zero-sum representative ($\mathbf{1}^\top z = 0$) is unique. Concretely, $z^*_u = \log p^\star_u - \overline{\log p^\star}$ where $\overline{\log p^\star} := \frac{1}{|V|}\sum_u \log p^\star_u$. The two distinct values are

$$z^*_{f(v)} = \log\alpha - \overline{\log p^\star}, \qquad z^*_u = \log\tfrac{1-\alpha}{|V|-1} - \overline{\log p^\star} \text{ for } u \neq f(v).$$

Their difference is $\log \frac{\alpha(|V|-1)}{1-\alpha}$. Since $\sum_u z^*_u = 0$ and exactly one entry is large, $\|z^*\|_2 \le |z^*_{f(v)} - z^*_u| = \log\frac{\alpha(|V|-1)}{1-\alpha} \le \log(|V|/(1-\alpha))$. Squaring gives the bound. □

Lemma B.3 (Per-column convergence). *Run GD on F_v with step size $1/L$ ($L = 1$) from $z^{(0)} = 0$. Then for any $T \ge 1$,*

$$\sup_{u\in V} \left|q(z^{(T)})_u - p^\star_u\right| \;=\; \widetilde{O}\left(\sqrt{\tfrac{1}{T}}\right),$$

with $\widetilde{O}$ hiding $\mathrm{polylog}(|V|)$.

Proof. By Lemma B.1, F_v is convex and 1-smooth. Standard convergence (Bubeck, 2015, Theorem 3.3) for the last GD iterate from $z^{(0)} = 0$ gives

$$F_v(z^{(T)}) - F_v(z^*) \;\le\; \frac{2L\|z^*\|_2^2}{T}.$$

By Lemma B.2, $F_v(z^*) = 0$ and $\|z^*\|_2^2 = O(\log^2|V|)$, so $F_v(\bar z^{(T)}) \le O(\log^2|V|/T)$. Therefore, we have

$$D_{\mathrm{KL}}\big(p^\star \,\|\, q(\bar z^{(T)})\big) \;=\; F_v(\bar z^{(T)}) \;\le\; O\left(\frac{\log^2|V|}{T}\right) \;=\; \widetilde{O}(1/T).$$

Pinsker gives $\|q(\bar z^{(T)}) - p^\star\|_{\mathrm{TV}} \le \sqrt{\frac{1}{2}D_{\mathrm{KL}}} = \widetilde{O}(1/\sqrt{T})$, and $\sup_u |q_u - p^\star_u| \le \|q - p^\star\|_{\mathrm{TV}}$. □

Proof of Theorem 3.1. Apply Lemma B.3 per column. Coverage and uncovered cases match the population identification (Lemma A.3). □

C. Proof of the path-accuracy comparison

Proof of Corollary 3.2. By Lemma A.3, the effective rollout kernel under estimator $\diamond$ is $P^\star(\cdot|v)$ on $C_\diamond$ and U_V on $V \setminus C_\diamond$. By the teacher's α-accuracy, $P^\star(f(v)|v) = \alpha$ at every covered vertex on target's path. Under U_V, the forward neighbor $f(v)$ is hit with probability exactly $1/|V|$.

Joint. The coverage facts in the setup (C_B = first d_B vertices, C_A = trunk) and $d_B + r_{\mathrm{s}} \ge \Delta_B$ implies that the whole path for task B is covered in the joint training, so every step is teacher:

$$\mathrm{Succ}(W^J) = \alpha^{\Delta_B}.$$

Target-only. Target covers the first d_B vertices of arm B; the remaining $\Delta_B - d_B$ steps are uncovered. Per-step accuracy is α on the prefix and at most q on the rest:

$$\mathrm{Succ}(W^B) = \alpha^{d_B}(1/|V|)^{\Delta_B - d_B}.$$

Aux-only. Aux covers the trunk (last r_{s} steps); the first $\Delta_B - r_{\mathrm{s}}$ steps on arm B are uncovered:

$$\mathrm{Succ}(W^A) = \alpha^{r_{\mathrm{s}}}(1/|V|)^{\Delta_B - r_{\mathrm{s}}}.$$

□

D. Experimental details

This section provides implementation details and full experimental configurations for reproducibility.

D.1. GPT-2 setup

Dataset. Both tasks use a shared symbolic vocabulary: `[GOAL]`, `[STATE]`, `[STEP]`, `[EVAL]`, `[RES]`, `[BACKTRACK]`, `[BUDGET]`, `[END]`. A trace alternates between state expansion, step execution, and verification. Invalid steps trigger backtracking and budget reduction. Despite identical syntax, Countdown performs arithmetic search, while 3-SAT performs logical propagation, sharing a DFS-like structure.

Training details. We use a 3-layer GPT-2 (1.7M parameters) trained from scratch using `nanochat`[1]. Pretraining uses 4,000 steps over 1.05B tokens with batch size 262K and sequence length 1024 on 8 NVIDIA RTX A5000 GPUs. RL fine-tuning uses REINFORCE on 3-SAT for 750 steps with binary verifier reward and a small termination bonus, also on 8 NVIDIA RTX A5000 GPUs.

Pretraining mixture. The pretraining mixture composition is given in Table 1 (main text). The only difference between conditions is the inclusion of full DFS traces from Countdown. No full 3-SAT DFS traces are included, making 3-SAT evaluation strictly zero-shot. Full DFS-trace examples for both tasks are shown in Figure 1 (main text); below we provide the corresponding partial-snippet samples used in both pretraining conditions.

Countdown snippet

```
[STATE] 6 13 19
[STEP] 19 - 6 [EVAL] 13 13
[STATE] 13 13
[STEP] 13 + 13 [EVAL] 26
```

3-SAT snippet

```
[STATE] !v2 | !v3 | !v4 & !v2 | !v3 | v4 & v1 | !v3 | v4
[STEP] v2 T [EVAL] !v3 | !v4 & !v3 | v4 & v1 | !v3 | v4
[STATE] !v3 | !v4 & !v3 | v4 & v1 | !v3 | v4
[STEP] v3 T [EVAL] !v4 & v1 | v4 & v4
[STATE] !v4 & v1 | v4 & v4
[STEP] v4 T [EVAL] F
```

Curriculum RL details. We evaluate RL curricula starting from both pretrained checkpoints. We compare: (i) single-task RL, (ii) forward curriculum (Countdown → 3-SAT), (iii) reverse curriculum. All runs use identical REINFORCE hyperparameters.

D.2. Qwen / GCD-Comp setup

Dataset. We adapt Omega (Sun et al., 2025) to construct two tasks: GCD and Comp (composition of root-finding and GCD). We modify the problem construction to ensure meaningful problem difficulty for easy and hard regimes. GCD difficulty is controlled by the magnitude of the input pair (a, b), while Comp difficulty is controlled jointly by the polynomial's rational-root search space and the magnitude of the GCD inputs.

Examples. Easy variants of GCD and Comp are shown in Figure 4 (main text). The corresponding hard variants are below.

[1] https://github.com/karpathy/nanochat

GCD hard

Prompt: Calculate the greatest common divisor of 28967 and 14276.
Answer: 83

Comp hard

Prompt: Given the polynomial $5x^3 + 19x^2 - 109x + 21$, determine its largest root in the form n/m. What is the greatest common divisor of $14060 \cdot |n|$ and $69217 \cdot |m|$?
Answer: 19

Training details. We use Qwen2.5-7B-Instruct with GRPO via `verl`. LoRA ($r = 32$) is applied throughout. Training uses 16 rollouts per prompt with an KL regularization coefficient of 0.001, and all hyperparameters are identical across conditions to ensure a fair comparison. Evaluation uses pass@1 over 16 samples with deterministic grading. All Qwen2.5-7B-Instruct runs are conducted on 4 NVIDIA H100 GPUs.

D.3. AsyncHow setup

Dataset. AsyncHow (Lin et al., 2026) defines procedural reasoning over DAGs, represented as: (i) graph format, or (ii) natural language steps. Both encode the same underlying partial order. We use the official 1373/225 train-test split similar to (Agarwal et al., 2023). In addition, we define: (i) easy NL: graph size (total number of nodes and edges) ≤ 14 and (ii) hard NL: otherwise. This stratification controls reward density in RL training.

Training details. Follow (Lin et al., 2026), we train Qwen2.5-1.5B with GRPO using identical hyperparameters across conditions, on 4 NVIDIA H100 GPUs. All curriculum and single-stage runs are strictly compute-matched.

Full results. Table 2 reports the full results across all conditions, including the hard-NL and full-NL curriculum. The main text focuses on the easy-NL curriculum but the same patterns hold across these additional conditions.

Table 2. Full AsyncHow results.

Condition	NL pass@1	Graph pass@1
Graph-only	5.2	81.6
NL-only	62.5	22.5
Graph + NL (full)	**71.9**	**82.7**
Graph → NL (easy)	67.5	68.1
NL (easy) → Graph	45.5	80.7
Graph → NL (hard)	35.3	72.3
Graph → NL (full)	53.8	68.8