UNDERSTANDING THE BENEFITS OF SCAFFOLDS THROUGH COMPUTATIONAL SEPARATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference-time harnesses, i.e., external procedures that call a model many times and keep state across calls, can substantially improve model capabilities. In this paper, we investigate the benefits of using such harnesses and the extent to which these benefits can be internalized through training. More precisely, we conceptualize a harness $\mathcal{A}$ as an arbitrary wrapper around a base model M and attempt to distill a student model from the harness's traces. Empirically we find that, even with simple scaffolds and small models, the inference-time harness significantly outperforms a distilled student. We then prove that such a gap is due to a fundamental representational limitation: inference-time harnesses can compute a closure of their per-call primitive that is NL-complete even when the primitive is a trivial AC^0 lookup, and that no fixed-depth forward pass can represent under the widely believed conjecture that $\mathsf{NL} \not\subseteq \mathsf{TC}^0$. Our theory suggests that expanding the representational capacity of the student relative to the task at hand, e.g., by enlarging the model, making the task easier, or lengthening its chain-of-thought, is necessary to close the gap, which we verify empirically. Finally, we scale to an agentic harness on competition math, where the same gap reappears. Overall, our results show that a harness confers on the base model a profound gain in representational power, one that training a same-class student on its traces cannot transfer back into the weights.

1 INTRODUCTION

Much of the recent progress on challenging reasoning and agentic tasks depends on the workflow surrounding a language model. An *inference-time harness* (or *scaffold*) keeps the underlying model fixed, but calls it repeatedly, maintaining state and choosing later queries based on earlier answers. Harnesses range from sampling and selecting complete solutions (Wang et al., 2022; Cobbe et al., 2021) to searching over partial solutions (Yao et al., 2023), backtracking with a verifier (Rohatgi et al., 2026), coordinating agents (Du et al., 2023), and incorporating tools and environment feedback (Yao et al., 2022).

There is something puzzling about how elementary some of these strategies are—and why they should improve the performance of already strong models. For example, early systems achieving IMO gold-level performance (Huang & Yang, 2025) and autonomous mathematics research systems like Aletheia (Feng et al., 2026) use designed, but simple loops of maintaining a candidate set of solutions, expanding them, and pruning them by verification occasionally. Why should a model capable of sophisticated mathematical reasoning benefit so much from having such a simple strategy imposed from outside?

Moreover, considering that simplicity of design is often desirable, the fact that carefully designed harnesses are needed to achieve strong performance raises the question of whether one can expect a model to learn such strategies on its own, thereby internalizing the benefits of a harness. Indeed, Cundy & Ermon (2024) train models to perform backtracking themselves and Gandhi et al. (2024) to search by learning from serialized search traces. It is unclear, however, when one can expect this internalization to be successful and when a practitioner would be better off simply using a harness at inference-time.

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.

There is likely more than one reason why a harness can improve performance: for example, it may increase the effective capacity of the underlying model, provide a form of error correction, or provide implicit decomposition of the target task into semantically meaningful subtasks (Zhang et al.). While these intuitions are plausible, turning them into a predictive theory that can guide harness design requires precise hypotheses that can be tested quantitatively and, ideally, proved under suitable assumptions. The prevalence and success of harnesses make this an increasingly important task.

In this paper, we do examine the success of inference-time harnesses through a computational lens: we show a model capable of executing each individual step of a harness may nevertheless be unable to represent the computation performed by the composition represented by the harness. We first investigate this hypothesis experimentally, then give a theoretical setting in which such internalization is *provably* impossible under standard computational modeling assumptions of the base model.

We first perform controlled experiments on Dyck languages (a formal language of valid nested parentheses) and Sokoban (a maze-like puzzle game), where we can train a model from scratch and vary the difficulty of generalization. We train a base model on ground-truth demonstrations, then augment it with a harness: either a principled backtracking algorithm (VGB) introduced by Rohatgi et al. (2026), or A* search with model-pruned expansions (Hart et al., 1968; Lehnert et al., 2024). Together, the harness and base model specify a teacher whose behavior we attempt to reproduce with a student of the same size and model class as the base model. The student is trained on the teacher's traces, either from scratch or by fine-tuning the base model. In both cases, a substantial gap remains, particularly out of distribution. Changing the amount and composition of training data, the trace format, or the number of training iterations does not close this gap in the settings we test.

Why does this training fail to reproduce the teacher's behavior? One possible explanation is that a harness fundamentally increases the representational capacity of the base model, so that reproducing the harness requires a larger model than executing its individual calls. We investigate this by increasing the student's size while keeping the teacher fixed: sufficiently large students match and eventually exceed their teacher's performance. We also test a complementary hypothesis that a simpler harness should be easier for the original student to internalize. Specifically, we replace verifier-controlled backtracking with a rule that deletes a fixed number of tokens and regenerates. The same-sized student can match this simpler teacher.

Model size is not the only reasonable consideration: a student may also need more "reasoning steps" to carry out the harness's computation. Indeed, recent work has shown that increasing the length of a chain of thought can be understood to mimic the representational capacity of a much deeper model (Merrill & Sabharwal, 2024; Li et al., 2024; Merrill & Sabharwal, 2026; Amiri et al., 2025). In particular, on modular arithmetic, we find that training a model to use a longer chain of thought lets it match and eventually exceed the accuracy obtained by augmenting a shorter-chain model with verifier-guided search. These experiments suggest that successful internalization depends on the student's size, the number of reasoning steps it can take, and the complexity of the strategy it must reproduce.

These observations motivate a formally provable *representational capacity* separation. We consider directed s–t reachability (Jones, 1975), the task of path-finding in a graph. To model a search harness, we think of vertices as reasoning states and edges as valid transitions. The base model only answers local edge queries, which are adjacency-matrix lookups computable by constant-depth Boolean circuits (AC^0). The harness maintains a search frontier and visited set, using these queries to perform breadth-first search and decide reachability in polynomial time.

Internalizing the harness requires the student to decide reachability from the graph itself. Under standard precision assumptions, fixed-depth transformers of polynomial size can be simulated by constant-depth threshold circuits (TC^0), even with logarithmic-length chain-of-thought (Li et al., 2024; Merrill & Sabharwal, 2026). Since directed reachability is NL-complete, the widely believed computational complexity separation $\mathsf{NL} \not\subseteq \mathsf{TC}^0$ implies that no such student can reproduce the harness on all inputs. The separation thus captures a concrete obstacle to internalization: composing local decisions through an external search procedure can exceed the students' representational capacity.

Finally, we examine whether the same pattern persists with an agentic harness around a pretrained language model. We use Branching (Bateni et al., 2025), which organizes model calls into a tree:

leaves propose solutions independently, and internal nodes synthesize new solutions from their children. Using Qwen3-1.7B on competition mathematics (DAPO-Math-17k), we flatten the teacher's runs into transcripts and LoRA fine-tune the same backbone to reproduce them in a single autoregressive rollout. As the branching factor d and depth L increase, the teacher's accuracy improves while the student's eventually plateaus, widening the gap. Increasing the LoRA rank yields only modest gains, whereas enlarging the backbone closes the gap. These results support the same interpretation as our controlled experiments and theory: internalizing a harness may require a larger model than the one used in the harness.

1.1 Our contributions

Student of the same size cannot internalize natural inference-time harnesses. We generate training data from traces of a model M used inside a harness, and perform supervised finetuning (SFT) of M on them. Across two natural harnesses: a principled backtracking algorithm (VGB) (Rohatgi et al., 2026) and A* search (Hart et al., 1968); a set of natural, extensive ablations varying the amount of traces, training iterations, trace format, amount of out-of-distribution data—the resulting student never closes the gap to its teacher.

The gap closes with relative expressive power of the student. We show empirically that the gap closes as the student's expressive power grows relative to the task—for example a larger student, an easier scaffold, or a longer chain-of-thought can all close this gap.

Formal representational capacity obstruction to internalizing a harness. We consider directed s–t reachability on a graph as a paradigmatic problem modeling path-finding. A base model that answers local edge queries—an AC^0 computation—can be used by a polynomial-time harness to run breadth-first search and decide reachability. On the other hand, a student with an $O(\log n)$-length chain-of-thought provably stays inside TC^0. By contrast, under standard precision assumptions, a fixed-depth, polynomial-size student remains in TC^0 even with an $O(\log n)$-length chain-of-thought. Thus, assuming $\mathsf{NL} \not\subseteq \mathsf{TC}^0$, no such student can internalize the harness.

A similar gap appears with an agentic harness. We distill the Branching harness (Bateni et al., 2025) on competition math using Qwen3-1.7B. The harness reaches 43.6% accuracy with 85 model calls, while a single-pass student saturates near 36%. Increasing the LoRA rank improves the student by only 2.2 percentage points; increasing its backbone from 1.7B to 4B closes the gap.

1.2 Preliminaries and notation

Teacher, scaffold, and student. Throughout the paper we fix a base autoregressive model M (the *base model*), trained on clean demonstrations of a task with standard next-token prediction. An *inference-time harness* (or *scaffold*) $\mathcal{A}$ is any algorithm that can call M as an oracle. The resulting teacher is $\mathcal{A}[M]$. A run of $\mathcal{A}$ produces a *trace*—the full sequence of calls and answers. The specific form of this trace will vary with the choice of scaffold. A *student* S is a model trained by next-token prediction on these traces. We ask whether S can *internalize* $\mathcal{A}$—i.e. match the accuracy of $\mathcal{A}[M]$ from one forward pass.

Computational complexity classes. We recall several definitions of uniform circuit classes. AC^0 denotes constant-depth, polynomial-size circuits with unbounded fan-in AND/OR/NOT gates. TC^0 adds unbounded fan-in threshold (majority) gates. Throughout the formal analysis, these families are logspace-uniform: a deterministic procedure generates the circuit from 1^n using $O(\log n)$ workspace. The classes L and NL consist of languages decided using deterministic and nondeterministic logarithmic space, respectively. We recall the following standard inclusions:

$$\mathsf{AC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P}.$$

2 A Same-Size Student Cannot Recover the Teacher's Performance

We begin with empirical observations derived in simple, but clean settings in which we can train a student model from scratch, and perform thorough sweeps and ablations across various relevant

aspects of the setup. Concretely, we consider harnesses inspired by search algorithms (principled backtracking based on (Rohatgi et al., 2026) and A*), for the tasks of generating valid strings of nested parentheses (i.e. Dyck languages) and Sokoban (a maze-like puzzle game) respectively—and ask whether distillation can transfer the harness's gains to a student of the same size as the base model. We find that students trained from scratch perform far below even the base models; students fine-tuned from the base models also cannot reach the teachers' gains.

2.1 SETUP

We use two tasks with an easy-to-control in-distribution (ID) / out-of-distribution (OOD) instances.

1. **Dyck**: sequences of properly nested brackets, with every opening bracket matched by a closing bracket of the same type. We use 4 bracket types and sequences of length 64. The ID distribution draws bracket types with probabilities $[0.1, 0.2, 0.3, 0.4]$, while the OOD distribution reverses these probabilities to $[0.4, 0.3, 0.2, 0.1]$.
2. **Sokoban**: a puzzle in which a player moves through a grid and pushes a box onto a target, with walls blocking the movement. We use a 5×5 grid with one player, one box, one target, and up to three walls, with actions $\{U, D, L, R\}$. ID instances have short solutions, while OOD instances are biased toward longer ones (reversed solution-length buckets).

We consider the following two scaffolds:

1. **VGB** (Rohatgi et al., 2026). The base model proposes an action $a \sim M(\cdot \mid s)$, a verifier scores it, and the step is committed or retracted (<BT>) according to the Metropolis-style ratio
$$p_{\text{retract}} \;=\; \max\big(\varepsilon,\; 1 - \widetilde{V}(s,a)/\widetilde{V}(s)\big).$$
Here $\widetilde{V}$ is the *policy-conditional* success probability—how likely the base model is to produce a valid solution from the current prefix—so the ratio measures the progress the proposed action makes:
$$\widetilde{V}(s) = \mathbb{E}_{\tau\sim M(\cdot|s)}\big[\mathbf{1}[\text{valid}(s\circ\tau)]\big], \qquad \widetilde{V}(s,a) := \widetilde{V}(s\circ a).$$
2. **A* search** (Hart et al., 1968) maintains a frontier of candidate states and repeatedly expands the state with the smallest $f = g + h$. Here g is the number of moves taken so far, and h is the Manhattan distance from the box to the target. At each expansion, we retain only the top-k=2 successor actions according to base model's logits. The trace records when a node enters the frontier (<CREATE>) and when it is expanded (<CLOSE>), along with its g, h, f values; it ends with the resulting <PLAN>. This search-log serialization follows Lehnert et al. (2024).
The trace prepends a quantized verifier bucket before each action or <BT> event, so the student sees the signal that drove every backtrack.

We defer the exact Metropolis derivation, verifier instantiations, and trace formatting to Appendix B.1.

Both student and teacher models are 2-layer Transformers with d_{model}=128 and 4 heads ($\approx$ 0.68M parameters), so the student is the *same size* as the model the teacher. The base model is trained on clean ID demonstrations. Students are trained in two ways: (i) *from scratch* on a pool of traces and (ii) *fine-tuned* from the base model on a mixture of traces and clean data. We report accuracy on held-out ID and OOD splits under greedy/top-p decoding (gen); unless otherwise noted, the teacher figure is the full scaffold $\mathcal{A}[M]$.

2.2 EXPERIMENTAL RESULTS

We first show that in both tasks, the student cannot recover the teacher's accuracy. We then perform ablations on the training data distribution, the number of training iterations, and the amount of training data that gives strong evidence that the gap between the student and the teacher is not caused by training dynamics.

Student Cannot Recover Teacher Model. Figure 1 compares the base model, the teacher $\mathcal{A}[M]$, and the two students in the three (algorithm, domain) settings. A student trained from scratch on the teacher's traces falls below the base model both ID and OOD, likely because the traces are a much

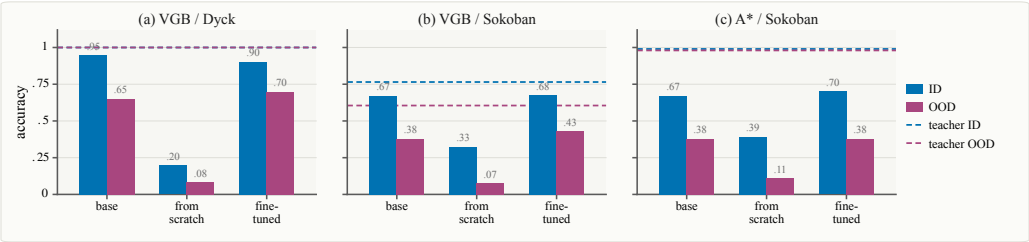


Figure 1: **Same-size students cannot recover the teacher.** *ID (blue) / OOD (magenta) accuracy of the base model and the two students; dashed lines denote the teacher.*

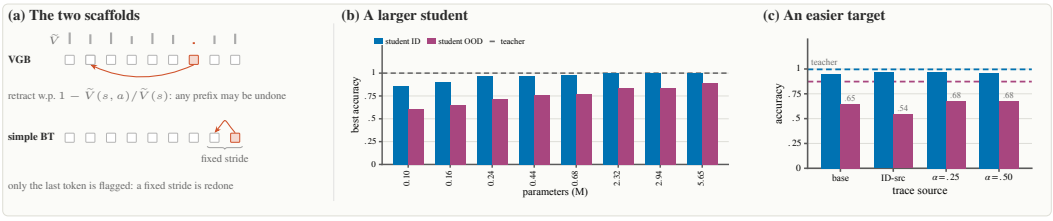


Figure 2: **Bigger students and a simpler scaffold narrow the gap.** *(a) The two scaffolds. (b) Accuracy against the student's parameter count. (c) Accuracy against the distribution the traces were drawn from; dashed lines mark the teacher.*

harder learning target than clean task data. A student fine-tuned from the base model stays close to the base model on both splits, but well below the teacher.

Ablations. We give strong evidence the gap is not due to a failure of training: we sweep various relevant aspects in the training setup, and the same-size student always stays below the teacher. Full grids, including four trace formats, are in Appendix B.3.

• *Fine-tuning sweeps.* Starting from the base model we vary, one axis at a time: the *clean-data ratio* mixed with VGB traces (0–100%), the number of *fine-tuning iterations* (3K–8K), the *number of traces* (1K–20K traces), and the *training-source mixture* $P_\alpha = (1-\alpha)P_{\text{ID}} + \alpha P_{\text{OOD}}$ of the puzzles the traces are drawn from ($\alpha \in \{0, 0.25, 0.5, 0.75\}$). Across all four axes the student tracks the base model and stays below the teacher dashed line (Figure 6, Appendix B.3.1). Adding more data, more iterations, and richer source mixtures change the student only marginally.

• *From-scratch sweeps.* Holding the student at the fixed teacher scale (0.68M), we sweep the remaining data and optimization knobs of the from-scratch training: random *seeds*, the training-source α-mix, and an *OOD-source* variant where the training traces themselves come from OOD instances. Across all of these sweeps, the gap between the student and the teacher remains (Figure 9, Appendix B.3.2).

3 The Student–Teacher Gap Closes as the Student's Expressive Power Increases

In Section 2 we systematically performed various changes to the training setup, but the student was always taken to be in the *same* model class as the model its teacher wraps. In this section, we remove that constraint and investigate the effect of allowing the student's expressive power to grow—relative to the hardness of the task. We find that the gap is not an intrinsic property of trace distillation: it shrinks, and eventually vanishes, along many axes that increase the student's representational capacity. We consider three ways of increasing the student's expressive power relative to the scaffolded teacher: enlarging the student's size (Section 3.1), replacing the scaffold with a simpler one (Section 3.2), and lengthening the student's chain-of-thought (Section 3.3).

3.1 Bigger Students Catch Up

We first consider increasing the student's expressive power by directly increasing its number of parameters.

Setup. Section 2 fixed the student at the teacher scale ($\approx$ 0.68M). Here we sweep 12 scales over two orders of magnitude in parameter count (0.10M–11.5M) and ask whether a *bigger* student in-

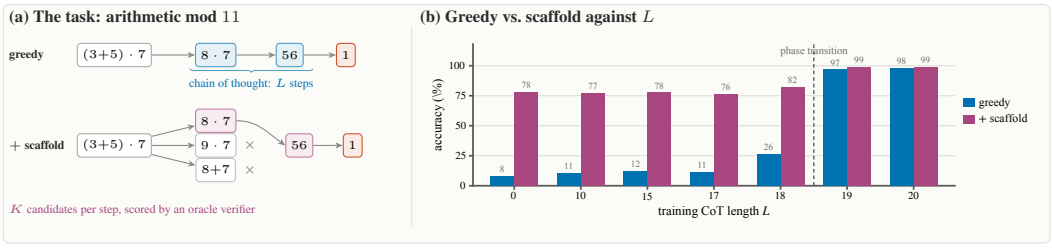


Figure 3: **A long enough chain-of-thought matches the scaffold.** *(a) The task. (b) Accuracy against the training chain-of-thought length L; the dashed line marks the phase transition.*

ternalizes the scaffold better. Each (scale, clean-ratio, iteration) cell is a fresh student trained on the same pool of traces as Section 2, evaluated on the held-out ID/OOD split. We report the per-scale best accuracy.

Results. As the student grows, its gap to the teacher shrinks on both ID and OOD: on VGB-Dyck (Figure 2b) and on A*- and VGB-Sokoban (Figure 10, Appendix C.1).

3.2 A Simpler Backtrack Is Internalized

Next, we consider making the target task easier: the teacher's scaffold is replaced by a simpler one, which implicitly increases the student's expressive power relative to the scaffolded teacher. We again find that the gap between the student and the teacher shrinks.

Setup. VGB and A*-search both maintain unbounded search state—the number of steps the teacher takes and the amount of history grow with the problem. If we instead consider a harness which is *bounded and local*, the student becomes larger in a relative sense. We test the simplest such scaffold on Dyck: single-step backtracking (**BT**), where a verifier flags the last token as erroneous and the model retracts a fixed stride and regenerates. The distilled student learns to emit its *own* `<BT>` tokens autoregressively—internalizing the policy with no external verifier—and we vary the source distribution of the distilled traces, an α-mixture between the ID and OOD bracket distributions.

Results. On ID, the student recovers the teacher's accuracy (Figure 2c), whereas a same-size student distilled from VGB on Dyck stays below its teacher (Section 2, Figure 1). On OOD, the student does not fully recover the teacher's accuracy, and how close it gets depends on the distribution the traces are drawn from: traces from the ID distribution alone recover little, while mixing in a small fraction of OOD traces (α=0.25) brings the student much closer to the teacher (Figure 2c). With this choice of training distribution, the gap between student and teacher is smaller than when the student learns VGB. The full grid, including single-pass accuracy and synthetic-BT fine-tunes, is in Table 2 (Appendix C.2).

3.3 Longer Chain-of-Thought

Prior theoretical work has theoretically shown that increasing the length of the chain-of-thought also increases a student's expressive power (Merrill & Sabharwal, 2024; Li et al., 2024). In this section, we consider a modular arithmetic task and show that the longer the student's chain-of-thought, the smaller the gap to the scaffolded teacher.

Setup. We consider modular arithmetic over $\mathbb{Z}_{11}$ (Feng et al., 2023), built up from numbers, arithmetic operators, and parentheses. A reduction step evaluates the leftmost subexpression that is ready to be computed and replaces it with its value modulo 11, until we are left with a single number. This rule gives each expression a prescribed chain of n_{ops} steps. We train a depth-3 transformer to produce only the first L steps of that chain and vary L to control how much of the computation it learns to perform. At inference time, the scaffold samples K=16 candidate next steps from the model. An oracle checks which candidates match the next valid reduction; the scaffold commits a valid candidate, or backtracks if all 16 fail.

Results. Figure 3b (n_{ops}=20) compares, at each training chain-of-thought length L, the model decoded greedily with the same model used in the scaffold. With the scaffold ("+ scaffold" bars), accuracy stays high at every L. Without it ("greedy" bars), a model trained on a short chain-of-thought falls far below the scaffold; its accuracy rises with L, sharply across the phase transition marked by the dashed line between L=18 and L=19, and at L=19 and L=20 it matches the scaffold.

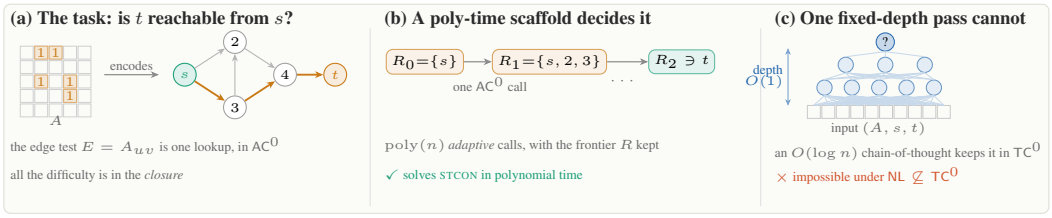


Figure 4: **Illustration of the directed s–t reachability task.** *(a) The path-finding task. (b) With an AC^0 edge oracle, a harness decides it in polynomial time. (c) A model with a bounded chain-of-thought cannot.*

The experiment therefore confirms that increasing the student's expressive power through a longer chain-of-thought closes its gap to the scaffolded teacher. The task definition, scaffold, and the n_{ops}=10 curve are in Appendix C.3.

4 An Expressivity Separation Between Scaffold and Student

The experiments in Sections 2 and 3 empirically show that a student trained on a scaffold's traces cannot match its teacher's accuracy when the student is only as expressive as the teacher model. Once the student's expressive power grows relative to the teacher's, however—through a larger model, a longer chain-of-thought, or an easier task—the gap between the two narrows. The natural hypothesis is therefore that the student cannot even *represent* what the computation of the scaffold. This section shows this hypothesis is formally true under natural mathematical modeling abstractions from complexity theory used to understand the representational power of Transformers (Merrill et al., 2022; Li et al., 2024).

4.1 Setup: Path-Finding in a Graph

We use directed s–t reachability (STCON, (Jones, 1975)) as a paradigm mathematical abstraction for the task of path-finding in a graph. An instance consists of a directed graph with adjacency matrix $A \in \{0,1\}^{N\times N}$, a source s, and a target t. The task is to decide whether there is a path $s = v_0 \to v_1 \to \cdots \to v_T = t$. Checking whether one proposed move is valid in the graph is easy: the local edge predicate $E_A(u,v) = A_{uv}$ is computable in AC^0. Indeed, $E_A(u,v) = \bigvee_{i,j\in[N]}(A_{ij} \wedge [u = i] \wedge [v = j])$, which is a constant-depth circuit of polynomial size because each equality test is an AND of $O(\log N)$ literals and the outer OR has N^2 terms. Reachability asks whether some sequence of such moves leads to t, so it requires combining many local answers.

Under standard complexity separation assumptions, there is a gap between the computational difficulties of checking for the validity of a single move and connectivity of s and t: while checking an edge is in AC^0, deciding whether a path exists is NL-complete even under AC^0 reductions (Immerman, 1987a). Hence, assuming NL $\not\subseteq$ TC^0, no TC^0 circuit can solve this task. This is helpful because our lower bound uses an expressivity upper bound for the student: under standard precision and uniformity assumptions, a fixed-depth transformer can be simulated by a TC^0 circuit (Merrill et al., 2022; Merrill & Sabharwal, 2023). The scaffold uses the model only for the simpler query $E(A,u,v) = A_{uv}$, while the student must decide reachability from the full input (A,s,t). Thus, even if we grant the student every computation in TC^0, the assumed separation NL $\not\subseteq$ TC^0 rules out solving all reachability instances.

In fact, the same machinery will be able to handle even chain-of-thought of logarithmic length. To that end, we'll define mathematically a model allowing chain-of-thought. Fix a finite alphabet Σ containing a padding symbol $\perp$, and an integer $c \geq 1$, and set $T_c(n) = \lceil c \log_2(n+1) \rceil$. Shorter runs are padded to this length, and the final predicate ignores the padding.

Definition 1 (CoT with TC^0 verification)**.** *A language L lies in $\exists\mathrm{CoT}_{T_c(n)}(\Sigma, \mathsf{TC}^0)$ if there are step predicates $V_{n,i}(x, z_{<i}, z_i)$, for $1 \leq i \leq T_c(n)$, and a final predicate $B_n(x,z)$ such that*

$$x \in L \iff \exists z \in \Sigma^{T_c(n)} : \quad B_n(x,z) \wedge \bigwedge_{i=1}^{T_c(n)} V_{n,i}(x, z_{<i}, z_i).$$

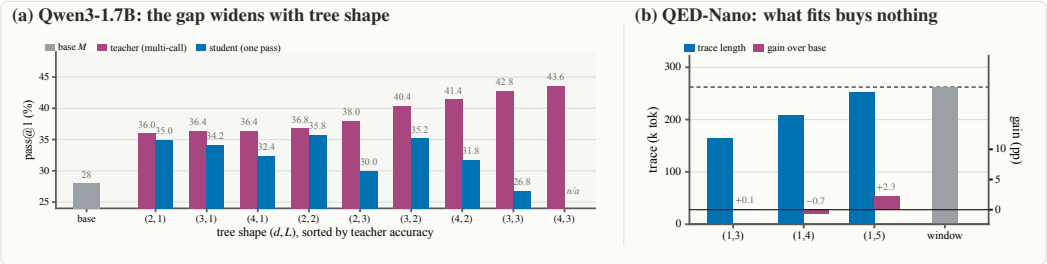


Figure 5: **Larger trees widen the gap.** *(a) Teacher and student accuracy on Qwen3-1.7B per tree shape (d, L). (b) Trace length and gain over the base model on QED-Nano.*

The predicates have a common constant depth bound and a common polynomial size bound in n. They are jointly logspace-uniform: one logarithmic-space procedure, given $(1^n, i)$, generates the circuit for $V_{n,i}$, and another, given 1^n, generates the circuit for B_n.

This abstraction includes deterministic autoregressive generation: $V_{n,i}$ checks that z_i is the token prescribed by the model on the preceding prefix, and B_n checks whether the final answer is yes. Allowing any transcript satisfying these checks can only enlarge the class of student computations.

4.2 The Separation

First, we show that adding logaritmically bounded CoT results in a model still in TC^0, using the same transcript-enumeration argument as Merrill & Sabharwal (2025, Theorem 4 and Appendix E):

Lemma 1 (Short CoT remain in TC^0)**.** *For every fixed finite alphabet Σ and integer $c \geq 1$,*

$$\exists \mathrm{CoT}_{T_c(n)}(\Sigma, \mathsf{TC}^0) \subseteq \mathsf{TC}^0.$$

There are only $|\Sigma|^{T_c(n)} = \mathrm{poly}(n)$ possible transcripts. A circuit checks each transcript in parallel and accepts if one passes every check, increasing the depth only by two. Appendix D.2 contains the complete proof.

Theorem 1 (Scaffold–student separation)**.** *Assume $\mathsf{NL} \not\subseteq \mathsf{TC}^0$. For* STCON *on an N-vertex graph given by its adjacency matrix A, the following hold:*

(i) The local predicate $E(A, u, v) = A_{uv}$ lies in AC^0. Given N, s, t and oracle access to E_A, a deterministic polynomial-time breadth-first-search scaffold decides reachability using at most N^2 oracle calls and $O(N)$ bits of external state.

(ii) For every fixed finite alphabet Σ and integer $c \geq 1$,

$$\text{STCON} \notin \exists \mathrm{CoT}_{T_c(n)}(\Sigma, \mathsf{TC}^0), \qquad T_c(n) = \lceil c \log_2(n+1) \rceil,$$

where n is the encoded input length and the predicates satisfy Definition 1.

Part (i) is the standard breadth-first-search algorithm. For part (ii), Lemma 1 would put a successful student in TC^0. Composing it with the uniform AC^0 reductions to STCON would then put every language in NL in TC^0, contradicting the assumption. The full proof is in Appendix D.3.

The theorem can be seen as giving an example of a natural task on which a scaffold simulating a search procedure combines elementary model calls into a computation that no student in the stated class can reproduce on all inputs, even with $O(\log n)$ chain-of-thought tokens. This is a representational obstruction to internalization. Commensurate with the experiments, longer chains of thought or increase in depth can change asymptotic expressive power (Merrill & Sabharwal, 2024; 2025).

5 Agentic Scaffolding

This section moves from simple, controlled tasks (Dyck, Sokoban) to more realistic ones: a **Branching** scaffold (Bateni et al., 2025) applied to competition mathematics problems. We show that the gap between student and teacher widens as the number of calls grows.

Setup. The base model M is Qwen3-1.7B (`enable_thinking=False`); the task is DAPO-Math-17k, with correctness judged by `math-verify` on the last `\boxed{...}`. We distill from a 1500-problem pool of traces (cached pass@1 ≈ 0.30) and evaluate on a disjoint 500-problem set (pass@1 ≈ 0.31).

Branching scaffold. The Branching scaffold calls M in a depth-L tree: the leaves call M on the question at temperature 1.0, and each internal node calls M once to synthesize a *fresh* solution from its children. The canonical $\mathbf{ks}=[3,3]$ tree uses 9 leaves $\rightarrow$ 3 aggregators $\rightarrow$ 1 root (13 calls, no verifier). The distillation-knob ablations use the smaller $\mathbf{ks}=[3]$ variant (3 leaves $+1$ root, 4 calls); the shape sweep varies the topology of the full tree. We flatten the transcript (`## Attempt` / `## Final Solution`) and LoRA fine-tune M to emit the whole trace, and hence the answer, in one autoregressive rollout, keeping only root-correct runs.

Experimental Result. We distill Branching trees of increasing size into a same-size student, varying the number of children $d \in \{2,3,4\}$ and the depth $L \in \{1,2,3\}$ (Figure 5a; full grid in Figure 14). When d and L are small (left of Figure 5a), the student recovers most of the teacher's gain over the base model in one autoregressive rollout. This agrees with Section 3.2 (Figure 2c), where a simpler backtracking rule narrows the teacher–student gap. As d and L grow (right of Figure 5a), the teacher keeps improving as calls are added, while the unscaffolded student does not, and it collapses once the tree reaches depth three. Thus, larger teacher trees are harder to reproduce with the tested same-size students.

Ablation. We compare changes to the student backbone, the number of trainable parameters, and the data mixture (full results in Appendix E.1).

• *Backbone size has the larger effect.* We compare enlarging the backbone with increasing LoRA rank and with full fine-tuning at fixed backbone size. Moving from the 1.7B to the 4B backbone closes the gap and surpasses the teacher (Figure 15a), whereas raising the LoRA rank, or fine-tuning every weight, makes little difference (Figure 15b).

• *The tested clean-data mixtures do not close the gap.* The student tolerates about half clean data, then starts to fall back toward the clean-data baseline, and no mixture can reache the teacher's performance (Figure 13a).

A harness on a strong prover relative to model size. Unlike the off-the-shelf Qwen3-1.7B used above, QED-Nano (Qu et al., 2026) is a 4B prover built specifically for Olympiad proofs, trained on solutions from a 685B model and then refined by reinforcement learning, which makes it unusually strong relative to its size. After RL, the released checkpoint is used in a multi-call scaffold at evaluation time, and its strongest reported results use that scaffold. We put Branching on that same RL checkpoint and sweep the tree shape, scoring both arms of the same trace on 300 problems from the model's own RL corpus. Across the shapes that stay within seven calls in total, none achieves a gain whose confidence interval is above zero (Figure 5b, Appendix E.2). A clear gain does appear once we go to d=3, L=3, but by then the estimated total trace length is more than five times the context window considered for distillation. Within this sweep, the shape with a clear gain also has the longest estimated trace. This creates an obstacle to distilling the complete trace within that context budget. We do not train a QED-Nano student in this experiment.

6 Conclusion

We studied when a model can internalize the benefits of an inference-time harness. Across controlled tasks and agentic scaffolds, we find that same-size students remain below their teachers in our experiments, while larger students, longer chains of thought, and simpler scaffolds can narrow or close the gap. We formalize a representational obstruction that explains these observations: under $\mathsf{NL} \not\subseteq \mathsf{TC}^0$, a scaffold simulating a search procedure exceeds the expressive power of a short-CoT student class. We thus clarify computational benefits of scaffolds.

AI use statement

We did not use AI to plan or design the experiments in this work. However, we used AI tools for coding and for writing assistance. We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

REFERENCES

Alireza Amiri, Xinting Huang, Mark Rofin, and Michael Hahn. Lower bounds for chain-of-thought reasoning in hard-attention transformers. 2025.

MohammadHossein Bateni, Vincent Cohen-Addad, Yuzhou Gu, Silvio Lattanzi, Simon Meierhans, and Christopher Mohri. Algorithmic thinking theory. *arXiv preprint arXiv:2512.04923*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

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

Chris Cundy and Stefano Ermon. Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking. In *International Conference on Learning Representations*, volume 2024, pp. 3387–3407, 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. 2023.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. volume 36, pp. 70757–70798, 2023.

Tony Feng, Trieu H Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang-hyun Kim, Federico Pasqualotto, et al. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Yichen Huang and Lin F Yang. Winning gold at imo 2025 with a model-agnostic verification-and-refinement pipeline. *arXiv preprint arXiv:2507.15855*, 2025.

Neil Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4): 760–778, 1987a. doi: 10.1137/0216051.

Neil Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4): 760–778, 1987b. doi: 10.1137/0216051.

Neil D Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. 2024.

Yanhong Li, Anej Svete, Ashish Sabharwal, and William Merrill. Efficiently representing algorithms with chain-of-thought transformers. *arXiv preprint arXiv:2606.19697*, 2026.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *International Conference on Learning Representations*, volume 2024, pp. 11911–11943, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. volume 36, pp. 46534–46594, 2023.

Will Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3186.

Will Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. volume 38, pp. 95315–95339, 2026.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations*, volume 2024, pp. 7690–7706, 2024.

William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Parsa Mirtaheri, Ezra Edelman, Samy Jelassi, Eran Malach, and Enric Boix-Adserà. Let me think! a long chain of thought can be worth exponentially many short ones. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5561.

Yuxiao Qu, Amrith Setlur, Jasper Dekoninck, Edward Beeching, Jia Li, Ian Wu, Lewis Tunstall, Aviral Kumar, et al. Qed-nano: Teaching a tiny model to prove hard theorems. *arXiv preprint arXiv:2604.04898*, 2026.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan Foster. Taming imperfect process verifiers: A sampling perspective on backtracking. In *International Conference on Learning Representations*, volume 2026, pp. 128647–128722, 2026.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8352–8370, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. volume 36, pp. 8634–8652, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Nikolaos Tsilivis, Nirmit Joshi, Marko Medvedev, Julia Kempe, and Nati Srebro. Learning through internalization. *arXiv preprint arXiv:2606.20937*, 2026.

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. 2022.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. volume 37, pp. 50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. volume 36, pp. 11809–11822, 2023.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

Alex L Zhang, Zhening Li, and Omar Khattab. The mismanaged geniuses hypothesis, 2026.

APPENDIX: CONTENTS

A Related Work **14**

B Supplementary Details for Section 2 **15**

B.1 Scaffold Details 15

B.2 Experimental Details 16

B.3 Full Ablation Grids 16

C Supplementary Details for Section 3 **18**

C.1 Model-Size Sweep (Section 3.1) 18

C.2 Simple-BT Distillation (Section 3.2) 20

C.3 Long-CoT on Modular Arithmetic (Section 3.3) 20

D Supplementary Details for Section 4 **21**

D.1 The local query and the hardness of reachability 21

D.2 Proof of Lemma 1 21

D.3 Proof of Theorem 1 22

E Supplementary Details for Section 5 **23**

E.1 Branching distillation on Qwen3-1.7B 23

E.2 The QED-Nano checkpoint and its harness 23

E.3 Corpus 24

E.4 Shapes 24

E.5 Generation, arms, and grading 24

E.6 Results and caveats 25

A Related Work

Agents and harnesses. An inference-time harness that can call a fixed language model as an oracle. Examples include interleaving reasoning with tool use and environment feedback (Yao et al., 2022), agent–computer interfaces for software engineering (Yang et al., 2024), iterative self-feedback and reflection (Madaan et al., 2023; Shinn et al., 2023), multi-agent debate (Du et al., 2023), branch-solve-merge decompositions (Saha et al., 2024), and aggregation trees over sampled solutions (Bateni et al., 2025). Recent mathematical systems likewise use strong models in verification-and-refinement loops (Huang & Yang, 2025; Feng et al., 2026; Qu et al., 2026). Informal conjectures for why such simple procedures help strong models include an implicit decomposition of the task into meaningful subtasks (Zhang et al.). We aim to provide more precise, quantitative and theoretical answers to this question. We also cover a complementary aspect: rather than designing a harness, we ask whether its benefit can be moved into the weights of a model from the same class, and we identify expressive power as one obstacle.

Expressive power of transformers and chain-of-thought. Circuit-complexity representational analyses place fixed-depth saturated and log-precision transformers in TC^0 (Merrill et al., 2022; Merrill & Sabharwal, 2023). Intermediate reasoning tokens can increase expressive power (Feng et al., 2023; Li et al., 2024; Merrill & Sabharwal, 2024), and some tasks require long chains of thought (Amiri et al., 2025). In particular, Merrill & Sabharwal (2025, Theorem 4) show that $O(\log n)$ CoT steps remain within logspace-uniform TC^0, while growing depth can enable directed graph connectivity. We use their transcript-enumeration argument in Lemma 1. Li et al. (2026) study how sufficiently long CoT can implement standard algorithms, including graph search. Closely related to our formal example, Mirtaheri et al. (2025) separate polynomial-length sequential CoT from polynomially many independently sampled constant-length CoTs with majority voting on an undirected-connectivity task, under a circuit-complexity conjecture. Our theorem combines existing short-CoT upper bounds and classical directed reachability hardness to formalize harness internalization; our experiments study distillation of explicit search and aggregation traces.

Verifiers and inference-time search. Sampling many solutions and selecting among them with a verifier or a majority vote trades inference compute for accuracy (Cobbe et al., 2021; Wang et al., 2022; Brown et al., 2024), and process reward models that score each step give finer-grained feedback than outcome verifiers (Lightman et al., 2024). Allocated well, test-time compute can let a smaller model outperform a larger one on some problems (Snell et al., 2025). Search over partial solutions includes model-guided tree search (Yao et al., 2023; Hao et al., 2023), classical planning such as A* search (Hart et al., 1968), and backtracking guided by an imperfect process verifier (Rohatgi et al., 2026). A complementary line trains models to carry out the search themselves: from serialized search traces (Gandhi et al., 2024; Lehnert et al., 2024), with explicit backtracking actions (Cundy & Ermon, 2024), or by distilling multi-step inference procedures into one student rollout (Yu et al., 2024). Our experiments distill exactly such traces, and show that when the student comes from the same model class as the base model, a gap to the scaffolded teacher persists; it narrows as the student's expressive power grows relative to the task. Closest to our question, Tsilivis et al. (2026) study how a transformer can absorb an explicit procedure into its weights. Our setting differs in two ways. First, they internalize the model's own chain-of-thought tokens, while we internalize a harness that calls the model many times and keeps state across calls. Second, they prove that gradient descent with a curriculum that gradually removes the chain-of-thought learns to compute sparse parities in a single forward pass, while we show an obstacle in expressive power for a student of the same class.

B SUPPLEMENTARY DETAILS FOR SECTION 2

This appendix collects the scaffold definitions, the experimental setup, and the full ablation grids behind Section 2.

B.1 SCAFFOLD DETAILS

This appendix gives the exact definitions of the two inference-time algorithms used in Section 2, together with their trace format.

B.1.1 VGB (PROBABILISTIC BACKTRACKING)

VGB (Rohatgi et al., 2026) replaces a hard threshold rule ($V < 0.5 \Rightarrow$ retract) with a Metropolis-style probabilistic backtracking that degrades gracefully under a noisy verifier. At a prefix s, the base model proposes $a \sim M(\cdot \mid s)$; the action is committed or a backtrack token <BT> is emitted according to

$$p_{\text{retract}} \;=\; \max\Big(\varepsilon,\; 1 - \tfrac{\widetilde{V}(s,a)}{\widetilde{V}(s)}\Big), \qquad \begin{cases} \text{commit } a & \text{w.p. } 1 - p_{\text{retract}}, \\ \texttt{<BT>} & \text{w.p. } p_{\text{retract}}. \end{cases}$$

The score $\widetilde{V}$. Following Rohatgi et al. Eq. (3), V is the *policy-conditional success probability* of the proposal model M from the current prefix:

$$\widetilde{V}(s) \;:=\; \mathbb{E}_{\tau\sim M(\cdot|s)}\big[\mathbf{1}[\text{valid}(s\circ\tau)]\big] \in [0,1], \qquad \widetilde{V}(s,a) \;:=\; \widetilde{V}(s\circ a).$$

We use two instantiations:

- *MC oracle.* $\widehat{V}_K(s) \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}[\text{valid}(s\circ\tau_k)]$ with $\tau_k \sim M(\cdot \mid s)$. We use $K{=}1$ at trace-generation time (a cheap Bernoulli sample) and $K{=}1000$ for diagnostics/evaluation.
- *Learned $\widehat{V}$.* A small `VerifierHead` trained by BCE on $(s, \text{outcome})$ pairs collected from base model rollouts (paper Eq. (4)); a single forward pass at inference replaces the K-sample MC estimate.

Trace format. Each step contributes two tokens: a quantized verifier bucket <V$_q$> (10 buckets) prepended to the action or <BT> it triggered, giving the student the explicit signal that triggered backtracking (vocabulary $|V|{=}27$):

$$\underbrace{\texttt{<board>}}_{\text{prompt}} \;\Big|\; \underbrace{\texttt{<V}_q\texttt{>}\ \texttt{<action_or_BT>}\ \cdots\ \texttt{<V}_q\texttt{>}\ \texttt{<action_or_BT>}}_{\text{search log (2 tokens / step)}} \;\Big|\; \underbrace{\texttt{<EOS>}}_{\text{end}}.$$

The four trace-format cells in the ablations toggle whether the verifier score is continuous or binary ({c,b}) and whether the bucket token is present ({nov,v}), giving `c_nov`, `c_v`, `b_nov`, `b_v`.

B.1.2 A* SEARCH (LM-PRUNED)

The A* scaffold runs standard A* search with a Manhattan-distance heuristic h and priority $f = g + h$, but at every expansion the successors are pruned to the top-$k{=}2$ legal actions under the base model's logits. Plans remain optimal and the traces shrink $\sim$36% relative to unpruned A* search:

$$\underbrace{\text{pop } s \text{ with min } f}_{\text{A* core}} \;\rightarrow\; \underbrace{M(s) \text{ top-}k}_{\text{LM prune}} \;\rightarrow\; \text{push children with } f' = g{+}1 + h(s').$$

Trace format. Each puzzle is serialized in an extended vocabulary ($|V|{=}45$ = base Sokoban tokens + <CREATE>/<CLOSE>/<PLAN> + number tokens for g, h, f and coordinates):

$$\underbrace{\texttt{<board>}}_{\text{prompt}} \;\Big|\; \underbrace{\big(\texttt{<CREATE>}\ pr\ pc\ br\ bc\ g\ h\ f \;\mid\; \texttt{<CLOSE>}\ pr\ pc\ br\ bc\big)^*}_{\text{search log}} \;\Big|\; \underbrace{\texttt{<PLAN>}\ a_1 a_2 \cdots \texttt{<EOS>}}_{\text{final plan}}.$$

At inference the student greedy-decodes the entire trace and we read off the plan after <PLAN>; the base model is never queried. We report `plan_emit` (did the student emit <PLAN>?) and `acc` (does the plan solve the puzzle?).

B.2 Experimental Details

Architecture. All base models, teacher backbones, and same-size students are decoder-only Transformers with 2 layers, d_{model}=128, 4 attention heads ($\approx$ 0.68M parameters). The capacity sweeps of Section 3 and Appendix B.3 vary this backbone; see Section 3 for the scale grid.

Dyck. Balanced sequences over 4 bracket types, length 64 with BOS/EOS (total length 66). A sequence is valid iff every opening bracket is closed in the correct order. ID bracket-type probabilities are $[0.1, 0.2, 0.3, 0.4]$; OOD reverses them to $[0.4, 0.3, 0.2, 0.1]$. The base model is trained on 300K clean ID sequences with next-token cross-entropy: batch size 32, AdamW, learning rate 3×10^{-4}, 100 warmup steps, 3000 iterations. Backtracking uses stride 4 and up to 50 backtracks (q=50); decoding is top-p=0.9, temperature 1.0. The learned verifier is a 2-class linear classifier on the base model's hidden states, trained on 500 error prefixes, reaching 87.3% token-level accuracy; the oracle verifier is `dyck.accept_prefix()`.

Sokoban. A 5×5 grid with 1 player, 1 box, 1 target, and up to 3 walls; action space $\{U, D, L, R\}$, and the backtrack token pops the last action (stride 1). Each puzzle is tokenized as <BOS> ⟨25 cells, row-major⟩ <SEP> ⟨actions⟩ <EOS>, vocabulary 16, maximum length $\sim$ 43. ID draws solution-length buckets [3–5, 6–8, 9–11, 12–15] with probabilities $[0.4, 0.3, 0.2, 0.1]$; OOD reverses them to $[0.1, 0.2, 0.3, 0.4]$ (longer solutions). Training: AdamW, learning rate 3×10^{-4}, 3000 iterations, 100 warmup steps, cosine decay. Evaluation uses 300 held-out puzzles per {ID, OOD} split with metrics `gen top_p` and `bt q=4 top_p`; the oracle verifier is a BFS over the Sokoban state space (`sokoban.accept_prefix`).

Student training regimes. Students are trained either (i) *from scratch* on the trace pool with next-token cross-entropy (loss masked on the board prompt), or (ii) *fine-tuned* from the base model on a mixture of clean data and traces. Unless otherwise noted the default fine-tuning mixture is 70% clean + 30% traces, 5K iterations. At inference the student greedy-decodes a single pass under the `gen` and `bt q=50` (Dyck) / `bt q=4` (Sokoban) samplers with no external verifier.

B.3 Full Ablation Grids

This appendix gives the complete sweeps summarized in Section 2.2. All sweeps are run over the four trace-format cells {`c_nov`, `c_v`, `b_nov`, `b_v`} on both Sokoban and Dyck; we show the representative cells below. Dashed lines mark the corresponding LM+VGB / LM+A* teacher in every panel.

B.3.1 Fine-tuning the Base Model (4 cells)

Figure 6 is the headline grid: starting from the base model we sweep, one axis at a time, every knob a practitioner would reach for.

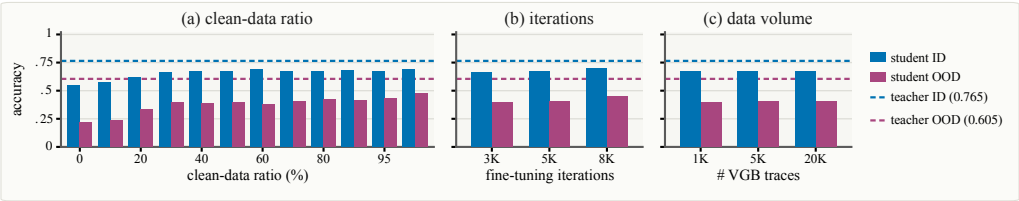


Figure 6: **No fine-tuning knob reaches the teacher.** *Fine-tuning a same-size student on VGB traces (Sokoban, cell `b_v`): sweeping clean-data ratio, iterations and trace count leaves the student below the teacher (dashed) at every setting.*

The axes are:

- **Clean-ratio sweep.** Fraction of clean base model traces mixed with scaffold traces swept from 0% to 100% in steps of 10% (plus a 95% point), at 5K iterations on the full data.
- **Iteration sweep.** Fine-tuning iterations $\in \{3\text{K}, 5\text{K}, 8\text{K}\}$ at clean $= 70\%$, full data.
- **Data-volume sweep.** Number of scaffold traces $\in \{1\text{K}, 5\text{K}, 20\text{K}\}$ at clean $= 70\%$, 5K iterations.

- **Training-source α-mix.** Training puzzles drawn from $P_\alpha = (1-\alpha)P_{\text{ID}} + \alpha P_{\text{OOD}}$ with $\alpha \in \{0.25, 0.5, 0.75\}$ (plus the pure ID α=0 baseline), re-running the clean-ratio sweep for each.

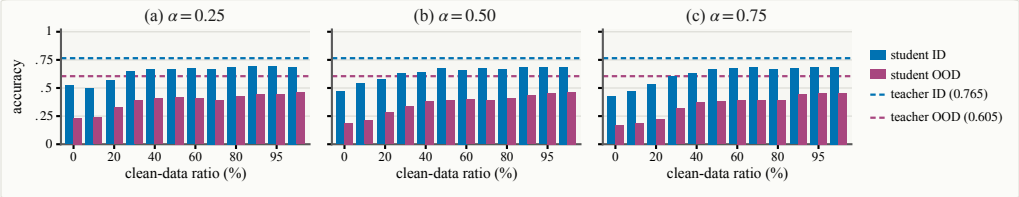


Figure 7: **No training-source mixture reaches the teacher.** *Training-source α-mix sweep for VGB fine-tuning on Sokoban (cell* `b_v`*): re-running the clean-ratio sweep at each α leaves the student below the teacher (dashed) throughout.*

Reading the grids. Three things are worth pointing out to rule out the obvious alternative explanations for the gap.

The clean-ratio curve rises and then flattens, and it flattens at the base model accuracy. On VGB (Figure 6a) accuracy climbs steeply from 0.545 ID at no clean data to ≈ 0.67 by a 30% clean ratio, and then moves by less than two points over the remaining 70% of the sweep. That plateau is at the base model accuracy, not at the teacher's: mixing in clean data recovers the loss from training on traces only—but nothing beyond it. The A* grid (Figure 8a) makes the same point more obviously, because there the trace format is harder and the collapse without clean data is nearly total (0.02 ID at 0%).

More optimization and more data do not help. Iterations (3K $\rightarrow$ 8K) move the student by 3.5 points ID and 5.5 OOD, and total amount of traces (1K $\rightarrow$ 20K, a twentyfold increase) moves it only by half a point. This makes it unlikely that data paucity is an issue.

Changing where the traces come from does not help. Re-running the whole clean-ratio sweep at each α (Figure 7) shifts the OOD column by a few points—more OOD-sourced puzzles do recover a little OOD accuracy, as one would expect—but every panel saturates at the same place relative to its teacher. The student's ceiling seems to be governed by something that isn't affected by the training distribution.

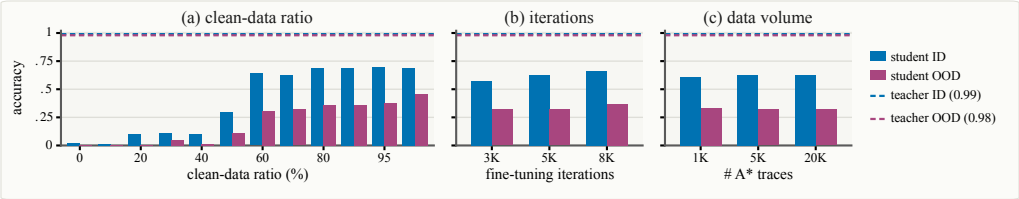


Figure 8: **Fine-tuned A* students stay at the base model.** *Fine-tuning the base model on A* traces (Sokoban): across clean-data ratio, iterations and data volume the student tracks the base model and never approaches the teacher (dashed).*

B.3.2 From scratch (2 cells: `c_nov`, `b_nov`)

For the from-scratch regime, holding the student at the fixed teacher scale, we sweep:

- **Seed variance.** 5K iterations, seeds $\{0, 1, 2, 3\}$.
- **OOD-source training.** 5K iterations on traces generated from OOD-source puzzles, probing whether the student learns the search algorithm or memorizes ID puzzle statistics.

- **Training-source α-mix.** The same P_α mixture as above with $\alpha \in \{0.25, 0.5, 0.75\}$, crossed with the clean-ratio sweep.

The axis of *student size* is studied separately in Section 3.1 (where larger students do begin to close the gap); the from-scratch figures below include a coarse small/medium/large capacity check, which saturates well short of the teacher and should not be read as the scaling result of Section 3.1.

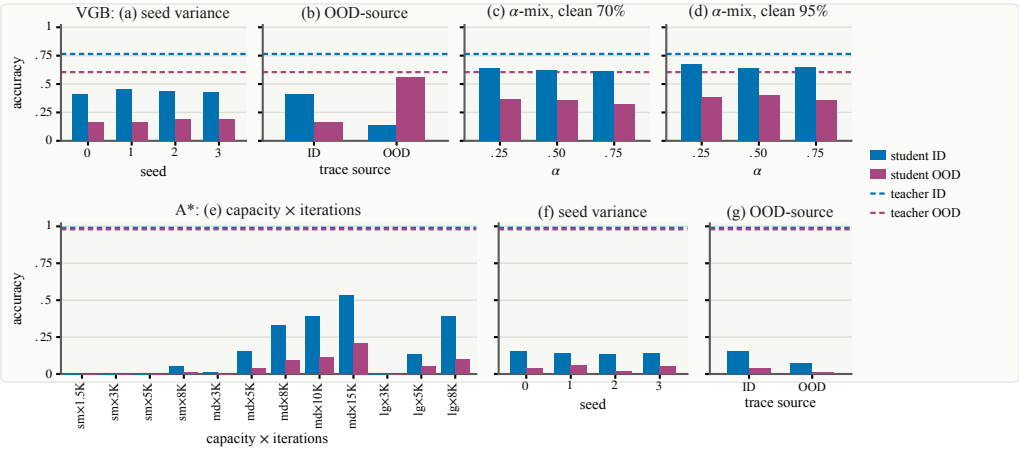


Figure 9: **From-scratch students stay far below the teacher.** *From-scratch students at the fixed* $0.68M$ *scale, VGB above and A* search below: no seed, no trace source and no coarse capacity setting lifts either student to its teacher (dashed, VGB* $0.765/0.605$ *and A** $0.99/0.98$*).*

C Supplementary Details for Section 3

This appendix expands the three experiments of Section 3.

C.1 Model-Size Sweep (Section 3.1)

Scales. Each student is a fresh from-scratch model trained on the same trace pool as Section 2, evaluated on the 200-puzzle ID/OOD split with the teacher overlaid. Table 1 lists the 12 configurations (d_{model}, L, H).

Student scale configurations

scale	tiny	xsmall	small	s_plus	teacher	t_plus
(d, L, H)	32/2/4	48/2/4	64/2/4	96/2/4	128/2/4	160/2/8
params	0.10M	0.16M	0.24M	0.44M	0.68M	0.98M
scale	mini	m_m1	m_m2	medium	m_plus	large
(d, L, H)	192/3/8	224/3/8	256/3/8	256/4/8	320/4/8	384/6/8
params	1.77M	2.32M	2.94M	3.73M	5.65M	11.5M

Table 1: *The* 12 *student scales swept in Section 3.1 (plus* `large`*). VGB is run on* 8 *scales (*`tiny`*–*`m_plus`*); A* search and the Dyck panel use the full set.*

Per-scale best accuracy (deferred panels). Figure 2b in the main text shows VGB/Dyck; Figure 10 gives the two Sokoban settings.
Both curves are monotone, but they end in different places. On A* search the student improves from 0.34 ID at `tiny` to 0.86 at `mini`, closing the teacher gap from 0.65 to 0.13 but never crossing it. On VGB/Sokoban the student passes its teacher between `t_plus` and `m_plus` and ends 0.18 ID and 0.23 OOD *above* it. The two settings differ in how hard the scaffold is, not in the trend.

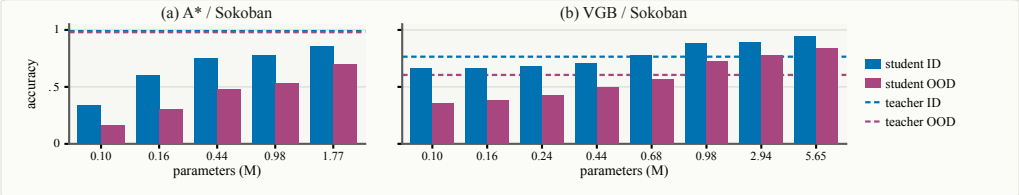


Figure 10: **Bigger students close the gap on Sokoban.** *Per-scale best accuracy on the two Sokoban settings: accuracy rises monotonically with parameter count and the teacher gap closes.*

Ratio- and iteration-resolved sweeps. The per-scale numbers above are maxima over the clean ratio and the checkpoint, so they could in principle hide a scale that only wins at one "lucky" setting. Figure 11 rules that out by resolving the same sweeps along both axes.

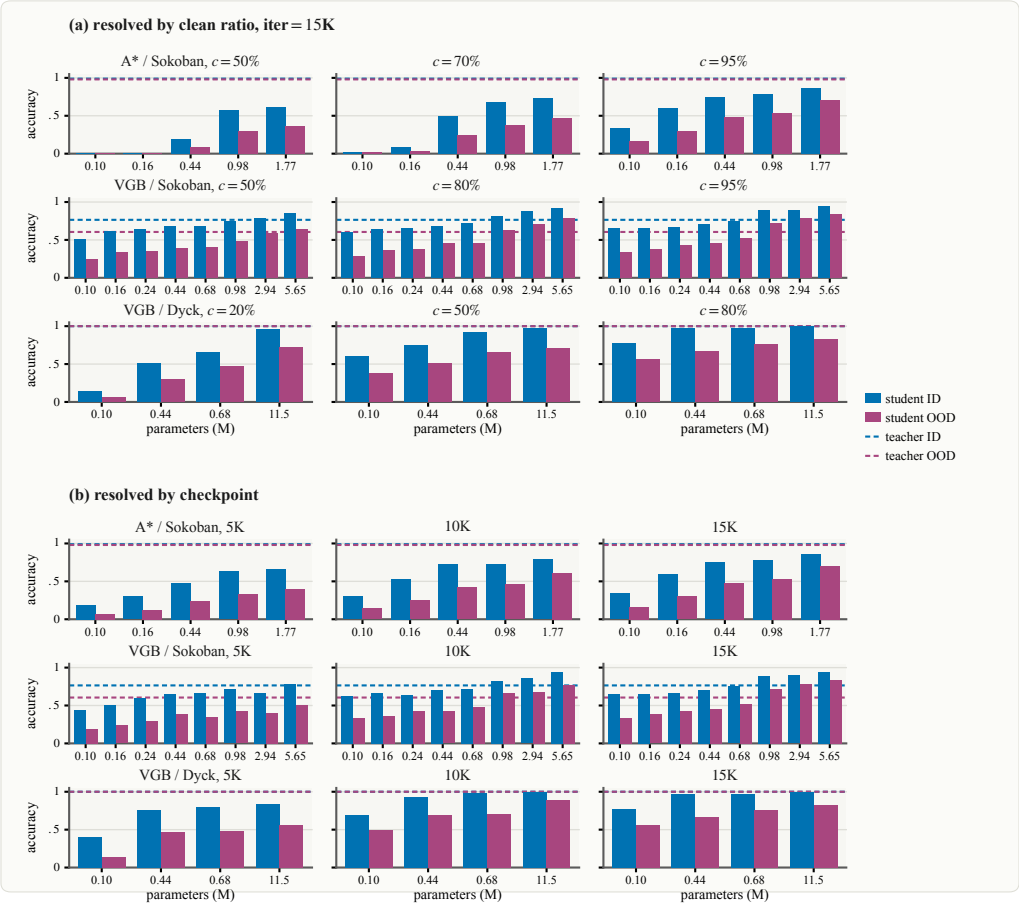


Figure 11: **The scaling trend holds at every clean ratio and checkpoint.** *The same sweeps resolved along both axes, A* / VGB-Sokoban / VGB-Dyck top to bottom within each block: accuracy is monotone in model size at every clean ratio and every checkpoint, and larger models saturate earlier.*

Every one of the eighteen panels is monotone in model size, for both ID and OOD, so the scaling trend is a property of the sweep and not of a particular cell. Two secondary effects are visible. The clean ratio and the model size are *substitutes* at the low end and not at the high end: on A* search at c=50% the two smallest students score essentially zero and only recover once the ratio reaches 95%, whereas the largest student is already at 0.61 with c=50%. And larger models saturate earlier in training—on VGB/Sokoban the 5.65M student gains 0.16 ID between 5K and 10K iterations and

only 0.005 from 10K to 15K, while the 0.10M student is still moving at 15K. Neither effect changes the ordering by size at any setting.

C.2 Simple-BT Distillation (Section 3.2)

Setup. On Dyck, the BT teacher is the base model paired with a verifier that flags the last token as erroneous; backtracking deletes a stride of 4 tokens and regenerates, with up to q=50 backtracks and top-p=0.9 decoding. The verifier is either learned (a linear probe on the base model's hidden states, 87.3% token accuracy) or an oracle (`dyck.accept_prefix`). The student is fine-tuned on filtered BT traces (`bt_count=1`) mixed with clean data; we vary the *source* distribution of the traces by an α-mixture between the ID and OOD bracket-type distributions.
Full grid. Table 2 reports both the single-pass (`gen`) and the backtracking-enabled (`bt`) accuracy for the distilled students, plus the synthetic-BT fine-tunes `mode6{a,b,c}` (OOD-error / random / ID-error injection). The single-pass student already matches the base model on ID; enabling its learned `<BT>` and choosing a mild-OOD trace source recovers most of the OOD gap.

Simple-BT distillation on Dyck

	reference		`ft_bt1`, trace source			synthetic-BT injection		
	base	oracle-BT	ID	α=0.25	α=0.50	OOD-err	random	ID-err
`gen` ID	0.947	—	0.936	0.957	0.942	0.947	0.977	0.840
`gen` OOD	0.650	—	0.133	0.470	0.313	0.250	0.637	0.450
`bt` ID	—	0.996	0.966	0.970	0.957	0.957	0.963	0.933
`bt` OOD	—	0.873	0.543	0.677	0.680	0.637	0.683	0.697

Table 2: *Full simple-BT distillation grid on Dyck. The main-text bar chart (Figure 2c) uses the* `bt` *column for the* `ft_bt1` *rows.*

C.3 Long-CoT on Modular Arithmetic (Section 3.3)

Task. Following Feng et al. (2023, Alg. 1), each problem is an arithmetic expression mod p=11 with n_{ops} operators and a unique reduction chain, where each CoT step rewrites the leftmost reducible sub-expression. The model is a 3-layer causal transformer (d=256, H=4, $\approx$ 2.5M parameters), trained on $\langle \text{problem} = \text{step}_1 = \cdots = \text{step}_L = \text{answer} \rangle$ with the CoT *truncated* to the first L steps, sweeping $L \in \{0, 2, \ldots, n_{\text{ops}}\}$. All evaluations use 1000 held-out problems.
Scaffold. The inference-time scaffold samples K=16 candidate next-steps, scores each with an oracle step-verifier, commits the best, and backtracks (blacklisting the popped step) whenever every candidate scores below threshold. Figure 12 shows the verifier loop and the easier n_{ops}=10 curve; the n_{ops}=20 curve is in the main text (Figure 3).

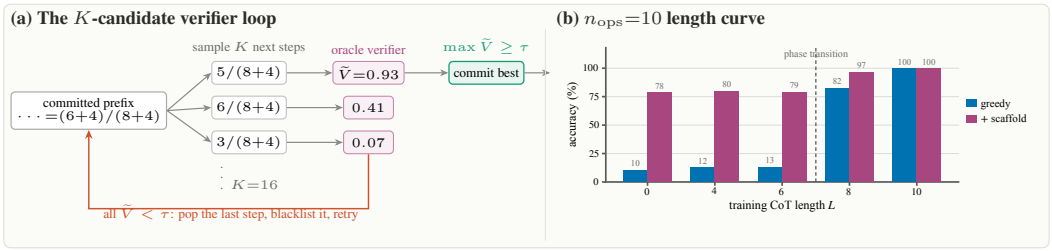


Figure 12: **The same phase transition on an easier task.** *The K=16 verifier-backtracking scaffold **(a)**, and the same phase transition on the easier n_{ops}=10 task **(b)**.*

D SUPPLEMENTARY DETAILS FOR SECTION 4

This appendix proves Lemma 1 and Theorem 1. Throughout, circuit uniformity means logspace uniformity. The input length is $n = \Theta(N^2)$, Σ is fixed and finite, and $T_c(n) = \lceil c \log_2(n+1) \rceil$ for a fixed integer $c \geq 1$.

D.1 THE LOCAL QUERY AND THE HARDNESS OF REACHABILITY

For $A \in \{0,1\}^{N \times N}$ and $u, v \in [N]$, the local predicate is

$$E(A, u, v) = \bigvee_{i,j \in [N]} \big(A_{ij} \wedge [u = i] \wedge [v = j] \big).$$

Writing each equality as a conjunction of binary literals gives a constant-depth Boolean circuit with polynomially many gates and wires. A logarithmic-space procedure can generate the gates and their wiring by enumerating (i, j), so $E \in \mathsf{AC}^0$. The notation $E_A(u, v)$ fixes the graph and makes the two queried vertices explicit.

Directed STCON is NL-complete under uniform projection translations, and hence under uniform AC^0 many-one reductions (Immerman, 1987b, Corollary 4.7). The standard configuration-graph construction explains the strength of the reduction. For a fixed nondeterministic logspace machine and input x, its polynomially many configurations become vertices; an edge records one legal machine transition. After fixing the two configurations, the legality of that transition depends on at most one input bit. In a standard normal form, each output adjacency bit is therefore a constant, an input bit, or its negation, and its wiring is computable in logspace. The source and target encode the initial configuration and a designated accepting configuration.

This reduction strength is what Theorem 1 uses. An arbitrary logspace reduction could itself perform work outside TC^0, whereas a uniform AC^0 reduction can be composed with a TC^0 decision circuit while preserving constant depth, polynomial size, and logspace uniformity.

D.2 PROOF OF LEMMA 1

Proof. Let $L \in \exists \mathrm{CoT}_{T_c(n)}(\Sigma, \mathsf{TC}^0)$, with predicates $V_{n,i}$ and B_n as in Definition 1. Let d and $p(n)$ be their common depth and polynomial size bounds. Since the alphabet and c are fixed,

$$M(n) := |\Sigma|^{T_c(n)} \leq |\Sigma|(n+1)^{c \log_2 |\Sigma|} = \mathrm{poly}(n).$$

Enumerate all transcripts as $z^{(1)}, \ldots, z^{(M(n))}$. For each transcript, form the circuit

$$\mathrm{Acc}_j(x) = B_n\big(x, z^{(j)}\big) \wedge \bigwedge_{i=1}^{T_c(n)} V_{n,i}\big(x, z^{(j)}_{<i}, z^{(j)}_i\big).$$

Each predicate is evaluated after hardwiring its transcript inputs. This does not increase its depth or size. Add one AND gate for each transcript and one final OR:

$$C_n(x) = \bigvee_{j=1}^{M(n)} \mathrm{Acc}_j(x).$$

The resulting circuit has depth at most $d + 2$ and size at most $M(n)(T_c(n) + 1)p(n) + O(M(n))$, which is polynomial in n.

For uniformity, the integer c is fixed and $T_c(n)$ is logspace-computable. Both a transcript and its index use $O(\log n)$ bits, so a logarithmic-space generator can enumerate the transcripts, produce their hardwired bits, and invoke the jointly uniform generators for $V_{n,i}$ and B_n. It can then add the AND and OR wiring using counters of $O(\log n)$ bits. Thus (C_n) is logspace-uniform. By Definition 1, $C_n(x) = 1$ exactly when $x \in L$, establishing $L \in \mathsf{TC}^0$. □

This is the transcript-enumeration argument used for logarithmic-length transformer CoT by Merrill & Sabharwal (2025, Theorem 4 and Appendix E). Joint uniformity and the explicit computable ceiling ensure that the construction also satisfies the uniformity convention of this paper. The argument uses the polynomial number of possible transcripts; it does not establish a sharp threshold for the power of longer chains of thought.

D.3 Proof of Theorem 1

Proof. Part (i). The preceding construction shows $E \in \mathsf{AC}^0$. Run breadth-first search using E_A to test existence of edges. Store a visited set R, a current frontier F, and a next frontier F' as N-bit characteristic vectors. Initially, $R = F = \{s\}$. At each iteration, set

$$F' = \{v \in [N] \setminus R : \exists u \in F \text{ with } E_A(u, v) = 1\},$$

by querying the pairs in $F \times ([N] \setminus R)$, and update $R \leftarrow R \cup F'$ and $F \leftarrow F'$. Stop when F is empty, and accept exactly when $t \in R$.

After k iterations, R contains exactly the vertices at directed distance at most k from s, and F contains those first reached at that distance. This invariant proves correctness, including the case $s = t$. A vertex enters the frontier at most once, so each ordered pair is queried at most once, giving at most N^2 oracle calls. The three characteristic vectors and the loop counters use $3N + O(\log N) = O(N)$ bits. The loop has at most N iterations and all its bookkeeping takes polynomial time.

Part (ii). Suppose, for some fixed Σ and integer $c \geq 1$, that $\mathrm{STCON} \in \exists \mathrm{CoT}_{T_c(n)}(\Sigma, \mathsf{TC}^0)$. Lemma 1 then gives $\mathrm{STCON} \in \mathsf{TC}^0$. For any $L \in \mathsf{NL}$, let f_L be the uniform AC^0 many-one reduction from L to STCON supplied by projection completeness (Immerman, 1987b). Composing a TC^0 circuit for STCON with the circuits for the output bits of f_L gives a circuit for L. The composition preserves constant depth, polynomial size, and logspace uniformity. Hence $\mathsf{NL} \subseteq \mathsf{TC}^0$, contradicting the assumption. □

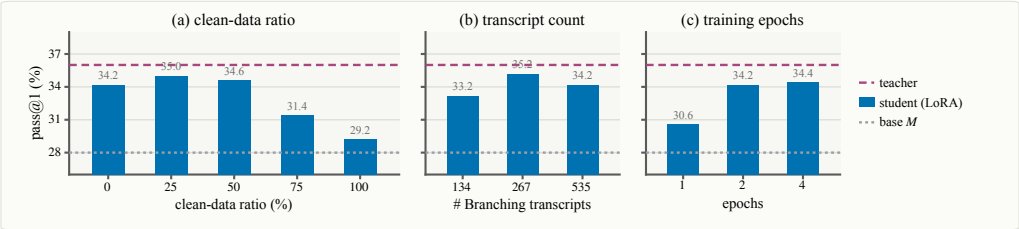


Figure 13: **No distillation knob reaches the teacher.** *Branching ablations: clean-data ratio, transcript count, and training epochs (teacher dashed, base dotted).*

E Supplementary Details for Section 5

This appendix collects everything deferred from Section 5: first the distillation study on Qwen3-1.7B, then the shape sweep on the QED-Nano RL checkpoint.

E.1 Branching distillation on Qwen3-1.7B

Distillation knobs. Figure 13 sweeps the three knobs of the distillation itself, at the canonical **ks**=[3] shape. Diluting the Branching transcripts with in-distribution self-distilled data is tolerated up to about half: the 0/25/50% cells sit in a 0.8-pp band and peak at 25% (35.0%), after which accuracy falls away toward the pure-clean baseline (29.2% at 100%). Transcript count is flat above ~ 250 examples, and training elbows at epoch 2. None of the three reaches the teacher's 36.0%.
The full (d, L) grid. Figure 14 puts the compute of each tree shape next to what it buys. Panel (a) is the call count $\sum_{i=0}^{L} d^i$, which runs from 3 at $(2, 1)$ to 85 at $(4, 3)$; panel (b) is the accuracy of both arms, sorted by the teacher, and is the panel reproduced in the main text. Reading them together is the point: the teacher's accuracy tracks compute almost monotonically over a $28\times$ range of budgets, while the student's does not track it at all.

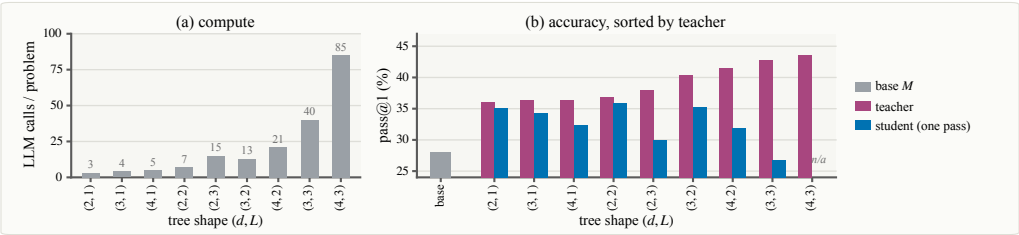


Figure 14: **Teacher accuracy tracks compute; the student's does not.** *The full Branching (d, L) grid: compute per problem **(a)** and accuracy sorted by teacher **(b)**.*

What closes the residual gap. Figure 15(a,b) separates two things that are easy to confuse. Raising the LoRA rank from 16 to 64, or going to full fine-tuning, moves the student from 31.8% to 33.6% and then 34.0%—a 7.4-pp gap to the teacher remains even when every weight is trainable. Enlarging the backbone from 1.7B to 4B moves it from 35.2% to 41.6%, past the teacher's 40.0%. What is missing is capacity in the forward pass, not trainable parameters in the fine-tune.

E.2 The QED-Nano checkpoint and its harness

QED-Nano (Qu et al., 2026) is a 4B Olympiad prover built from `Qwen3-4B-Thinking` in two training stages, and only afterwards wrapped in a harness at test time; nothing about the harness is trained. The supervised stage uses `FineProofs-SFT`, roughly 7.5k solutions over 4.3k problems distilled from DeepSeek-Math-V2 (685B). The reinforcement stage uses `FineProofs-RL` with a rubric-based outcome reward, the rubrics written by Gemini 3 Pro and the grading during training done by `GPT-OSS-20B`. The released checkpoint at `revision main` is the model after both stages, and is what we wrap.

The paper reports the harness as the larger half of the story. Wrapping the finished model in *Recursive Self-Aggregation* (RSA) lifts IMO-ProofBench from 40.0% to 56.9% and ProofBench from 44.9% to 62.6%, at $21.8\times$ the tokens, about 2.0M per problem. RSA keeps a population of n candidate proofs and rewrites the whole population every round: a *solver* produces n independent proofs, a *merger* then writes each new proof from k randomly drawn members of the current population, and after T rounds a pairwise *selector* runs a knockout tournament over the survivors. The same model plays all three roles, differing only in system prompt.

That configuration is exactly what a student cannot absorb: at 39 calls its trace runs to roughly 1.5M tokens, several times any context window we could train into. We therefore test **Branching**, the minimal shape—independent leaves and a single aggregation—whose traces are short enough that distillation is at least conceivable, and ask whether the gain survives the shrink.

E.3 Corpus

We evaluate on 300 problems drawn from `FineProofs-RL` (5,227 rows), the model's *own* RL corpus. Three properties make it the right choice, and all three are necessary. The authors have already excluded the 2025 contest season and fuzzy-matched the corpus against IMO-ProofBench, so the problems are decontaminated with respect to the benchmark the harness was tuned on. Every row ships a per-problem `rubrics` field, without which a free-form proof cannot be scored at all. And the corpus is large enough that a 300-problem sample is not the whole of it.

Problems are taken from the difficulty band `reward_mean` $\in [0.10, 0.60]$, where `reward_mean` is the base model's average reward over 128 rollouts and is therefore a measured difficulty rather than an editorial one. Below that band the harness cannot rescue the problem; above it a single call already answers it and there is nothing for the harness to add. The selection is deterministic—sorted, then taken at equal spacing—so every re-run scores the same problems.

E.4 Shapes

We write a shape as (rounds, block size), so the number of leaves is the block size raised to the number of rounds. The scaffold is configured to produce one aggregation per round rather than one merge per solution, and the pairwise selector is disabled, so a trace contains the tree and nothing else. $(1, 6)$ and $(2, 2)$ are deliberately matched at 7 calls: the budget is held fixed and the shape is the only variable, wide-and-flat against narrow-and-layered.

The five tree shapes we run

shape	$(1,4)$	$(1,5)$	$(1,6)$	$(2,2)$	$(3,3)$
leaves	4	5	6	4	27
aggregations	1	1	1	2+1	13
calls	5	6	7	7	40
median trace	208k	253k	309k	—	$\approx$ 1.4M

Table 3: *The five tree shapes we run, with the median whole-trace length each produces.*

Trace length is the character count of every call's reasoning and answer divided by 2.47; the $(3, 3)$ entry is extrapolated from the shorter shapes rather than measured. A $(2, 2)$ tree consumes each solution exactly once per round, which the stock block sampler does not guarantee, so that arm uses a partitioning sampler instead.

E.5 Generation, arms, and grading

Generation. The backbone is served by vLLM in bfloat16 with an fp8 KV cache and a 73,728-token model length, 65,536 tokens per call, temperature 1.0, one run per problem. The solver and merger prompts are the repository's own, unmodified.

Two arms from one trace. Both arms are read out of the same generation. The *base* arm is the first solver call's own answer—a single direct inference, so the baseline costs nothing extra—and the *harness* arm is the root aggregate, identified structurally as the one aggregation output that no later step consumes. The two arms therefore share the problem, the run, the server and the sampling temperature, and pair per problem with no drift between runs. Answers are the assistant messages that are not reasoning traces, so no string surgery is needed to separate thinking from the proof.

Grading. A local `gpt-oss-20b` judge scores each answer against the problem's rubric; a score is the sum of the rubric's points out of 7, and an answer passes at ≥ 6, so every percentage we report is

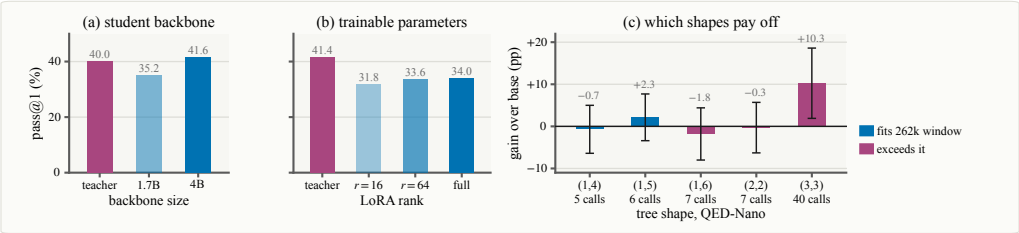


Figure 15: **Capacity, not trainable parameters, closes the gap.** *On Qwen3-1.7B, capacity closes the gap and trainable parameters do not* ***(a,b)****; on QED-Nano, only the* 40*-call shape clears zero, and its trace is far past the window* ***(c)****.*

a pass rate and not a mean score. Answers are truncated to 100,000 characters before grading, since degenerate output tokenizes badly and would otherwise fail the request and be silently dropped as missing data. A rubric item that comes back `null` is treated as missing rather than as zero. Intervals are paired bootstrap over 6000 resamples with the same indices drawn for both arms.

E.6 Results and caveats

Three caveats, in order of how much they matter.

(i) The length filter is post-treatment. Trace length includes the aggregation calls, so a merge that runs to the token ceiling both overflows the window and scores zero, and the filter reads six calls of information while the base arm uses one. All numbers reported here are therefore unfiltered.

(ii) The baseline is a single sample, not a deployment baseline. The base arm uses the scaffold's own solver prompt rather than the benchmark's evaluation prompt, so this measures aggregation against one sample of the same model.

(iii) No seed replication. One run per problem at temperature 1.0, and a single grading pass.