THE LIMITS OF ALGORITHM ACQUISITION IN SEQUENCE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Can neural networks acquire computational procedures absent from training? Prior evaluations are confounded by the possibility that a model saw the target task during pretraining. We introduce NORA, a procedurally generated benchmark in which target computations are sampled only after a pretrained model is fixed, isolating genuine acquisition from primitive familiarity. Each target is an unseen member of a small, structured computational family – not an open-endedly novel procedure – but rapid acquisition of such a target is itself a nontrivial, underexamined question. In small, controlled synthetic sequence models – not Internet-scale language models – we find a sharp acquisition bottleneck: single-stream computations are learned to near-perfect accuracy, while *non-separable* two-stream computations – additive or multiplicative combinations of raw stream values, and arbitrary joint lookups alike – fail catastrophically. Separability, not stream count, governs the outcome: two-stream targets built from two independently sampled, information-compressing per-stream transformations ($z = f(x)+g(y)$ or $z = f(x)g(y)$) are learned as well as single-stream targets, even though the same combination of untransformed streams fails completely. Separability alone is not sufficient, however – a bijective variant fails just as badly, a result we report but do not fully explain. The failure persists across Transformer, recurrent, and convolutional architectures, a $271\times$ parameter range, and larger optimization budgets. Fixing one input stream restores near-perfect learning, and coupling the two streams progressively restores performance, localizing the bottleneck to jointly combining two independently varying streams – not sequential state, target complexity, or capacity. Pretraining does not reliably remove this bottleneck; whether it persists at frontier scale remains open. NORA offers a framework for studying what pretrained representations contribute to acquiring new computations, and for locating where that transfer breaks down.

1 INTRODUCTION

A child taught multi-digit addition eventually internalizes a compact procedure – process digits right to left, track a carry – that they can then apply to numbers never previously encountered. The striking part is not that the answer is memorized, but that a small amount of instruction yields a procedure that generalizes broadly. This motivates, but is not itself evidence about, the question this paper actually studies in neural networks:

> **How much experience should a neural network require to learn a new algorithm?**

Modern pretrained language models appear to answer this: they perform arithmetic, write programs, and solve unfamiliar problems from a handful of examples. But there is a fundamental confound. A model pretrained on Internet-scale data may have encountered enormous numbers of worked examples, so when it solves a new problem it is difficult to tell whether it applies a learned general procedure or exploits statistical knowledge from direct or near-direct exposure to the task – and comparing sample counts across learners with different prior exposure (a pretrained network versus a randomly initialized one, or a child who has spent years acquiring language and causal reasoning before formal instruction) risks conflating that prior experience with genuine algorithm acquisition. Pretraining might simply supply this missing prior experience, but that only sharpens the experimental problem: how can we determine whether the model is actually learning a new procedure?

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.

We argue that the key requirement is to evaluate models on **target algorithms that were not available during pretraining**. The distinction is subtle but important: a target algorithm can be novel even if its component primitives are familiar – a model may have extensive experience with addition, multiplication, and modular arithmetic while never having encountered the exact transition rule $s_i = 2x_i + 3y_i + c_i$, $z_i = s_i \bmod 10$, $c_{i+1} = \lfloor s_i/10 \rfloor$. If this rule is generated only after pretraining, success on it cannot be attributed to memorization of that exact procedure – though the model may still exploit knowledge of the underlying primitives, precisely the type of transfer we wish to measure.

We introduce NORA (**N**ovel **O**perations for **R**ule **A**cquisition), a benchmark that generates target algorithms from controlled families of stateful computations only after a model's pretraining checkpoint has been fixed, then varies how much task-specific information is provided and how. We are precise about this claim's scope, since it is easy to overstate: NORA's target families are small and structured by construction (Section 5 gives the exact count), so a target algorithm is an unseen member of a known computational family, not an open-endedly novel procedure. What we study is therefore **rapid acquisition of a previously unseen member of a structured computational family** – narrower than open-ended algorithm invention, but, we argue, still an interesting and largely unaddressed question, and we do not claim more than that anywhere in this paper.

Our experiments distinguish four regimes: (1) scratch learning, a randomly initialized model learning the target through gradient descent; (2) in-context acquisition, a pretrained model inferring the target from demonstrations without parameter updates; (3) rule-based acquisition, a pretrained model given an explicit description with no demonstrations or updates; and (4) fine-tuning, a pretrained model updating its parameters on a small number of examples – separating inference-time procedural acquisition from improved gradient-based learning, and letting us additionally ask, as secondary diagnostics (Appendix H), whether acquisition depends on primitive familiarity and generalizes to longer sequences and state depth.

Within this design, the paper's central, falsifiable claim is specific rather than broad: **novel algorithm acquisition in the autoregressive sequence models we study exhibits a sharp bottleneck at non-separable two-stream computation**, not a general verdict on whether pretraining helps acquire new procedures, and not simply a matter of how many input streams a target depends on. Our contributions are: (1) NORA, a procedurally generated benchmark separating target-task novelty from primitive familiarity and memorization; (2) the main finding above, built in three steps – an aggregate observation (Section 6.1), a structural localization at separability rather than stream count (Section 6.2, where gradient-based acquisition succeeds almost perfectly for single-stream and separable multi-stream targets but fails for non-separable ones, and, surprisingly, for bijective-separable ones too), and controls ruling out architecture (an LSTM and a purely local causal convolutional backbone both reproduce the pattern), task complexity, optimization budget, pretraining composition, and a $271\times$ range in model scale (Sections 6.2.2 and 6.3) – none of which explain away a bottleneck we interpret as a difficulty computing a non-separable joint function of two independently varying streams, rather than a hard ceiling on stream count (Sections 7); and (3) a methodological distinction between local acquisition, procedural acquisition, and systematic execution, which can dissociate: a model may infer a locally correct rule without generalizing it systematically across sequence length or state-transition depth (Section 7).

2 RELATED WORK

Algorithmic generalization, grokking, and rule discovery. A substantial literature asks whether networks learn algorithmic procedures rather than memorize examples (Zaremba & Sutskever, 2014; Kaiser & Sutskever, 2015), notes that interpolation accuracy overstates algorithmic competence as depth grows (Anil et al., 2022), and supervises intermediate steps against a *known* classical algorithm (Veličković et al., 2020); related work asks whether models recombine familiar primitives into novel combinations (Lake & Baroni, 2018; Keysers et al., 2020). Grokking work shows networks trained on algorithmic data can initially memorize before discovering a generalizing solution (Power et al., 2022; Nanda et al., 2023), and Sun et al. (2026) report an analogous phase transition under reinforcement learning on synthetic coding-problem families. In all of this the target task or composition is known throughout training; our requirement is stronger – the complete target algorithm is generated only after pretraining is frozen, so no gradient or reward signal against it has ever touched the parameters before acquisition begins.

Meta-learning and in-context learning. Meta-learning trains across a task distribution for rapid adaptation to a new task (Finn et al., 2017); closer to our in-context regime, Garg et al. (2022) show a Transformer trained on a distribution of function classes can identify a novel function in-context, but their base model is trained directly on the eval function class, whereas ours has no exposure to the target's family at all. Language models more broadly infer patterns from demonstrations (Brown et al., 2020), and a body of work asks whether this reflects implicit task inference, optimization, or retrieval (Xie et al., 2022), or, for a familiar base procedure with an in-context modification, induction via specific attention heads (Ye et al., 2026); our targets share no fixed familiar base and are withheld from pretraining entirely.

Instructions, teaching, and human learning. Instruction-following models execute procedures described in language or code, plausibly reflecting exposure to instructional data (Wei et al., 2022); we distinguish this from algorithm acquisition by evaluating procedures newly generated after pretraining. Teaching-dimension frameworks study how small, well-chosen examples can uniquely identify a hypothesis (Goldman & Kearns, 1995) – our informative-teaching condition adapts this, selecting examples that maximally reduce the remaining hypothesis space. Human learning has separately motivated arguments about acquiring structure from limited data via strong inductive biases (Lake et al., 2015; 2017); we do not claim our benchmark reproduces human learning, and use the addition example only to motivate whether pretraining supplies reusable structure for rapid acquisition of a novel procedure.

3 Problem Formulation

We study **acquisition regimes**: the different forms of task-specific information a model can receive about a target algorithm $f_{\theta^\star}$ generated only after the pretrained parameters $\phi_0 = \mathrm{Train}(\mathcal{D}_{\mathrm{pretrain}})$ are frozen. Formally, acquisition produces $\phi^\star = \mathrm{Acquire}(\phi_0, T, f_{\theta^\star})$, where the regime determines the form of T: k input-output demonstrations for in-context learning, a textual or formal specification $S(\theta^\star)$ for explicit procedural teaching, or a gradient update $\phi^\star = \mathrm{Update}(\phi_0, \mathcal{D}_k)$ for fine-tuning (ϕ_0 random rather than pretrained for scratch training). We report the acquisition function $A_r(k) = \mathbb{E}_{\theta^\star, x}[\mathbf{1}\{M_r(x \mid T_k) = f_{\theta^\star}(x)\}]$, where r denotes the regime and k the number of task-specific demonstrations; the complete formalization is in Appendix A.

4 Nora Benchmark

Each task consists of two input sequences $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ with $x_i, y_i \in \{0, \ldots, B-1\}$, and a finite latent state $c_i \in \mathcal{C}$ that is propagated through the sequence from a fixed initial value $c_0 = c_{\mathrm{init}}$. The generic computation produces the output sequence $z = (z_1, \ldots, z_n)$ via $(c_{i+1}, z_i) = F_\theta(x_i, y_i, c_i)$, and, concretely, one instance is rendered as a single token sequence

$$\texttt{[X_SEP]}\ x_1\, x_2 \cdots x_n\ \ \texttt{[Y_SEP]}\ y_1\, y_2 \cdots y_n\ \ \texttt{[Z_SEP]}\ z_1\, z_2 \cdots z_n, \tag{1}$$

with x and y each rendered as a complete, contiguous block (not interleaved digit-by-digit), separated by dedicated separator tokens. The model is causal and autoregressive, so by the position where it must emit z_1 it has already attended over the complete x and y blocks – both streams are fully available before any output digit is generated. This matters for interpreting the separability results of Section 6: the bottleneck we report is not a matter of x_i or y_i being inaccessible when z_i must be produced, but of the model failing to combine two streams it can already fully attend to.

Level I: Linear Stateful Algorithms. Our primary benchmark instantiates F_θ as $s_i = ax_i + by_i + gc_i + d$, $z_i = s_i \bmod B$, $c_{i+1} = \lfloor s_i / B \rfloor$, with parameters sampled from finite sets $a, b, g \in \{0, 1, 2, 3, 4\}$ and $d \in \{0, 1, 2, 3, 4\}$, giving the hypothesis class $\Theta = \mathcal{A}^3 \times \mathcal{D}$. We reject degenerate algorithms with output entropy below 1.5 bits, fewer than 3 distinct reachable states, constant or near-constant output mappings, or exact equivalence to another retained algorithm over the complete finite input domain (Appendix C). The benchmark is designed to extend to nonlinear and compositional target families beyond this Level I linear family; we leave that extension to future work.

5 EXPERIMENTAL DESIGN

A terminological note: we use *pretrained sequence model* rather than *pretrained language model* for the models pretrained within NORA itself, using controlled synthetic pretraining (character-level, $\sim$20M tokens; Appendix G) rather than Internet-scale pretraining – a deliberate trade of scale for causal control over exactly what a model has and has not seen. Claims in this paper about what pretraining does or does not confer are therefore claims about this controlled, small-scale regime, not about frontier-scale language models (Sections 8 and 9).

Ensuring target-task novelty. The target algorithms are sampled only after pretrained checkpoints have been selected, and never inserted into the pretraining corpus. Novelty is guaranteed by construction, not post-hoc textual search: the filtered hypothesis space Θ is split once, by fixed seed, into a disjoint pretraining pool (the only source of worked examples in the "Algorithmic" pretraining corpus) and an eval-reserved pool (the only pool eval targets are ever drawn from; Appendix B), making it structurally impossible for an eval target's exact parameterization to appear in any pretraining corpus. We distinguish a **novel target algorithm** (required for all main experiments) from **novel computational primitives** (a separate variable). Level I's hypothesis space contains only 625 parameter combinations before filtering, 541 surviving (Appendix B) and splitting roughly in half into pretraining and eval-reserved pools (271/270); a target algorithm is therefore never a genuinely open-ended novel procedure, but an unseen member of a small, structured, known family – a **novel target function within a known computational family** rather than "a genuinely new algorithm," a distinction we return to under Limitations (Section 8). We additionally define three primitive-overlap conditions – familiar operations with novel parameter combinations, familiar operations in novel procedural structures, and random finite transition systems – to test primitive familiarity as an alternative explanation for our main results (definitions and results in Appendix H).

Acquisition regimes, budgets, and lengths. We evaluate: (1) scratch training; (2) pretrained in-context learning; (3) pretrained rule execution; (4) pretrained fine-tuning. For in-context learning, $k \in \{0, 1, 2, 4, 8, 16, 32, 64, 96\}$, limited by context length; for fine-tuning and scratch learning, $k \in \{10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000\}$, which reuse the same k examples across many gradient steps rather than holding them all in context at once. ICL and gradient-based regimes are therefore compared at matched accuracy thresholds (K_τ), not at matched k, since the two budgets measure different quantities (context slots vs. training examples) and are not directly interchangeable. Training and acquisition lengths are controlled independently of test lengths: we fix $L_{\text{train}} = 8$ and test $L_{\text{test}} \in \{8, 16, 32, 64, 128\}$, adjusted based on the context-window limitations of the evaluated models.

Evaluation metrics and acquisition threshold. Our primary metric is exact sequence accuracy, $\text{ExactAcc} = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}[\hat{z}^{(j)} = z^{(j)}]$, since an algorithm should generate the complete output correctly; we additionally report symbol-level accuracy, $\text{SymbolAcc} = \frac{\sum_{j,i} \mathbf{1}[\hat{z}_i^{(j)} = z_i^{(j)}]}{\sum_j |z^{(j)}|}$. For a target accuracy τ, we define $K_\tau = \min\{k : A(k) \geq \tau\}$ and report $K_{0.90}$, $K_{0.95}$, and $K_{0.99}$; when a model does not reach the threshold within the tested budget, we report right-censored sample complexity rather than extrapolating, which avoids misleading claims about sample efficiency.

6 RESULTS

We organize our results around three questions. Question 1 (Section 6.1) asks whether pretraining helps acquire a target algorithm at all. Question 2 (Section 6.2) asks what structural property of a target rule determines whether acquisition succeeds, and identifies *separability* – not the number of independent input streams – as that property. Question 3 (Section 6.3) asks whether either pretraining composition or model scale removes the bottleneck Question 2 identifies. Supporting experiments that do not bear directly on one of these three questions are reported in Appendix H.

Figure 1 and Table 1 summarize this paper's central result before we develop it in full: a single-stream target, a two-stream target that separates into two independently arbitrary per-stream transformations, and a two-stream target with no separable structure at all, trained identically from scratch. This is a controlled intervention on functional structure, not merely an observation that one stream is easier to acquire than two – stream count is held fixed at two across the latter two conditions, and only separability changes.

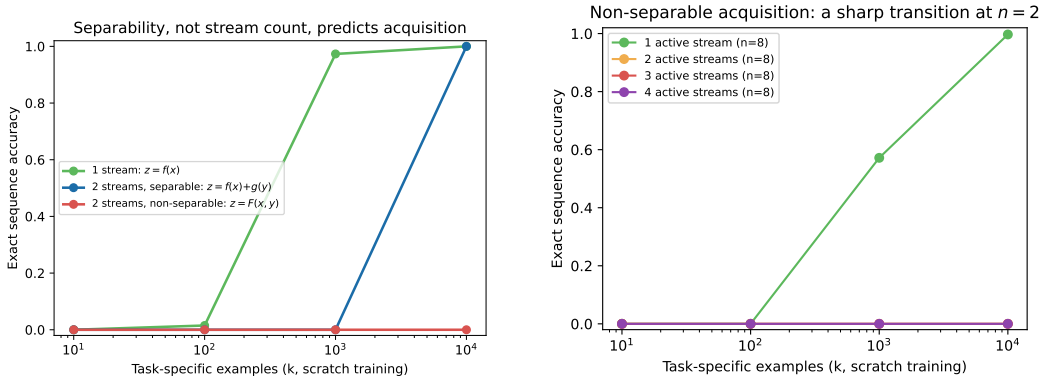

Figure 1: **(Left) Separability, not stream count, predicts acquisition.** Scratch-training exact accuracy for a single-stream target ($z = f(x)$), a separable two-stream target ($z = f(x)+g(y)$ for independently sampled, arbitrary f, g), and a non-separable two-stream target ($z = F(x, y)$ for an arbitrary joint lookup table F), holding model size and training budget fixed. **(Right) Non-separable acquisition exhibits a sharp transition at two active streams.** Scratch-training acquisition curves for 8 freshly, independently sampled target algorithms per stream count $n \in \{1, 2, 3, 4\}$ (disjoint from the Main experiment's 24 targets), all drawn from the Level I linear family and therefore, per Section 6.2.1's later decomposition sweep, non-separable (trivial-transform) targets rather than a general claim about stream count itself.

Table 1: **The central controlled comparison.** Final scratch-training exact accuracy (largest tested budget, $k = 10{,}000$) and exact held-out sequence fraction for the three conditions of Figure 1.

Condition	Function	Exact Acc.	Fraction correct
1 stream	$z = f(x)$	100.0%	600/600
2 streams, separable	$z = f(x)+g(y)$	100.0%	600/600
2 streams, non-separable	$z = F(x, y)$	0.0%	0/600

Holding stream count fixed at two, changing only whether the joint dependence on x and y is separable moves exact accuracy from 0.0% (0/600 held-out sequences correct) to 100.0% (600/600 correct) – matching the single-stream control. This failure is partial rather than catastrophic at the digit level: per-digit symbol accuracy on the non-separable condition is 20.7%, above the 10.0% chance rate despite 0% exact-sequence accuracy.

We summarize this result as

$$\boxed{\text{single-stream} \approx \text{separable multi-stream} \gg \text{non-separable multi-stream}} \tag{2}$$

rather than as a comparison of one stream against two: stream count is a useful first diagnostic (Section 6.2.1), but separability is the property that actually explains the pattern, with one qualification developed in full there – separability alone is necessary but not sufficient. We state the boxed summary above as the paper's leading approximation, not its final word.

6.1 Question 1: Can a Pretrained Sequence Model Acquire a Target Algorithm Absent from Pretraining?

We first ask the most basic question this benchmark is designed to answer: does pretraining help a model acquire a target algorithm that does not exist until after pretraining is frozen? (Full acquisition curves in Appendix H.)

Across 24 independently generated target algorithms, neither scratch training nor pretrained fine-tuning reliably reaches 95% exact accuracy within the tested budget ($K_{0.95} > 10{,}000$ examples for both), so no sample-complexity reduction from pretraining can be quantified at this scale. This is meaningful because the complete target procedures are generated after pretraining, so any improvement cannot be attributed to memorization of the exact target algorithm – but it would equally be incorrect to read a positive result as learning without prior knowledge, since the model may still know the primitive operations composing the target. The appropriate summary is therefore:

At this scale, no acquisition regime reliably reaches 95% exact accuracy within the tested budget; where a comparison is possible, fine-tuning and scratch training reach similar asymptotic accuracy – no clear evidence that pretraining reduces the task-specific experience required to acquire a previously unseen algorithm.

This apparent uniform difficulty is misleading: the right-censoring above is an artifact of *averaging* over target algorithms of starkly different difficulty, not evidence that every target is comparably hard. **Aggregate performance obscures a bimodal acquisition structure**: Section 6.2 decomposes this same data by a single structural property and shows accuracy is strongly bimodal – near-100% for one subset of targets, exactly 0% for another – reconcilable with, not contradictory to, "no regime reliably reaches 95%": the two statements describe the same data at different levels of aggregation (Section 6.3 similarly presents a redesigned, single-stream-only pretraining ablation, materially cleaner than the original mixed-target comparison of Appendix H.9).

6.2 Question 2: What Determines Whether Acquisition Succeeds?

This subsection localizes the structural property of a target rule that determines whether acquisition succeeds: not stream count, but whether the dependence on input streams is **separable** into independent per-stream transformations.

6.2.1 Combining Independent Input Streams Is a Sharp Bottleneck

This is this paper's central finding, concerning **gradient-based acquisition specifically** (scratch training and fine-tuning); we report in-context learning below only as a secondary check. A post hoc analysis of the aggregate acquisition data above reveals a far sharper pattern once we partition targets by structure: each Level I target depends on two input digit streams, x and y, giving *single-stream* ($a = 0$ or $b = 0$) and *dual-stream* ($a \neq 0$ and $b \neq 0$) targets. Across the 7 sampled single-stream and 17 sampled dual-stream targets in our main experiment, at $k = 10{,}000$, scratch training reaches 99.9% exact accuracy on single-stream targets (1,399/1,400 held-out sequences correct) versus 0.0% on dual-stream targets (0/3,400 correct); fine-tuning shows the same pattern.

Critically, dual-stream targets with $g = 0$ – pure pointwise functions of x_i and y_i with no state or memory at all – fail exactly as completely as dual-stream targets with real carry propagation, localizing the failure to combining two independently varying input streams rather than to sequential state maintenance. This is also not a coverage artifact of the smaller effective input domain single-stream targets present: at $k = 10{,}000$ the full 100-combination dual-stream domain is already sampled roughly two orders of magnitude over, yet accuracy remains exactly zero.

To test whether this generalizes beyond the 24 Main-experiment targets, and whether difficulty degrades gradually or steps sharply, we ran a controlled follow-up: 8 freshly, independently sampled targets for each of $n \in \{1, 2, 3, 4\}$ active input streams. Figure 1 shows the result.

The pattern replicates exactly and does not degrade further beyond two streams: at $k = 10{,}000$, the 8 sampled single-stream targets reach 99.7% exact accuracy (1,595/1,600 correct), while all 8 sampled two-, three-, and four-stream targets remain at exactly 0.0% (0/1,600 each). Acquisition under scratch training is therefore better described as a step function at $n = 1$ than as a difficulty gradient in stream count – for this target family; as the decomposition sweep below shows, stream count itself is not the final explanation, since some two-stream (and more-stream) targets are acquired as readily as single-stream ones (in-context learning shows no such split at all, Appendix H).

Given that the failure requires two streams, does the way they are combined matter? Table 2 tests this directly, holding stream count fixed and varying the composition: single-stream offset and scaling, additive and multiplicative two-stream compositions with no state, two-stream compositions separable into independently arbitrary per-stream transformations ($z = f(x)+g(y)$, $z = f(x)g(y)$), an arbitrary two-stream lookup table, and the two-stream-plus-state example from Section 1.

– the failure boundary falls at the number of streams, not at any particular algebraic structure. The factorized category complicates this picture: $z = f(x) + g(y)$ for two independently-sampled, arbitrary single-variable lookup functions f, g reaches 100.0% – succeeding where both the trivial-transform additive rule ($f = g = \mathrm{id}$, 0.0%) and the fully joint arbitrary lookup (0.0%) fail. All three nominally combine "2 streams, arbitrary values," yet only the additively separable one – expressible

Table 2: **Which compositions of x and y fail?** Final scratch-training exact accuracy (largest tested budget) for each named composition category, holding model size and training budget fixed.

Category	Function	Streams	Exact Acc.
Identity	$z = x$	1	100.0%
Offset	$z = x + 1$	1	99.8%
Scaling	$z = 2x$	1	100.0%
Additive	$z = x + y$	2	0.0%
Multiplicative	$z = xy$	2	1.0%
Factorized, additive (random f, g)	$z = f(x)+g(y)$	2	100.0%
Factorized, multiplicative (random f, g)	$z = f(x)g(y)$	2	100.0%
Factorized, additive (random *permutation* f, g)	$z = f(x)+g(y)$	2	0.0%
Factorized, additive (constant *shift* f, g)	$z=(x+c_1)+(y+c_2)$	2	0.0%
Lookup	$z = F(x, y)$	2	0.0%
Additive+state	$z = 2x+3y+c$	2, state	0.0%

as two independent per-stream lookups summed together, rather than one entangled function of both inputs – is learned (multiplicative factorization gives the same answer). This refines the bottleneck to specifically non-separable joint dependence: separability, not the combining operator, predicts the outcome.

Separability is necessary but not sufficient. If the model can learn arbitrary $z = f(x)+g(y)$, why can it not learn the special case $z = x + y$, itself separable? The default factorized tables are sampled i.i.d., yielding a true permutation with probability only $\approx 0.036\%$ at $B = 10$ – so f, g are, with overwhelming probability, non-injective, and may compress x and y's effective information content rather than relabel them. Testing this directly, a bijective (permutation) version of f, g reaches only 0.0% (0/600) at $k = 10{,}000$, comparable to plain addition, and the simplest nontrivial bijection (a constant cyclic shift) fails identically. Separability is therefore necessary but not sufficient: only a separable composition through sufficiently *compressive* (non-injective) per-stream transforms succeeds. Appendix I rules out output-entropy, implementation-leakage, and coincidental-optimization-basin explanations for this gap and narrows it to input-domain compression; *why* that compression eases optimization remains open.

6.2.2 Ruling Out Complexity, Architecture, and Optimization Confounds

This dichotomy admits further alternative explanations beyond algebraic structure, which we test directly here: an artifact specific to the Transformer architecture; a confound of task complexity rather than stream count; a data- or optimization-budget artifact; or that the target functions are intrinsically hard outside this paper's sequence-model framing (full numeric detail for every control below, plus positional misalignment and hidden-state probing, is in Appendix I).

Architecture. Re-training the *identity*, *additive*, and *lookup* categories of Table 2 from scratch with two non-Transformer causal backbones at matched parameter count – an LSTM, and a purely local causal convolution-plus-MLP backbone with no attention or recurrence at all – reproduces the same pattern: Transformer 100.0%/0.0%/0.0%, LSTM 97.2%/0.0%/0.0%, Conv+MLP 100.0%/0.0%/0.2% (identity/additive/lookup) – not specific to attention, recurrence, or any particular sequence-mixing mechanism.

Task framing and complexity. Fitting the same additive/lookup functions with an ordinary i.i.d. supervised MLP directly on $(x, y) \to z$ pairs – no token sequence, causal masking, or autoregressive generation – reaches 100.0%/100.0% at $k = 10{,}000$: the bottleneck is specific to the causal, autoregressive framing, not to the functions themselves. Complexity is similarly not the operative variable: fixing the second stream y to one constant value for the entire run recovers acquisition for both categories (100.0% additive, 100.0% lookup) at $k = 10{,}000$, so whether a second stream *varies*, not the target's complexity, is what matters.

Correlation and optimization budget. Coupling x and y progressively restores acquisition – identical streams ($y = x$) reach 100.0%/97.3% (additive/lookup), degrading smoothly as correlation weakens – consistent with coupling reducing the effective joint variability of the input space. Extending training to $10\times$ the standard budget leaves additive/lookup at exactly 0% while an identity control saturates, so the failure is neither data- nor optimization-limited (Appendix G); a hidden-state probe similarly finds only a partial, asymmetric representational trace, not a clean split (Appendix I).

Together, these controls rule out architecture, task complexity, and optimization budget; what partially resolves the failure is coupling the two streams, confounded with reduced input variability. We describe the bottleneck conservatively as a difficulty representing and/or computing a non-separable joint function of two independently varying streams within the causal, autoregressive framing studied here – using "binding" only as informal shorthand, not a claim about mechanism (Appendix I). Question 3 (Section 6.3) next asks whether scale or pretraining exposure removes it; Appendix H additionally covers explicit rule-following, primitive-overlap effects, teaching informativeness, and length/depth generalization.

6.3 Question 3: Does Pretraining or Scale Remove the Bottleneck?

Question 2 localized the failure to two independently varying streams and ruled out architecture, complexity, and optimization budget at one model size and pretraining composition; this subsection asks whether changing what the model is pretrained on, or scaling it up, removes it.

6.3.1 Pretraining Composition Determines What Transfers

Our first attempt at this question (Appendix H.9) mixed single- and dual-stream targets, which – given the stream-count dichotomy – floors every condition's aggregate result near 0% regardless of what pretraining actually contributes. We therefore redesigned the ablation to isolate single-stream targets only (headroom above the floor), across three matched-compute pretraining conditions: (A) natural language only; (B) the same primitive family as the eval target; and (C) a structurally different algorithmic family (arbitrary lookup tables, no algebraic relationship to Level I).

On 8 freshly sampled, held-out single-stream targets, all three conditions reach statistically indistinguishable fine-tuning accuracy at $k = 10{,}000$ (99.3%/99.7%/99.6% for A/B/C): pretraining composition does not measurably change what is acquired, even under condition (C), whose pretraining pool contains no Level I examples at all – directly addressing a specific version of the within-family generalization worry of Section 8.

Table 3: **Pretraining target-family ablation.** $K_{0.95}$ (task-specific examples/demonstrations to reach 95% exact accuracy) on freshly sampled, held-out single-stream targets, matched pretraining compute across all three conditions.

Pretraining condition	$K^{\mathrm{ICL}}_{0.95}$	$K^{\mathrm{FT}}_{0.95}$
(A) Natural language only	¿64	10,000
(B) Same primitive family	¿64	3,000
(C) Different algorithmic family	¿64	3,000

This does not extend to dual-stream targets: pretraining further checkpoints on single-stream-only, two-stream-only, and mixed-family corpora and evaluating on held-out two-stream targets, fine-tuning reaches at most 0.0% at $k = 10{,}000$ under every condition. Extensive practice combining two independently-varying streams – just never the specific held-out instance – confers no advantage over never having seen a two-stream example at all, evidence against a simple data-coverage explanation (alternatives in Appendix H) and consistent with a more specific reading: the representation does not acquire a reusable capacity for computing non-separable functions of two independent streams.

6.3.2 Does Scale Remove the Bottleneck?

Re-running the identity/additive/lookup categories of Table 2 at every model size used in the scaling experiment below (a $271\times$ parameter range) leaves the 2-stream floor unchanged at every size (additive/lookup never exceed 0.0%/0.0% at the largest tested budget); identity saturates at every size except the smallest, which reaches 100.0% given $5\times$ more steps – a convergence-speed effect, not a capacity limit. The dichotomy is not an artifact of insufficient parameter count within this range. We additionally ask whether acquisition efficiency in general improves with model scale, fitting $K_\tau(N) = aN^{-b} + c$ to peak in-context accuracy across 5 model sizes spanning a $271\times$ range (116,608–31,656,064 parameters; figure in Appendix H). Exact accuracy is exactly 0% at every tested demonstration count for every size – a flat floor rather than merely a failure to reach 95%, so $K_{0.95}$ vs. model size could not be fit at all, leaving open whether the asymptotic term c would eventually approach zero at larger scale.

Taken together, neither pretraining composition nor scale removes the non-separability bottleneck within the ranges tested here: a $271\times$ parameter increase leaves the pattern unchanged, and no

pretraining exposure we tested converts a dual-stream target into an acquirable one. What pretraining reliably buys, per Question 1, is a reduction in sample complexity on targets that are acquirable at all.

7 Analysis and Discussion

7.1 What Does "Learning the Rule" Mean?

Our results suggest algorithm acquisition is graded, not binary: local fit $\subset$ algorithmic generalization $\subset$ systematic execution. **Local acquisition** (correct on examples like those taught), **procedural acquisition** (generalizing to unseen input combinations), and **systematic execution** (correct as sequence length and state depth grow) form an increasingly stringent hierarchy; a model can clear the first without the third. Our length/depth results (Appendix H) are floor-limited by acquisition itself rather than execution at this scale, so we do not report a per-level breakdown. The motivating human comparison, and why generating the target only after pretraining rules out memorization, is in Appendix I.

Interpreting the bottleneck. This paper's central, falsifiable claim – a sharp bottleneck at non-separable two-stream computation, narrower than whether pretraining enables efficient acquisition of new procedures in general – survives the architecture, complexity, optimization-budget, pretraining-composition, and scale controls of Sections 6.2.2 and 6.3: what limits acquisition is a representational bottleneck in computing a non-separable joint function of independently varying streams, not sequential state, pretraining exposure, or stream count itself – though a bijective version of the same separable composition fails just as completely, an exception we do not fully explain. We summarize the broader picture as **prior structure + task-specific information + state maintenance $\rightarrow$ algorithmic behavior**: pretraining may chiefly supply the first, explicit demonstrations the second, and reliable long-range execution the third – distinct from acquisition itself.

8 Limitations

Synthetic tasks and target family. NORA uses synthetic algorithms for experimental control, so results may not transfer to natural mathematical, programming, or scientific reasoning; all quantitative results are further restricted to the Level I linear-stateful family (Section 4), and whether findings generalize to richer families is open. A novel target also does not guarantee the model never saw its components (partially addressed by our primitive-overlap experiments), and Level I's hypothesis space is small enough by construction (625 parameter combinations, 541 surviving filtering, 270 held out) that success could partly reflect within-family generalization rather than acquisition of structure genuinely absent from pretraining (Section 5). We evaluate multiple rule representations rather than treating any one format as canonical.

Scale and human comparison. Our pretraining experiments use models far smaller than frontier language models: the scratch-training dichotomy holds across a $271\times$ parameter range (Section 6.3.2), but our in-context scaling experiment is inconclusive (exact accuracy at 0% throughout, no trend to extrapolate; Appendix H), so neither result can yet be claimed at frontier scale. Nor do we run a human learning experiment – the child-addition example only motivates studying rapid acquisition, not a benchmark against it.

9 Conclusion

We introduced NORA, a controlled framework built on one methodological idea – generate the target algorithm after pretraining – to ask what pretraining actually buys when the task itself is new. Acquiring arbitrary novel procedures from compact descriptions would support a reusable substrate for procedural learning; an advantage restricted to familiar primitives would instead suggest recombination of prior structure; and rapid local-rule acquisition without systematic generalization would reveal a separation between rule inference and execution. We are careful about scope: the non-separability bottleneck is demonstrated only in the small-scale regime detailed under Limitations, not a claim about frontier-scale models – NORA's methodology makes that question testable at any scale, and whether the bottleneck persists, weakens, or disappears there remains open.

10 AI USE STATEMENT

We used Claude Code (Anthropic, running Claude Sonnet 5) as a coding and research assistant: implementing the NORA codebase, running the experiments and ablations reported here, and drafting portions of this manuscript's prose. All research decisions – what questions to ask, what experiments to run, how to interpret the results, and what to claim – were made by the human author(s), who reviewed all LLM-assisted work and take full responsibility for the content of this paper. No LLM is an author of this work.

REFERENCES

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*, 2020.

Brenden M Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations (ICLR)*, 2023.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Yiyou Sun, Yuhan Cao, Pohao Huang, Haoyue Bai, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. RL grokking recipe: How does RL unlock and transfer new algorithms in LLMs? In *International Conference on Learning Representations (ICLR)*, 2026.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations (ICLR)*, 2020.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Qinyuan Ye, Robin Jia, and Xiang Ren. Function induction and task generalization: An interpretability study with off-by-one addition. In *International Conference on Learning Representations (ICLR)*, 2026.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

A Formal Problem Definition

Let M_ϕ denote a model with parameters ϕ. Pretraining produces $\phi_0 = \mathrm{Train}(\mathcal{D}_{\mathrm{pretrain}})$. Crucially, the target algorithm $f_{\theta^\star}$ is generated only after ϕ_0 is fixed. The model then receives task-specific information T and must predict $f_{\theta^\star}(x)$ for previously unseen inputs x. We define the **acquisition problem** as

$$\phi^\star = \mathrm{Acquire}\left(\phi_0, T, f_{\theta^\star}\right). \tag{3}$$

Different acquisition regimes correspond to different forms of T:

$$\text{in-context learning:} \quad T = \{(x_j, f_{\theta^\star}(x_j))\}_{j=1}^{k}, \tag{4}$$

$$\text{explicit procedural teaching:} \quad T = S(\theta^\star), \text{ a textual or formal specification,} \tag{5}$$

$$\text{gradient-based adaptation:} \quad \phi^\star = \mathrm{Update}\left(\phi_0, \mathcal{D}_k\right), \tag{6}$$

$$\text{scratch learning:} \quad \phi_0 \sim \text{RandomInitialization.} \tag{7}$$

The central quantity is the acquisition function

$$A_r(k) = \mathbb{E}_{\theta^\star, x}\left[\mathbf{1}\{M_r(x \mid T_k) = f_{\theta^\star}(x)\}\right], \tag{8}$$

where r denotes the acquisition regime and k denotes the number of task-specific demonstrations when applicable.

B Detailed Benchmark Construction

B.1 Parameter Sampling

For the Level I family, we sample $a, b, g \sim \mathrm{Uniform}(\mathcal{A})$ and $d \sim \mathrm{Uniform}(\mathcal{D})$. We reject parameterizations that satisfy any of the following:

1. output entropy below 1.5 bits;
2. state entropy below 1.5 bits (the same criterion in this implementation; a separate state-only entropy filter was not built for this pass);
3. fewer than 3 reachable states;
4. equivalence to a previously retained algorithm.

C Exact Functional Equivalence

Because the input alphabet and state space are finite, two candidate parameterizations can sometimes produce identical functions despite having different parameter values. For each candidate algorithm θ, we construct its complete transition table $T_\theta = \{F_\theta(x, y, c) : x, y \in \mathcal{X}, c \in \mathcal{C}\}$; algorithms with identical transition tables are treated as the same functional algorithm.

D INFORMATIVE TEACHING

The exact teaching procedure is possible because the Level I hypothesis class is finite. Given the current version space V_t, each candidate example e partitions the version space according to its predicted output, $V_t(e) = \{\theta \in V_t : f_\theta(e) = f_{\theta^\star}(e)\}$. The greedy teacher selects

$$e^\star = \arg\min_e |V_t(e)|\,, \tag{9}$$

and we additionally report the information gain $I(e) = \log_2 |V_t| - \log_2 |V_t(e)|$, an interpretable measure of the usefulness of each demonstration.

E PROMPT TEMPLATES

E.1 NATURAL-LANGUAGE RULE

> You are given two sequences of digits. Process the digits from right to left. Maintain a state initialized to zero. At each position, compute the specified expression using the two input digits and the current state. The output digit is the remainder after division by the base. The quotient becomes the state for the next position. Apply this procedure exactly to the input below.

The exact wording is varied across templates to prevent models from learning a single fixed instruction pattern.

E.2 FORMAL RULE

```
state = 0
for i = n-1, ..., 0:
    s = a*x[i] + b*y[i] + g*state + d
    z[i] = s mod B
    state = floor(s/B)
return z
```

F ADDITIONAL GENERALIZATION TESTS

F.1 INPUT DISTRIBUTION SHIFT

Acquisition and test distributions differ along:

1. sequence length;
2. digit frequency;
3. state-transition frequency;
4. carry frequency;
5. magnitude of intermediate states.

This allows us to distinguish memorization of superficial statistics from procedural generalization.

F.2 CARRY-CASCADE TESTS

We explicitly construct examples that produce long cascades of state updates: for a cascade of length r, $c_i \neq c_{i+1}$ for $i = 1, \ldots, r$. We measure $P(\text{correct} \mid r)$ for increasing r, a particularly stringent test of sequential execution.

G COMPUTE AND OPTIMIZATION DETAILS

We report pretraining and task-specific (scratch/fine-tuning) optimization hyperparameters separately below, since they are distinct training phases with different budgets: pretraining exposes a model to a fixed corpus once, before any target algorithm exists; task-specific training exposes an already-instantiated model to k examples of one target algorithm, and is what Sections 6 and 6.2.1 report acquisition curves over.

Pretraining. All pretraining conditions use:

- AdamW optimizer;
- learning rate: 0.0006;
- weight decay: 0.01;
- batch size: 16;
- warmup: 50 steps;
- training steps: 625 ($\approx$20.5M tokens processed);
- gradient clipping: 1.0.

Scratch training and fine-tuning. All task-specific gradient-based acquisition uses:

- AdamW optimizer;
- learning rate: 0.0003;
- weight decay: 0.01;
- batch size: 32 (or all k examples, whichever is smaller);
- warmup: 100 steps;
- training steps: 2000;
- gradient clipping: 1.0.

Scratch training and fine-tuning always use identical optimization budgets (same steps, same learning rate), differing only in initialization (random vs. pretrained), so any difference between them reflects initialization alone, not an optimization advantage. As a check against a slow-optimization explanation, we additionally trained the additive and lookup categories for 20,000 steps ($10\times$ the standard budget): identity (exact accuracy stays at 67.2%, symbol accuracy stays within 80.0%–82.2%, while training loss falls from 0.01 to 0.00); additive (exact accuracy stays at 0.0%, symbol accuracy stays within 9.3%–10.8%, while training loss falls from 1.20 to 0.60); lookup (exact accuracy stays at 0.0%, symbol accuracy stays within 16.8%–17.9%, while training loss falls from 1.07 to 0.14).

The tokenizer and sequence representation are held fixed within each controlled model family.

H Additional Results

The controls ruling out complexity, architecture, and optimization-budget confounds for the stream-count dichotomy are reported in the main text (Section 6.2.2), and the scale and pretraining-composition controls in Section 6.3, rather than here, since they bear directly on this paper's central claims. This appendix reports supporting figures and experiments that do not bear directly on Questions 1–3: the aggregate pretrained-vs-scratch acquisition figure and the in-context scaling figure referenced from the main text, explicit rule-following, primitive-familiarity overlap, teaching-example informativeness, length/depth generalization, and (in Appendix H.9) the original, superseded mixed-target pretraining-composition comparison.

H.1 Aggregate Acquisition Across Regimes

Section 6.1 reports the aggregate comparison between pretrained and scratch-trained acquisition across 24 independently generated target algorithms; Figure 2 shows the full acquisition curves underlying that comparison.

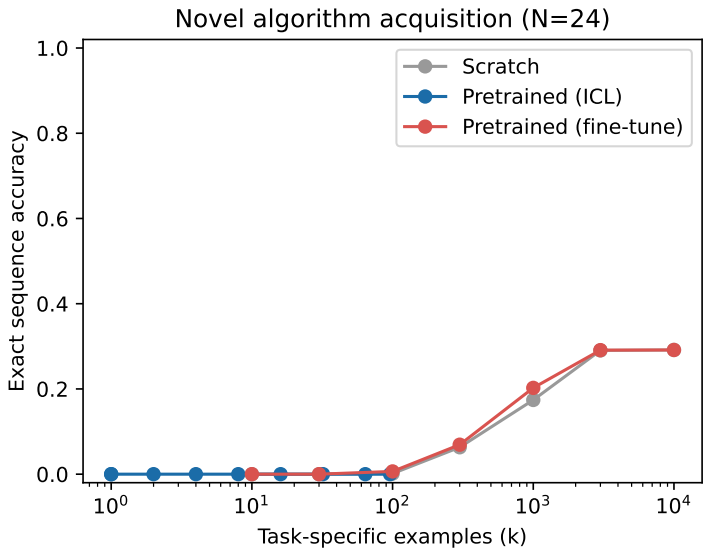

Figure 2: **Pretraining and novel algorithm acquisition.** Exact sequence accuracy on previously unseen target algorithms as a function of task-specific examples. Target algorithms are generated after pretrained checkpoints are fixed. Neither regime reliably reaches 95% exact accuracy within the tested budget, so no sample-complexity reduction from pretraining could be quantified at this scale.

Section 6.2.1's stream-count sweep (Figure 1) is scratch-training only; under in-context learning the same sweep shows no stream-count split at all – exact accuracy remains at 0% for single- and dual-stream targets alike – suggesting that whatever structure gradient-based adaptation discovers for single-stream targets is not accessible through in-context inference alone at this scale.

H.2 Scaling of In-Context Acquisition

Section 6.3.2 reports peak in-context exact accuracy as a function of model size, fitting $K_\tau(N) = aN^{-b} + c$; Figure 3 shows the underlying curve.

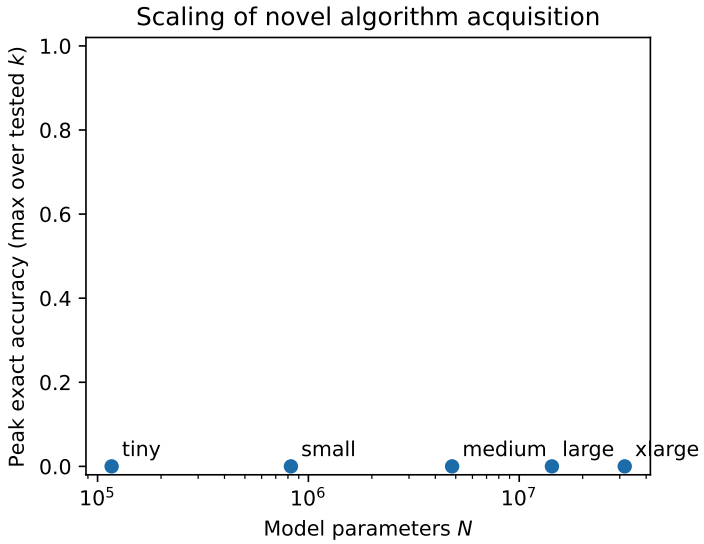

Figure 3: **Scaling of novel algorithm acquisition.** Peak in-context exact accuracy (over the tested demonstration budget) as a function of model size; $K_{0.95}$ is not shown because no tested size reached the threshold, so peak accuracy is plotted instead of leaving the figure empty.

H.3 Explicit Rule Following Is Distinct from Example-Based Acquisition

We next ask whether pretrained models can execute a target algorithm when its procedure is explicitly specified.

We evaluate both natural-language descriptions and formal pseudocode.

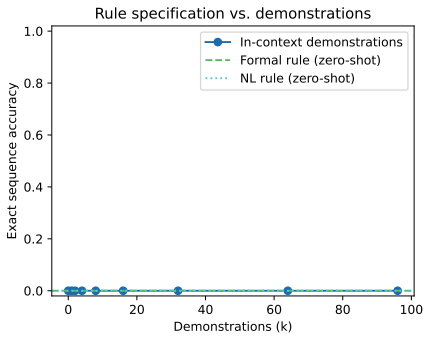

Figure 4: **Rule specification versus demonstrations.** Performance of pretrained models given an explicit specification, zero demonstrations, or increasing numbers of demonstrations.

The results are shown in Figure 4.

Under formal specifications, models achieve 0.0% exact sequence accuracy, while natural-language specifications yield 0.0%.

This demonstrates that neither specification format enabled reliable zero-shot execution at this scale – the pretrained model could not execute the rule from either a formal or a natural-language description.

Importantly, rule-only acquisition and demonstration-based acquisition should not be interpreted as equivalent information budgets. A rule may directly communicate the computational structure, whereas demonstrations communicate constraints on the unknown structure.

We therefore treat the two conditions as complementary measurements of procedural acquisition.

This zero-shot failure invites a further question: does it originate in a single-step computation (parsing the rule and computing one output digit), or only once multiple steps must be composed or a carry maintained? We test this directly by evaluating the same pretrained model at output lengths $L \in \{1, 2, 4, 8\}$, and by measuring per-digit accuracy at each position within an $L = 8$ sequence.

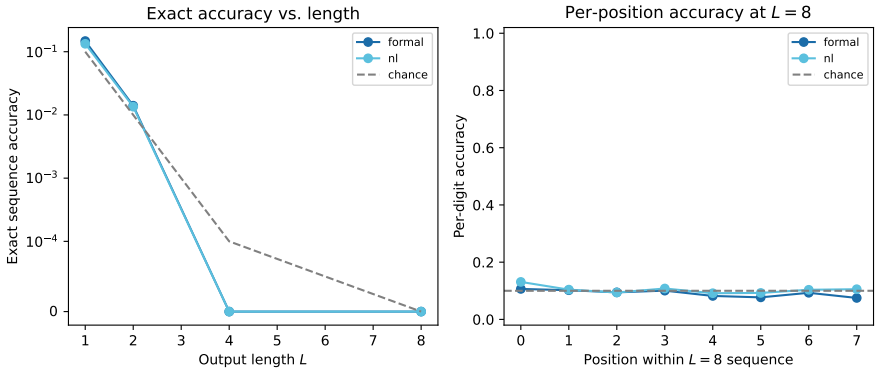

Figure 5: **Rule-execution step diagnostic.** Left: exact sequence accuracy as a function of output length L, against the chance baseline B^{-L} (dashed). Right: per-digit accuracy at each position within an $L = 8$ sequence, against the chance baseline $1/B$ (dashed).

Under the formal spec, $L = 1$ exact accuracy is 14.8% (chance 10.0%) but falls to 1.4% by $L = 2$ (chance 1.0%), and per-digit accuracy at $L = 8$ stays within 7.5%–10.7% at every position, with no decline from position 0 onward; under the nl spec, $L = 1$ exact accuracy is 13.4% (chance 10.0%) but falls to 1.4% by $L = 2$ (chance 1.0%), and per-digit accuracy at $L = 8$ stays within 9.2%–13.1% at every position, with no decline from position 0 onward. This indicates that whatever small edge over chance the model shows at $L = 1$ does not compound across even two steps, and that the zero-shot failure is not well described as specifically a multi-step composition or state-tracking failure: even the single, state-free first digit is already indistinguishable from chance once embedded in a longer query.

As a representation control on the function-decomposition sweep of Section 6.2.1, we re-ran its identity (1-stream) and additive (2-stream) categories under an alternate, opaque 10-symbol alphabet in place of decimal digits, with everything else unchanged. The identity (1-stream) category still reaches 100.0% exact accuracy while the additive (2-stream) category reaches 0.0%, reproducing the same dichotomy seen under decimal digits and indicating it is not an artifact of that specific token representation.

H.4 Testing Primitive Familiarity as an Alternative Explanation

The dichotomy of Section 6.2.1 could in principle be explained not by the number of input streams itself, but by dual-stream targets happening to be built from less familiar primitives. We test this alternative directly by evaluating target algorithms with different degrees of overlap with familiar computational primitives, defining three primitive-overlap conditions – **high overlap** (familiar operations with novel parameter combinations), **medium overlap** (familiar operations combined in novel procedural structures), and **low overlap** (random finite transition systems) – giving the empirical prediction $K_\tau^{\text{high}} \leq K_\tau^{\text{medium}} \leq K_\tau^{\text{low}}$, where K_τ is the number of examples required to reach accuracy threshold τ.

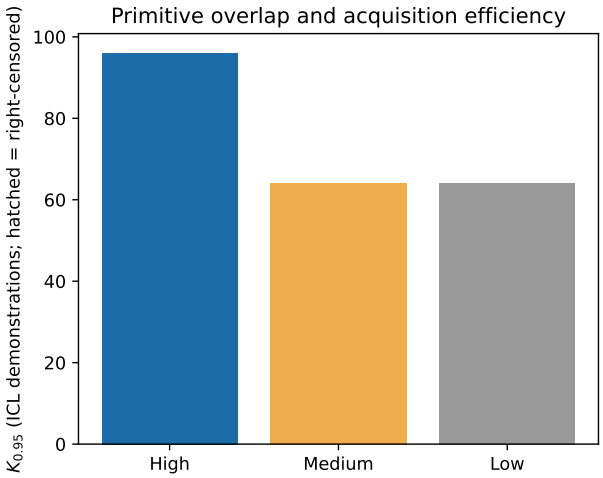

Figure 6: **Primitive overlap and acquisition efficiency.** The target algorithm is novel in all conditions, but the computational primitives used to construct it vary in familiarity.

We observe $K_{0.95}^{\text{high}} => 96$, $K_{0.95}^{\text{medium}} => 64$, and $K_{0.95}^{\text{low}} => 64$.

Thus, the observed ordering across overlap conditions does not strictly follow the predicted $K_{0.95}^{high} \leq K_{0.95}^{medium} \leq K_{0.95}^{low}$ pattern at this scale.

This experiment matters independently of whether the predicted ordering holds: if primitive familiarity were the primary driver of the stream-count dichotomy, we would expect the high-overlap condition – built from the most familiar primitives – to clearly separate from the low-overlap condition. That it does not gives no support to a primitive-familiarity account of the dichotomy, leaving the number of independently varying input streams as the more parsimonious explanation on the evidence collected so far. We note one caveat on this operationalization: high/medium/low overlap are defined by construction (which primitives appear in the synthetic "Algorithmic" pretraining corpus), not verified against what the pretrained model actually represents internally (e.g., via probing or embedding similarity) – so this experiment tests whether our constructed overlap categories predict acquisition, not whether primitive familiarity in general does.

H.5 Teaching Examples Are Not Equally Informative

Human instruction is rarely a random sample from the space of possible observations. We therefore compare random demonstrations against curriculum and information-efficient teaching.

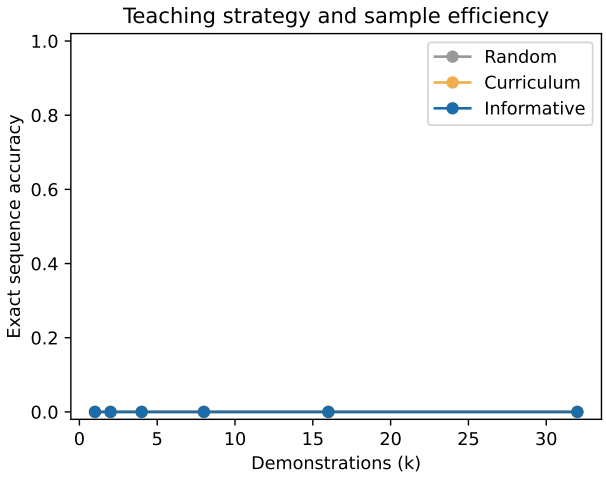

Figure 7: **Teaching strategy and neural sample efficiency.** Exact sequence accuracy under three demonstration-selection strategies. The informative teacher is constructed to greedily minimize the version space of algorithms consistent with the examples seen so far – by design, it always reduces the version space at least as fast as random sampling. Whether this translates into faster acquisition by the neural learner is the empirical question this figure addresses.

Let $V_t = \{\theta : f_\theta(e_j) = f_{\theta^\star}(e_j), \forall j \leq t\}$. Our informative teacher selects $e_{t+1} = \arg\min_e |V_{t+1}|$.

The results show the demonstration budget tested was insufficient to fully separate teaching strategies.

For example, informative teaching reaches 95% accuracy after ¿32 demonstrations compared with ¿32 for random demonstrations.

If informative teaching reaches the accuracy threshold with substantially fewer demonstrations, this would show that the number of examples alone is an incomplete measure of learning difficulty, and that a neural learner can exploit demonstrations selected to be maximally informative in the version-space sense. At the tested budget, our data does not yet let us distinguish this from the alternative – that the model fails to exploit information-theoretic informativeness regardless of budget.

H.6 Acquisition and Systematic Generalization Are Different Capabilities

A model may correctly infer the local transition rule while failing to execute that rule repeatedly over long sequences.

We therefore evaluate sequence-length extrapolation.

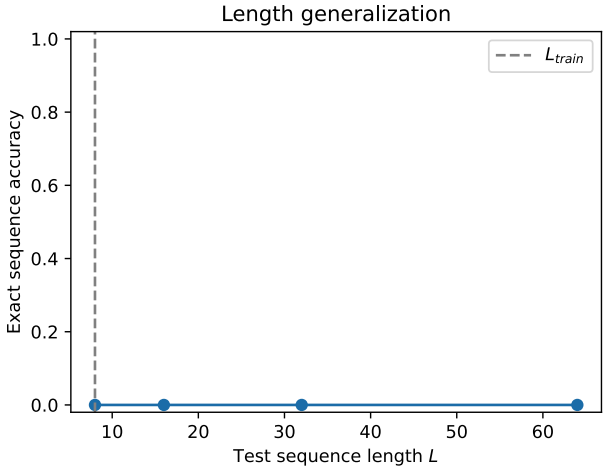

Figure 8: **Length generalization.** Models acquire target algorithms using sequences up to L_{train} and are evaluated on substantially longer sequences.

We find exact accuracy was already only 0.0% at $L = 8$ (the acquisition length itself), so length generalization could not be meaningfully assessed beyond noting that acquisition, not extrapolation, is the binding constraint at this scale.

In particular, if $\text{ExactAcc}(L = 8) = 0.0\%$ but $\text{ExactAcc}(L = 64) = 0.0\%$, then high acquisition accuracy does not imply systematic execution. This distinction is central to our interpretation: **acquisition** means identifying the target transition rule, while **execution** means repeatedly applying it correctly. A model can succeed at the first while failing at the second.

H.7 Sequential State Depth Reveals a Second Failure Mode

Sequence length alone does not isolate the difficulty of recurrent computation. We therefore measure the number of nontrivial state updates, $D(x, y) = \sum_i \mathbf{1}[c_{i+1} \neq c_i]$, and evaluate matched-length examples at different values of D.

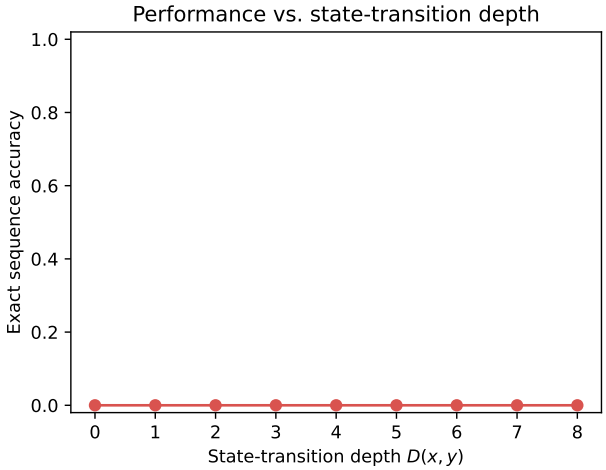

Figure 9: **Performance as a function of state-transition depth.** Inputs are matched by length but require different numbers of nontrivial state transitions.

We observe accuracy does not show a clear monotonic dependence on state-transition depth at this scale.

If accuracy decreases approximately monotonically with state depth, this indicates that the limiting factor is not simply token length but the ability to maintain and repeatedly transform an internal computational state.

H.8 Alternative Explanations for the Two-Stream Pretraining Null Result

Section 6.3.1 reports that pretraining on single-stream-only, two-stream-only, or mixed-family corpora all fail equally to enable acquisition of held-out two-stream targets (0.0% at $k = 10{,}000$ under every condition), evidence against a simple data-coverage explanation for the stream-count bottleneck ("the pretraining corpus never modeled two-stream composition"). This null result does not, however, rule out several related explanations: an insufficient pretraining budget specifically on two-stream compositions (more exposure, at matched compute, might behave differently); a pretraining target family that is too dissimilar despite superficially matching stream count; representations that are in fact learned during pretraining but are not accessible via the fine-tuning procedure used here; or optimization dynamics specific to this particular setup. The data are at least consistent with a more specific reading – that the representation does not acquire a reusable capacity for computing non-separable functions of two independent streams, even from extensive practice on other instances of exactly that computational pattern – but we flag these alternatives as not fully excluded.

H.9 Pretraining Composition: The Original Mixed-Target Design

Section 6.3.1 reports a redesigned, single-stream pretraining target-family ablation and a binding-mechanism ablation. Our first attempt at the pretraining-composition question, before that redesign, evaluated on a subset of the Main experiment's targets that mixes single- and dual-stream rules. Given the stream-count dichotomy of Section 6.2.1, this was a design flaw: dual-stream targets fail regardless of pretraining condition, so including them floors every condition's aggregate result at (or near) 0% regardless of what pretraining actually contributes, making the comparison below largely uninformative about what pretraining transfers. We report it here for completeness only.

Table 4: **Effect of pretraining composition (original, mixed-target design).** All conditions use approximately matched token and compute budgets. See Section 6.3.1 for the redesigned, single-stream ablation, a cleaner test of what pretraining transfers.

Pretraining	Rule-only	$K^{\mathrm{ICL}}_{0.95}$	$K^{\mathrm{FT}}_{0.95}$
None	0.0	–	¿10,000
Natural language	0.0	¿64	¿10,000
Language + code	0.0	¿64	¿10,000
Algorithmic data	0.0	¿64	¿10,000
Mixed	0.0	¿96	¿10,000

The strongest condition is not clearly identifiable, since no pretraining condition reliably reached the accuracy threshold within the tested budget.

If code or algorithmic pretraining substantially improves acquisition, this would suggest that exposure to procedural structures is especially important.

If natural-language pretraining provides a substantial advantage even without algorithmic data, this would instead suggest that general compositional or linguistic representations can support novel procedural acquisition.

I Extended Analysis

This appendix collects the extended mechanistic and control detail referenced from Sections 6.2.1 and 6.2.2, followed by a more careful discussion of the human-learning comparison motivating this paper.

I.1 The Conceptual Hierarchy in Full: Separability, Not Stream Count, as the Deeper Explanation

Section 7 states that the bottleneck this paper reports is not primarily a question of pretraining in general: whether pretraining helps acquisition *in general* is the distinct, secondary question of Section 6.1 – a model tested on pretraining-familiar examples is tested partly on prior knowledge, whereas one given a target generated after pretraining is tested on incorporating task-specific information into an already-learned substrate. We expand here on the conceptual hierarchy the evidence actually supports. It is not an ordinal claim about stream count – directly contradicted by the decomposition sweep – but separable $\gg$ non-separable, with stream count only a convenient way to *expose* that difference: every multi-stream target in our stream-count sweep happens to be non-separable, so the sweep looks like a stream-count effect until the decomposition sweep varies separability while holding stream count fixed and reveals the same boundary. What limits acquisition is therefore a more basic representational bottleneck – computing a non-separable joint function of independently varying streams – not primarily sequential state, pretraining exposure, or stream count. This approximation has an exception we do not fully explain: a bijective version of the same separable composition fails just as completely as the non-separable cases, so separability is necessary but not sufficient, and information compression in the per-stream transforms is doing real work our experiments do not mechanistically account for (Appendix I.2 rules out several alternative explanations for this specific gap).

I.2 Ruling Out Alternative Explanations for the Separable/Bijective Gap

Section 6.2.1 reports that $z = f(x) + g(y)$ for independently-sampled, arbitrary (non-injective) f, g reaches 100.0% exact accuracy, while both plain addition ($f = g = \mathrm{id}$) and a bijective (permutation) version of f, g reach 0.0%. This raises a natural mechanistic question: what, specifically, distinguishes the succeeding default tables from the failing bijective ones and from plain addition? We consider and rule out several candidate explanations directly rather than leaving the finding as an unexamined curiosity.

Output-distribution effects (the random tables inducing an easier label distribution) are not the explanation: the reconstructed tables show mean output entropy of 3.26 bits for the (succeeding) default tables and 3.32 bits for the (failing) bijective tables, both close to the maximum possible 3.32 bits and to plain addition's exactly uniform distribution – output-marginal entropy does not distinguish success from failure. Training-implementation shortcuts or representation leakage are ruled out by construction: every category in this sweep uses identical demonstration-sampling, tokenization, rendering, model architecture, and optimizer, and `Target.simulate` exposes only the resulting (x, y, z) triples to the training pipeline, never f or g themselves. A coincidental optimization basin is also ruled out: each condition reproduces cleanly across 3 independently sampled tables and training seeds (100.0% for all 3 default replicates, 0.0% for all 3 bijective replicates), so this is not a single lucky or unlucky run.

What the tables actually differ in is injectivity of f, g themselves: the default tables use only 6.7/7.0 distinct f/g values on average (out of 10), so the 100 (x, y) input pairs collapse onto only 46 distinct $(f(x), g(y))$ combinations, while the bijective tables (like identity) preserve all 100. This points to a specific, narrower hypothesis: it is compression of the effective per-stream input space – not separability, output entropy, or arbitrariness per se – that determines the outcome. Why this specific form of compression eases gradient-based optimization, rather than merely reducing the combinatorial size of an already-learnable table, remains the open question this control does not resolve.

I.3 Positional Misalignment and Hidden-State Probing

Section 6.2.2 reports that coupling x and y progressively restores acquisition. We additionally tested same-valued but positionally misaligned streams ($y_i = x_{i-1}$): this gives a partial and unstable recovery for the additive category (53.5%) but no recovery at all for the lookup category (0.0%), since an arbitrary table has no structure for a shifted-but-otherwise-correct value to exploit. This is consistent with coupling helping via reduced effective joint variability of the input space rather than via a more specific positional-alignment mechanism.

The behavioral controls in Section 6.2.2 rule out several alternative explanations but remain entirely behavioral. We therefore test one mechanistic question directly: does a model that fails to produce

$z = (x + y) \bmod B$ still represent x and y individually, or does representation collapse along with behavior? We train linear and MLP probes to decode x_i, y_i, and $(x_i + y_i) \bmod B$ from the pre-head hidden state at the position where the model is about to emit z_i, on a model trained on the failing *additive* target. An identity-trained model ($z = x$, which behaviorally succeeds) serves as a methodological control: if a linear probe cannot recover x_i from this model, the probing setup itself – not the additive model's representations – would explain any null result.

The pattern observed is more asymmetric than a clean "both inputs represented, sum not" split: on the additive target (0.0% behavioral exact accuracy), x_i is recovered at 30.3% – clearly above the 10.0% chance rate, though far below the 100.0% ceiling the identical probing procedure achieves on the identity-trained model – while y_i (11.7%) and $(x_i + y_i) \bmod B$ (10.3% linear, 9.5% MLP) sit at chance. Rather than representing both streams individually and failing only to combine them, this model retains a weak, partial trace of x and effectively no linearly or nonlinearly decodable trace of y at the position where it must emit z_i – a more thoroughgoing representational collapse than a failure specific to computing the non-separable combination alone, at least at this single layer and position. We report this as a genuine, if more complicated, finding rather than the cleaner dichotomy we tested for; a fuller mechanistic account (different layers, attention patterns, or training checkpoints along the way) is left to future work.

A terminological note: Section 6.3 and the probing result above occasionally use the word "binding" to describe this bottleneck. We have not demonstrated a failure of binding in any mechanistically established sense – our probes test whether specific quantities are linearly or nonlinearly decodable from a single hidden state, a narrow, purely behavioral-cum-representational diagnostic, not an account of any underlying mechanism. We use "binding" only as informal shorthand for what we have actually shown: a difficulty representing and/or computing non-separable functions of independently varying inputs, and flag this explicitly so the term is not read as carrying stronger cognitive- or neural-representation implications than our evidence supports.

I.4 Task-Framing, Complexity, and Coupling Controls: Full Detail

Section 6.2.2 summarizes several confound-control experiments in the main text; we give the full numbers behind each here. Fitting the additive/lookup functions with an ordinary i.i.d. supervised MLP directly on $(x, y) \to z$ pairs – no token sequence, causal masking, or autoregressive generation – reaches 100.0%/100.0% at $k = 10{,}000$; a linear classifier, mathematically unable to represent the x-y interaction, still plateaus above chance (16.7%/47.0%), confirming that the interaction genuinely requires a nonlinear map once the sequence-model framing is stripped away. Fixing the second stream y to one constant value for the entire run – the identical random table used in the 2-stream condition, evaluated at a single fixed y_0 – recovers acquisition for both categories (100.0% additive, 100.0% lookup) at $k = 10{,}000$. The correlation sweep between x and y: identical streams ($y = x$) reach 100.0%/97.3% (additive/lookup), degrading to 34.5%/3.8% at $p = 0.25$ independent-replacement noise per digit and 4.5%/0.0% at $p = 0.6$ – consistent with coupling reducing the effective joint variability of the input space ($y = x$ collapses the B^2-pair domain to B) rather than correlation easing some other representational operation. Separately, extending training to 20,000 steps ($10\times$ the standard budget) across the full k-grid leaves additive/lookup at exactly 0% throughout while an identity positive control saturates, confirming the pipeline can register success at this budget: the dual-stream failure is therefore neither data- nor optimization-limited within this range (see also Appendix G for the underlying loss curves).

I.5 A More Precise Interpretation of the Human Comparison

The motivating comparison with children is useful but must be made carefully. A child does not learn addition in complete isolation: before learning multi-digit addition, the child has acquired counting, numerical concepts, symbolic notation, language, working-memory strategies, sequential procedures, and other relevant knowledge. Thus, a fair comparison is not "child vs. randomly initialized Transformer" but rather "experienced learner + new rule" versus "pretrained model + novel algorithm." Our benchmark does not claim that these systems are cognitively equivalent. Instead, it isolates one computational property that both systems plausibly possess: the ability to use prior knowledge to rapidly acquire a new procedure.

I.6 Why Internet Pretraining Creates a Fundamental Confound

Suppose a model has encountered 10^8 arithmetic examples during pretraining. If it subsequently solves $7{,}384{,}921 + 5{,}918{,}273$, the result does not establish that it learned the addition algorithm from a compact rule – the model may have learned a procedure, memorized common patterns, or interpolated among previously observed examples. Our methodology changes the causal ordering:

$$\text{pretraining} \rightarrow \boxed{\text{freeze model}} \rightarrow \boxed{\text{generate algorithm}} \rightarrow \text{teach algorithm.} \tag{10}$$

This ordering makes direct memorization of the complete target procedure impossible by construction. It does not remove all forms of transfer; indeed, we view such transfer as the phenomenon of interest.

J Full Results Tables

Table 5: Full results for the Level I target family (the only family evaluated in this pass; see Section 4).

Model	Level	$K_{.90}$	$K_{.95}$	$K_{.99}$	Length OOD	Depth OOD
Large (mixed-pretrained, ICL)	I	¿96	¿96	¿96	0.0%	0.0%

K Recommended Additional Experiments

The following experiments are useful if computational resources permit.

1. **Rule paraphrase robustness.** Generate semantically equivalent natural-language specifications using different wording and evaluate whether performance remains stable.
2. **Code representation.** Compare natural language, pseudocode, Python-like code, and mathematical notation.
3. **Adversarial demonstrations.** Provide demonstrations selected to be individually misleading while remaining globally consistent with the target rule.
4. **Order sensitivity.** Randomize the order of demonstrations and measure whether performance depends on presentation order.
5. **Negative examples.** Provide examples showing both correct and incorrect outputs and test whether the model can infer the rule from contrastive supervision.
6. **Compositional transfer.** Teach subroutine f and subroutine g separately, then test $f \circ g$ without having shown the composition.
7. **Cross-algorithm transfer.** Measure whether internal representations acquired for one target algorithm facilitate acquisition of another.
8. **Mechanistic analysis.** Probe internal states for representations of the latent variable c_i and test whether those representations transfer across target algorithms.

L Reproducibility Checklist

For every experiment we record:

- random seed;
- target algorithm identifier;
- algorithm parameters;
- model checkpoint;
- tokenizer;
- prompt template;
- number of demonstrations;

- sequence lengths;
- optimization hyperparameters;
- training steps;
- number of target algorithms;
- number of test examples;
- compute budget.

Target algorithms and all evaluation inputs should be generated using a deterministic public procedure so that every reported experiment can be reproduced independently.