WHAT DOES RLVR LEARN? BINDING, TABLE ACCESS, AND COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

When reinforcement learning with verifiable rewards (RLVR) raises accuracy, what has the model learned? The same accuracy gain can reflect different outcomes: binding a new name to a familiar operation, fitting an arbitrary table of input–output pairs, or composing operations into unseen programs. We introduce Alien DSLs, executable languages whose symbol assignments and operation tables are sampled after pretraining, allowing us to evaluate these outcomes separately. Under outcome-only GRPO, exhaustive input evaluation supports partial binding between new names and familiar operations. Within the tested training budget, random-table accuracy plateaus near 30%, revealing a fitting bottleneck before transfer is assessed. Dense supervision largely resolves fitting, yet accuracy on the same observed cells drops from 96.5% to 13.7% when the query form changes. High training accuracy therefore does not ensure access to an observed table entry. For composition of renamed familiar operations, continued execution-trace supervision outperforms trace-initialized GRPO on held-out programs in smaller models. In the larger model, both recipes exceed an oracle per-program constant baseline on unseen operation pairs, unseen templates, and deeper programs. These transfer gains recur across two sequentially preregistered waves totaling sixteen fresh languages under a frozen training protocol. Together, these results show why fitting and transfer must be evaluated separately before interpreting an RLVR accuracy gain as evidence of a newly acquired capability.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves performance on reasoning tasks by rewarding correct answers (Shao et al., 2024; DeepSeek-AI, 2025). What a model learns from this signal remains contested. Some studies find that RLVR mainly increases the probability of solutions already available to the pretrained model (Yue et al., 2025); others report gains on previously unsolved problems or unseen compositions (Liu et al., 2025; Yuan et al., 2026). Interpreting these findings requires distinguishing what is learned. Binding a new name, fitting a new operation table, and combining known operations can all improve accuracy, but support different claims about capability acquisition.

Standard math and code benchmarks make these claims difficult to separate. A new problem may require only operations encountered in pretraining, and learning a new symbol assignment need not entail learning a new computation (Wu et al., 2023; Wei et al., 2023). Conversely, failure on a held-out query may reflect poor training fit, difficulty accessing a learned answer, or insufficient information in the examples. Evaluating capability acquisition therefore requires specifying what is new, what the training examples determine, and what the test measures.

We introduce *Alien DSLs*, executable expression languages over sixteen values, each with eight primitives and symbol assignments sampled after pretraining. An exact interpreter supplies rewards, and the finite input domain permits exhaustive evaluation of every held-out program. We vary the transfer target in three tests (Fig. 1). *Binding* assigns new names to familiar operations. *Table learning* changes the operation tables and tests whether an answer learned in one expression remains accessible in another. *Composition* retains familiar operations under new names and changes their combinations. Composition uses familiar operations, not newly generated random tables.

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

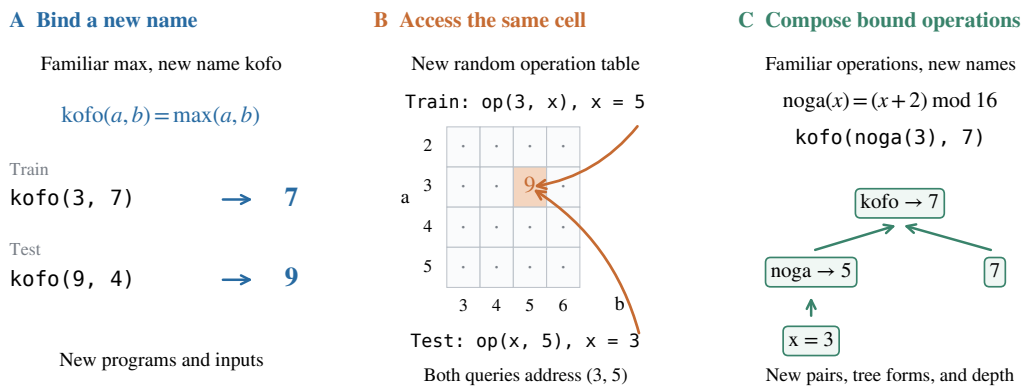

Figure 1: **Three tests with distinct transfer targets.** (A) Binding associates new names with familiar operations. (B) Two expressions address the same cell of a new random table, separating fitting from query access. (C) Composition combines renamed familiar operations in held-out programs; it does not combine newly generated random tables. Values are schematic; B shows only a table submatrix.

The first two tests distinguish learning an operation from accessing its outputs. Outcome-only GRPO learns partial semantic bindings from 0.5B to 7B, exceeding an oracle per-program constant under exhaustive evaluation. A separate audit of held-out predictions also recovers the true operation for a subset of primitives. Random tables are harder to fit with outcome rewards. Dense supervision largely resolves fitting, yet accuracy remains low when the same table cells are queried in another form.

Composition depends on both initialization and the training protocol. At 0.5B and 3B, continued execution-trace supervision outperforms trace-initialized GRPO on held-out programs. At 7B, both trace-initialized GRPO and continued trace supervision exceed the oracle per-program constant on all four held-out splits. Two sequential preregistered waves reproduce the 7B gains on sixteen fresh languages. Transfer also persists when the initialization traces are presented in reverse execution order. Together, these tests separate what training examples identify, which answers remain accessible, and where familiar operations transfer.

2 Alien DSLs

Each random seed defines a language with four unary and four binary primitives. All languages share a grammar whose programs are expression trees over integer literals, an input variable x, and these primitives. For example, `kofo(noga(x), 7)` first applies `noga` to x, then applies `kofo` to the result and 7. All values lie in $\mathbb{Z}_{16}$. An interpreter executes each program exactly. Appendix A specifies the grammar, operation pools, and data splits.

Binding and transformed tables. Three conditions change the representation of the operations selected by each seed. In the *familiar* condition, descriptive names identify their operations. In the *nonsense* condition, new names are assigned to the same operations. In the *conjugated* condition, a shared random permutation π changes the operation tables. For a binary operation f, the new operation is

$$g(a, b) = \pi\big(f(\pi^{-1}(a), \pi^{-1}(b))\big). \tag{1}$$

Conjugation preserves algebraic structure and full-table output frequencies up to relabeling by π. The new tables therefore represent the same algebra, but recovering the hidden permutation requires additional information. The nonsense-to-conjugated comparison changes both novelty and sample complexity.

The depth-1 training set contains 360 examples. Evaluation holds out seven binary programs per primitive and also tests new inputs to trained programs. Each unary primitive has only one depth-1 program, so every unary program appears in training and the held-out-program test is binary-only. A constraint-solver audit finds a unique hypothesis consistent with the training data for every population

language in both the nonsense and conjugated conditions. This identifiability result is relative to the specified generative family.

Random tables and query forms. A second family removes the shared permutation. Each binary operation has independently sampled bijections over its first argument, one for each value of the second argument. The covered split trains all `op(c,x)` programs on all sixteen inputs and tests `op(x,c)`. Every test query therefore addresses a table cell observed during training, expressed in a different form. A partial-coverage split further separates access to observed cells from inference about unobserved cells (§4.3).

Composition. All composition experiments use the *nonsense* condition: new symbol assignments for familiar operations, with no conjugated or random tables. We train on depth-1 and depth-2 programs and evaluate four held-out sets: *seen-pair/template* (E1), *new-pair* (E2), *new-template* (E3), and *depth-3* (D3). E1 targets new programs with training-observed pairs and templates; E2 and E3 withhold a pair or abstract-syntax-tree template, respectively. At 0.5B, we filter E1 to enforce that both witnesses occur in training, excluding 13 of 384 candidates independently of model outputs. The frozen 3B and 7B evaluations instead retain the nominal E1 set; we mark these scores E1† throughout. These scores describe a different evaluation population from the witness-filtered set. These are syntactic holdouts: distinct programs can still compute the same function. Appendix A.5 quantifies that overlap and gives the realized split sizes.

3 Experimental Setup

Models and optimization. We use Qwen2.5-Instruct at 0.5B, 1.5B, 3B, and 7B (Qwen Team, 2024). Outcome-reward training uses group relative policy optimization (GRPO; Shao et al., 2024) with a binary exact-match reward, 16 completions per prompt and sampling temperature 1.0. The distributed configuration determines the effective generation batch. We apply no KL penalty unless specified. Binding and conjugated-table runs use 2,000 updates, except at 3B, which uses 1,500. After 7B pilot runs collapsed, we reduced the learning rate to 10^{-6}. Cross-scale results therefore compare separately tuned protocols. Appendix B reports prompts, optimizer settings, compute, and seed counts.

Supervision controls. The main composition experiment compares three continuations of a shared oracle-trace SFT checkpoint: GRPO, continued oracle-trace SFT, and rejection-imitation on verifier-passing model samples. Every arm uses the full depth-2 training pool. GRPO and continued SFT share the phase-2 learning rate and update count. At 7B, rejection-imitation uses one sampling round. Its sampling code omits the distributed world-size factor used by GRPO, so equal verifier-call budgets are not established. The arms therefore differ in both supervision information and total compute. Pretrained and alternative SFT checkpoints test the contribution of trace initialization. Appendix E reports the earlier experiment with disjoint SFT and RL halves separately.

Evaluation. Our primary evaluation uses greedy decoding and exact match on all sixteen inputs of each held-out program. We call this the *all-input audit*. For each program, the baseline always predicts its most frequent true output, determined from the evaluation labels. This oracle per-program constant is stronger than a global constant, but does not bound all input-dependent heuristics. We report accuracy or *excess*: model accuracy minus baseline accuracy. We label earlier sampled-input results separately because they use a global constant baseline. For the random-table study, we evaluate training fit and held-out transfer with the same free-generation protocol.

Statistical analysis. The language seed is the unit of analysis. We first average results over optimizer seeds within each language, then compute paired differences, confidence intervals, and two-sided t-tests across languages. Most experiments use eight languages; we report exceptions and exclusions with their results. Exploratory tests are unadjusted unless stated. Preregistered experiments specify their hypothesis families in advance and apply Holm correction (Holm, 1979). The two 7B waves each use eight fresh languages and two optimizer seeds per arm. The second wave was registered after the first result was known; pooling all sixteen languages is therefore descriptive. Appendix C details the evidence status of each experiment and the limits of distribution-free inference at this sample size.

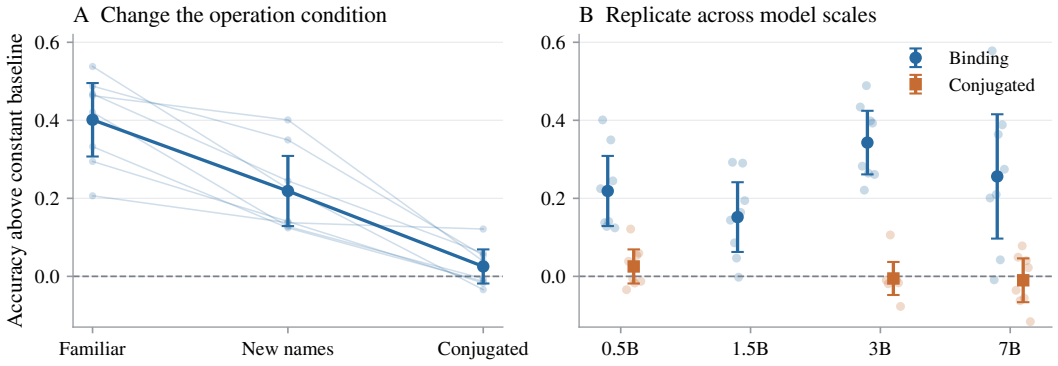

Figure 2: **Binding transfers more reliably than conjugated-table learning.** Left: paired language-level excess at 0.5B across the three conditions. Right: binding and conjugated-table excess across scales. Small points denote languages; large markers and bars show means and 95% confidence intervals of paired excess ($n = 8$). Training protocols differ across scales.

4 Binding and Accessing New Tables

4.1 Rewards learn partial semantic bindings

Can correctness rewards teach a model to associate new names with familiar operations? Models exceed the oracle constant at every tested scale (Fig. 2). At 0.5B, all-input accuracy reaches 0.450 against an oracle constant baseline of 0.231. All 24 trained checkpoints exceed the baseline, and the language-level paired effect is positive on all eight seeds. The result replicates at 1.5B, 3B, and 7B; Appendix D gives the condition-wise accuracies. Accuracy also improves on new inputs to trained programs, ruling out memorization of exact training examples as a complete explanation.

Exceeding the constant baseline does not show that the model interprets each symbol consistently. We fit an operation assignment to the model's predictions on half of each primitive's held-out programs, then evaluate it on the remaining half. On the evaluation half, the decoded assignment agrees with 64% of predictions and recovers the true operation for 43% of primitives, compared with chance $1/11$. These results support partial semantic binding, with substantial residual execution errors. Correctness-independent rewards, including a density-matched shuffled verifier, remove the binding gain (Appendix G).

Additional runs initialized from OLMo-2 (Team OLMo, 2024), Llama-3.1 (Grattafiori et al., 2024), Qwen2.5-Math-7B (Yang et al., 2024), and R1-Distill-Qwen-7B (DeepSeek-AI, 2025) also show binding gains. The pattern varies across model families: SmolLM2 (Allal et al., 2025) fails to bootstrap under the tested protocol, and R1-Distill-1.5B does not exceed the strict baseline. Appendix D reports the complete family comparison and format-based exclusions.

4.2 New-table learning is limited by fitting

Replacing familiar operations with conjugated tables sharply reduces transfer (Fig. 2). At each tested main scale (0.5B, 3B, and 7B), the all-input excess is statistically indistinguishable from zero. None of the additional model-family effects survives multiplicity correction. The uncertainty still allows positive effects: the main-scale confidence intervals extend as high as 0.069 excess.

Three controls examine possible sources of failure. Training examples uniquely identify each conjugated language within the generator family. Oracle-trace supervision on the same examples does not produce a detectable effect at the matched budget, so sparse outcome rewards alone do not explain the difficulty. Providing the exact permutation enables detectable, though inaccurate, execution under some recipes. For example, at 3B the codebook-given SFT model reaches 0.290 against 0.220, whereas its hidden-codebook counterpart reaches 0.253 without a detectable excess. These results show that both acquisition and execution can limit performance; they do not isolate a single causal bottleneck (Appendix G).

Table 1: **Fitting and transfer for random tables.** A common free-generation evaluation measures training accuracy and held-out query accuracy at 0.5B ($n = 8$ languages per row); intervals are 95% CI half-widths across languages. The held-out baseline is 0.0625. SFT fits the table more fully, but neither path transfers most observed cells. Additional budgets and warm starts appear in Appendix I.

Training recipe	Train accuracy	Held-out accuracy
GRPO, 2k updates	0.077 ± 0.008	0.070 ± 0.006
GRPO, 40k updates	0.301 ± 0.023	0.083 ± 0.011
Rejection-SFT, four rounds	0.055 ± 0.007	0.061 ± 0.008
Oracle SFT, 12 epochs	0.075 ± 0.010	0.067 ± 0.007
Oracle SFT, 48 epochs	0.951 ± 0.047	0.123 ± 0.023
Oracle SFT, 48 epochs → GRPO	0.984 ± 0.016	0.126 ± 0.025

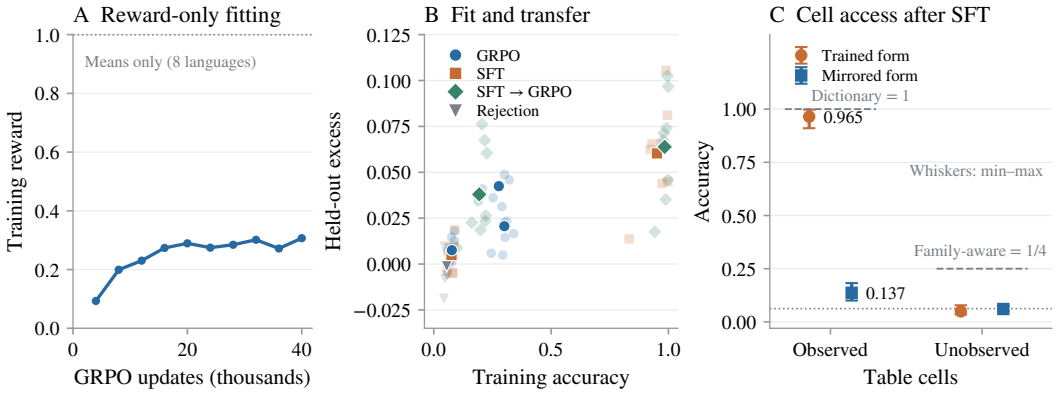

Figure 3: **Fitting and access are distinct bottlenecks.** (A) Mean sampled training reward during 40k-update GRPO runs ($n = 8$); the frozen curve does not retain dispersion. (B) Free-generation fit and held-out excess; faint points denote languages and large markers arm means (two languages at intermediate GRPO checkpoints, eight otherwise). (C) Partial-coverage SFT separates observed/unobserved cells and trained/mirrored forms ($n = 8$). Points show means; whiskers show language minima and maxima, not CIs. The dotted line is uniform guessing ($1/16$); dictionary and family-aware predictions provide reference levels.

The random-table family provides a complementary test without a shared codebook. Every evaluation cell is observed during training. The preregistered reward-only run yields a small positive excess, with strict-audit 95% CI $[0.002, 0.012]$. This is evidence of a small transfer gain, substantially below the binding gain on the same split.

Extending reward-only training to 40,000 updates improves fit but leaves it near 0.30 (Table 1). Dense SFT reaches 0.951 training accuracy. Transfer increases alongside fit across the measured training paths, but remains low even after dense SFT (Fig. 3A–B). Continuing GRPO after dense SFT improves training accuracy to 0.984 while changing held-out accuracy only from 0.123 to 0.126. Other exploration or optimization procedures may yield larger transfer gains.

4.3 High table fit does not ensure access in another form

A covered-split failure can reflect poor fit or difficulty accessing an observed cell. To distinguish these failures, we train a partial-coverage model with dense SFT and classify evaluation queries by their canonical cell $(\texttt{op}, a, b)$. Queries address observed or unobserved cells in the trained or mirrored form. Canonical indexing is necessary because a held-out expression can still refer to a cell seen through another expression.

Accuracy is 0.965 on observed cells in the trained form and 0.137 on the same cells in the mirrored form (Fig. 3C). A dictionary built from the training cells answers both forms perfectly. Thus high accuracy on observed cells does not ensure access through another query form. The asymmetry

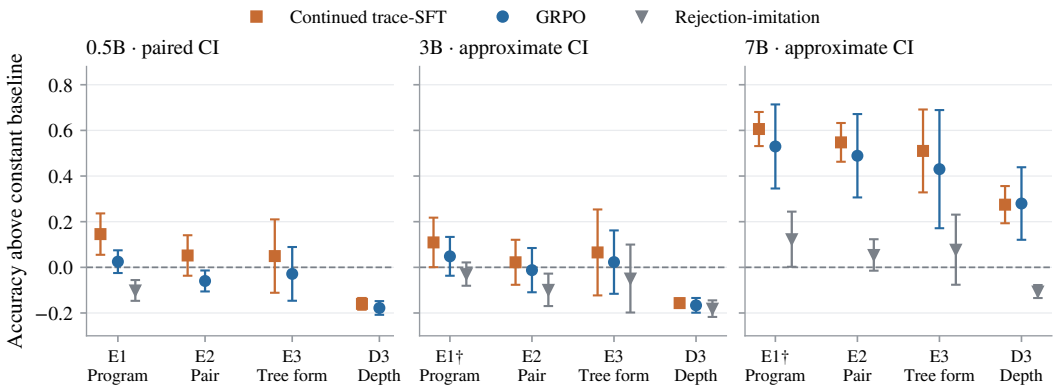

Figure 4: **Composition depends on supervision and scale.** All-input excess over the oracle per-program constant in the original eight-language suite. Bars are 95% intervals across languages, computed from paired effects at 0.5B and reconstructed from rounded paired t-statistics at 3B/7B. E1 is witness-filtered only at 0.5B; E1† retains nominal candidates at larger scales. The 0.5B rejection audit covers E1 only.

resembles the reversal curse (Berglund et al., 2024), although here the required answer is the same operation-table entry rather than an inverse relation.

Unobserved cells remain near uniform guessing. Because each table column is a bijection, the twelve observed entries leave four possible outputs for a missing entry; family-aware prediction can attain expected accuracy $1/4$. The measured accuracies show no benefit from that residual structure. Predicting these cells requires inference under residual uncertainty; it is distinct from retrieving an answer already observed in training.

5 Composing Renamed Familiar Operations

We next test whether models compose familiar operations under new names. Outcome-only training from the pretrained checkpoint does not produce a detectable excess on the held-out composition splits at 0.5B or 3B under the original sampled-input protocol. The 7B from-base control also falls below the stronger all-input baseline. At 0.5B, failure begins on trained depth-2 programs, before any held-out composition is required (Appendix E). We therefore compare training continuations from a checkpoint initialized with execution traces.

5.1 Continued trace supervision is stronger at smaller scales

The main comparison starts GRPO, continued oracle-trace SFT, and rejection-imitation from the same composition-trace checkpoint. At 0.5B, GRPO improves the sampled-input endpoint but does not exceed the all-input E1 baseline. Continued SFT exceeds that baseline and has a paired advantage over GRPO of 0.121 (95% CI $[0.071, 0.171]$; $t(7) = 5.73$). At 3B, its advantage is 0.061 on nominal E1† ($t(7) = 3.61$). The paired contrasts directly test the SFT advantage over GRPO.

Rejection-imitation does not yield a detectable E1 excess at either smaller scale (Fig. 4). Only continued oracle-trace SFT receives new correct intermediate steps during phase 2. The advantage therefore applies to the full supervision recipe at the tested update count; it cannot be attributed to the optimization objective alone.

5.2 Both trace-initialized recipes transfer at 7B

At 7B, both GRPO and continued trace-SFT exceed the baseline on all four evaluated sets, including depth-3 programs (Table 2). The initial trace-SFT checkpoint has no detectable positive excess. Both continuations improve on it in every language, but their final difference is imprecisely estimated. We

Table 2: **7B composition on original and fresh languages.** Mean all-input excess; E1† is the nominal, unfiltered program split. The final two rows pool two sequential waves ($n = 16$ languages; two optimizer seeds per arm). Their 95% CI half-widths summarize language variation descriptively. Per-wave results and initialization controls appear in Appendix I.

Training condition	E1†	E2	E3	D3
Original suite ($n = 8$)				
From-base GRPO	-0.174	-0.175	-0.182	-0.211
Trace-SFT initialization	+0.027	-0.045	-0.005	-0.157
$\rightarrow$ GRPO	+0.530	+0.489	+0.430	+0.280
$\rightarrow$ continued trace-SFT	+0.606	+0.548	+0.510	+0.275
$\rightarrow$ rejection-imitation	+0.123	+0.054	+0.077	-0.106
Sequential fresh waves, descriptive pooled summary ($n = 16$)				
Trace-SFT $\rightarrow$ GRPO	0.546 ± 0.091	0.490 ± 0.093	0.542 ± 0.098	0.251 ± 0.069
Trace-SFT $\rightarrow$ continued SFT	0.615 ± 0.049	0.580 ± 0.043	0.553 ± 0.089	0.275 ± 0.050

therefore do not infer a general SFT advantage at this scale. Rejection-imitation trails both recipes, but its smaller documented sampling budget prevents a compute-matched interpretation.

The first fresh-language wave freezes the successful 7B protocol and confirms both recipes on eight new languages. A second wave, registered after the first result, adds eight different languages under the same protocol. Both waves show positive mean excess on each split for each recipe. In the pooled summary, seven effects are positive on all sixteen languages; GRPO's depth-3 effect is positive on fifteen. The pooled Holm-adjusted paired-test p-values are at most 1.2×10^{-6}, but are descriptive because the extension followed inspection of the first wave. The replication supports the chosen 7B recipes; it does not isolate model size from per-scale protocol differences.

5.3 Trace order and retention qualify the result

Alternative initializations test which trace properties matter. Answer-only initializations yield no statistically detectable positive mean excess. Atomic traces produce transfer only in individual languages. An exploratory 7B arm instead reverses the order of the composition trace lines before initialization, then applies the same GRPO phase. It still transfers across all four sets, with mean depth-3 excess 0.246 across eight languages. Thus innermost-first ordering is not required under this recipe. This result does not establish equivalence between orders or identify step content as the sole cause (Appendix I).

The recipes also differ in benchmark retention. Across eight 7B checkpoints per arm, continued trace-SFT reduces GSM8K accuracy by 0.248, compared with 0.052 for GRPO. The paired retention advantage of GRPO is 0.195 ($t(7) = 3.69$), measured on a shared set of 150 questions. This is evidence of a recipe-dependent trade-off on that probe, rather than a complete measure of retained general capability (Appendix J).

6 Discussion and Limitations

High training accuracy can coexist with poor access to observed answers, whereas familiar operations can transfer to new compositions. The framework separates these outcomes through tests of binding, table fit and access, and composition. Neither finding alone establishes acquisition of a new computational primitive.

Scope of the comparisons. Paired languages and controlled queries strengthen comparisons within each test. Across tests, the operation families differ in both familiarity and information content. Across scales and model families, the training protocols were tuned separately. Oracle traces provide intermediate steps unavailable from outcome rewards, and the initialization controls do not fully separate trace content from format. The observed differences therefore characterize the tested protocols; they do not identify model size or the optimization objective as the sole cause. Likewise, limited random-table fit after 40k updates does not establish an intrinsic limit on RL.

Scope of generalization. Our domain contains sixteen values, a fixed grammar, and programs of depth at most three. Most exploratory analyses reuse eight language seeds. Fresh-suite confirmation supports only its preregistered contrasts. Exceeding the oracle constant rules out that baseline, but leaves input-dependent shortcuts possible. Moreover, 29.6–53.4% of nonconstant held-out programs compute a function represented in training (Appendix A.5). The composition results thus establish transfer to new expressions, not exclusively to new functions. The semantic-assignment audit provides behavioral evidence without identifying the model's internal algorithm. Finally, the small output space makes raw pass@k particularly permissive: uniform guessing already achieves $1 - (15/16)^k$. An increase in this metric alone cannot establish capability expansion.

7 Related Work

Capability expansion under RLVR. Studies using large-k evaluations disagree on whether RLVR expands a pretrained model's capabilities. Yue et al. (2025) find substantial overlap with base-model solutions, whereas prolonged training and synthetic maze experiments report success beyond the base model's measured performance (Liu et al., 2025; Shen et al., 2025). Wen et al. (2025) also evaluate intermediate reasoning through CoT-Pass@k; Cai et al. (2026) combine teacher guidance with curriculum RL. Studies of empirical support, spurious rewards, and contamination further qualify the interpretation of accuracy gains (Wu et al., 2025a; Shao et al., 2025; Wu et al., 2026). We separate three transfer targets using task instances generated after pretraining: symbol binding, table access, and composition of renamed familiar operations.

Synthetic tasks and compositional transfer. Procedural environments supply verifiable problems for RL training (Stojanovski et al., 2025; Lacombe et al., 2025; Xie et al., 2025; Zhao et al., 2025a). Yuan et al. (2026) compare RL with imitation of verified model-generated traces on string-function composition. Cheng et al. (2025) also compare further SFT, LoRA, and RL on the same composition samples, with templated reasoning traces for SFT. Kong et al. (2026) study reusable modules and routing interfaces learned from compound reasoning traces. We evaluate binding, random-table access, and compositional execution within one experimental framework. The composition comparison adds scale and initialization controls, while distinguishing continued oracle-trace supervision from imitation of verifier-passing samples. Appendix H compares these tasks and supervision regimes.

Execution and access. Program-execution models (Zaremba & Sutskever, 2014; Reed & de Freitas, 2016) and compositional benchmarks (Lake & Baroni, 2018; Keysers et al., 2020; Kim & Linzen, 2020; Hupkes et al., 2020) distinguish fitting examples from systematic transfer. Counterfactual tasks and symbol tuning probe dependence on familiar semantics (Wu et al., 2023; Wei et al., 2023); execution analyses examine limits of multi-step composition (Dziri et al., 2023). Our same-cell tests hold the required answer fixed while changing its query form. This design connects knowledge storage and extraction (Allen-Zhu & Li, 2024) to the directional asymmetry studied by Berglund et al. (2024), without assuming a shared internal mechanism. Our longer runs test whether the fitting gap persists with more training. Reports of delayed generalization caution against interpreting a finite-budget failure as impossibility (Power et al., 2022).

8 Conclusion

Alien DSLs separate three questions about post-training: learning symbol meanings, fitting and accessing new tables, and composing renamed familiar operations. Models learn partial bindings from outcome rewards. Dense supervision largely resolves table fitting, but leaves a large gap in access across query forms. At 7B, both trace-initialized GRPO and continued trace supervision transfer across the tested composition splits, with a larger GSM8K retention loss under continued supervision. Claims about capability acquisition should therefore distinguish what a model fits, how it accesses learned answers, and where it transfers.

AI USE STATEMENT

Generative AI tools assisted with research code, analysis of experimental results, and manuscript preparation. Their use included statistical and source-consistency checks, literature and citation checks, drafting and editing, figure and table preparation, and simulated reviewer feedback. AI assistance was limited to coding, analysis, and writing; it was not used to originate the research ideas or design the experiments. The authors are responsible for the paper's claims, code, and reported results.

ETHICS STATEMENT

This work studies capability acquisition in small open-weight models on a fully synthetic formal-language domain; no human subjects, personal data, or deployment-facing systems are involved. Retention probes (App. J) measure changes on selected benchmarks after training. We see no direct ethical risks beyond the general dual-use considerations of improving RLVR understanding.

REPRODUCIBILITY STATEMENT

The accompanying source contains the language generator, training and evaluation code, preregistrations, and frozen aggregate statistics. The figure renderer and performance tables use these frozen statistics; a separate generator audit reproduces split sizes and function overlap. Full recomputation additionally requires the underlying per-run artifacts, which are not included in the current source package. Appendix B specifies prompts, parsing, decoding, and training settings, and Appendix I records the sequential confirmation protocols. The pooled sixteen-language summary is descriptive, and its frozen record does not contain the underlying language-level vectors.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, et al. SmolLM2: When smol goes big — data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 1067–1077, 2024. URL https://proceedings.mlr.press/v235/allen-zhu24a.html.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "A is B" fail to learn "B is A". In *International Conference on Learning Representations (ICLR)*, 2024.

Pengxiang Cai, Tianchen Fang, Xiaohan Li, Qingyuan Zeng, Guocong Li, and Jintai Chen. Curriculum reinforcement learning can incentivize reasoning capacity in LLMs beyond the base model. *arXiv preprint arXiv:2606.22317*, 2026.

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yulin Chen, He He, and Chen Zhao. The unlearnability phenomenon in RLVR for language models. *arXiv preprint arXiv:2605.16787*, 2026.

Sitao Cheng, Xunjian Yin, Ruiwen Zhou, Yuxuan Li, Xinyi Wang, Liangming Pan, William Yang Wang, and Victor Zhong. Atomic skills are the prerequisite: When reinforcement learning synthesizes compositional reasoning, and when it only amplifies. *arXiv preprint arXiv:2512.01970v3*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/chu25c.html.

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

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015. doi: 10.1126/science.aaa9375.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning LLMs on new knowledge encourage hallucinations? In *Proceedings of EMNLP*, 2024. doi: 10.18653/v1/2024.emnlp-main.444.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Xiao Hu, Xingyu Lu, Liyuan Mao, YiFan Zhang, Tianke Zhang, Bin Wen, Fan Yang, Tingting Gao, and Guorui Zhou. Why distillation can outperform Zero-RL: The role of flexible reasoning. *arXiv preprint arXiv:2505.21067*, 2025.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020. doi: 10.1613/jair.1.11674.

Daniel Keysers, Nathanael Schärli, Nathan Scales, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*, 2020.

Minwu Kim, Anubhav Shrestha, Safal Shrestha, Aadim Nepal, and Keith Ross. Reinforcement learning vs. distillation: Understanding accuracy and capability in LLM reasoning. *arXiv preprint arXiv:2505.14216*, 2025.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of EMNLP*, 2020. doi: 10.18653/v1/2020.emnlp-main.731.

Lingjing Kong, Xin Liu, Guangyi Chen, Martin Q. Ma, Xiangchen Song, Yuekai Sun, Mikhail Yurochkin, Taylor W. Killian, Ruslan Salakhutdinov, Kun Zhang, Eric P. Xing, and Zhengzhong Liu. From reasoning traces to reusable modules: Understanding compositional generalization in language model reasoning. *arXiv preprint arXiv:2606.18089*, 2026.

Valentin Lacombe, Valentin Quesnel, and Damien Sileo. Reasoning core: A scalable RL environment for LLM symbolic reasoning. *arXiv preprint arXiv:2509.18083*, 2025.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. RL squeezes, SFT expands: A comparative study of reasoning LLMs. *arXiv preprint arXiv:2509.21128*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Core Francisco Park, Zechen Zhang, and Hidenori Tanaka. New news: System-2 fine-tuning for robust integration of new knowledge. *arXiv preprint arXiv:2505.01812*, 2025.

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.

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

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20*, 2020.

Scott Reed and Nando de Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*, 2016.

Qihan Ren, Peng Wang, Ruikun Cai, Shuai Shao, Dadi Guo, Yuejin Xie, Yafu Li, Quanshi Zhang, Xia Hu, Jing Shao, and Dongrui Liu. Rethinking generalization in reasoning SFT: A conditional analysis on optimization, data, and model capability. *arXiv preprint arXiv:2604.06628*, 2026.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, et al. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Minghe Shen, Zhuo Zhi, Chonghan Liu, Shuo Xing, Zhengzhong Tu, and Che Liu. Does RLVR extend reasoning boundaries? Investigating capability expansion in vision-language models. *arXiv preprint arXiv:2511.00710*, 2025.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025.

Team OLMo. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Jerry Wei, Le Hou, Andrew Lampinen, Xiangning Chen, Da Huang, Yi Tay, Xinyun Chen, Yifeng Lu, Denny Zhou, Tengyu Ma, and Quoc V. Le. Symbol tuning improves in-context learning in language models. *arXiv preprint arXiv:2305.08298*, 2023.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. *arXiv preprint arXiv:2506.14245*, 2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why RLVR may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025a.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, et al. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. In *AAAI Conference on Artificial Intelligence*, 2026. doi: 10.1609/aaai.v40i40.40687.

Yongliang Wu et al. On the generalization of SFT: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*, 2025b.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.

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

An Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Lifan Yuan, Weize Chen, Yuchen Zhang, Ganqu Cui, Hanbin Wang, Ziming You, Ning Ding, Zhiyuan Liu, Maosong Sun, and Hao Peng. From f(x) and g(x) to f(g(x)): LLMs learn new skills in RL by composing old ones. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/eeae43a68515325cad64c0f54b2d0c70-Abstract-Conference.html.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

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

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

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: RL post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025b.

A Language Construction and Data Splits

A.1 Grammar and operation pools

Programs are expression trees over $\mathbb{Z}_{16}$. Leaves are literals in $0, \ldots, 15$ or the input variable x, and every generated program contains x. A language contains four unary and four binary primitives. Names are sampled consonant–vowel strings filtered against English words.

The unary pool contains modular addition by $k \in \{1, \ldots, 15\}$; modular multiplication by $k \in \{3, 5, 7, 9, 11, 13\}$; modular negation; complement $15 - a$; the two-bit-half swap $4(a \bmod 4) + \lfloor a/4 \rfloor$; squaring modulo 16; and $\lfloor a/2 \rfloor$. The binary pool contains modular addition, subtraction, and multiplication; bitwise AND, OR, and XOR; maximum and minimum; absolute difference; $\lfloor (a + b)/2 \rfloor$; and $(a + b + 1) \bmod 16$. Conjugation applies a single sampled permutation to all primitives. For unary f, $g(a) = \pi(f(\pi^{-1}(a)))$; for binary f, $g(a, b) = \pi(f(\pi^{-1}(a), \pi^{-1}(b)))$.

A.2 Depth-1 and composition splits

The depth-1 universe contains four unary programs and 132 binary programs of the forms $p(x, c)$, $p(c, x)$, and $p(x, x)$. All unary programs train. Seven binary programs per primitive are held out, giving 28 held-out programs (set B). Training uses twelve inputs per unary program and three per binary program, for 360 examples. Set C tests trained programs on unused inputs.

Mixed-depth training adds 256 depth-2 programs with two inputs each, for 872 examples in total. E1 tests new programs with observed primitive pairs and templates; E2 holds out an ordered primitive pair; E3 holds out an abstract-syntax-tree template; and D3 tests depth-3 programs. H tests unused inputs to depth-2 programs assigned to the SFT half in the original disjoint-half experiment. Those programs are unseen by the no-trace RL arm of that experiment.

The original sampled-input evaluation contains 28 B examples, 96 C examples, 96 examples each in E1/E2 before witness filtering, up to 96 E3 examples, 48 D3 examples, and 128 H examples per language. E3 uses up to 48 programs from the eligible template pool, which can be smaller; each program contributes two sampled inputs. Table 3 gives witness-filtered generator counts, which need not equal the nominal larger-scale evaluation counts. The all-input audit expands every evaluated program to all sixteen inputs. B therefore contains 448 audited examples. The generator's split and interpreter checks comprise 31 unit tests.

A.3 Family-relative identifiability

A hypothesis consists of injective assignments of four unary names to the 26-operation unary pool and four binary names to the 11-operation binary pool, together with a permutation π for the conjugated condition. For nonsense languages, π is the identity. A constraint solver enumerates the hypotheses consistent with the 360 training rows. With $q = \pi^{-1}$, each example imposes $f_{\sigma(p)}(q(\text{args})) = q(y)$.

The solver uses arc consistency and all-different pruning over the sixteen domains of q, then backtracks over the remaining assignments. It discards only assignments that violate a constraint. For all eight population languages in both conditions, it finds exactly one solution, equal to the generating hypothesis. Thus the training data determine all 448 held-out answers within this family (statistics key `identifiability`). This conclusion assumes the generator family is known. A learner without that prior faces a larger hypothesis space; the informed-prior controls explicitly describe the construction.

The hidden permutation carries $\log_2(16!) \approx 44.3$ bits. The name assignment carries $\log_2(26 \cdot 25 \cdot 24 \cdot 23) + \log_2(11 \cdot 10 \cdot 9 \cdot 8) \approx 31.4$ bits. These counts describe latent information, not equal learning difficulty.

A.4 Witness filtering

E1 is sampled before the depth-2 training programs. Its nominal construction therefore does not guarantee that every pair and template appears in the realized training set. In the original suite, 13 of 384 candidate programs violate at least one condition: three pair violations and twelve template violations, with overlap. The 0.5B item-level aggregation removes these programs using only the

training data and recomputes the baseline. The 3B, 7B, and fresh-wave aggregates instead use the unfiltered nominal E1 set (E1†). Their frozen summaries cannot recover witness-filtered scores. A generator audit finds 9 of 384 unwitnessed candidates in the first fresh suite; it supplies no corrected model scores. All larger-scale E1 values below refer to the nominal set, including tables where the dagger marks this distinction. The remaining splits are unaffected by this E1 reporting issue.

Splitting training between SFT and RL reduces witness coverage: 49 and 57 candidate E1 programs, respectively, lack witnesses. On the subset witnessed in both halves (31–42 programs per language), trace-SFT→RL reaches 0.238 against a constant baseline of 0.139. Its paired advantages are 0.085 over SFT-only ($t(7) = 5.61$) and 0.098 over no-trace RL ($t(7) = 3.84$). This robustness check applies to the original sampled-input experiment, not the stronger all-input criterion used for the main composition claims.

A.5 Program novelty and functional overlap

The composition splits withhold program strings, ordered primitive pairs, or templates. They do not remove all equivalent input–output functions. To quantify this distinction, we regenerate the original and fresh suites with the full-pool protocol (256 depth-2 training programs). For each program p, we compute its function signature $(p(0), \ldots, p(15))$ and compare it with the signatures of all depth-1 and depth-2 training programs. E1 is restricted by the same separate pair/template witness rule. Tables 3 and 4 report this generator-only audit; it does not use model predictions or change any evaluation score.

Table 3: **Realized composition-program counts.** E1 counts follow witness filtering; E3 counts depend on the eligible template pool. E2 and D3 each contain 48 programs per language. Sampled E3 evaluation uses two inputs per program; the all-input audit uses sixteen. Counts are regenerated from the current source, not recovered from archived evaluation files.

Original suite			Fresh suite		
Language	E1	E3	Language	E1	E3
1111	46	48	1212	48	48
2222	44	24	2323	48	48
3333	47	22	3434	46	13
4444	46	22	4545	45	13
5555	48	12	5656	48	40
6666	45	48	6767	46	13
7777	48	34	7878	48	38
8888	47	15	8989	46	20

Table 4: **Function overlap with training programs.** Numerators count held-out programs with a matching sixteen-input signature; denominators count all programs in the indicated subset, pooled across eight languages. The nonconstant subset removes functions already solved by a per-program constant. These are program counts, not model accuracies or language-level effect estimates.

Suite	Subset	E1	E2	E3	D3
Original	All programs	204/371	170/384	128/225	133/384
Original	Nonconstant	179/343	152/358	101/189	103/348
Fresh	All programs	197/375	183/384	128/233	138/384
Fresh	Nonconstant	175/349	164/356	115/219	115/355

Nonconstant overlap ranges from 29.6% to 53.4% across suite–split combinations. Overlap does not imply that every answer was observed: training uses only three inputs per binary depth-1 program and two per depth-2 program. Recognizing a familiar function in a new expression may itself require transfer. These splits nevertheless do not establish generalization exclusively to new functions. Separating performance on overlapping and disjoint functions requires item-level predictions, which are absent from this snapshot. The supplied `audit_splits.py` and `split_audit.json` record the generator audit's settings, source hashes, and per-language counts. This audit changes neither training data nor performance statistics.

B Training Protocols and Compute

B.1 Optimization and replication

GRPO uses 16 completions per prompt, temperature 1.0, top-p 1.0, ten warmup steps, and a constant learning rate. The default completion cap is 160 tokens, or 256 with a brevity instruction at 1.5B and above. Unless stated otherwise, $\beta = 0$ and no reference model is used. Training uses TRL 0.19.1 (von Werra et al., 2020) and Transformers 4.53.2 (Wolf et al., 2020). We use full bf16 fine-tuning, AdamW (Loshchilov & Hutter, 2019) with weight decay set to zero, clipping $\epsilon = 0.2$, normalized group-relative advantages, and gradient-norm clipping at 1.0. Models up to 3B use one H100-80GB; 7B runs use eight H100s with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020).

The original trace-SFT phase uses learning rate 10^{-5}, a cosine schedule, and three epochs over 616 examples, giving 60 optimizer updates at batch size 32. In the full-pool factorial, phase-2 continued SFT and GRPO use 1,500 matched updates and the same learning rate. The 3B rejection comparison matches 96k verifier calls. At 7B, rejection-imitation uses one round rather than three because retained distributed-engine buffers caused out-of-memory failures. A round-count control at 0.5B and 3B changes E1 by only 0.003 and 0.006.

Table 5: **Main experiment families.** Counts are languages × optimizer seeds per condition. The original suite is 1111–8888; fresh suites and their exact seeds are specified in the preregistrations.

Experiment	Scale	Replication	Training examples
Binding	0.5B / 1.5B / 3B / 7B	$8 \times \{3, 2, 2, 1\}$	360
Conjugated tables	0.5B / 3B / 7B	$8 \times \{3, 2, 1\}$	360
Familiar control	0.5B	8×2	360
Mixed-depth reward-only	0.5B / 3B	$8 \times \{3, 2\}$	872
Disjoint-half trace arms	0.5B	8×2	d1 + d2 half
Oracle / informed-prior controls	0.5B	8×2	360
Exact-codebook controls	0.5B / 3B	8×2	360
Random covered split	0.5B	8×2	1,088
Full-pool factorial	0.5B	8×2	full d1 + d2
Fresh disjoint-half replication	0.5B	8×2	d1 + d2 half
Fresh 7B waves (each)	7B	8×2	full d1 + d2

B.2 Prompts, traces, and parsing

The system message is: "You are an expert at reasoning about unfamiliar formal languages." The standard user message specifies the integer domain, a program, and its input, followed by the instruction to reason briefly and finish with `Answer: <integer>`. For example:

```
The program below is written in a formal language whose
operations you may not know.  All values are integers 0-15.
'x' is the input.
Program:  jivo(poko(x, 3), 12)
Input:  x = 9
Determine the integer value (0-15) the program evaluates to.
Think step by step, briefly.  Then give your final answer on
the last line in exactly this format:
Answer:  <integer>
```

Oracle traces contain one line per primitive application in innermost-first order, followed by the answer. An illustrative trace for this program is:

```
poko(9, 3) = 6
jivo(6, 12) = 12
Answer:  12
```

Manual-bearing prompts prepend each primitive's base operation. The informed-prior version additionally states that all operations share one unknown permutation and explains conjugation. The exact-codebook version prints both the permutation and its inverse as sixteen-entry tables. The generator and prompt construction code specify the exact strings for each variant.

The parser reads only the last non-empty line. It accepts `Answer: <integer>` or a boxed integer on that line; otherwise the completion is incorrect. Training samples at temperature 1.0. Greedy and all-input evaluations use temperature zero. The pass@8 and mean@8 probes use eight samples at temperature 1.0.

B.3 Compute accounting

The documented 7B configuration uses eight processes, per-device batch size 2, and gradient accumulation 8. Under TRL 0.19.1's default generation-batch calculation, this gives $8 \times 2 \times 8 = 128$ completions, or eight distinct prompts, per optimizer update. At 1,500 updates the nominal total is 192k completions. In contrast, the rejection branch generates only on rank zero and computes its total as $1500 \times 2 \times 8 = 24$k, omitting world size. Thus its documented sampling budget is eight times smaller. Runtime overrides and actual invocation records are unavailable, so these are source-implied counts, not verified execution totals. Claims of matched 7B verifier calls are unsupported; the rejection comparison is a recipe comparison at reported update counts.

The following table records the original experiment portfolio. Node-hours are estimates from wall-clock durations, not normalized training FLOPs. Later controls and extensions are described with their budgets in the corresponding sections. Full per-run reaggregation requires the raw artifacts, which are not bundled with the current source snapshot.

Table 6: **Original compute portfolio.** Cells and budgets follow the frozen run accounting; node-hours are wall-clock estimates. "Seeds" denotes optimizer seeds per language. Semicolons separate recipes within a row. Later extensions are described in Appendix I.

Condition	Scale	Cells	Seeds	Budget	Node-h
Binding	0.5B	24	3	2k	≈ 1
Binding	1.5B	16	2	2k	≈ 5
Binding	3B	16	2	1.5k	≈ 6
Binding	7B	8	1	2k	≈ 23
Familiar	0.5B	16	2	2k	≈ 0.7
Conjugated	0.5B	24	3	2k	≈ 1
Conjugated	3B / 7B	16 / 8	2 / 1	1.5k / 2k	$\approx 6/14$
Mixed-depth	0.5B / 3B	24 / 16	3 / 2	2k / 1.5k	$\approx 1.5/6$
Disjoint-half RL arms	0.5B	16; 16	2	1.5k	≈ 1.6
Prolonged conjugated	0.5B	16	1	8k	≈ 3
Meta train; adapt	0.5B	2; 16	2; 1	4k; 0.5k	≈ 0.7
Synthesis	0.5B	8	1	1.5k / 3k	≈ 1.2
Synthesis, manual	0.5B	2	1	1.5k	≈ 0.2
RSFT	0.5B	3	1	128k samples	≈ 0.5
Oracle SFT	0.5B	64	2	3 / 12 ep	≈ 0.7
Informed, RL; SFT	0.5B	16; 16	2	2k; 12 ep	≈ 1.3
Codebook, RL; SFT	0.5B	16; 16	2	2k; 12 ep	≈ 1.9
Codebook, SFT; RL	3B	16; 16	2	12 ep; 1.5k	≈ 30.5

Budgets are optimizer updates unless marked as epochs (ep) or samples. The single-process reference uses 64 completions per update; the documented 7B configuration implies 128 (Appendix B). Two synthesis cells run for 3k updates. The 3B codebook campaign additionally contains eight walltime-limited cells excluded from aggregate results; its compute estimate includes those runs.

C Statistical Analysis and Evidence Scope

Optimizer seeds are averaged within each language. Tests compare per-language differences with zero using a two-sided paired t-test, with $df = n - 1$. For eight languages, the 95% interval is $\bar{d} \pm 2.365\, s_d/\sqrt{8}$. Intervals on excess use paired model-minus-baseline differences, not separate uncertainty estimates for model and baseline. Cross-family comparisons use the number of languages remaining after the reported exclusions.

Unless a correction is specified, exploratory tests are unadjusted. The six cross-family conjugated tests form a post-hoc Holm family; none survives, with the smallest corrected p approximately 0.19. The

Table 7: **Evidence status of the main findings.** Preregistration fixes the stated experiment and analysis, not later exploratory extensions.

Finding	Status	Scope and selection
Binding across scales and families	Exploratory replication	Per-scale/family tuning; failed starts and format exclusions reported.
Small random-table excess	Preregistered	Fresh eight-language suite; primary Holm family of three tests.
Continued SFT vs. GRPO at 0.5B	Preregistered factorial	Sampled-input endpoint preregistered; all-input audit added later.
Random-table fit and access	Exploratory	Budgets and diagnostics extended after inspecting earlier results.
7B composition	Exploratory, then confirmed	Fresh-suite confirmation freezes both trace-initialized recipes; eight Holm-corrected tests.
7B initialization controls	Exploratory	Original suite only; no fresh-suite test of trace necessity.

original 7B composition family contains two trace-initialized recipes and four splits, with maximum corrected $p = 0.0085$. Its first fresh-suite confirmation has maximum corrected $p = 0.000321$. The pooled two-wave summary uses $n = 16$, $df = 15$, and CI multiplier $2.13145/\sqrt{16}$; it is descriptive because the second wave was added after inspecting the first.

For effects with eight aligned signs, the exact two-sided sign test gives $2/2^8 = 0.0078125$. This provides a distribution-free check for each effect, but cannot reject all eight confirmatory tests jointly after Holm correction: the minimum corrected value is 0.0625. The joint confirmatory statement therefore uses the preregistered paired t analysis. The per-effect sign check and the family-wise paired t-tests therefore support different statements.

Most exploratory analyses reuse the original eight languages, creating adaptive-reuse risk (Dwork et al., 2015). Fresh seed suites confirm specified findings but do not eliminate that risk for every comparison. Frozen statistics retain per-language values for many analyses, with the pooled wave 42/43 exception described below. The figure renderer reads this file without modifying it. Rendering the figures does not reproduce the underlying evaluations.

D Binding Audits and Cross-Family Results

D.1 Shortcut baselines and semantic consistency

Table 8: **Measured outcome-only conditions.** All-input held-out binary accuracy and oracle per-program constant, averaged over eight languages. Figure 2 shows paired excess and its uncertainty. Each row is a measured condition; model-scale protocols differ.

Operation condition	Scale	Accuracy	Constant
Familiar names	0.5B	0.638	0.237
New names (binding)	0.5B	0.450	0.231
New names (binding)	1.5B	0.383	0.231
New names (binding)	3B	0.574	0.231
New names (binding)	7B	0.487	0.231
Conjugated tables	0.5B	0.246	0.220
Conjugated tables	3B	0.215	0.220
Conjugated tables	7B	0.210	0.220

At 0.5B, all-input binding accuracy is 0.450. The comparator accuracies are 0.231 for the oracle per-program constant, 0.229 for copying x, 0.184 for a per-primitive constant, 0.124 for a global constant, and 0.069 for a training-marginal sampler. All 24 checkpoints exceed the per-program constant, with language-level $t(7) = 5.76$.

A revealed-semantics decoder selects the pool operation that best explains each primitive's held-out predictions. Fitting and evaluating on the same predictions gives agreement 0.66 and true-operation recovery 0.42. Splitting each primitive's programs into fitting and evaluation halves gives agreement 0.64 and recovery 0.43, with 95% CI $[0.33, 0.53]$ for recovery across languages. Chance recovery is $1/11$ because the evaluated primitives are binary.

An oracle affine comparator, $(\alpha x + \beta) \bmod 16$ fitted separately to each program's true outputs, reaches 0.67. Thus the constant baseline is not an upper bound on input-dependent heuristics. The decoded assignment provides additional evidence of partial semantic learning, rather than treating one accuracy threshold as conclusive.

Base sampled reward density (mean@8 on held-out B) is 0.0672 for familiar names, 0.0115 for nonsense names, and 0.0102 for conjugated tables. The latter two are close, but reward-density similarity alone does not control every aspect of training difficulty.

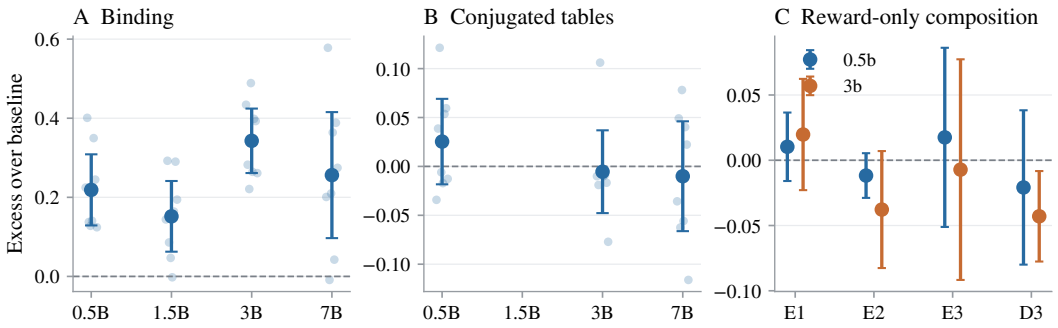

Figure 5: **Scale replication and reward-only composition.** Binding and conjugated panels use all-input excess over per-program constants. The composition panel uses the original sampled-input protocol and its global constant. The composition values therefore use a different evaluation protocol. Bars are paired 95% intervals across eight languages.

D.2 Additional model starts

Table 9 summarizes the additional all-input audits. OLMo-2-1B also gives random-covered accuracy 0.064 against 0.0625, with excess CI $[-0.002, 0.006]$. Learning rates and token caps in the cross-family comparisons were selected through preliminary probes. They test whether a pattern recurs under an adapted recipe, rather than whether model family alone causes the difference.

Table 9: **Additional model starts.** All-input binding accuracy and conjugated excess over each arm's per-program baseline. Sample sizes differ where format/bootstrap exclusions apply. None of the conjugated tests survives the reported cross-family Holm correction.

Model	Binding	n	Conjugated excess	n
OLMo-2-1B	0.538	8	+0.028	8
OLMo-2-7B	0.582	8	+0.053	8
OLMo-2-13B	0.601	8	+0.042	8
Llama-3.1-8B	0.460	6	−0.023	7
Qwen2.5-Math-7B	0.632	6	−0.007	5
R1-Distill-Qwen-7B	0.684	8	+0.042	8
R1-Distill-Qwen-1.5B	0.204	8	−0.050	8

For OLMo-2 (Team OLMo, 2024), we use the brief prompt with a 256-token cap and learning rates of 3×10^{-6} at 1B and 10^{-6} at 7B/13B. We use the latter settings for Llama-3.1-8B (Grattafiori et al., 2024). Three of the sixteen Llama runs are excluded after zero-reward collapse; another run completes after resuming from a checkpoint. Qwen2.5-Math (Yang et al., 2024) fails to bootstrap in nine of sixteen brief-protocol runs because outputs exceed the cap. With a 1,024-token cap and no brevity instruction, eleven cells complete the bootstrap, yielding the sample sizes in Table 9.

R1-Distill-Qwen (DeepSeek-AI, 2025) also requires the 1,024-token protocol because its initial reasoning blocks are truncated at 256 tokens. At 7B, mean completion length decreases from approximately 990 to 130 tokens in the first 200 updates, followed by binding gains. The 1.5B start improves training reward but remains below the strict binding baseline. SmolLM2-1.7B-Instruct (Allal et al., 2025) fails to bootstrap binding under both probed learning rates, even at 4,000 updates: sampled-input B accuracy is 0.183 against 0.165, $t(7) = 0.59$. Its lower initial parse rate is a possible contributor, not an established explanation.

E Composition Controls and Execution Audits

E.1 Original disjoint-half experiment

The original experiment splits trained depth-2 programs between SFT and RL. It compares trace-SFT only, no-trace RL, and their sequential combination. Because the combined arm sees both halves, the experiment changes both supervision and program coverage. The full-pool comparison in the main text holds program coverage fixed.

Table 10: **Disjoint-half composition at 0.5B.** Sampled-input accuracy, with two optimizer seeds averaged within each of eight languages. H is trained-program/new-input evaluation for the trace arms but unseen-program evaluation for no-trace RL. Statistical comparisons use each set's constant baseline.

Training arm	B	H	E1	E2	D3
Trace-SFT only	0.460	0.167	0.150	0.104	0.066
No-trace RL	0.460	0.163	0.143	0.151	0.104
Trace-SFT → RL	0.607	0.239	0.225	0.173	0.112

Only the combined arm exceeds the E1 constant baseline in this experiment, with $t(7) = 3.32$. Its paired advantages over SFT-only and no-trace RL are 0.074 ($t = 4.28$) and 0.081 ($t = 4.40$). The no-trace arm nevertheless exceeds the H baseline (0.163 vs. 0.121, $t = 3.05$). Reward-only training therefore transfers to some unseen programs, even though it does not pass the witness-controlled E1 test.

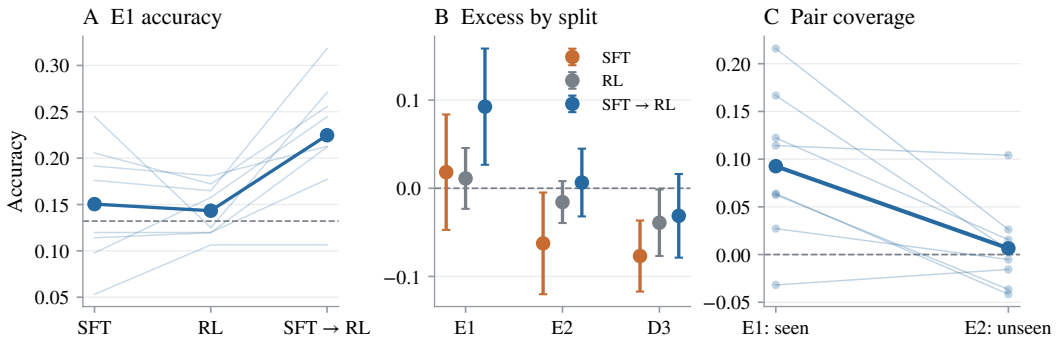

Figure 6: **The original disjoint-half comparison.** (A) E1 accuracy by language and arm. (B) Excess over each split's constant, with paired 95% intervals. (C) Per-language E1 and E2 excess after trace-SFT→RL. These sampled-input results precede the full-pool and all-input controls; they do not establish that RL is necessary for composition.

E.2 Where reward-only composition fails

A checkpoint audit at 0.5B gives model/constant accuracy $0.511/0.113$ on trained depth-1 rows, $0.305/0.131$ on trained depth-2 rows, $0.194/0.127$ on new inputs to those depth-2 programs, and $0.128/0.160$ on jointly witnessed E1 programs. On the weaker subset where pair and template are each witnessed but not necessarily together, accuracy is $0.155/0.164$. Failure therefore appears both in training fit and in transfer.

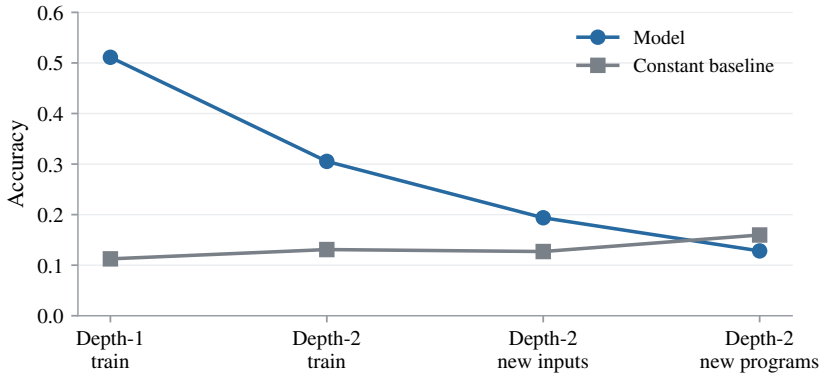

Figure 7: **Accuracy decreases before and after distribution shift.** The checkpoint audit compares increasingly demanding evaluations at 0.5B. The baseline is recomputed for each set. These are descriptive set-level accuracies, not matched examples along a causal trajectory.

A single-fault analysis classifies wrong depth-2 answers by whether they are consistent with skipping an operation, copying the input, or substituting an operation. Counts are corrected for accidental matches in the sixteen-value answer space. For no-trace RL on E1, excess skip-pattern counts are 1,117 for omitting the outer operation and 985 for omitting the inner one; input-copying excess is 56. Trace-trained arms have three to four times smaller skip excesses. These errors are consistent with omitted operations, but do not identify the model's internal computation.

E.3 Trace order and behavioral faithfulness

A 0.5B SFT control holds programs, examples, epochs, and tokenizer fixed while varying the target: innermost-first steps, the same steps in outermost-first order, or an answer alone. The two step formats are token-matched. Sampled-input E1 accuracy is 0.210, 0.155, and 0.099, respectively. The innermost-first advantage over outermost-first is 0.055 on E1 ($t = 2.47$), 0.040 on E2 ($t = 4.33$), and 0.119 on E3 ($t = 2.93$). Trace order affects this 0.5B SFT endpoint. The later 7B outermost-first initialization control tests a different model scale and subsequent GRPO continuation (Appendix I.10).

The available step checker validates each parsed primitive application against the DSL operation tables. It does not receive the original query tree or input, and does not verify that the lines cover the query or chain intermediate values correctly. Its output is therefore a local step-correctness diagnostic, not a certificate of complete program execution or internal reasoning.

A separate frozen summary (`faithfulness_denoms`) reports answer / trace-valid / jointly-correct rates of $0.425/0.374/0.363$ for continued SFT and $0.302/0.224/0.218$ for GRPO at 0.5B. At 7B, the corresponding rates are $0.919/0.909/0.909$ and $0.898/0.884/0.883$; the depth-3 jointly-correct rates are 0.582 and 0.500. These values retain the archived diagnostics and their original denominators. They have not been revalidated as measures of complete-trace correctness. The release lacks their item-level records and a verified producer for the separate summary. We therefore do not use them as evidence that successful answers require faithful execution. No-trace RL emits no parseable steps, which similarly provides no direct evidence about its internal computation.

F Learning Dynamics and Protocol Selection

At learning rate 3×10^{-6}, the 0.5B–3B policies usually converge to short in-domain outputs within several hundred updates. The same rate destabilizes the 7B protocol: across eight collapsed runs, average training reward falls from 0.179 at the first logged point to 0.047 at the last, and final held-out parse rate averages 0.25. Reducing the rate to 10^{-6} stabilizes the main 7B runs. The 1.5B model, however, under-trains at that lower rate.

These observations motivate the separate scale protocols. They do not support a scaling law at fixed optimization settings. Endpoint evaluations use the final checkpoint from each run; checkpoints

are not selected for the best evaluation score. Several 0.5B conditions show reward decline after an earlier peak.

G Controls for Table Learning

G.1 Dense supervision and codebook availability

The oracle-trace control uses the same 360 depth-1 rows as reward training, with eight languages and two optimizer seeds at each of three and twelve SFT epochs. It totals 64 cells. At three epochs, nonsense accuracy is 0.328 against 0.231 ($t = 2.11$); conjugated accuracy is 0.168 against 0.220 ($t = -3.52$). At twelve epochs, binding is stronger but conjugated excess remains undetected (Table 11). The paired excess gap is 0.338 ($t = 5.48$, eight positive signs).

Table 11: **Conjugated-table controls.** All-input accuracy and paired t against the per-program baseline 0.220, averaged over eight languages. The 0.5B nonsense reference under twelve-epoch SFT reaches 0.596 against 0.231.

Scale	Condition	Accuracy	$t(7)$
0.5B	GRPO, hidden codebook	0.246	1.37
	Oracle SFT, 12 epochs	0.248	1.73
	Construction stated, GRPO	0.257	1.88
	Construction stated, SFT	0.251	1.65
	Exact codebook, GRPO	0.267	3.14
	Exact codebook, SFT	0.267	3.27
3B	GRPO, hidden codebook	0.215	−0.30
	SFT, hidden codebook	0.253	2.02
	SFT, exact codebook	0.290	7.12
	GRPO, exact codebook	0.234	0.75

The informed-prior prompt names the true base operations and explains that they share a hidden permutation. At 0.5B, this changes GRPO and SFT accuracy by only 0.012 ($t = 1.23$) and 0.003 ($t = 0.35$) relative to their uninformed counterparts. Providing the exact permutation instead produces positive effects against the baseline under both recipes. The resulting accuracy remains low, so codebook access does not make execution reliable.

The 0.5B exact-codebook excess lies inside the hidden-codebook GRPO interval $[-0.018, 0.069]$. This comparison alone therefore cannot separate failed acquisition from inaccurate execution after acquisition. At 3B, codebook-given SFT reaches 0.290, while hidden-codebook SFT reaches 0.253 without a detectable excess. Codebook-given GRPO exceeds its matched hidden twin by 0.019 ($t = 2.95$), yet does not itself exceed the constant baseline. The controls show recipe-sensitive execution, not a universal ordering of internal learning stages.

At 7B, exact-codebook GRPO reaches 0.225 against its 0.234 baseline ($t(5) = -0.24$). Two of eight languages collapse and are excluded from that comparison. The paired difference from the hidden-manual twin is -0.057 ($t(5) = -1.86$). A separate 3B protocol removes the brevity instruction and allows 512 completion tokens. Its codebook-given accuracy is 0.371 versus 0.202 for the hidden twin, but the paired 0.169 difference is imprecise ($t(7) = 1.50$, five positive signs).

An inference-time intervention adds the codebook to already trained hidden-codebook 3B models. Accuracy decreases to 0.098 from 0.215. Because the new prompt is out of distribution for those checkpoints, this does not prove that they lack a latent executor. The 3B training campaign also includes walltime-limited cells: the aggregated GRPO results use a checkpoint-resume rerun rather than pooling those incomplete runs. Conjugated tables were not run at 1.5B; this was a compute-allocation choice made before that wave.

G.2 Correctness-independent rewards

Shao et al. (2025) report reasoning gains from spurious rewards in some model families. We test whether similarly uninformative rewards can produce binding gains in Alien DSLs. Under

Bernoulli(0.5) rewards independent of correctness, 0.5B B accuracy is 0.022 versus the 0.165 constant baseline ($t = -8.47$), compared with 0.464 under exact verification. A second control shuffles true rewards across completions within each reward call, preserving the positive count and density while removing answer–reward correspondence. B accuracy is 0.138 ($t = -1.03$), and C excess is 0.034 ($t = 1.11$). Neither control reproduces the binding gain.

G.3 Exploratory rejection-SFT and synthesis

On a single language (seed 202), GRPO reaches B accuracy 0.393 at a 128k-sample generation budget. Rejection-SFT on verified base samples reaches 0.000, or 0.036 after tuning to one epoch. The matched comparison has eleven GRPO-only correct examples and none in the opposite direction, giving exact McNemar $p = 0.00098$. This is a single-language observation.

The inverse synthesis task asks the model to write a program consistent with input–output examples, then verifies all sixteen inputs. Base pass@8 is zero in the evaluated cells. Reward-only training exceeds a trailing reward threshold of 0.05 after 290, 520, 710, 860, and 1,170 updates for seeds 202, 505, 606, 303, and 404; seed 101 never crosses it. The strongest seed's five correct held-out functions all belong to the maximum family, including one constant-equivalent function. This concentration is consistent with template identification and does not establish general program induction. Two manual-bearing runs reach held-out accuracy 0.133 and 0.000.

H Relation to Other Post-Training Comparisons

Table 12: **Closest comparisons address different transfer targets.** The table describes the relevant experimental contrasts, rather than ranking methods across incompatible tasks. Oracle traces and verifier-selected model traces provide different supervision.

Study	Learning and supervision contrast	Transfer target
Yuan et al. (2026)	Atomic string skills followed by compositional RL or rejection fine-tuning	Held-out string-function compositions, greater depth, and cross-task transfer
Cheng et al. (2025)	Atomic parametric/contextual skills followed by further SFT, LoRA, or RL on identical composition samples	Integration of stored and contextual facts on new facts and reasoning paths
Kong et al. (2026)	Compound traces, RL, and different relationships between SFT/RL composition sets	Recovery and reuse of atomic skills and routing interfaces
Alien DSLs	Outcome rewards, oracle traces, and rejection-imitation under controlled task variants	Symbol binding, same-cell query access, and syntactic composition with all-input audits

The identifiability questions also differ. Kong et al. (2026) give sufficient conditions for identifying latent modules and composing their interfaces. Our solver asks whether the observed input–output rows identify the generating language within a specified family. It does not recover or identify a language model's internal modules. Similarly, our E1 witness rule checks observed program pairs and templates; it is not a test of all latent-interface conditions in that theory.

The composition comparisons differ in the supervision each training arm receives. Yuan et al. (2026) compare outcome RL with rejection fine-tuning on synthetic string transformations and find greater compositional transfer from RL. Our rejection-imitation comparison is qualitatively compatible with that observation. Continued oracle-trace SFT is an additional comparator with a different information source; its success at smaller scales shows that reinforcement is not required when that supervision is available.

Other comparisons show that the relative performance of SFT and RL depends on what supervision supplies and how transfer is tested. Chu et al. (2025) report stronger RL generalization to rule and visual variants, while Ren et al. (2026) identify conditions under which extended reasoning SFT generalizes across domains. Kim et al. (2025) distinguish accuracy gains from large-k capability estimates, and Hu et al. (2025) study a setting where distillation outperforms zero-RL. Trajectory

analyses (Matsutani et al., 2025) and modifications to the SFT objective (Wu et al., 2025b) further examine how the two training methods differ. Our experiments match update counts but retain differences in supervision information, so they do not establish a general ranking of objectives.

The random-table results leave several explanations for limited RL fitting open. Prior work studies entropy collapse (Cui et al., 2025) and examples that remain difficult despite available correct rollouts (Chen et al., 2026). We do not test those mechanisms directly. Short SFT runs fail to fit the tables, whereas longer runs largely succeed, showing that the fitting result depends on the protocol and budget. Related knowledge-acquisition studies examine slow learning of new facts (Gekhman et al., 2024), failures to extract learned information (Allen-Zhu & Li, 2024), and training methods that improve knowledge integration (Park et al., 2025). Our same-cell comparison isolates a change in query form within a finite operation table. Generating task semantics after pretraining also complements studies that control pretraining mixtures (Zhao et al., 2025b) or audit benchmark contamination (Wu et al., 2026).

I Preregistered Experiments and Fitting Extensions

I.1 Original preregistered family

The original preregistration fixes three primary comparisons before job submission: random-table excess, the full-pool RL increment over trace-SFT initialization, and the fresh-suite disjoint-half contrast against SFT-only. It specifies deterministic seed suites and a Holm correction over three tests. The associated 0.5B campaign contains 144 cells, with approximately 4.7 node-hours of compute. The later all-input composition audit is exploratory, even where the original sampled-input endpoint was preregistered.

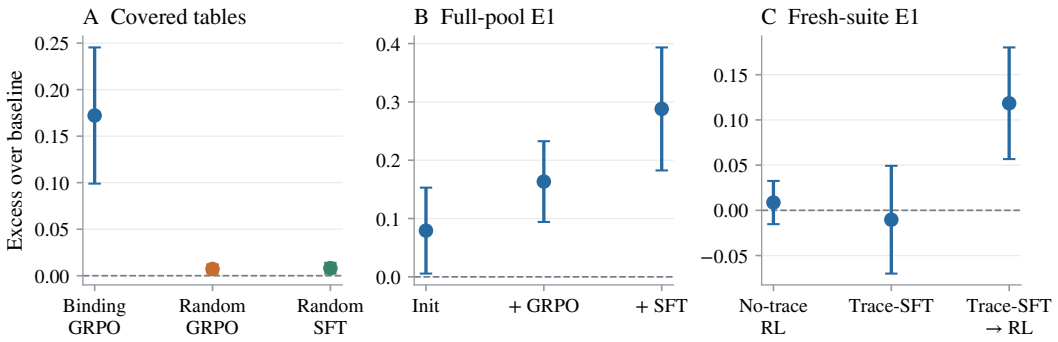

Figure 8: **Original preregistered comparisons.** (A) Covered-split all-input excess. (B) Full-pool E1 excess under the sampled-input protocol. (C) Fresh-suite E1 excess under the disjoint-half sampled-input protocol. Bars are paired 95% intervals across eight languages. Each panel uses its own stated baseline.

I.2 Covered random tables

The random family uses seeds 31111–38888 and two optimizer seeds. Every unary primitive is a random bijection. For each binary primitive, each second-argument column is an independent random bijection. Training contains all `op(c,x)` programs at all sixteen inputs, plus unary examples, for 1,088 rows. The 1,024 binary test rows use `op(x,c)`. Every test cell is covered, and every held-out program has constant baseline $1/16$.

The pretrained model scores 0.022 on the strict audit. Reward-only training reaches 0.070, with $t(7) = 3.44$, corrected $p = 0.019$, and excess CI $[0.002, 0.012]$. Matched oracle SFT reaches 0.071 ($t = 3.34$). The same-split binding reference reaches 0.404 against 0.232; its paired excess advantage over random-table GRPO is 0.165 ($t = 5.44$). The preregistration did not fix a numerical equivalence margin. We therefore report a small positive effect and its interval, rather than declaring an absence of learning.

Approximately 49.8% of reward-only GRPO groups have nonzero within-group reward variance across the sixteen runs (range 43.5–54.1%). The mean within-group reward standard deviation is 0.162. Many groups therefore provide a reward contrast, although this alone does not establish that their gradients support efficient table fitting.

I.3 Additional fitting budgets and training paths

A post-hoc sweep varies learning rate over $\{3 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}\}$, group size over $\{16, 32\}$, and budget up to four times the original. Faster settings collapse or remain near chance, with strict endpoints 0.056–0.062. Extending the original recipe to 20k updates on two languages gives strict excesses 0.046 and 0.038. At 40k across all eight languages, free-generation training fit is 0.301 and strict excess is 0.021 ($t = 4.21$, eight positive signs). The two languages measured at both budgets change from $0.046/0.038$ to $0.040/0.038$ excess.

At 3B, two exploratory 8k runs fit 0.126 and 0.309 of rows, with excesses 0.015 and 0.004. Two 7B runs reach fit 0.329 and 0.393, with excesses 0.031 and 0.023. One 7B run is retried at learning rate 5×10^{-7} after collapse at 10^{-6}. These two-seed probes are not population-scale confirmation.

Table 13: **Additional unified-instrument endpoints.** Fit and transfer are free-generation exact match. The 20k row has two languages; all other rows have eight. The same-split binding reference uses its own baseline.

Recipe	Fit	Transfer	Excess	n
Binding reference, 2k GRPO	0.522	0.402	+0.170	8
Random GRPO, 20k	0.277	0.105	+0.042	2
12-epoch SFT $\rightarrow$ 4k GRPO	0.194	0.100	+0.038	8
48-epoch SFT $\rightarrow$ 4k GRPO	0.984	0.126	+0.064	8

Four-round rejection-SFT uses 128k verifier calls, matching the original 2k GRPO generation budget. Its sampled hit rate rises from 0.007 to 0.032, but final free-generation fit is only 0.055, with transfer excess -0.001 and parse rate 0.78. After a twelve-epoch SFT warm start, 4k GRPO updates raise strict transfer from 0.071 to 0.098 ($t = 3.86$ for the increment). After a 48-epoch warm start, strict transfer changes by only 0.002 ($t = 0.64$); 90–98% of GRPO groups then have zero reward variance. This last diagnostic concerns reinforcement after fitting, not its ability to acquire previously unfit rows.

I.4 Unified evaluation and fit–transfer association

The fitting campaign initially logged sampled rewards for RL and token accuracy for SFT. The unified evaluation remeasures both training and held-out rows using greedy free generation, the same chat template, and the same exact-match parser. Parse rates are 0.95–1.00 except for rejection-SFT. Dense SFT's token accuracy of 0.999 corresponds to free-generation fit 0.951, so high token accuracy is not being substituted for task accuracy in the main table.

A regression over nine random-family arms and 60 language-level points predicts excess from fit, training path, and their interaction. With language-clustered standard errors, the fit coefficient is 0.065 ($t = 6.30$), the interaction is undetected ($t = -0.80$), and the path offset is 0.009 ($t = 2.61$). Training fit was not randomly assigned, and the regression contains only eight language clusters, limiting the reliability of asymptotic inference. We interpret the positive fit–transfer association descriptively, not as proof of a common causal curve.

A teacher-forced answer-token probe scores 0.048 on rows that the 40k GRPO models miss under free generation, versus 0.980 on rows they answer correctly. This suggests that the tested readout does not readily recover the missing answers. Because the probe conditions on free-generation success and tests only one readout, it cannot establish that answer information is absent internally (historical key `storage_certificate`).

I.5 CANONICAL CELLS AND EXPRESSION CHANGES

A balanced-orientation control trains half of the constants in each argument position, then evaluates mirrored forms. Dense SFT has token accuracy 1.0 on training but held-out accuracy 0.127 against 0.128 ($t = -0.03$); reward-only excess is -0.063. Merely exposing both syntactic forms does not remove the transfer gap in this control.

The partial-coverage experiment trains twelve of sixteen constants in `op(c,x)` form. A naive expression-based split mixes observed and unobserved cells: 75% of mirrored-form queries refer to canonically observed entries. We therefore classify every query by $(\texttt{op}, a, b)$ before aggregating. Table 14 reports the pure groups.

Table 14: **Canonical-cell audit after dense SFT.** Mean exact-match accuracy across eight languages. The dictionary abstains on unobserved cells. The $1/4$ comparator is the expected family-aware accuracy given four remaining values per column.

Cell status	Predictor	Trained form	Mirrored form
Observed	Model	0.965	0.137
	Training-cell dictionary	1.000	1.000
Unobserved	Model	0.049	0.061
	Uniform guessing	0.0625	0.0625
	Family-aware prediction	0.250	0.250

I.6 FULL-POOL FACTORIAL AND STRICT COMPOSITION AUDIT

The full-pool factorial uses the original eight languages and two optimizer seeds. Sampled-input E1 accuracy is 0.211 after trace-SFT initialization, 0.295 after GRPO, and 0.420 after continued trace-SFT, against 0.132. The preregistered RL increment is 0.084 ($t = 3.53$, corrected $p = 0.019$). Continued SFT exceeds GRPO by 0.125 ($t = 6.06$), and sampled-input E2/E3 also exceed their baselines. D3 does not.

Evaluating the same endpoints over all inputs gives a more restricted account of transfer. At 0.5B, E1 accuracy is 0.426 after continued SFT against 0.281 ($t = 3.80$), but 0.305 after GRPO ($t = 1.18$). Continued-SFT E2/E3 effects remain positive without reaching significance, and all D3 effects are negative. The fresh disjoint-half replication also fails the strict E1 threshold after GRPO (0.270 against 0.263, $t = 0.19$). These distinctions explain why the main text uses the all-input criterion consistently.

At 3B, nominal all-input E1† accuracy is 0.396 for continued SFT, 0.335 for GRPO, and 0.258 for rejection-imitation. The paired continued-SFT advantage over GRPO is 0.061 ($t = 3.61$), and the GRPO advantage over rejection is 0.078 ($t = 3.39$). All three D3 excesses are below baseline: -0.157, -0.167, and -0.181. One rejection cell is rerun after a batch-size fault with unchanged effective batch size.

A continued-SFT dose series at 0.5B gives sampled-input E1 accuracy 0.392, 0.420, and 0.433 at 750, 1,500, and 3,000 updates. The first increment is 0.028 ($t = 2.47$), while the second is 0.013 ($t = 1.46$). D3 remains at its constant baseline at every budget. This is a post-hoc budget comparison, not part of the original primary family.

I.7 7B PHASE-2 EFFECTS AND INITIALIZATION CONTROLS

At 7B, a re-instantiated phase-1 checkpoint reproduces the original sampled-input endpoints and shows no positive all-input detection on any split. GRPO and continued SFT each produce positive phase-2 gains on all eight languages. Their final E1† difference, 0.076 in favor of SFT ($t = 1.01$), is imprecisely estimated. Rejection-imitation has a positive but much smaller E1 effect and remains below baseline on D3.

Answer-only initializations show no statistically detectable positive mean excess on any split, although some E2/E3 language effects are positive. The atomic-trace mean effects are also undetected; seed 2222 nevertheless transfers on all four splits. From-base GRPO has no positive E1/E2/D3 language

Table 15: **Alternative initializations before the same 7B GRPO phase.** All-input excess averaged over eight languages. Parentheses give the number of positive language-level effects. Positivity alone is not a significance test.

Initialization	E1†	E2	E3	D3
Atomic answers	−0.099 (0)	−0.119 (1)	−0.139 (1)	−0.187 (0)
Atomic traces	−0.006 (3)	−0.018 (2)	−0.001 (3)	−0.144 (1)
Composition answers	−0.082 (0)	−0.123 (1)	−0.116 (2)	−0.167 (0)
Composition traces	+0.530 (8)	+0.489 (8)	+0.430 (7)	+0.280 (7)
Outermost-first traces	+0.544 (8)	+0.455 (8)	+0.425 (7)	+0.246 (7)

effects and one positive E3 effect. Accordingly, the evidence supports reliable transfer under composition-trace initialization, not a categorical statement that every alternative fails on every language and split.

I.8 Fresh-suite 7B confirmation

The confirmatory suite uses seeds 1212, 2323, 3434, 4545, 5656, 6767, 7878, and 8989, with two optimizer seeds per arm. Both arms use the frozen 7B full-pool protocol, learning rate 10^{-6}, 1,500 phase-2 updates, and the same all-input instrument. The design specifies intent-to-treat handling, infrastructure retries, and a single lower-rate retry after reward collapse. None is needed: all 32 cells complete without collapses, reruns, or exclusions.

Table 16: **Preregistered secondary comparison on fresh languages.** Paired continued-SFT minus GRPO excess, after averaging optimizer seeds within each language ($n = 8$). All 95% intervals include zero; this is not evidence of equivalence. Recomputed from rounded frozen language aggregates.

Split	Held-out structure	SFT−GRPO	95% CI
E1†	Nominal new programs	+0.034	[−0.039, 0.108]
E2	New primitive pairs	+0.082	[−0.041, 0.206]
E3	New templates	−0.017	[−0.096, 0.061]
D3	Depth 3	+0.013	[−0.071, 0.097]

Table 18 reports primary arm means by wave; Table 16 gives the first wave's preregistered secondary contrast. Table 17 reports the range across the two optimizer seeds within each language. GRPO ranges reach 0.3581 on E2, so similar arm means should not be read as similar optimization stability. The source package includes a script for recomputing these estimates from frozen language aggregates, and a CSV linking each contributing value to its JSON source path.

Table 17: **Optimizer-seed variability in the confirmation.** Maximum minus minimum all-input excess across the two optimizer seeds, within each language. These ranges are descriptive, not confidence intervals.

	GRPO				Continued trace-SFT			
Language	E1†	E2	E3	D3	E1†	E2	E3	D3
1212	0.016	0.115	0.081	0.078	0.025	0.023	0.026	0.125
2323	0.117	0.191	0.118	0.105	0.001	0.009	0.031	0.018
3434	0.035	0.074	0.077	0.072	0.004	0.031	0.000	0.008
4545	0.233	0.358	0.029	0.284	0.047	0.016	0.029	0.130
5656	0.012	0.013	0.019	0.208	0.004	0.012	0.023	0.074
6767	0.031	0.262	0.072	0.081	0.008	0.046	0.058	0.070
7878	0.135	0.017	0.100	0.214	0.003	0.019	0.002	0.029
8989	0.010	0.014	0.013	0.042	0.022	0.019	0.050	0.005

The largest Holm-corrected paired-test p in wave 42 is 0.000321 across two arms and four splits. Every language effect is positive for all eight comparisons. Exact sign tests give 0.0078125 per effect,

with the family-wise limitation described in Appendix C. This wave confirms the two recipes on new languages; it does not repeat the initialization controls.

I.9 Sequential replication and descriptive pooling

The second wave (43) uses seeds 1313, 2424, 3535, 4646, 5757, 6868, 7979, and 8181, with two optimizer seeds per arm. Its amendment was registered after the first wave's result and before the second wave's jobs. All settings are carried over. Because this extension followed inspection of wave 42, we report the pooled result descriptively rather than as a fixed-sample confirmatory test.

Table 18: **7B transfer in two sequential fresh-language waves.** Mean all-input excess over the per-program constant; eight languages per wave. Wave 43 means are recovered as twice the rounded pooled mean minus the rounded wave 42 mean, so their final digit is approximate. E1† denotes the nominal split.

Wave	Continuation	E1†	E2	E3	D3
42	GRPO	0.594	0.533	0.626	0.283
42	Continued trace-SFT	0.629	0.615	0.609	0.296
43	GRPO	0.497	0.448	0.457	0.218
43	Continued trace-SFT	0.601	0.544	0.497	0.253

In wave 43, seven of the eight effects are positive on all eight languages; GRPO depth-3 is positive on seven. In the pool, the corresponding sign counts are $16/16$ and $15/16$. The maximum Holm-adjusted paired-test value is 1.19×10^{-6}. The exact two-sided sign-test values are 3.05×10^{-5} and 5.19×10^{-4} before correction; their Holm-adjusted values are 2.44×10^{-4} and 5.19×10^{-4}. These checks summarize the observed pool; they do not account for the decision to extend the study after the first wave.

The frozen pooled record contains means, standard deviations, test statistics, and sign counts, but no per-language vectors or optimizer-seed ranges for wave 43. Thus the manuscript can reconstruct pooled intervals from summary statistics, but cannot independently reaggregate wave 43 or estimate its paired between-arm contrast. The phrase "preregistered family" retained in that historical JSON record should be read in light of the sequential history above.

I.10 Outermost-first composition traces at 7B

Wave 44 is exploratory: eight original languages and one optimizer seed, initialized with the same composition-step lines in outermost-first order before the standard 7B GRPO phase. E1†/E2/E3/D3 mean excess is $0.544/0.455/0.425/0.246$, with paired $t(7) = 8.71/7.22/4.69/4.36$ and positive effects on $8/8$, $8/8$, $7/8$, and $7/8$ languages. The order no longer follows forward execution, yet the final models transfer on every split at the population level. This excludes strict innermost-first ordering as a necessary condition for this recipe. It does not isolate the benefit of step content from all other initialization properties or demonstrate equivalent performance between trace orders.

I.11 Fresh-suite disjoint-half replication

The earlier replication uses fresh seeds 21111–28888, the original disjoint-half protocol, and a fixed witness rule that excludes 12 of 384 candidates. Sampled-input E1 accuracy is 0.128 for SFT-only, 0.147 for no-trace RL, and 0.257 for trace-SFT→RL, against 0.139. The combined arm exceeds SFT-only by 0.129 ($t = 7.83$, corrected $p = 0.0003$) and no-trace RL by 0.110 ($t = 3.30$). This sampled-input result is specific to its evaluation protocol; the stricter all-input audit does not detect positive excess for trace-initialized GRPO at 0.5B.

J SAMPLING, RETENTION, AND ADDITIONAL PROBES

J.1 PASS@k IN A SMALL ANSWER SPACE

With sixteen possible answers, independent uniform guessing gives $1-(15/16)^k$. For $n \geq k$ samples with c correct answers, the unbiased estimator is $1 - \binom{n-c}{k}/\binom{n}{k}$ (Chen et al., 2021). We collect 128 completions at temperatures 0.2, 0.7, and 1.0.

At temperature 1.0, base nonsense pass@k rises from 0.013 at $k = 1$ to 0.583 at $k = 128$. The trained curve starts at 0.473 and grows more slowly, eventually falling below the base curve. Parse failures keep the base below the uniform-guessing curve. This crossover does not by itself identify whether the trained model acquired a new operation.

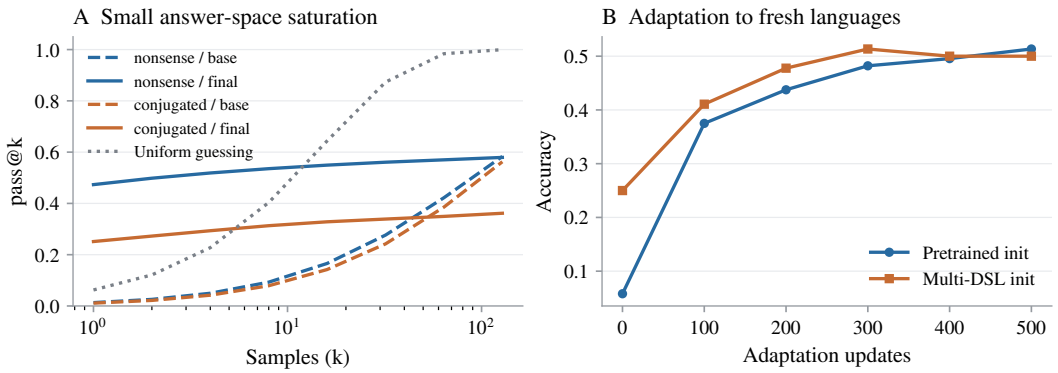

Figure 9: **Sampling and adaptation probes.** (A) Base and trained pass@k at temperature 1.0, with uniform guessing for reference. (B) Adaptation to fresh languages from the pretrained model or a model trained on eight other languages. These are auxiliary evaluations, not the main all-input composition test.

J.2 RETENTION AND ATTENTION IMPLEMENTATION

Retention probes use 150 GSM8K items (Cobbe et al., 2021), 300 MMLU items (Hendrycks et al., 2021), and approximately 100k WikiText-2 tokens (Merity et al., 2017). We remeasure retention with default SDPA after finding an eager-attention numerical fault at 7B: base GSM8K accuracy is 0.79 with SDPA versus 0.21 with eager attention. On one checkpoint, SDPA minus eager DSL accuracy is 0.0000 on E1†, -0.0013 on E2, -0.0039 on E3, and $+0.0052$ on D3. These small differences do not establish equivalence across implementations.

With SDPA, GSM8K changes at 0.5B are -0.010 for reward-only training and -0.022 for trace composition; the 3B change is -0.001. MMLU changes lie in $[0.002, 0.006]$ and perplexity changes in $[0.02, 0.15]$. Probe resolution is limited: per-checkpoint binomial 95% half-widths are approximately 0.08 on GSM8K and 0.06 on MMLU, precluding an equivalence claim.

Across eight 7B language-trained checkpoints per arm, GSM8K averages 0.532 after continued trace-SFT and 0.728 after GRPO, against 0.780 for the base model. GRPO's paired advantage is 0.195, $t(7) = 3.69$. MMLU and perplexity show no detected change at this resolution. SFT also produces shorter outputs, but this association does not explain the retention loss.

J.3 LONGER CONJUGATED-TABLE TRAINING AND META ADAPTATION

An 8k-update conjugated-table probe compares plain GRPO with KL regularization $\beta = 0.005$ and reference resets every 400 updates. Sampled-input endpoints are 0.250 and 0.241, respectively, compared with the original 2k reference 0.262. Neither extension produces an improvement in this probe.

For meta adaptation, we train on the eight population languages for 4k updates, then adapt to eight fresh seeds 9111–9888. This initialization starts at 0.250 zero-shot accuracy versus 0.058 for the

pretrained model, and reaches 0.4 at the 100-update rather than 200-update checkpoint. At 500 updates, accuracy is 0.500 versus 0.513 ($t(7) = -0.55$). Held-out depth-2 accuracy is $0.145/0.171$ for multi/pretrained initialization against 0.137; D3 is $0.113/0.100$ against 0.126. The adaptation advantage concerns early binding, not robust compositional transfer.