Calibrated Tool Use via Learned Adapters to Compiled Neural Programs

Anonymous authors
Paper under double-blind review

Abstract

Language models typically call a tool with a single set of arguments, even when several readings of the input remain plausible. This discards useful information, and as a result, reinforcement learning for tool use must estimate credit by sampling many trajectories. We propose SPACE-LLM: Sampling-free Probabilistic Adapters for Calibrated Execution in Large Language Models. Learned adapters over LLM hidden states predict distributions over the arguments of a compiled neural program, a symbolic program written as tensor operations that execute inside the model's own forward pass. Retaining these argument belief distributions allows inference to combine support for an answer across ambiguous readings and enables training to compute analytic gradients from a ground truth answer. We show that maintaining belief distributions at inference, rather than committing to one reading, raises accuracy and improves calibration without a calibration-specific loss. Using only answer-level supervision, we observe comparable or stronger out-of-distribution generalization than fully supervised finetuning and a comparable Brier score, from a single pass without sampling or a recalibration step. On identical training sets, SPACE matches or exceeds GRPO accuracy with a lower Brier score more than 50 times faster, and the tool runs inside the model's forward pass. These advantages are limited to domains in which maintaining the factored belief state is tractable, the answer carries enough information to credit each argument, and the order of operations is known in advance.

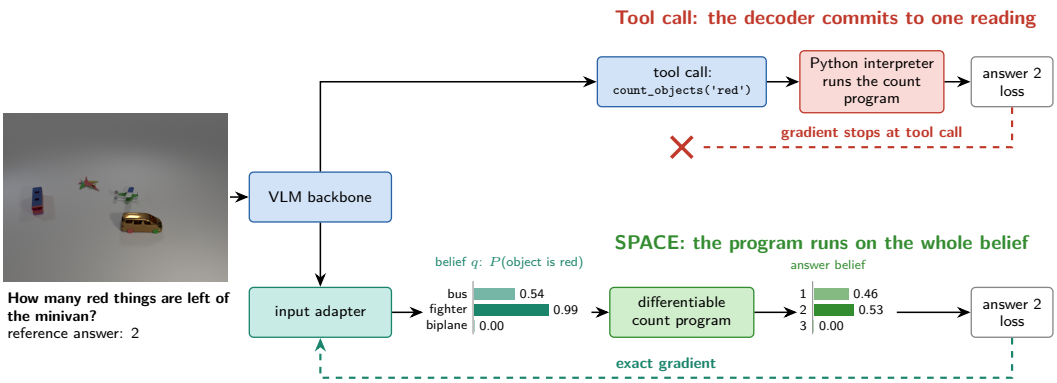

Figure 1: Keeping the belief distribution lets the loss backpropagate into an ambiguous reading. A text tool call commits to one reading, so no gradient reaches the adapter; SPACE runs the compiled program on the whole belief, so the gradient does.

1 Introduction

A language model typically calls a tool by writing one set of arguments (Gao et al., 2023; Chen et al., 2023; Schick et al., 2024) into a symbolic external function. This premature commitment to one set of arguments throws away the model's uncertainty about how to best call the tool because

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.

the probability mass on alternative arguments may actually support a different answer. For example, suppose a model has a tool with three possible arguments A, B or C. The model has confidence .40 for A, .35 for B, and .25 for C. Executing the tool with A produces an incorrect answer, while B and C both result in the correct answer. With greedy sampling, the model would execute the tool with A, arriving at the incorrect answer. If instead the model could somehow execute all possible tool arguments simultaneously, it would see that B and C arrive at the same answer. It could then sum the probabilities of each tool result and answer with the correct result.

Our approach maintains this argument probability distribution and integrates the computation into the training process. A task-specific program schema is compiled into fixed tensor operations, a compiled neural program in the sense of Weng et al. (2024), that executes inside the language model's forward pass rather than in an external code interpreter. To do this, a learned adapter network predicts a distribution over every argument from a middle layer of the LLM, and the compiled program runs on those distributions. A learned output adapter projects the program's result into a reserved token position in the residual stream so that the LLM's upper layers answer in natural language (Figure 2). Because every operation is differentiable, the answer loss returns an analytic gradient to each argument, and the adapter trains from answers alone. With standard discrete argument selection, small changes in the model's argument probabilities leave the arguments and tool output unchanged, so a loss on that output has zero gradient almost everywhere. Because we maintain and propagate the uncertainty end-to-end, the output has a more informative gradient that can correct specific errors in tool argument selection.

Figure 1 shows this on a multimodal scene from the Super-CLEVR benchmark (Li et al., 2023a). The normal tool call commits to the bus being red, and nothing the loss says can reach the model's uncertain 0.54 belief that the bus is red. Score-function methods such as REINFORCE and GRPO sample whole argument sets, reward those whose answer is right, and raise their probability (Williams, 1992; Shao et al., 2024); straight-through estimators commit in the forward pass and ignore the commitment in the backward pass (Bengio et al., 2013). Both replace the exact derivative with a noisy or biased estimate, and with group-centered rewards a group whose samples all earn the same reward supplies no reward signal (Appendix A).

Our contributions are:

1. We introduce SPACE-LLM, a novel neurosymbolic LLM architecture, which trains from answer supervision to learn effective and calibrated tool calling behavior. SPACE-LLM executes tools in a single forward pass by integrating compiled neural programs into its architecture (Figures 2 and 3).
2. We isolate the value of maintaining belief argument distributions by comparing two readouts of the same trained weights: *Soft*, which propagates belief distributions through the compiled program, and *Commit*, which replaces each with a one-hot vector. Soft regularly beats Commit in domains where there is ambiguous problem state.
3. We compare SPACE against supervised finetuning (SFT) and Group Relative Policy Optimization (GRPO) at equal training data and clock time. In certain domains, SPACE trains 50 to 85 times faster than GRPO at equal data, generalizes out of distribution as well as or better than SFT, and is calibrated comparably to SFT, without argument labels, sampling or recalibration. We characterize where these advantages hold and where they fail (Figure 5).

2 Related work

Symbolic programs can be compiled into transformer weights (Weiss et al., 2021; Lindner et al., 2023; Shaw et al., 2025) or attached to pretrained models (Weng et al., 2024; Vergara-Browne & Soto, 2025; Saldyt & Kambhampati, 2024), and networks can also learn programs (Graves et al., 2014; Zhang et al., 2026). Neurosymbolic methods combine neural beliefs with symbolic rules and constraints and can learn from answers alone (Manhaeve et al., 2018; Li et al., 2023b; Xu et al., 2018; van Krieken et al., 2023; Minervini et al., 2020; Lu et al., 2022; Berant et al., 2013; Guu et al., 2017; Gaunt et al., 2016), for example by training a relation extractor on CLUTRR to feed a differentiable logic reasoner (Zhang et al., 2023). Concept bottlenecks expose intermediate concepts that need not be literal (Koh et al., 2020; Margeloiu et al., 2021; Mahinpei et al., 2021; Marconato et al., 2023). Language models call tools by writing arguments as text or code (Gao et al., 2023; Chen et al., 2023;

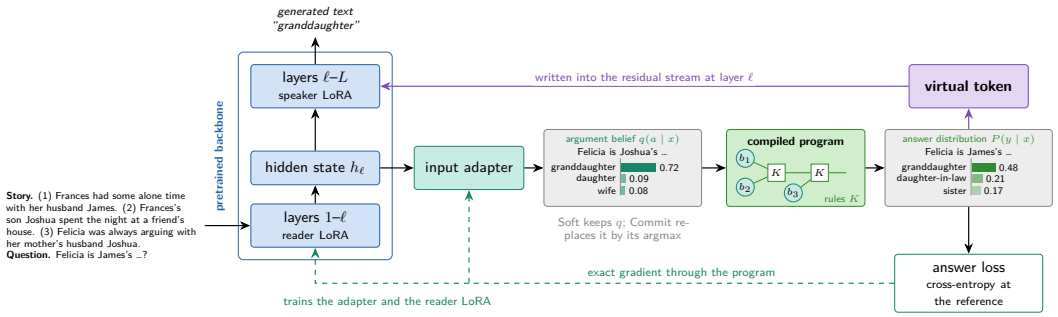

Figure 2: The input adapter reads intermediate LLM/VLM states, which reader LoRAs in the lower layers train for extraction, and supplies argument beliefs to a fixed compiled program. A learned virtual-token bridge writes the program's result into a reserved residual-stream position, and speaker LoRAs train the upper layers to read it before the native language-model head generates text. Bars show the adapter's belief for one sentence of the CLUTRR (Sinha et al., 2019) story in Figure 7 and the relation distribution the program computes from all three sentences.

Schick et al., 2024; Gupta & Kembhavi, 2023; Surís et al., 2023; Hao et al., 2023), and modular and attribute-based systems join perception to structured computation (Dugan et al., 2024; Andreas et al., 2016; Yi et al., 2018; Lampert et al., 2014; Atzmon & Chechik, 2018). These calls commit to one set of arguments in text, and symbolic programs cannot read language or images themselves, so SPACE joins the two with a usable gradient.

3 Methods

3.1 The SPACE approach

We use two Vision Language Model (VLM) backbones, Qwen3.5-4B (Qwen Team, 2026) and Gemma-4-E4B-it (Gemma Team, 2026). A prompt enters the backbone normally. At layer ℓ an adapter predicts a probability distribution for arguments into a symbolic program, which executes on them in the same graph. The result enters the residual stream as a token at layer ℓ, and layers $\ell + 1$ through L and the native head produce the answer text (Figure 2). Reading, execution and result injection therefore happen within one prefill pass, and answer-supervised training backpropagates through that pass. By contrast, a typical tool call generates arguments token by token, parses them, executes outside the model, and returns the result in another model call. The following five steps describe the SPACE forward pass.

A program as fixed tensor operations Each domain's symbolic program is implemented as fixed tensor operations with no trainable parameters and no external interpreter (Appendices E–I describe the program for each dataset). We compile each solver from a human-readable schema specifying its domains, rules, operations, and execution bounds into lookup tables and index tensors that hand-written, domain-specific tensor kernels consume (Appendix A.2). The compiled program is checked against a discrete reference on one-hot inputs. For a deterministic program f and factored argument distribution $q(a \mid x) = \prod_i q_i(a_i \mid x)$, the exact answer probability is

$$P(y \mid x) = \sum_{a : f(a) = y} \prod_i q_i(a_i \mid x). \tag{1}$$

A discrete program returns one answer for one set of arguments. Equation 1 runs the program on every possible set of arguments at once, weights each by its probability under the adapter's beliefs, and sums the weight on each answer. The result is multilinear: it is linear in each argument distribution q_i when the others are held fixed, because each q_i appears exactly once in each product term. As a concrete example, a `bird_identifier` tool scores each species from one-hot readings of a bird's visual features with a fixed species–feature table T. SPACE replaces the one-hot readings with the adapter's belief distributions q and scores every species with one product $T^\top q$ (Figure 3). A softmax over these

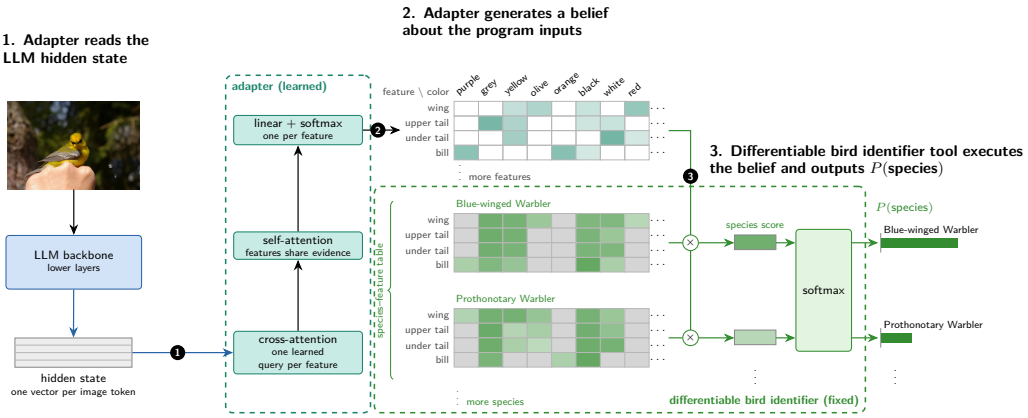

Figure 3: Example SPACE architecture for CUB-200 (Wah et al., 2011). (1) The adapter (teal, learned) reads the hidden state that the LLM backbone computes for the input. One learned query per feature cross-attends the image tokens, and the features share evidence by self-attention. (2) A linear layer with a softmax gives each feature a belief over its values; darker cells hold more probability. (3) The differentiable bird identifier (green, fixed) is simply a species–feature table which scores the belief for each species, and a softmax over the species scores gives $P(\text{species})$.

scores gives $P(\text{species})$, a relaxation of Equation 1. Table 1 lists what the program computes on each dataset.

An adapter fills the arguments The adapter predicts one distribution per program input from hidden states h_ℓ in a single pass, without recurrent decoding. Its architecture varies by domain, because each program expects different arguments read from different parts of the input (Appendices E–I). A middle layer, between 40% and 57% of the depth, is used for reading because its states provide contextual information useful for argument extraction, while the final layer is shaped toward next-token prediction. Layer probes informed the chosen read layer for each task (Appendix B.4).

LoRA below the read layer LoRA updates (Hu et al., 2022) in the layers producing h_ℓ move the representation toward one the adapter can extract.

Execution and the result token The program runs on the argument distributions and computes Equation 1 or a tractable relaxation of it, depending on the dataset (Table 1; Appendices E–I). In every case, minimizing $-\log P(y^\star \mid x)$ gives the exact derivative of the implemented objective without sampling. CLUTRR and Super-CLEVR add answer-free auxiliary losses on the adapter's beliefs (Appendices E and G). A learned output adapter projects the the answer distribution into a reserved token position at layer ℓ, allowing upper attention layers to cache and read it like an ordinary token. Appendix A.3 describes the shared-memory write needed by Gemma.

Reading the result and retaining language The upper layers and native language head continue from the result token. We fit an additional speaker LoRA above the read layer, together with the output adapter, while holding the reader LoRA and argument adapter fixed. We train the speaker LoRA to verbalize the program's result token: on training prompts, the result token carries a result chosen independently of the prompt, so the speaker learns to state that result rather than its own answer to the prompt. We measure WikiText-103 (Merity et al., 2017) perplexity on the base backbone and after fitting this interface to verify that the backbone still models language (Appendix J).

Readout and training comparisons We compare the performance of SPACE using the learned argument distributions (Soft) against selecting its most probable value as a one-hot vector (Commit). We contrast the training dynamics against traditional SFT and GRPO, which train LoRAs of the same rank (16) and alpha (32) on the language layers. All three methods share the same program, differing only in how the model is integrated with it and how it is trained to use it. SFT and GRPO call the

program through the backbone's native tool-call template, which the model already uses from its post-training (Appendix B.1). We compare the methods at equal data (iso-data) and at equal training clock time (iso-clock). Iso-clock compares the performance of each approach after a set time on one Nvidia H100, making a best effort to fully utilize the GPU. We use an iso-clock comparison as opposed to an iso-FLOPs comparison to better represent realistic hardware costs to LLM users. We score each method's answer distribution with the multiclass Brier score (Brier, 1950), which measures calibration and discrimination together, and Appendix D separates the two with a CORP decomposition (Dimitriadis et al., 2021). Appendix B.2 lists the budgets and the checkpoints each method uses.

Algorithm 1 gives the two training stages and Algorithm 2 gives inference, with the notation they use summarized in Appendix A.

3.2 Datasets

To evaluate SPACE across modalities and program structures, we study five benchmarks: CLUTRR (Sinha et al., 2019) for kinship relation composition over text, GSM8K (Cobbe et al., 2021) for multistep arithmetic word problems, CUB-200 (Wah et al., 2011) for attribute scoring over bird photographs, Super-CLEVR (Li et al., 2023a) for scene-graph execution over rendered scenes, and SAT-Bench (Wei et al., 2025) for bounded satisfiability search over natural-language logic puzzles. For testing out of distribution (OOD) performance, we restrict training on CLUTRR chain lengths and Super-CLEVR scene sizes and evaluate performance on held-out longer chains and larger scenes. SFT receives argument labels, GRPO receives answer rewards, and SPACE receives only answer supervision for each benchmark.

CLUTRR Family stories ask for a relation between two people (Sinha et al., 2019). All methods receive the same public chain skeleton and composition rules. The deduplicated training pool has 12,038 stories with two to four relations; each method trains on 32,768 samples drawn from this pool. The test pool contains 222 stories at those lengths and 826 with five to ten relations; the longer chains are out of distribution (OOD).

CUB-200 Photographs of 200 bird species (Wah et al., 2011) are scored against a fixed species–attribute table of 28 categorical attributes. The comparison budget is 40 epochs over the 5,395 training images. GRPO was not run to this exposure because of training cost. The official test contains 5,794 images of the same species.

Super-CLEVR Synthetically rendered vehicle scenes with visual questions about objects, parts, counts and relations (Li et al., 2023a). Each method trains on 8,192 questions from a pool of 64,000 restricted to scenes with three to six objects and count answers of at most three. Test has 16,042 questions in this range and 10,355 questions out of distribution, on scenes with eight to ten objects and count answers up to ten.

GSM8K Multistep arithmetic word problems (Cobbe et al., 2021), which require natural language reasoning over operation choices and operand references, including references to intermediate results. All methods share 3,396 original training questions, those whose reference calculation compiles to a program of at most ten steps, and train on 8,192 samples from them. All 1,319 official test questions are used for evaluation.

SATBench Natural-language conditions ask whether a satisfying assignment exists (Wei et al., 2025). Each method trains on the first 3,200 questions of a frozen training order, whose condition sets are disjoint from the 2,100-question test pool.

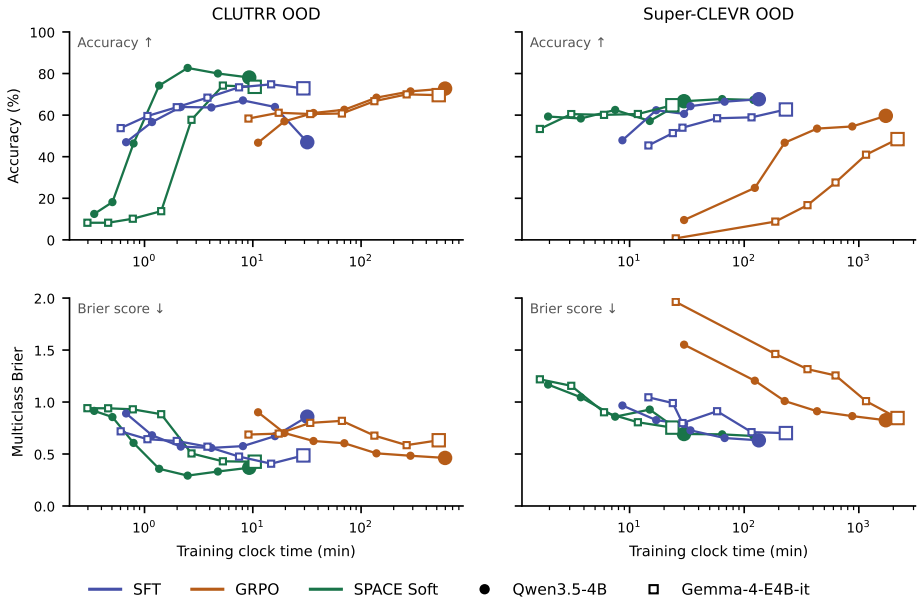

Figure 4: Out-of-distribution (OOD) accuracy (top) and multiclass Brier score (bottom; lower is better) against recorded training clock time on one H100. The SPACE curves use the Soft readout, which runs the compiled program on the argument distributions. Large markers are the equal-data checkpoints: 32,768 samples on CLUTRR and 8,192 on Super-CLEVR. Large SPACE Soft accuracy markers show end-to-end accuracy. Each curve is one fitted seed. Brier intervals are in Table 9. Accuracy by chain length is in Figure 8.

4 Results

The results are largely binary: either SPACE achieves SFT-like training dynamics from answer-only supervision, or it fails to learn entirely. On CLUTRR, CUB-200 and Super-CLEVR, SPACE comes within about one point of SFT's test accuracy or exceeds it without program supervision (Figures 4 and 5). On SATBench and GSM8K, SPACE falls well behind both text baselines. Table 7 reports all scores, with additional experiment controls in Appendix C.1.

SPACE-compatible domains On CLUTRR, CUB-200 and Super-CLEVR, SPACE trains from answers alone yet approaches or exceeds the test accuracy of argument-supervised SFT (Figure 4). On CLUTRR at 32,768 samples, SPACE Soft scores 0.781 OOD on Qwen and 0.738 on Gemma, matching or exceeding SFT and exceeding GRPO by 4–5 points on both backbones. This order holds when checkpoints are selected on in-distribution validation data. The gap shrinks to within the paired intervals upon sampling eight calls and voting over the legal answers, except for SFT on Qwen (Appendix C.1). On Super-CLEVR, SPACE leads SFT on the compositional questions that the compiled program extends beyond the training range, while SFT retains its advantage on part-attribute questions for which it alone has perception labels (Table 19). On Gemma, CUB-200 scores 0.817 Soft and 0.809 Commit at the fixed 40-epoch training budget (Figure 5). On CUB-200, SPACE acts as a species classifier inside the VLM. With a frozen backbone, it scores 1.1–2.3 points above a linear probe on the same features, and a shuffled attribute table lowers accuracy by 2.3–11.4 points (Appendix C.1). With a speaker fitted to each SPACE Soft checkpoint, 99–100% of generated answers with a result token state the program's result on all ten task–backbone pairs, and the generated text's accuracy is within 0.2 points of the program readout on the same questions (Table 20).

Without argument labels, SPACE Soft reaches SFT-level or lower Brier scores on CLUTRR and CUB-200; SFT scores lower on Super-CLEVR (Figure 4). The text baselines reach comparable calibration only after sampling eight calls or fitting a temperature on development data (Appendix C.1), or after

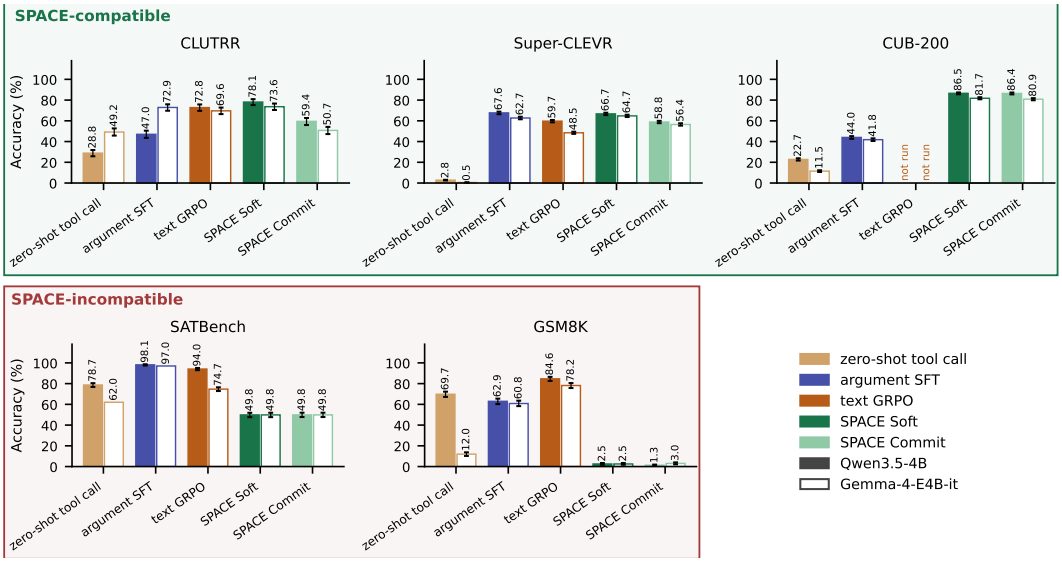

Figure 5: Test accuracy by method on each dataset. SPACE results show full end-to-end accuracy. Error bars are 95% bootstrap intervals for one fitted seed. Every data configuration is specified in Table 7.

zeroing confidence on failed calls (Appendix D.1). SPACE Soft has lower Brier point estimates than GRPO on CLUTRR and Super-CLEVR, although some paired intervals include zero (Appendix C.1).

For ambiguous inputs such as photographs or complicated family stories, the Soft readout matches or outperforms Commit. On CLUTRR, committing drops OOD accuracy from 0.781 to 0.594 on Qwen, with the gap widening as chain length increases (Figure 8); most of these Commit errors are edge labelings with no legal composition, and a legal-MAP decoder, which runs the program on the single most probable legal labeling, matches Soft's accuracy (Table 2). On Super-CLEVR, the OOD gain is 7.75 points by using Soft. On CUB-200, answer-only training moves some of the adapter's visual attention onto the relevant parts of each photograph without any spatial supervision (Figure 11), and on Gemma Soft outperforms Commit in accuracy and has a slightly lower Brier score.

SPACE-incompatible domains On SATBench, the adapter's top candidate readings of the puzzle all give the same verdict, so the task loss has zero gradient in the recorded runs. SPACE Soft scores 0.498 on Qwen and 0.498 on Gemma; Commit has the same chance accuracy on both backbones. This falls below both argument SFT at 0.981 and text GRPO at 0.940. The zero shot tool call baseline already answers 0.787 on the test pool. The calibration benefit does not transfer where learning fails, and Soft's Brier score is about 1.05 on SATBench and GSM8K for both backbones, far above the text baselines.

GSM8K presents a different failure mode: the adapter must not only read the numbers from the text but also recover the order in which arithmetic operations are applied and which operands they reference. The answer is consistent with many operation wirings at once, and the analytic gradient cannot single out the correct sequence from the many that happen to produce the same result. SPACE Soft collapses to 0.025 on Qwen, far below SFT at 0.629, GRPO at 0.846 and a zero-shot tool call with thinking at 0.864 (Table 13).

Training and inference cost SPACE trains in a single forward and backward pass per question, while GRPO samples eight complete calls before every update, so SPACE reaches GRPO's final OOD accuracy more than 50 times faster on both CLUTRR and Super-CLEVR (Figure 4). On CLUTRR, SPACE reaches the equal-data checkpoint in about ten minutes while GRPO takes about nine hours, and on Super-CLEVR, about thirty minutes versus 29 to 36 hours (Table 5). On CUB-200, GRPO was not feasible within the 40-epoch training budget that SFT and SPACE required, so we compare it only at equal clock time with the earlier frozen-backbone SPACE reader, the time that reader needed

for its full run, where its Brier score is 0.90/1.42 against 0.26/0.38 for that reader's Soft readout. On CLUTRR, both Soft and Commit score the full test set in 28 seconds, compared with roughly three minutes for the text methods. All clock measurements are on one H100 and exclude evaluation and checkpoint writing (Appendix B.2).

5 Discussion

Generalization and in-distribution accuracy SPACE keeps the advantage of GRPO (learning without argument demonstrations) without its compute cost, and the compiled program supplies the composition beyond the training range. SFT leads in distribution on most tasks, because it trains on argument labels that SPACE never sees, but its accuracy falls off outside the training range on the questions the program composes.

Gradient quality and credit assignment The analytic gradient carries more information per answer than a sampled reward. Because the program carries every argument distribution through to the answer, the answer loss returns a separate signal to each argument in proportion to how much the answer depends on it. When the reader is unsure between two readings, the gradient moves probability toward the reading that agrees with the answer. A sampled method instead draws one complete assignment, observes a single reward, and must infer from many such draws which argument was at fault. We isolate this effect by training the same adapter with sampled rewards: on CLUTRR it reaches about 13% long-chain accuracy against 79–86% for exact training (Appendix E), and on CUB-200 binary reward stays near 1% while a shaped reward reaches 62% with 24 times more program executions, against 69% for exact training (Appendix F). This is credit for agreeing with the answer, not supervision of the correct reading. It is only as sharp as the dependence of the answer on each argument, so when many assignments produce the same answer, the gradient spreads over all of them, and it also credits coincidental assignments that produce the right value. Where the answer does not identify an argument, the adapter can predict argument values that are not literally true of the input but still lead the fixed program to the right answer (Appendix F).

Calibration For a true answer marginal, SPACE Soft makes the Bayes decision under the adapter's specified distribution, and its confidence is that answer's probability under the model. A typical language model performs the linguistic reading and the numerical or logical computation in the same network, and its output probability mixes uncertainty about what the text says with error in how the computation was carried out. Training on answers sharpens that output as a whole, so the confidence it reports no longer tracks either source. SPACE separates the computation and linguistic networks. Uncertainty enters only at reading the input into program arguments, and the fixed computation propagates that uncertainty under the semantics in Table 1. This calibration is conditional on argument beliefs that match the input, and a confidently wrong reading is propagated as faithfully as a correct one. Answer-trained CLUTRR edge beliefs mostly match the reference relations, but mean argument entropy separates Soft's right and wrong answers only weakly (Table 3). Where answer-only training does not learn the readings, SPACE has no calibration benefit (Table 7). SPACE trains on the log loss of the answer, which is a proper scoring rule for the answer distribution. That loss rewards reading distributions whose computed answer probabilities are calibrated, though not each reading distribution on its own, since different readings that produce the same answer receive the same loss. SFT gets the same kind of signal only from argument labels. GRPO maximizes expected reward, which a deterministic policy maximizes, so nothing in its objective keeps sampled answers spread in proportion to uncertainty. A known program is what lets answer-only training produce SFT-level calibration. Commit discards the reading uncertainty, so a wrong one-hot answer receives the full Brier penalty.

Learning through a model of the domain The compiled program plays a role analogous to a known simulator in model-based reinforcement learning. Learning through its computation can replace sampled credit with derivatives, but someone must supply the domain model. SPACE requires these domain priors by design. On CLUTRR, the program knows that every answer is a legal relation, so it renormalizes over legal readings, as constrained decoding does, rather than scoring illegal readings as a failure answer that is never correct. At 2,048 training questions, removing Super-CLEVR's lexical priors drops OOD accuracy from 62% to 11% (Appendix C.1). Standard tool use trades this structured gradient for a common text interface to many tools. SPACE's style of execution therefore

provides a more specialized alternative to broad tool use in exchange for more favorable training dynamics.

5.1 Limitations

The reading adapter is the central bottleneck, and its difficulty depends on how much of the program's structure the input defines. Local readings such as a relationship in CLUTRR or visual attributes in CUB are straightforward. Super-CLEVR lies in between: the adapter must compose intermediate values, but it recovers the composition from answers because each argument is used only once. GSM8K requires the adapter to recover the structure of the arguments itself, and SATBench supplies no task-loss gradient in the recorded runs, because its top candidate readings agree on the verdict and make the answer mixture locally constant. Each new tool also has an engineering cost: we write its tensor kernels by hand and compile only their wiring into the network, and tool selection or composition is outside this study.

Even where the program is tractable in principle, maintaining the factored belief state may not be feasible in practice. The belief is factored across arguments, so it cannot represent correlated ambiguity, such as two readings that are plausible only together. On GSM8K and SATBench the state space is too large for exact enumeration, so the program retains only the top-K most probable values at each step, an approximation whose gradient is not the exact derivative of the full marginal. The failures on those two datasets are therefore confounded, the poor performance may stem from the approximate gradient, from the adapter's inability to recover the program structure, from the sparse training signal, or from some combination of all three.

We make a best effort to isolate the sample efficiency and compute efficiency of SPACE with known limitations. We compare one seed of each SFT and GRPO baseline against three SPACE seeds. The methods use different learning-rate schedules, so Appendix C.1 also reports checkpoints selected on in-distribution validation data, with the other experiment controls.

5.2 Future work

A potential next step is to read more structured arguments, potentially using the language model's chain of thought to expose the operation order that a parallel adapter misses. Sparse verdicts such as SATBench also motivate training signals that identify useful clause-level changes. For compositional tasks such as GSM8K, this approach may require not just a compiled neural program, but a neural virtual machine, which can search for the operation sequence given a belief state.

6 Conclusion

Maintaining belief distributions improves inference over commitment on several tasks, and analytic gradients outperform sampled argument credit in local-reading tasks. At equal data exposure, SPACE generalizes to longer argument chains better than argument SFT and trains faster than GRPO on both tested VLM backbones. The benefit is conditional on tractable uncertainty computation and learnable arguments. SPACE is practical where readings are both local and ambiguous, and the input supplies the composition. Standard LLM training is more effective where the model must recover the program structure itself or the answer carries too little information to credit each argument. These results demonstrate where internal symbolic computation can help and where a more capable language-to-program interface is still needed.

AI use statement

We used generative AI tools as coding and writing assistants throughout this work. AI assistants helped write and refactor experiment, training and analysis code, and helped draft and edit the text of this paper. We did not use generative AI tools to generate research ideas, to select or interpret results, or to produce any reported number; every figure, table and macro in this paper is generated from recorded run artifacts by the scripts released with the code. All AI-assisted code was reviewed by the authors and is covered by the behavioral tests described in the reproducibility statement, and

all AI-assisted text was read and revised by the authors. We take responsibility for the final content of this work, including its text, claims and artifacts.

Ethics statement

This work uses only public benchmarks (Sinha et al., 2019; Cobbe et al., 2021; Wah et al., 2011; Li et al., 2023a; Wei et al., 2025) under their released licenses, and it involves no human subjects, no private data and no new data release. CUB-200 contains photographs of birds, not people. The method reports a calibrated confidence with each answer, which is intended to make a wrong answer easier to detect, but that calibration is conditional on argument beliefs that match the input, so a confidently misread input is reported as confidently as a correct one; this limit is stated in Section 5. We see no other ethical concerns specific to this work.

Reproducibility statement

The released code includes the compiler, both readouts, behavioral tests, the timestamped configuration of every reported run and its per-example scored predictions, from which one script recomputes every table cell (Appendix B.4). All metrics retain the declared evaluation pools, and paired summaries consume saved correctness flags. Appendix B.4 records training recipes; Appendices E and I record training-overlap checks.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Yuval Atzmon and Gal Chechik. Probabilistic AND-OR attribute grouping for zero-shot learning. In *Conference on Uncertainty in Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1806.02664.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, pp. 1533–1544, 2013.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023. URL https://arxiv.org/abs/2211.12588.

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, 2021. URL https://arxiv.org/abs/2110.14168. arXiv:2110.14168.

Timo Dimitriadis, Tilmann Gneiting, and Alexander I. Jordan. Stable reliability diagrams for probabilistic classifiers. *Proceedings of the National Academy of Sciences*, 118(8):e2016191118, 2021.

Owen Dugan, Donato M. Jiménez-Benetó, Charlotte Loh, Zhuo Chen, Rumen Dangovski, and Marin Soljačić. OccamLLM: Fast and exact language model arithmetic in a single step. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.06576.

Christopher A. T. Ferro. Fair scores for ensemble forecasts. *Quarterly Journal of the Royal Meteorological Society*, 140(683):1917–1923, 2014. doi: 10.1002/qj.2270.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2211.10435.

Alexander L. Gaunt, Marc Brockschmidt, Rishabh Singh, Nate Kushman, Pushmeet Kohli, Jonathan Taylor, and Daniel Tarlow. TerpreT: A probabilistic programming language for program induction, 2016. URL https://arxiv.org/abs/1608.04428.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770. arXiv:2607.02770.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines, 2014. URL https://arxiv.org/abs/1410.5401. arXiv:1410.5401.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Gupta_Visual_Programming_Compositional_Visual_Reasoning_Without_Training_CVPR_2023_paper.html.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of ACL*, pp. 1051–1062, 2017.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8fd1a81c882cd45f64958da6284f4a3f-Abstract-Conference.html.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5338–5348, 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014. URL https://hannes.nickisch.org/papers/articles/lampert13attributes.pdf.

Zhuowan Li, Xingrui Wang, Elias Stengel-Eskin, Adam Kortylewski, Wufei Ma, Benjamin Van Durme, and Alan Yuille. Super-CLEVR: A virtual benchmark to diagnose domain robustness in visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14963–14973, 2023a.

Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. In *Proceedings of the ACM on Programming Languages (PLDI)*, 2023b.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled Transformers as a Laboratory for Interpretability, November 2023. URL http://arxiv.org/abs/2301.05062. arXiv:2301.05062 [cs].

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Shengyao Lu, Bang Liu, Keith G. Mills, Shangling Jui, and Di Niu. R5: Rule discovery with reinforced and recurrent relational reasoning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=2eXhNpHeW6E.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models, 2021. URL https://arxiv.org/abs/2106.13314.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2019.

Emanuele Marconato, Stefano Teso, Antonio Vergari, and Andrea Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/e560202b6e779a82478edb46c6f8f4dd-Abstract-Conference.html.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended?, 2021. URL https://arxiv.org/abs/2105.04289. ICLR Workshop on Responsible AI.

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

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6938–6949, 2020. URL https://proceedings.mlr.press/v119/minervini20a.html.

Allan H. Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973.

Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5):64–79, 1972.

Qwen Team. Qwen3.5-4B. https://huggingface.co/Qwen/Qwen3.5-4B, 2026. Model card.

Lucas Saldyt and Subbarao Kambhampati. Augmenting language models with composable differentiable libraries, 2024. URL https://arxiv.org/abs/2407.04899. Related arXiv version titled Algorithmic Language Models with Neurally Compiled Libraries, arXiv:2407.04899.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL http://arxiv.org/abs/2402.03300. arXiv:2402.03300.

Peter Shaw, James Cohan, Jacob Eisenstein, Kenton Lee, Jonathan Berant, and Kristina Toutanova. ALTA: Compiler-Based Analysis of Transformers, June 2025. URL http://arxiv.org/abs/2410.18077. arXiv:2410.18077 [cs].

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4506–4515, 2019.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006 of *Proceedings of SPIE*, 2019.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11888–11898, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Suris_ViperGPT_Visual_Inference_via_Python_Execution_for_Reasoning_ICCV_2023_paper.html.

Emile van Krieken, Thiviyan Thanapalasingam, Jakub Tomczak, Frank van Harmelen, and Annette ten Teije. A-NeSI: A scalable approximate method for probabilistic neurosymbolic inference. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tomás Vergara-Browne and Álvaro Soto. Tracr-Injection: Distilling algorithms into pre-trained language models, 2025. URL https://arxiv.org/abs/2505.10719. arXiv:2505.10719.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. SATBench: Benchmarking LLMs' logical reasoning via automated puzzle generation from SAT formulas. *arXiv preprint arXiv:2505.14615*, 2025.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking Like Transformers, July 2021. URL http://arxiv.org/abs/2106.06981. arXiv:2106.06981 [cs].

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Kang Liu, and Jun Zhao. Mastering Symbolic Operations: Augmenting Language Models with Compiled Neural Networks, March 2024. URL http://arxiv.org/abs/2304.01665. arXiv:2304.01665 [cs].

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5502–5511, 2018. URL https://proceedings.mlr.press/v80/xu18h.html.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming. In *Findings of the Association for Computational Linguistics: ACL*, 2023. URL https://arxiv.org/abs/2305.03742.

Wentao Zhang, Liliana Hotsko, Woojeong Kim, Pengyu Nie, Stuart Shieber, and Yuntian Deng. Program-as-Weights: A Programming Paradigm for Fuzzy Functions, 2026. URL https://arxiv.org/abs/2607.02512. arXiv:2607.02512.

A Method details

A.1 Program semantics and training

The multilinear extension There are two ways to evaluate a program under an uncertain input. One can evaluate it at the mean input, or one can evaluate it in every world and average the answers. The mean input is usually a world that does not exist, since the average of a reading of 80 and a reading of 300 is a 190 that neither reading contains. Averaging answer indicators gives the exact distribution under the specified belief; its accuracy still depends on that belief. Every function on finite alphabets has exactly one extension that is affine in each input's belief and agrees with the function on one-hot inputs, its multilinear extension (Owen, 1972),

$$F(q_1, \ldots, q_n) = \sum_{a \in \mathcal{A}} \prod_{i=1}^{n} q_i(a_i)\ f(a). \tag{2}$$

Applied to the indicator of each answer, this is the answer marginal of Equation 1. For a compiled program that is affine in each original input belief, uniqueness makes agreement on one-hot inputs sufficient to certify agreement on every belief. One-hot agreement alone does not establish that premise: multiplying dependent intermediate marginals can pass every corner test and still assign probability to impossible outcomes, as in forward-marginal approximations (Gaunt et al., 2016). The extension is cheap when the program is a tree of lookups and sums, because each input is used once, the extension factorizes gate by gate, and the belief propagates at one contraction per gate. When an intermediate value is used twice, its belief has to be carried as a distribution and everything downstream has to be conditioned on it.

A chart of legal relation compositions The chain composition of CLUTRR carries a distribution over relation states for every span of the chain and fills them in by a dynamic program over bracketings, a polynomial-time chart rather than an enumeration of all edge labelings and trees. With the latent bracketing t, the chart computes

$$W_y(q) = \sum_{a,t} \mathbf{1}\{(a,t) \text{ is legal},\ f(a,t) = y\} \prod_i q_i(a_i), \qquad P_y(q) = \frac{W_y(q)}{\sum_{y'} W_{y'}(q)}. \tag{3}$$

For a positive denominator, each legal derivation contributes once, so labelings with more legal derivations can receive more weight. The numerator is multilinear; the normalized answer distribution generally is not. Commit replaces each edge belief with a one-hot vector at its most probable relation and retains the same sum over bracketings. Figure 7 draws the tensor contractions for one story. Table 1 lists what each program computes.

Table 1: What the Soft readout computes on each dataset.

Dataset	Soft computes	Exact Eq. 1?	Approximation
CLUTRR	Chart over legal bracketings	No	Legal-derivation weighting and normalization
CUB-200	softmax($s\,\overline{S}$)	No	Expected-score relaxation of the argmax tool
Super-CLEVR	Scene executor for each program	Yes, for each program	Top-8 programs; pruned and invalid mass not renormalized
GSM8K	Register value distributions	No	Top-K values per step; pruned mass dropped, failure kept
SATBench	DPLL for each formula	Yes, for each formula	Top-32 formulas; normalized within the beam

The Bayes decision and the disagreement set Write $\hat{a} = \arg\max_a q(a \mid x)$ for the committed input assignment, so that $f(\hat{a})$ is the Commit answer, and $y_{\mathrm{m}} = \arg\max_y P(y \mid x)$ for the Soft answer. We call the set of inputs x on which the two answers differ, $\{x : f(\hat{a}(x)) \neq y_{\mathrm{m}}(x)\}$, the disagreement set.

Remark 1 (Bayes decision)**.** *This is the standard Bayes decision under 0-1 loss. If the true assignment is distributed as $q(\cdot \mid x)$, then $y_{\mathrm{m}}(x)$ minimizes the expected 0-1 loss on the answer, and $f(\hat{a})$ minimizes it only when $\hat{a}$'s answer is also the most probable answer. If the marginal mode is unique, the readouts differ exactly when some other answer collects more belief than $f(\hat{a})$. Ties may produce different answers with equal risk.*

The proof in Appendix A.4 includes the tie case. The disagreement set is computed from the adapter's outputs alone, without a label, and outside it the two readouts have identical accuracy, so the entire

h_ℓ	hidden states at read layer ℓ of L	F	compiled program, fixed (Equation 1)
q_i	belief over argument i	P	answer distribution $P(\cdot \mid x)$
A_ϕ	argument adapter	θ	reader LoRA, layers 1 to ℓ
B_ψ	result adapter, writes result token r	ω	speaker LoRA, layers $\ell+1$ to L

Algorithm 1 SPACE training

Stage 1: read arguments from answers
1: **for** each prompt and answer $(x, y^\star)$ **do**
2: $h_\ell \leftarrow \text{Layers}_{1:\ell}(x; \theta)$
3: $q_i \leftarrow \text{softmax}\big(A_\phi(h_\ell)_i\big)$ for each i
4: $P \leftarrow F(q_1, \ldots, q_n)$
5: update ϕ, θ with $\nabla\big(-\log P(y^\star \mid x)\big)$

Stage 2: verbalize the result, ϕ, θ fixed
6: **for** each prompt and response **do**
7: h_ℓ, P as in lines 2–4
8: $r \leftarrow B_\psi(P)$, written into layer ℓ
9: update ψ, ω with the next-token loss

Algorithm 2 SPACE inference

Require: prompt x; readout Soft or Commit
1: $h_\ell \leftarrow \text{Layers}_{1:\ell}(x; \theta)$
2: $q_i \leftarrow \text{softmax}\big(A_\phi(h_\ell)_i\big)$ for each i
3: **if** readout is Commit **then**
4: $q_i \leftarrow \text{onehot}(\arg\max_v q_i(v))$ for each i
5: $P \leftarrow F(q_1, \ldots, q_n)$
6: $r \leftarrow B_\psi(P)$, written into layer ℓ
7: text $\leftarrow \text{Head}\big(\text{Layers}_{\ell+1:L}([h_\ell; r]; \omega)\big)$
8: **return** $\hat{y} = \arg\max P$, $P(\hat{y})$, text

effect of marginalizing is its wins minus its losses inside that set. The remark describes the rows of Table 1 that compute Equation 1 exactly; elsewhere the Soft answer is the mode of the computed distribution.

Readout controls Commit replaces each argument with its most probable value independently, so its tuple can have no legal execution. Table 2 separates that effect from marginalization. Legal MAP runs the program on the most probable legal assignment: on CLUTRR it is the same chart with max in place of sum, and on Super-CLEVR it is the most probable legal assignment among 128 samples, a lower bound on the exact legal MAP; on CUB-200 every tuple is legal, so it equals Commit. Vote@N draws N assignments from the beliefs, runs the discrete program on each and returns the most frequent answer. On CLUTRR, most Commit errors have no legal composition, and legal MAP matches Soft; on Super-CLEVR, Soft stays ahead of legal MAP and of voting.

Table 2: Accuracy of readouts of the same adapter beliefs (CLUTRR lengths 5–10, large Super-CLEVR scenes, frozen-backbone CUB-200 test). Illegal is the share of Commit errors with no legal composition (CLUTRR) or the share of questions whose committed scene fails a uniqueness condition (Super-CLEVR). Soft and Commit are recomputed from re-exported beliefs; on CLUTRR Qwen, 8 of 2,096 answers differ from the stored predictions behind Table 7. Paired intervals are in the diagnostics report.

Dataset	Backbone	Soft	Commit	Legal MAP	Vote@1	Vote@128	Illegal
CLUTRR	Qwen	0.780	0.598	0.781	0.567	0.648	0.765
CLUTRR	Gemma	0.738	0.507	0.749	0.465	0.591	0.735
CUB-200	Qwen	0.860	0.856	= Commit	0.847	0.860	—
CUB-200	Gemma	0.757	0.734	= Commit	0.701	0.757	—
Super-CLEVR	Qwen	0.666	0.588	0.600	0.497	0.575	0.153
Super-CLEVR	Gemma	0.645	0.564	0.587	0.469	0.539	0.164

The analytic gradient is credit per argument With answer supervision $y^\star$ the loss is $-\log P(y^\star \mid x)$, the negative marginal likelihood that weakly supervised semantic parsers maximize over latent programs (Berant et al., 2013; Guu et al., 2017). Because the extension is affine in each argument belief q_j, its derivative with respect to $q_j(v)$ is the program run with argument j clamped to the value v and every other argument left as a belief,

$$\frac{\partial F}{\partial q_j(v)} = \sum_{a:\, a_j = v} \prod_{i \neq j} q_i(a_i)\; f(a) = F(q_1, \ldots, e_v, \ldots, q_n). \tag{4}$$

Here e_v is the one-hot vector at v. This is the derivative of F, not of its negative log-likelihood. For an answer indicator, the log-loss derivative additionally divides by the answer probability. For the

conditioned chart in Equation 3, training differentiates both the answer weight and the normalizer. These derivatives avoid Monte Carlo sampling, but need not be nonzero or informative for every input. Under the argmax readout the answer is constant on each region of the simplex where the argmax is fixed, so differentiation through the discrete answer supplies no useful learning signal almost everywhere. Training can instead use a surrogate or sampled credit. The surrogate can be a straight-through estimator, which commits in the forward pass and pretends it did not in the backward pass (Bengio et al., 2013; Jang et al., 2017), or the score-function estimator (Williams, 1992), which samples whole assignments, rewards the ones whose answer is right and raises their probability, as GRPO does (Shao et al., 2024). With a group-centered baseline, a group in which every sample earns the same reward contributes zero score-function reward credit; entropy or KL regularization can still update the policy. If a fraction p of assignments is rewarded and a group has K independent samples, that happens with probability

$$p^K + (1-p)^K. \tag{5}$$

Once the adapter has become deterministic every group is silent. Appendix E measures this on CLUTRR and Appendix F on CUB-200.

Argument labels Answer log loss is proper for the answer distribution, not for each argument belief, so the credit above rewards readings that agree with the answer rather than the correct reading. Table 3 scores the argument beliefs of the answer-trained adapter against reference argument labels, which are used here for evaluation only. CLUTRR edge beliefs mostly match the reference relations, where the chain composition constrains each edge; frozen-backbone CUB-200 attribute beliefs are no more accurate than predicting each category's most frequent value, consistent with attributes acting as a code for species (Appendix F). Mean argument entropy predicts wrong answers only weakly on every dataset.

Table 3: Argument beliefs against reference argument labels, used for evaluation only: CLUTRR edge relations and frozen-backbone CUB-200 image attributes (partial labels retained). Majority predicts each category's most frequent training value. AUROC is how well mean argument entropy ranks wrong Soft answers above right ones (0.5 is chance), with a 95% interval.

Dataset	Backbone	Split	SPACE accuracy	SFT accuracy	Majority	Entropy AUROC
CLUTRR	Qwen	2–4	0.928	0.969	—	0.582 [0.457, 0.709]
CLUTRR	Qwen	5–10	0.904	0.910	—	0.642 [0.596, 0.688]
CLUTRR	Gemma	2–4	0.892	0.974	—	0.768 [0.658, 0.860]
CLUTRR	Gemma	5–10	0.869	0.891	—	0.636 [0.593, 0.680]
CUB-200	Qwen	Test	0.364	0.511	0.376	0.672 [0.653, 0.691]
CUB-200	Gemma	Test	0.376	0.487	0.376	0.627 [0.611, 0.644]
Super-CLEVR	Qwen	Large	—	—	—	0.548 [0.529, 0.566]
Super-CLEVR	Gemma	Large	—	—	—	0.562 [0.544, 0.578]

A.2 Compiling neural programs

Source language and compilation boundary Our compiler accepts human-readable schemas describing ordered categorical domains, public knowledge tables, composition rules, supported program skeletons, and register or search bounds. Following RASP's restricted programming primitives (Weiss et al., 2021), Tracr's named categorical representations (Lindner et al., 2023), ALTA's explicit transition rules (Shaw et al., 2025), and CoNN's compiled neural networks (Weng et al., 2024), these declarations are lowered into fixed tensors and bound to a library of domain-specific tensor execution kernels. Schema parsing and table construction occur before inference; execution consumes adapter-produced tensors inside the model's forward pass. Compilation uses no task examples, answer labels, or parameter fitting, and the learned adapters and LoRAs remain separate from it.

Finite functions and rules For a finite function $f : A_1 \times \cdots \times A_r \to B$, the compiler assigns an index to each declared domain element and constructs

$$T_{i_1,\ldots,i_r,j} = \mathbf{1}\{f(a_{1,i_1}, \ldots, a_{r,i_r}) = b_j\}.$$

Contracting this tensor with independent input distributions computes the function's multilinear extension. Functions can be supplied directly for compile-time enumeration or as sparse rules with an explicit default outcome. Domain order is preserved because it defines the interface to the learned

adapters. Exact local contractions alone do not guarantee an exact program marginal: the executor must also preserve dependencies between repeated uses of an argument.

Domain-specific lowering For CUB-200, compilation groups the public class–attribute table into categorical score tensors and padding masks. For CLUTRR, abstract family rules expand into gendered relation-composition tensors, direction mappings and an explicit invalid outcome, which the chart recurrence executes. For Super-CLEVR, the compiler type-checks the supported program skeletons and constructs filter selectors, attribute and answer indices, part mappings and slot masks for the scene-belief kernels. GSM8K uses a bounded arithmetic register schema: compilation emits operation codes, register indices and integer decimal-scaling tables for mantissa–exponent kernels, while number extraction and the constant-bank reader remain in the adapter. SATBench specifies bounded CNF and search registers; compilation emits register indices and output offsets for the tensor DPLL transitions of propagation, branching and backtracking. Compilation does not make an approximate executor exact: the GSM8K top-K register marginal and the SATBench formula mixture remain approximations (Appendices H and I).

Verification Artifacts store the schema and emitted tensors and reconstruct parameter-free solver modules. Equivalence checks compare against reference tensors and outputs, including one-hot and uncertain inputs, differentiable-interface gradients, invalid executions, search-budget exhaustion and serialization.

A.3 Result tokens

The compiled program returns its result to the language model through a learned result token. We reserve a token position after the public prompt and replace its residual state at the adapter layer with a learned projection of the computed result. The remaining layers attend to that state and generate through the original vocabulary head. The placeholder retains an ordinary token ID, including for Gemma's per-layer embeddings. Cached generation injects the result during prefill and then continues with the model's native decoder. The adapter and program receive only the prefix before the reserved position, including during teacher-forced response scoring. Reference answers never supply the result-token features.

Gemma's shared-key/value layers require an additional memory write when their key/value providers precede the read layer. We project the same result token with the frozen provider projections and its rotary position, and replace only the reserved slot in the shared attention memory. This restores a gradient path from future text tokens to the result bridge without changing the earlier adapter. Cached decoding retains that slot subject to the native sliding-window mask.

CLUTRR supplies its relation law, SAT its verdict mixture, and CUB its species-score distribution. GSM8K supplies fixed decimal-place features of the computed numeric answer. Super-CLEVR supplies its answer distribution from program and part adapters, using public predicted listings and frozen crop features, with nonzero answer mass normalized to sum to one before projection into the result token. Appendix J gives the readback training and its evaluation.

A.4 Proof of Remark 1

Let $A \sim q(\cdot \mid x)$ and $Y = f(A)$. For any answer y, $\Pr[Y = y] = \sum_{a:f(a)=y} q(a \mid x) = P(y \mid x)$. The expected 0-1 loss of answering y is $1 - P(y \mid x)$, minimized by $y_{\mathrm{m}} = \arg\max_y P(y \mid x)$. The argmax readout answers $f(\hat{a})$, whose loss is $1 - P(f(\hat{a}) \mid x)$; this equals the minimum exactly when $P(f(\hat{a}) \mid x) = \max_y P(y \mid x)$, and is larger otherwise. With a unique marginal mode, different readouts imply that some $y \neq f(\hat{a})$ has $P(y \mid x) > P(f(\hat{a}) \mid x)$. With tied maxima, the readouts can disagree at equal risk. If f is injective on the support of q, commitment is also optimal; agreement then requires compatible tie-breaking. Thus a strict risk improvement requires that several assignments share an answer. □

B Experimental protocol

B.1 Text baselines

Argument SFT and text GRPO call the same compiled program as SPACE. The tool schema is passed through the backbone's own chat template and the reply is the model's own tool-call block. Every tool takes values, never indices (Table 4). Thinking is off in training and evaluation. The model writes exactly one call and the compiled result of that call is the answer; a missing, second, malformed or truncated call and a failed execution count as errors, and nothing is repaired. SFT trains on the reference arguments written as that call. GRPO samples eight calls per question and rewards the calls whose compiled answer is correct; an invalid call receives no reward. Each dataset keeps the base model, LoRA, optimizer recipe, seed, question order and pool of its SPACE comparison. On CLUTRR, argument SFT uses AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-4} and batch 16, and GRPO uses learning rate 10^{-5} with eight questions per update. The CLUTRR GRPO recipe sets its learning rate from elapsed time over a schedule cap; because native-format updates take about twice as long, the cap is 12 hours instead of 6, and warm-up stays 432 seconds.

Table 4: Tool calls of the text baselines and of the zero-shot floor.

Dataset	Tool and argument	Generation cap
CLUTRR	`compose_relations`: one relation name per edge of the public chain	128
CUB-200	`bird_attributes`: a named value for each of the 28 attribute categories	384
Super-CLEVR	`scene_program`: a scene program and optional part queries	1,024
GSM8K	`calculate`: one arithmetic expression over the literal numbers	256
SATBench	`cnf`: one clause per condition, with named literals	4,096

B.2 Budgets

Equal data A training sample is one question shown to the model once, so repeated questions count each time; GRPO's eight generated rollouts are additional computation. Within a dataset and backbone, the comparison holds the public input, base model, eligible training IDs and sample count fixed, and every method uses the same seeded question order. SFT's argument labels are extra supervision. The iso-data counts are 32,768 samples on CLUTRR, 8,192 on GSM8K and Super-CLEVR, 3,200 on SATBench, and 40 epochs (215,800 samples) on CUB-200. Every method reports the checkpoint at exactly that count; no checkpoint is chosen by a score. Paired Soft/Commit readouts share every training update by construction.

Equal clock Each iso-clock mark is the full SPACE training clock of that cohort at its iso-data count, so every method is given the budget SPACE actually used: CLUTRR 556.12 seconds on Qwen and 625.02 on Gemma; GSM8K 200.22 and 225.85; Super-CLEVR 1,789.83 and 1,417.62; CUB-200 272.50 and 385.46, from the earlier frozen-backbone reader. The clock is initialization plus all training work, CUDA-synchronized, including GRPO rollout generation; evaluation and checkpoint writing are recorded separately. Both CUB marks include training-split feature extraction, which SPACE must pay before its head sees an image. Each method saves the first completed update crossing its mark and records the actual time, then continues to the iso-data count. No run replays a prefix. SATBench is compared at equal data only. Table 5 gives the full training clock of each method at the iso-data count. All clocks are on one H100. Equal clocks do not establish equal FLOPs, and the clocks do not isolate an executor speedup: prompt lengths, batches, backpropagation paths and rollouts all differ.

B.3 Evaluation

Pools Accuracy retains every declared test example, with invalid calls, failed executions and unsupported inputs counted as errors. On CLUTRR, ID is relation chains of length 2 to 4 and Test is chains of length 5 to 10; on Super-CLEVR, ID is scenes with 3 to 6 objects and count answers at most three, and Test is scenes with 8 to 10 objects and count answers up to ten. On these two datasets

Table 5: Full training clock in minutes at the iso-data count, excluding evaluation and checkpoint writing, on one H100. CUB-200 GRPO stops at the iso-clock mark of the frozen-backbone reader and has no iso-data run.

Dataset	Backbone	SPACE	Argument SFT	Text GRPO
CLUTRR	Qwen	9.27	31.84	596.67
CLUTRR	Gemma	10.42	29.32	521.40
GSM8K	Qwen	3.34	6.80	451.48
GSM8K	Gemma	3.76	5.96	349.95
Super-CLEVR	Qwen	29.83	134.15	1,719.32
Super-CLEVR	Gemma	25.54	231.01	2,176.04
CUB-200	Qwen	99.20	981.04	—
CUB-200	Gemma	166.03	1,980.11	—

the Test pool is out of distribution (OOD). CUB-200, GSM8K and SATBench have no established distribution shift, so their ID pool is the held-out development pool (599 images for CUB-200, 128 questions for GSM8K and SATBench). Its accuracy is therefore an in-distribution reading rather than a clean held-out score.

Intervals Each method and backbone has one training seed. Paired bootstrap intervals resample stories for CLUTRR, images for CUB, scenes for Super-CLEVR and questions for GSM8K and SATBench. They condition on the fitted models and do not measure training-seed variation.

Inference batch Greedy decoding and confidence move with padding and grouping, so each cohort fixes one evaluation batch for all of its evaluations. An evaluation that replays the training run's development predictions must reproduce them exactly.

B.4 Configurations

Table 6 records the SPACE argument adapters and their read locations. Reader LoRAs use rank 16 and alpha 32 below the read layer on every dataset. On CUB-200, these LoRAs train together with the argument adapter and score scale. Result-token readback trains a separate bridge and upper-layer speaker LoRAs with the argument adapter fixed (Appendix J).

Choice of read layer A frozen SAT extraction probe selected Gemma layer 17 of 42 by in-distribution held-out accuracy, without using the longer-condition results for selection. Arithmetic schema probes at layers 14, 16, 18 and 20 differed by at most 0.04, supporting the layer-18 choice. These probes used structural labels to inform layer selection only; the reported SPACE adapters and reader LoRAs are trained from answers only. On CUB-200 and Super-CLEVR, the read layer keeps the same depth fraction across backbones (Qwen 14 of 32 matches Gemma 18 of 42 on CUB-200; Gemma 24 of 42 matches Qwen 18 of 32 on Super-CLEVR) and is not chosen by any score. CLUTRR and GSM8K use layer 18 on both backbones.

Code, configurations and records The released code repository has one configuration file for each data build, training run and evaluation, and its path names the run. `configs/<dataset>/data/` holds the data builds. `configs/<dataset>/<backbone>/train/` holds `space.json`, `sft.json` and `grpo.json`, and `configs/<dataset>/<backbone>/eval/` holds the iso-data evaluations (`space.json`, `sft.json`, `grpo.json`), the iso-clock evaluations (`sft_clock.json`, `grpo_clock.json`), the zero-shot evaluations and the training-curve checkpoints (`curve/`). `configs/calibration/` holds the Brier-score evaluations. `records/runs.md` lists the training configuration, evaluation configuration and records of every reported run, `records/<dataset>/<backbone>/<run>/` holds the per-example scored predictions of each evaluation, and `scripts/results/make_table.py` recomputes every cell of Tables 7 and 13 from these records alone. Checkpoint weights and built datasets are not in the repository; its README gives the steps that rebuild them.

Table 6: SPACE argument adapters, read layers and training recipes.

Task	Read layer	Adapter and training
CLUTRR	18, both	Last prompt token; LayerNorm, Linear $2560 \to 256$, GELU, LayerNorm, and direction/relation/gender softmaxes of sizes 2/16/2. No additional cross-attention. LoRA lr 10^{-4}, adapter lr 10^{-3}, effective batch 16, answer loss through the chart. Contrast weight 1.
CUB-200	Qwen 14; Gemma 18	Live image states with reader LoRA (rank 16, alpha 32), 28 category queries, width 512, eight attention heads, cross-attention and two query self-attention layers. AdamW lr 3×10^{-4} for the adapter and reader LoRA and 10^{-2} for the score scale, OneCycle (Smith & Topin, 2019), effective batch 48, forty epochs. Gemma E4B for the main result; E2B for the frozen-backbone controls.
Super-CLEVR	Qwen 18; Gemma 24	Lexical program adapter: width 256, two layers. Part adapter: one query per named part over crop tokens, width 256, eight attention heads, two layers. Answer-only training, lr 10^{-5} for LoRA and 10^{-4} for the program adapter; part lr 5×10^{-4}. Batch eight, top-eight program mixture, coverage-loss weight 0.3.
GSM8K	18, both	Four latent queries per instruction in the prompt; the backbone's own attention provides their states. Typed operation and operand-pointer outputs, ten steps, top-K 32 and input support 32. LoRA lr 10^{-5}, adapter lr 10^{-4}, batch 16.
SATBench	Qwen 16; Gemma 17	Nine literal queries per condition, attending only over its token span; eight attention heads at backbone width, MLP width 512, one cross-attention step. Activity, sign and coordinate outputs. LoRA lr 2×10^{-4}, head lr 10^{-3}, effective batch eight, microbatch one; 400 updates and 3,200 presentations. Answer-only verdict loss through the top-32 mixture, no formula supervision; 3,600 input tokens.

C Full numeric results

Table 7 reports program accuracy and Brier scores. In Figures 4 and 5, the final SPACE Soft accuracy is end-to-end: it scores the generated answer text. Appendix B defines the baselines, the equal-data and equal-clock budgets, the evaluation pools and the confidence conventions.

Table 7: Accuracy and multiclass Brier score (lower is better). Accuracy retains its original pools. Brier uses 128 shared Test questions per dataset, 8 independent tool-call samples per question for SFT/GRPO/zero-shot with a finite-sample correction, and SPACE's complete declared answer distribution. Failures are retained. Iso-data checkpoints have matched training presentations; iso-clock accuracy uses the matched training clock. CLUTRR and Super-CLEVR Test are OOD. GSM8K SPACE Brier uses a disclosed full-distribution replay, not an exact reconstruction of the archived accuracy predictions. A dash marks an unevaluated or unavailable Brier cell.

		Qwen3.5-4B				Gemma-4-E4B-it			
		Iso-data			Iso-clock	Iso-data			Iso-clock
Dataset	Method	ID↑	Test↑	Brier↓	Test↑	ID↑	Test↑	Brier↓	Test↑
CLUTRR	SFT	**0.910**	0.470	0.859	0.726	0.838	0.729	0.486	0.680
	GRPO	0.815	0.728	0.462	0.467	0.793	0.696	0.632	0.569
	SPACE Soft	0.847	**0.781**	**0.368**	**0.781**	**0.892**	**0.738**	**0.426**	**0.738**
	SPACE Commit	0.757	0.594	0.719	0.594	0.707	0.507	0.953	0.507
	Zero-shot	0.320	0.288	1.334	—	0.730	0.492	0.781	—
CUB-200	SFT	0.451	0.440	0.683	0.440	0.439	0.418	0.706	0.418
	GRPO	—	—	—‡	0.272	—	—	—‡	0.122
	SPACE Soft	0.871	**0.866**	**0.353**	**0.866**	**0.801**	**0.817**	**0.351**	**0.817**
	SPACE Commit	**0.873**	0.864	0.374	0.864	**0.801**	0.809	0.375	0.809
	Zero-shot	0.264	0.227	1.036	—	0.119	0.115	1.427	—
Super-CLEVR	SFT	**0.813**	**0.676**	**0.632**	0.606	**0.757**	0.627	**0.701**	0.514
	GRPO	0.652	0.597	0.825	0.095	0.584	0.485	0.846	0.007
	SPACE Soft	0.705	0.666	0.693	**0.666**	0.690	**0.645**	0.754	**0.645**
	SPACE Commit	0.681	0.588	0.875	0.588	0.657	0.564	0.922	0.564
	Zero-shot	0.019	0.028	1.766	—	0.009	0.005	1.739	—
GSM8K	SFT	0.656	0.629	0.461	0.550	0.664	0.608	0.484	**0.604**
	GRPO	**0.875**	**0.846**	**0.218**	**0.699**	**0.797**	**0.782**	**0.326**	0.119
	SPACE Soft	0.016	0.024	1.058	0.024	0.016	0.025	1.067	0.025
	SPACE Commit	0.008	0.013	1.984	0.013	0.016	0.030	1.969	0.030
	Zero-shot	0.688	0.697	0.427	—	0.141	0.120	1.571	—
SATBench	SFT	**0.867**	**0.981**	0.085	**0.974**	**0.859**	**0.970**	**0.294**	**0.981**
	GRPO	**0.867**	0.940	**0.058**	0.792	0.625	0.747	0.369	0.621
	SPACE Soft	0.422	0.498	1.047	0.498	0.422	0.498	1.047	0.498
	SPACE Commit	0.422	0.498	1.047	0.498	0.422	0.498	1.047	0.498
	Zero-shot	0.734	0.787	0.357	—	0.516	0.620	0.634	—

C.1 Additional controls

Table 8 reports matched-budget CUB controls, eight-draw voting, symmetric ID-validation selection, and three SPACE seeds. Intervals are 95% bootstrap, paired unless noted, and condition on fitted checkpoints.

Probe and decoding checks. The frozen-backbone CUB controls share frozen features, trainable pooling and budget; the table helps, but high Qwen ablation accuracy does not establish literal attributes. Voting counts failures and breaks ties by first draw; legal voting drops invalid draws, with all-invalid questions wrong. At 512 questions, available SFT legal-vote gaps are unresolved, as are CLUTRR GRPO gaps (Q/G: $-2.93\,[-6.45, 0.59]$/$0.59\,[-3.71, 5.08]$); Super-CLEVR Qwen GRPO remains lower ($-8.01\,[-12.20, -3.89]$). Checkpoint selection uses 128 ID-dev questions, a 32,768-presentation ceiling and earliest-step ties; SFT selects its final checkpoints. Argument SFT on CLUTRR Qwen scores 0.726 OOD at its earlier equal-clock checkpoint and 0.470 at equal data while in-distribution validation still selects the later checkpoint, so the equal-data gap reflects continued training on lengths two to four rather than a ceiling on SFT; at equal clock SPACE Soft leads by 5.5 points.

Language and collapse checks. Base WikiText test perplexities are 12.75 [11.53,14.11] (Qwen) and 102.12 [88.07,119.21] (Gemma), under the existing protocol; these intervals are unpaired. Seed-0 adapter-GRPO replay reproduces all 1,048 predictions per backbone: length-2 zeros reflect sibling-only collapse (0/38 sibling targets versus 31/107 at length 3), not scoring error.

Table 8: Additional controls. Accuracies are percentages; Δ is control minus SPACE Soft, in percentage points with paired 95% bootstrap intervals. CUB slashed scores are Soft/Commit; its Δ uses Soft. Q/G denote Qwen/Gemma. Panel D gives mean ± sample SD across three seeds, not a confidence interval; Super-CLEVR Gemma includes one B200 and two H100 runs.

A. Frozen-backbone CUB: fixed 40 epochs; 5,794 test images

Control	Q accuracy	G accuracy	Δ Q [95% CI]	Δ G [95% CI]
Original Soft/Commit	85.99/85.57	75.73/73.40	—	—
Linear species head	84.88	73.40	$-1.10\,[-1.85, -0.35]$	$-2.33\,[-3.40, -1.24]$
MLP species head	84.09	60.13	$-1.90\,[-2.69, -1.10]$	$-15.60\,[-16.91, -14.31]$
Shuffled table	83.66/82.93	64.29/61.53	$-2.33\,[-3.11, -1.54]$	$-11.44\,[-12.69, -10.25]$
Random binary table	84.40/83.36	68.83/65.27	$-1.59\,[-2.38, -0.81]$	$-6.90\,[-8.03, -5.76]$

B. Vote@8/legal vote@8: 128 shared OOD questions

Text baseline	Q vote/legal	G vote/legal	Δ legal Q	Δ legal G
CLUTRR SFT	50.78/56.25	72.66/77.34	$-23.44\,[-32.81, -14.84]$	$+1.56\,[-5.47, +9.38]$
CLUTRR GRPO	72.66/77.34	64.06/70.31	$-2.34\,[-7.81, +2.34]$	$-5.47\,[-14.06, +2.34]$
Super-CLEVR SFT	62.50/62.50	57.81/57.81	$-2.34\,[-11.54, +6.82]$	$-7.03\,[-14.40, +0.00]$
Super-CLEVR GRPO	53.91/53.91	47.66/49.22	$-10.94\,[-17.74, -4.44]$	$-15.62\,[-22.40, -9.30]$

C. CLUTRR: ID-validation-selected checkpoints; 826 OOD questions

Method	Q OOD	G OOD	Δ Q	Δ G
SPACE	82.69	73.85	—	—
SFT	46.97	72.88	$-35.71\,[-39.47, -31.84]$	$-0.97\,[-4.36, +2.42]$
GRPO	73.12	66.71	$-9.56\,[-12.83, -6.30]$	$-7.14\,[-10.65, -3.63]$

D. Three seeds: Table 7 recipes and exposure

Dataset	Q Soft	Q Commit	G Soft	G Commit
CLUTRR (OOD)	78.81 ± 1.92	61.86 ± 2.48	74.82 ± 0.87	53.19 ± 4.91
CUB-200 (test)	86.39 ± 0.16	86.27 ± 0.22	81.77 ± 0.08	80.89 ± 0.17
Super-CLEVR (OOD)	65.82 ± 0.89	57.88 ± 0.96	65.07 ± 1.28	56.62 ± 1.32
GSM8K (test)	2.53 ± 0.76	1.79 ± 1.36	2.40 ± 0.12	1.92 ± 0.99
SATBench (test)	49.81 ± 0.00	49.79 ± 0.03	49.81 ± 0.00	49.83 ± 0.03

Calibration controls. At 128 questions, eight-draw modal-answer frequency (failures stay in the denominator) gives Soft minus text top-label CORP miscalibration of $+.035\,[+.002, +.086]$ for CLUTRR Gemma SFT, $+.032\,[+.001, +.076]$ for CLUTRR Qwen GRPO and $-.080\,[-.148, -.014]$ for Super-CLEVR Qwen GRPO; development-fitted temperature reduces failure-gated CLUTRR Gemma GRPO miscalibration from .024 to .006, so comparisons depend on readout. Multiclass Brier is separate: at 128 questions only CLUTRR Gemma resolves the Soft–GRPO ordering (Super-CLEVR Gemma: $-.092\,[-.203, .017]$); at 512 both CLUTRR gaps resolve ($-.114\,[-.179, -.049]$/$-.079\,[-.158, -.001]$, Q/G), while Super-CLEVR Qwen remains unresolved ($-.077\,[-.156, +.0001]$). SFT is lower on Super-CLEVR at 512 (Soft minus SFT: $.045\,[.005, .086]$/$.104\,[.056, .152]$, Q/G).

Failure mass. Under an edge-assignment diagnostic (failure iff no legal bracketing; uniform legal derivations within each assignment), 512-question CLUTRR Soft Brier is .737/.895 (Q/G), increasing by $.383\,[.318, .451]$/$.446\,[.379, .515]$ relative to that law conditioned on legality; top-label miscalibration increases by $.091\,[.064, .118]$/$.134\,[.098, .167]$. This differs from Equation 3's derivation-weighted law; Super-CLEVR already retains failure mass and is unchanged. At 2,048 answer-only presentations, removing the lexical-prior bundle reduces Qwen Super-CLEVR OOD accuracy from 62.30% to 11.33% ($\Delta = -50.98\,[-55.90, -45.88]$ points, 512 questions), supporting the need for supplied domain knowledge.

D Brier score

We evaluate the multiclass Brier score $\sum_c (p_c - \mathbf{1}[c = y])^2$ (Brier, 1950), including a single failure category with zero reference probability. This scores the full answer distribution and remains finite when the correct answer has zero probability; an all-failure forecast scores 2. Lower is better. It

measures overall probabilistic prediction quality, including calibration and discrimination (Murphy, 1973), rather than calibration alone. Because the rule is strictly proper, a forecaster obtains its best expected score only by reporting its actual answer distribution (Gneiting & Raftery, 2007). The score needs no separate confidence value, so one rule applies to SPACE's exact distribution and to the sampled text methods alike. For each dataset we select 128 shared Test questions by a fixed hash of public IDs. SPACE supplies its complete declared answer distribution; each text baseline supplies 8 independent native tool-call samples at temperature 1, without top-k or top-p truncation. We retain failed calls and repeated samples. With n_c samples in category c and $K = 8$, we use the fair multicategory Brier estimator (Ferro, 2014)

$$1 - \frac{2n_y}{K} + \sum_c \frac{n_c(n_c - 1)}{K(K - 1)}.$$

For K independent draws, $\mathbb{E}[n_y] = Kp_y$ and $\mathbb{E}[n_c(n_c - 1)] = K(K - 1)p_c^2$, so the estimator's expectation is the Brier score of the sampling distribution. Scoring the empirical frequencies n_c/K directly would overstate it by $(1 - \sum_c p_c^2)/K$, up to $1/K$, and so would penalize the most uncertain text forecasts most. Zero-shot samples use thinking off, except Gemma on Super-CLEVR, which uses thinking on. No calibrator is fitted to these scores; Appendix D.1 fits one to each method's top-label confidence only as a diagnostic. Paired 95% bootstrap intervals resample public question groups and condition on the fitted checkpoints; they are descriptive and are not adjusted for multiple comparisons. Accuracy retains the full original pools. SPACE's answer distribution retains its executor's conditioning convention: CLUTRR conditions on legal derivations, SAT Soft on its retained formula hypotheses, and CUB uses its score softmax. SAT Commit is a point mass at its hard verdict, not its formula confidence. GSM8K uses exact numeric answer classes and a disclosed full-distribution replay that differs from some archived predictions.

Brier comparison uncertainty Table 9 reports paired differences on the same 128 questions. Negative values favor SPACE Soft. These are descriptive 95% cluster-bootstrap intervals conditional on one fitted checkpoint, without multiplicity adjustment. Intervals crossing zero do not resolve an ordering; they do not establish equivalence. The 128-question subset supports only effects its uncertainty can resolve, not precise rankings of close methods. Independent native samples estimate the forecast for each question; they are not additional independent test questions.

Table 9: Multiclass Brier differences and paired 95% intervals; 128 shared questions per dataset. Negative values favor SPACE Soft, and an interval that contains zero means the difference is not statistically significant. An em dash marks CUB-200 GRPO, which stops at the equal-clock mark of the frozen-backbone reader and has no equal-data checkpoint.

Dataset	Backbone	Soft − SFT	Soft − GRPO
CLUTRR	Qwen	-0.491 [-0.649, -0.338]	-0.094 [-0.193, 0.003]
CLUTRR	Gemma	-0.060 [-0.199, 0.076]	-0.206 [-0.360, -0.060]
CUB-200	Qwen	-0.330 [-0.461, -0.197]	—
CUB-200	Gemma	-0.355 [-0.497, -0.210]	—
Super-CLEVR	Qwen	0.061 [-0.022, 0.145]	-0.132 [-0.271, 0.008]
Super-CLEVR	Gemma	0.053 [-0.046, 0.154]	-0.092 [-0.203, 0.017]
GSM8K	Qwen	0.597 [0.525, 0.667]	0.841 [0.762, 0.914]
GSM8K	Gemma	0.583 [0.513, 0.652]	0.741 [0.635, 0.838]
SATBench	Qwen	0.961 [0.793, 1.132]	0.989 [0.807, 1.170]
SATBench	Gemma	0.753 [0.560, 0.945]	0.678 [0.476, 0.884]

CUB-200 GRPO clock-budget Brier (frozen-backbone comparison) These results use the GRPO clock checkpoint of the frozen-backbone comparison and are separate from the equal-data comparisons above. Qwen: 0.903 [0.838, 0.965]. Gemma: 1.415 [1.306, 1.517].

D.1 Calibration analysis

For the trained CUB readers, the paired MCB comparisons against empirical eight-call confidence are not resolved on either backbone, although their multiclass Brier scores are lower than SFT's.

Table 10: Top-label CORP decomposition of each method's own confidence on the full test pools (CLUTRR lengths 5–10, large Super-CLEVR scenes). MCB is the miscalibration term (lower is better) for raw confidence, after setting failed calls to zero confidence (gated), and after the development-fitted map (dev); DSC is discrimination.

Dataset	Backbone	Method	Accuracy	MCB raw	MCB gated	MCB dev	DSC
CLUTRR	Qwen	SFT	0.470	0.307	0.061	0.016	0.031
CLUTRR	Qwen	GRPO	0.728	0.084	0.012	0.007	0.015
CLUTRR	Qwen	SPACE Soft	0.781	0.038	0.038	0.042	0.026
CLUTRR	Gemma	SFT	0.729	0.113	0.019	0.007	0.041
CLUTRR	Gemma	GRPO	0.696	0.113	0.014	0.003	0.023
CLUTRR	Gemma	SPACE Soft	0.738	0.048	0.048	0.036	0.015
Super-CLEVR	Qwen	SFT	0.676	0.073	0.016	0.003	0.000
Super-CLEVR	Qwen	GRPO	0.597	0.155	0.121	0.007	0.002
Super-CLEVR	Qwen	SPACE Soft	0.666	0.035	0.035	0.035	0.048
Super-CLEVR	Gemma	SFT	0.627	0.083	0.019	0.003	0.000
Super-CLEVR	Gemma	GRPO	0.485	0.267	0.138	0.006	0.019
Super-CLEVR	Gemma	SPACE Soft	0.645	0.038	0.038	0.028	0.054

Table 11: Multiclass Brier differences on 512 shared questions (a superset of the 128 in Table 9), with paired 95% intervals. Negative values favor SPACE Soft. Comparisons unresolved at 128 questions were extended, except Super-CLEVR Gemma GRPO, whose checkpoint finished later; — marks a comparison not re-sampled.

Dataset	Backbone	Soft − SFT	Soft − GRPO
CLUTRR	Qwen	—	-0.114 [-0.179, -0.049]
CLUTRR	Gemma	-0.031 [-0.107, 0.045]	-0.079 [-0.158, -0.001]
Super-CLEVR	Qwen	0.045 [0.005, 0.086]	-0.077 [-0.156, 0.000]
Super-CLEVR	Gemma	0.104 [0.056, 0.152]	—

Multiclass Brier mixes calibration with discrimination, so we also decompose each method's own top-label confidence with CORP (Dimitriadis et al., 2021): top-label Brier is MCB minus DSC plus UNC, where MCB is the reduction from isotonic recalibration on the test pool. SPACE reports the probability of its answer; the text baselines report the geometric-mean probability of the generated call. Table 10 gives MCB for raw confidence, after failed calls are set to zero confidence, and after the evaluator's development-fitted map, which also zeroes failed calls. SPACE's raw confidence needs almost no recalibration: MCB moves from 0.0354 to 0.0347 on Super-CLEVR Qwen and from 0.0476 to 0.0360 on CLUTRR Gemma. The raw sequence confidence of the text baselines is near 1 even on failed calls (mean 1.00 on the 154 failed CLUTRR Gemma SFT calls); after the failure gate and the development map they are as well or better calibrated, with SFT moving from 0.0732 to 0.0034 on Super-CLEVR Qwen and from 0.1127 to 0.0073 on CLUTRR Gemma. The comparison at equal accuracy therefore depends on the confidence readout. Across the 26 pairs of Figure 4 checkpoints within 0.05 accuracy, SPACE has lower, higher and unresolved MCB in 21/2/3 pairs for raw confidence, 6/16/4 after the failure gate and 0/26/0 after the development map. SATBench Soft confidence is near 1 on 2092 of 2100 questions at accuracy 0.498, so we exclude it from comparative claims. Figure 6 shows reliability diagrams. Table 11 extends the comparisons that Table 9 leaves unresolved to 512 questions: SPACE is lower than GRPO on CLUTRR for both backbones, and SFT is lower on Super-CLEVR for both backbones.

D.2 Zero-shot native tool calls

Table 13 gives the backbones without fine-tuning when the compiled program is offered as the tool of Table 4 through each model's own chat template. The model writes exactly one call, with thinking off or after at most 1,024 thinking tokens, and the compiled result of that call is the answer. A missing, malformed or truncated call and a failed execution count as errors. Every row uses the full declared pool.

Table 12: Top-label CORP decomposition of each method's own confidence on the full test pools (5,794 CUB test images). MCB is the miscalibration term (lower is better) for raw confidence, after setting failed calls to zero confidence (gated), and after the development-fitted map (dev); DSC is discrimination. — marks a checkpoint that does not exist.

Dataset	Backbone	Method	Accuracy	MCB raw	MCB gated	MCB dev	DSC
CUB-200	Qwen	SFT	0.440	0.308	0.275	0.002	0.001
CUB-200	Qwen	GRPO	—	—	—	—	—
CUB-200	Qwen	SPACE Soft	0.866	0.012	0.012	0.001	0.030
CUB-200	Gemma	SFT	0.418	0.329	0.279	0.002	0.001
CUB-200	Gemma	GRPO	—	—	—	—	—
CUB-200	Gemma	SPACE Soft	0.817	0.013	0.013	0.004	0.045

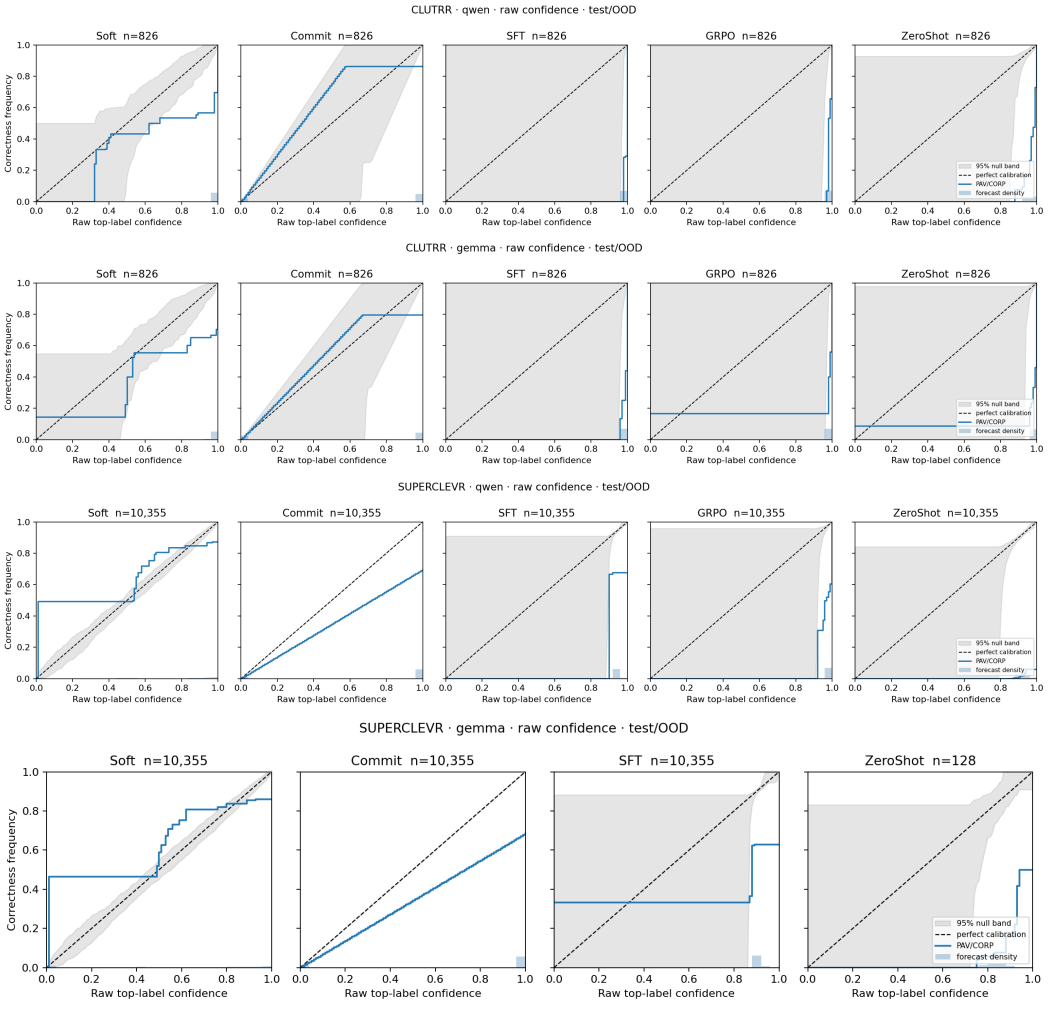

Figure 6: CORP reliability diagrams of raw top-label confidence on CLUTRR (lengths 5–10) and large Super-CLEVR scenes. Grey bands are 95% consistency bands under each method's calibrated null, not confidence bands; the histogram shows where forecasts occur.

Table 13: Zero-shot native tool call. ID and OOD follow Table 7.

Dataset	Backbone	Thinking	n	Accuracy	Valid calls
CLUTRR ID	Qwen	off	222	0.320	0.52
CLUTRR OOD	Qwen	off	826	0.288	0.51
CLUTRR ID	Qwen	on	222	0.680	0.76
CLUTRR OOD	Qwen	on	826	0.568	0.64
CLUTRR ID	Gemma	off	222	0.730	0.85
CLUTRR OOD	Gemma	off	826	0.492	0.66
CLUTRR ID	Gemma	on	222	0.608	0.66
CLUTRR OOD	Gemma	on	826	0.517	0.58
CUB-200 ID	Qwen	off	599	0.264	0.89
CUB-200 Test	Qwen	off	5794	0.227	0.89
CUB-200 ID	Gemma	off	599	0.119	0.38
CUB-200 Test	Gemma	off	5794	0.115	0.37
Super-CLEVR ID	Qwen	off	16042	0.019	0.06
Super-CLEVR OOD	Qwen	off	10355	0.028	0.10
Super-CLEVR ID	Gemma	off	16042	0.009	0.04
Super-CLEVR OOD	Gemma	off	10355	0.005	0.03
GSM8K ID	Qwen	off	128	0.688	0.98
GSM8K Test	Qwen	off	1319	0.697	0.98
GSM8K ID	Qwen	on	128	0.883	0.99
GSM8K Test	Qwen	on	1319	0.864	0.98
GSM8K ID	Gemma	off	128	0.141	0.16
GSM8K Test	Gemma	off	1319	0.120	0.14
GSM8K ID	Gemma	on	128	0.805	0.88
GSM8K Test	Gemma	on	1319	0.804	0.87
SATBench ID	Qwen	off	128	0.734	0.84
SATBench Test	Qwen	off	2100	0.787	0.84
SATBench ID	Gemma	off	128	0.516	0.55
SATBench Test	Gemma	off	2100	0.620	0.64

E CLUTRR

Data The split is `gen_train234_test2to10`, with 12,064 training stories with chains of length 2 to 4, 3,019 validation stories, and 1,048 test stories with chains of length 2 to 10, of which 826 have length 5 to 10 (185, 105, 155, 135, 124 and 122 stories at lengths 5 to 10). Names in the stories are bracketed. The equal-data pool removes 26 exact validation-input duplicates from the training split, leaving 12,038 stories. An input-only audit finds 27 validation prompts in the training pool after case-folding and whitespace normalization, but no such overlap in the test pool; the canonical test denominator is retained, including three repeated prompts within the test set.

Program The composition table is the dataset's own set of 64 rules over relation pairs, checked against every derivation in the training and validation splits (30,127 instances, 62 distinct rules observed, no mismatch). The 21 states are the 14 relations a sentence can state, six that arise only as compositions, and "no legal relation". The chart of Equation 3 fills a distribution over the 21 states for every span of the chain from the beliefs of its sentences, summing over the legal split points, and discards undefined compositions before the final normalization (Figure 7). The benchmark uses 18 answer relations; training indexes the target relation in the chart. On the reference relations the sum over legal bracketings keeps 0.95 to 1.00 of the mass on the reference answer out to length ten, while an average over all bracketings falls to 0.01, because most bracketings pass through a pair with no rule.

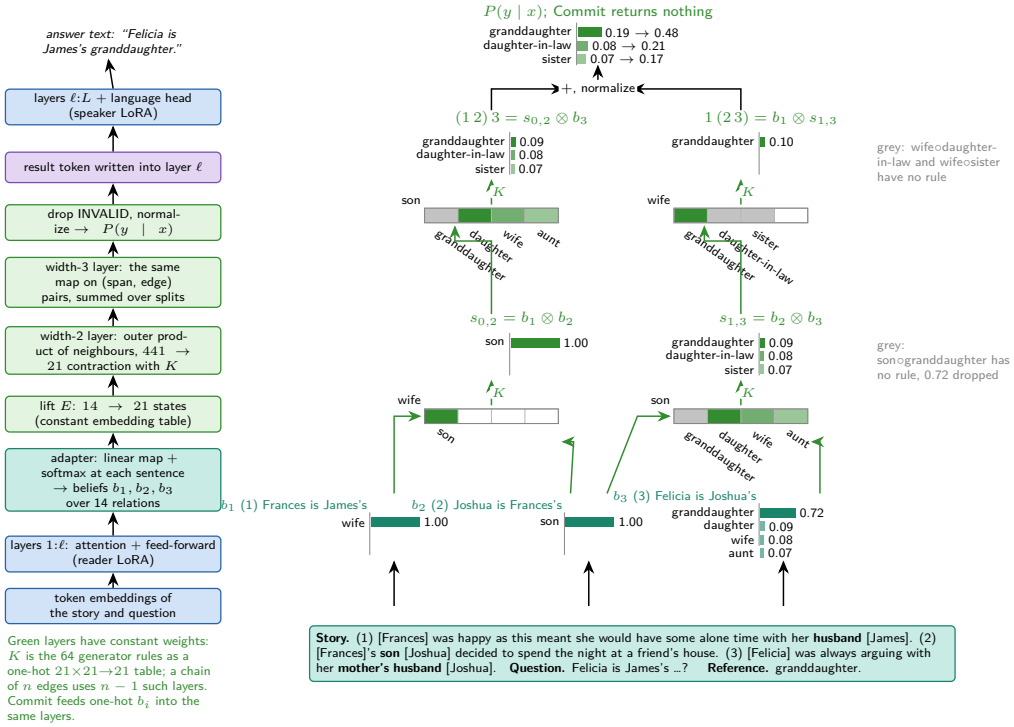

Figure 7: The compiled CLUTRR program as layers of the network (left, input at the bottom) and as the chart it computes on one test story (right). Left: the backbone reads the story, the adapter emits one belief per sentence at layer ℓ, and the green layers are the chart with constant weights: a lift to 21 relation states, one bilinear layer per span width contracting with the rule table K, and a normalization; the result token re-enters the backbone and the upper layers write the answer. Right: bars are probability vectors, grids are outer products of two beliefs, the arrow marked K contracts a grid with the rule table, grey cells have no rule, and the full span sums both bracketings. Only the input beliefs are learned.

Argument distributions and the contrast term Each sentence of the chain is read on its own. When a sentence states two relations of the chain, the whole story is the sentence's context; this

happens for about 45 percent of sentences and is why the prompt needs a 512-token budget. A 128-token budget truncates the query in 34 to 45 percent of sentences at lengths eight to ten and limits accuracy to 0.61. For a sentence with query names h (head) and t (tail), the adapter gives three distributions: a belief over the direction $d \in \{t \text{ is } h\text{'s}, h \text{ is } t\text{'s}\}$, the stated relation word s (16 values, the 14 relations plus *nephew* and *niece*, which appear only as the reverse of an aunt or uncle) and the tail's gender g. A fixed table $T[d, g, s, \ell]$ built from the dataset's relation store maps the triple to the edge relation ℓ (14 values). For the first direction $\ell = s$ when s is an edge relation, for the second ℓ is the inverse of s for a tail of gender g, and triples with no legal relation carry no mass. The belief over ℓ is the contraction of the three distributions with T, renormalized. The contrast term swaps h and t in the sentence, reads it again, and adds $\mathrm{KL}(d' \,\|\, \mathrm{flip}(d)) + \mathrm{KL}(s' \,\|\, s)$ with weight 1. Both halves are needed, since a uniform direction belief satisfies the first alone and a name-blind relation belief satisfies the second alone, which is the solution the collapsed sampled adapter reaches with a contrast loss of 0.000. A rule-based adapter that reads roles from position satisfies the term exactly; one that keys them on the names does not.

Equal-data results At 32,768 samples, SPACE Soft exceeds GRPO's long-chain accuracy by 5.33 percentage points on Qwen (paired 95% interval $[2.30, 8.35]$) and 4.24 points on Gemma ($[0.73, 7.87]$). GRPO exceeds SFT by 25.79 points on Qwen ($[22.15, 29.42]$), where SFT overfits the training lengths, and trails SFT by 3.27 points on Gemma ($[0.73, 5.81]$); Gemma SFT and Soft differ by 0.97 points ($[-2.30, 4.36]$). Figure 8 breaks these results down by chain length. Table 7 and Figure 4 use multiclass Brier on 128 shared OOD questions to compare complete answer distributions, including failures.

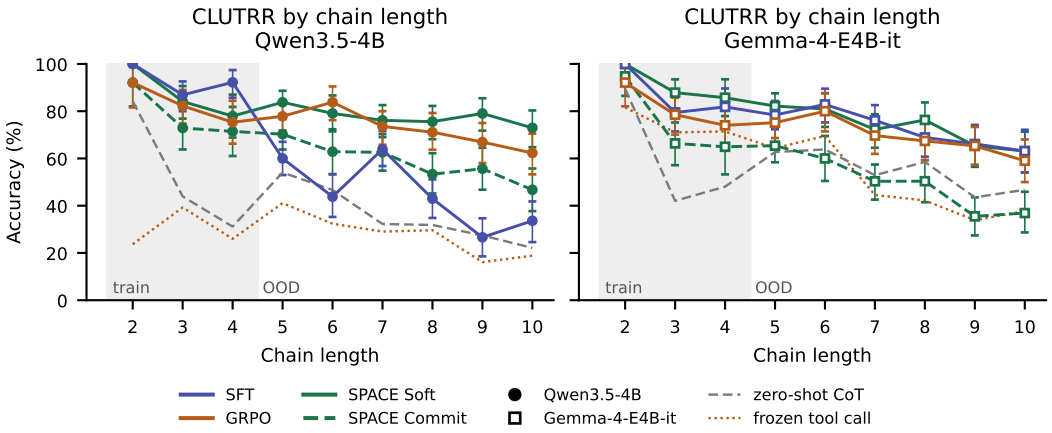

Figure 8: CLUTRR test accuracy by chain length at 32,768 training samples on both backbones. Shading marks the training lengths 2 to 4; lengths 5 to 10 are out of distribution. Error bars are 95% bootstrap intervals over test stories for one fitted seed.

Exact and sampled training of the same adapter The controls below use the original 12,064-story pool for two epochs with final-step reporting, separately from the equal-data comparison. Qwen has one saved exact seed; Gemma exact and contrast average two seeds, and sampled-adapter training has one seed per backbone. This control trains the same adapter by exact differentiation or sampled policy gradients, with the same data and schedule. The exact arm differentiates the full relation chart; the sampled arm draws eight assignments per story and rewards correct answers. Sampled training reaches about 0.13 on both backbones, compared with 0.86 and 0.79 for exact training. Figure 9 shows that the sampled adapter becomes nearly deterministic while accuracy remains low. Gemma ends with mean edge entropy 0.0026 nats and identical answers under both readouts. At the observed initial success rate near 0.01, Equation 5 gives about 92% of groups with uniform rewards, and once the adapter collapses, sampled assignments usually agree. This explains the lack of reward signal in these runs; it is not a claim that sampled training must fail.

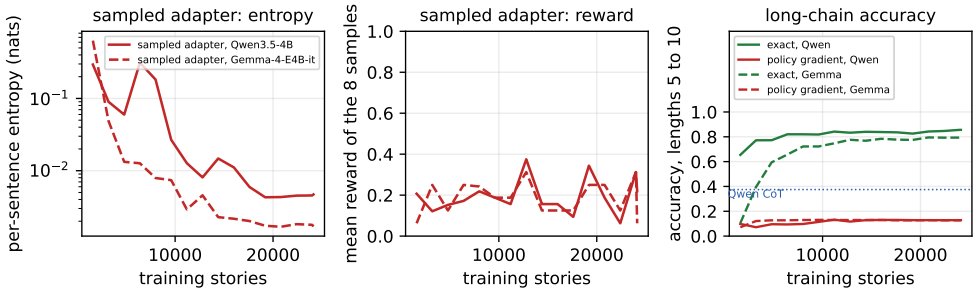

Figure 9: CLUTRR adapter training: edge entropy, sampled reward and long-chain accuracy by training samples, for exact and sampled gradients.

Accuracy by chain length Table 14 gives the test accuracy by chain length for the exact-gradient, contrast, sampled-adapter, answer-SFT and chain-of-thought controls on the original training pool.

Table 14: CLUTRR test accuracy by chain length. Means over seeds; lengths 2 to 4 are the training lengths.

backbone	training	2	3	4	5	6	7	8	9	10
Qwen3.5-4B	exact gradient	1.00	0.91	0.90	0.89	0.90	0.85	0.85	0.84	0.80
Qwen3.5-4B	exact + contrast	1.00	0.87	0.85	0.88	0.89	0.85	0.87	0.83	0.84
Qwen3.5-4B	adapter GRPO	0.00	0.29	0.10	0.16	0.10	0.12	0.10	0.14	0.13
Qwen3.5-4B	answer-SFT	1.00	0.95	0.94	0.44	0.47	0.38	0.48	0.32	0.49
Qwen3.5-4B	CoT + confidence	0.84	0.44	0.31	0.54	0.47	0.32	0.32	0.27	0.22
Gemma-4-E4B-it	exact gradient	1.00	0.85	0.86	0.84	0.84	0.77	0.77	0.80	0.72
Gemma-4-E4B-it	exact + contrast	1.00	0.86	0.85	0.86	0.86	0.82	0.80	0.83	0.78
Gemma-4-E4B-it	adapter GRPO	0.00	0.28	0.10	0.16	0.10	0.12	0.10	0.15	0.11
Gemma-4-E4B-it	answer-SFT	1.00	0.95	0.98	0.42	0.44	0.36	0.39	0.33	0.42
Gemma-4-E4B-it	CoT + confidence	0.89	0.42	0.48	0.63	0.64	0.53	0.59	0.44	0.47
Qwen3.5-4B	CoT, 3,072 tokens	0.74	0.40	0.25	0.46	0.39	0.39	0.41	0.27	0.27
Qwen3.5-4B	thinking CoT, 8,192 tokens	0.79	0.27	0.21	0.41	0.31	0.25	0.24	0.16	0.10

The chain-of-thought budget Table 15 reports the chain-of-thought baseline at three budgets, without a confidence instruction. The prompt holds the story with the brackets removed, the chain of names, the question, and a fixed answer line; decoding is greedy. A generation cut off by its budget counts as wrong. The failures are mostly generations that re-derive the chain and never stop, and the stories that do finish are answered well, in particular by the thinking mode, at a mean cost of 7,264 generated tokens per story.

Table 15: Qwen3.5-4B chain of thought without a confidence instruction, greedy decoding.

mode	budget	acc all (1,048)	acc 5–10 (826)	cut off	acc on finished long stories	mean new tokens	GPU s
no thinking, chain given	3,072	0.382	0.375	31%	0.582 ($n = 533$)	1,548	4,331
thinking, chain given	8,192	0.273	0.255	77%	0.901 ($n = 162$)	7,264	10,611
no thinking, no chain	768	0.287	0.274	48%	–	502	459

F CUB-200

Data and program The main comparison uses the original 5,395/599/5,794 training/development/test image split. For category j, value v and species c, a fixed table A_{cjv} supplies the score

$$\overline{S}_c = \frac{1}{28} \sum_{j,v} q_j(v) A_{cjv}, \qquad P(c \mid x) = \text{softmax}(s\overline{S})_c. \tag{6}$$

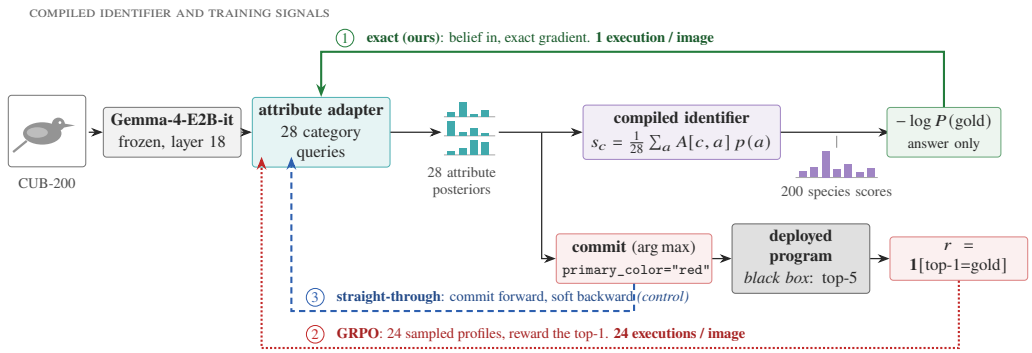

Figure 10: CUB identifier and frozen-backbone training controls. Attribute distributions enter an additive species-score table; exact gradients, straight-through gradients and sampled rewards provide three paths to the same adapter.

The learned scale s is saved with the adapter. Mean-match weights use attribute frequencies; log-likelihood weights use their logarithms (Figure 10).

Adapter Reader LoRAs (rank 16, alpha 32) train below layer 14 of Qwen3.5-4B or layer 18 of Gemma-4-E4B-it, and the vision encoder stays frozen. Category queries read these adapted image states and exchange information through two self-attention layers (Table 6). Training minimizes species cross-entropy without image-level attribute labels. Qwen images pass at native resolution under a budget of 401,408 pixels, above the largest CUB image.

Main comparison Both backbones train for 40 epochs (215,800 presentations) with log-likelihood weights and reader LoRAs. Table 7 uses the fixed final checkpoint, not a development-selected epoch. The program readouts score 0.866/0.864 (Soft/Commit) on Qwen and 0.817/0.809 on Gemma E4B over all 5,794 test images. Gemma's paired Soft-minus-Commit advantage is 0.86 percentage points (95% image-bootstrap interval 0.48–1.26). Figure 5 reports end-to-end Soft accuracy instead.

Text baselines Argument SFT uses the human image-attribute annotations as targets, with partial labels retained; species-derived values do not fill missing attributes. Text GRPO uses binary species-answer rewards with eight sampled calls per image. GRPO stops at the iso-clock mark of the frozen-backbone reader and was not run for 40 epochs because of training cost. On Gemma, its first update crossing the 385.46-second mark is update 5 (40 presentations, 404.65 seconds), where 60% of its calls are still malformed against the 28-attribute schema.

The Gemma SFT iso-data cell was evaluated at batch 64, while every other CUB Gemma cell used batch 8 with generation batch 1. Replayed against the training run's development predictions, the larger batch changes 56 of 599 development answers in both directions, and development accuracy moves from 0.4407 to 0.4391, a difference of 0.17 points.

Training controls The following controls use frozen Gemma E2B features, with fifteen percent of the training images reserved for development. Table 16 holds the adapter and 40-epoch schedule fixed and varies the training signal. Binary-reward GRPO rarely distinguishes sampled profiles; rank-shaped reward supplies more signal but requires all species scores, whereas the binary reward needs only correctness. Straight-through training uses score cross-entropy and selects the most probable attribute values as one-hot inputs in the forward pass.

Sampled-reward controls Table 17 varies the group size, entropy bonus and learning rate of binary-reward GRPO. The fraction of groups with identical rewards (Equation 5) is 0.92 initially and 0.86 at the final epoch; rank-shaped reward starts at 0.00 and ends at 0.58 as the predictions sharpen. Larger sample groups and entropy bonuses partly recover accuracy, at additional execution cost.

Table 16: E2B training controls, evaluated with each method's deployed readout: Soft for exact training and Commit for sampled and straight-through training.

Rule	Training signal	Dev	Test	Executions
Mean match	Exact gradient	0.691	0.694	203,800
Mean match	GRPO, binary reward	0.013	0.012	4,891,200
Mean match	GRPO, rank reward	0.619	0.620	4,891,200
Mean match	Straight-through	0.704	0.687	203,800
Log likelihood	Exact gradient	0.745	0.726	203,800
Log likelihood	GRPO, binary reward	0.016	0.013	4,891,200
Log likelihood	GRPO, rank reward	0.652	0.635	4,891,200

Table 17: E2B GRPO configurations with the mean-match identifier and Commit readout. All rows use binary rewards except the final rank-shaped control.

Configuration	Dev	Test	Executions
Default ($G = 24$, entropy 0.01, lr 3×10^{-4})	0.013	0.012	4,891,200
Entropy bonus 0.1	0.189	0.198	4,891,200
Entropy bonus 0.3	0.204	0.194	4,891,200
$G = 64$, entropy bonus 0.1	0.544	0.504	13,043,200
Learning rate 10^{-3}	0.008	0.005	4,891,200
Learning rate 10^{-4}	0.035	0.034	4,891,200
Rank-shaped reward	0.619	0.620	4,891,200

Synthetic attribute distributions Table 18 tests the identifier without a vision model by perturbing the fixed table's species profiles with Gaussian logit noise. These are constructed inputs to the program, separate from predictions on test photographs. Soft retains an advantage across the displayed input qualities; as the synthetic reading loses information, both accuracy and the gap decline. The mean-match rule itself need not identify every species even from a faithful profile.

Table 18: Synthetic inputs to the fixed CUB attribute table at four noise levels. Parentheses give attribute accuracy; Δ is Soft minus Commit.

	Mean match			Log likelihood		
Input quality	Soft	Commit	Δ	Soft	Commit	Δ
Perfect (1.000)	0.690	0.555	0.135	1.000	0.825	0.175
Good (0.623)	0.618	0.537	0.082	1.000	0.903	0.096
Poor (0.450)	0.461	0.370	0.091	0.954	0.819	0.135
Weak (0.272)	0.146	0.122	0.024	0.166	0.144	0.022

Attention and grounding Figure 11 shows category-query attention of the Qwen3.5-4B reader before and after species-label training. In the frozen-backbone Gemma-4-E2B controls, head-averaged attention is integrated in a Gaussian window one token cell wide around each visible annotated part, after subtracting the mean over category queries. A hit occurs when the query's own part receives the largest value. Hit rates are 0.221 for straight-through training, 0.204 for exact log-likelihood training, 0.185 for exact mean-match training, 0.143 for rank-reward GRPO, 0.088 for binary-reward GRPO and 0.089 without training. Shuffling category-to-part assignments and using the next image's parts give floors of 0.107 and 0.139. Part annotations score frozen attention maps; they do not supervise the adapter.

In the frozen-backbone controls, answer-only training does not ensure literal attributes: the mean-match adapter reaches 0.694, above the 0.690 reached by faithful profiles under that rule. Against the image annotations, the log-likelihood adapter's attribute accuracy is 0.364 on Qwen and 0.376 on Gemma, no higher than the 0.376 of predicting each category's most frequent value, against 0.511 and 0.487 for argument SFT (Table 3). Attributes can therefore act as a code for species, and the attention measurement alone does not establish the semantics of every predicted category.

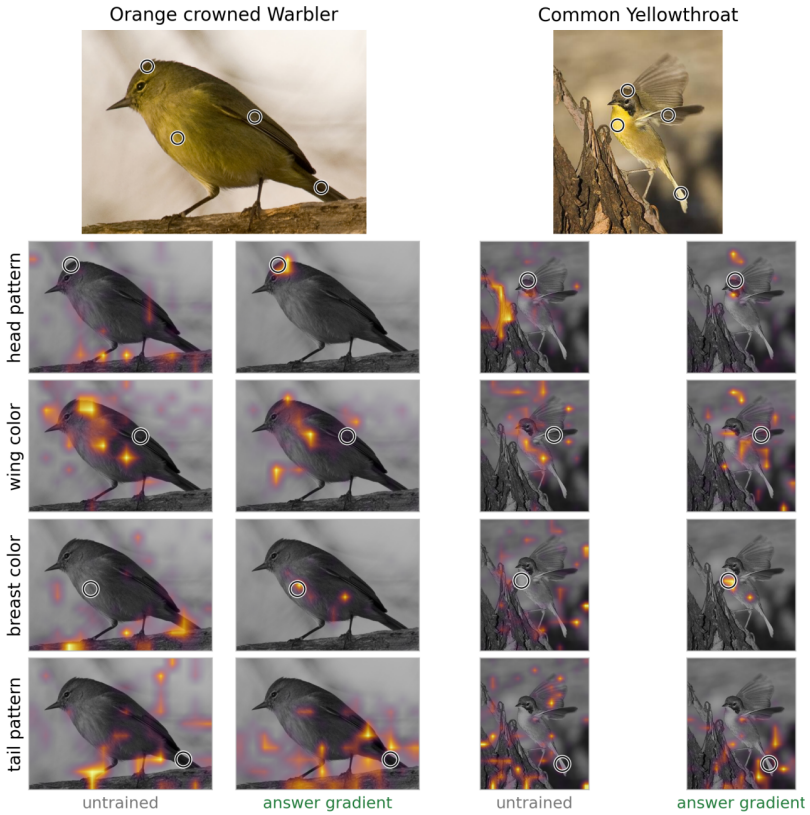

Figure 11: CUB-200 category-query attention of the Qwen3.5-4B reader before training (untrained) and after species-label training of its reader LoRA and adapter (answer gradient), at the fixed epoch-40 checkpoint. Maps show attention above the mean over 28 queries, scaled separately for each map, so brightness is not comparable across maps. After training, the maps focus more on the relevant spatial locations, without any supervised spatial or attribute labels.

G SUPER-CLEVR

Scenes and question families The source has 30,000 rendered scenes with 21 vehicle shapes, five categories, eight colors, two materials and two sizes. Scenes 0–23,999 train, 24,000–26,999 form development, and 27,000–29,999 form test. A generator constructs executable programs and renders questions from them in seven families: part query, count, existence, equality and comparison, plus deep query and count over relation chains. Object parts have independently colored regions that can be smaller than one image token. The part pool contains 210,000 training questions, 26,000 development questions and 26,305 test questions, generated separately from the answer-training test pools below over the same 3,000 test scenes. To test generalization out of distribution, answer training uses scenes with three to six objects and count answers at most three, while the larger-scene test has eight to ten objects and count answers up to ten. The equal-data study draws from a common 64,000-row pool, including 21,820 count and 42,180 non-count questions, with no inherited program-supervised weights.

Public objects and learned arguments A zero-shot Qwen object listing supplies boxes and attribute probabilities, with left/right and front/behind relations derived from box geometry. Every method uses the same listings. Part queries cross-attend vision tokens inside each box in relative coordinates, predicting a gate for inheritance from the body and categorical laws for color, material and size. The program adapter predicts a skeleton and its typed filter and relation slots. SPACE learns a fresh part adapter and a lexical program adapter from final answers in the common pool, without reference-program targets. The lexical adapter uses the 111-skeleton inventory but reads every slot

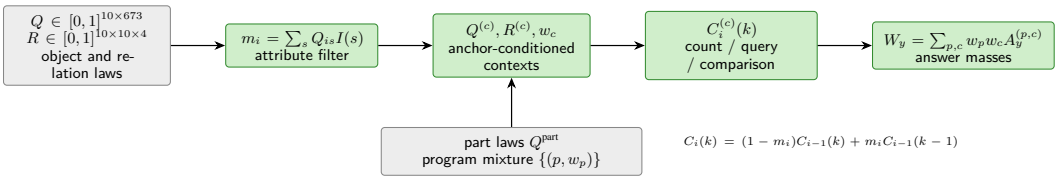

Figure 12: Scene-executor tensors. Attribute filters produce membership weights; conditioning on an anchor preserves dependencies for relations and shared attributes. Each context contributes an unnormalized answer mass, summed over contexts and retained programs. The count branch uses a Poisson-binomial recurrence.

value from the question: a slot points at one question token or at nothing, and a pointed token scores each value by the cosine between the backbone's frozen input embedding of the token and those of the value's surface forms, plus a learned correction. The surface forms are the value names, the size and material synonyms of the public Super-CLEVR synonym file (big, tiny, metallic, shiny, matte), the generator's category nouns (bike, motorcycle, airplane, plane) and plurals. Three priors use only the question and these forms. The pointer of block b prefers words followed by b relation words, which is the postfix order of the program, and the second half of a comparison is numbered after the first. A slot reads only a word that names a value of its own attribute group. A skeleton is masked unless it has one relate per relation word, one part key or role per part word and at least as many attribute slots as attribute words, a rule that holds for every reference program in training and development. A coverage loss with weight 0.3 asks each value word to be read by exactly one slot of the chosen skeleton, all 111 skeletons are scored in every update, and the part adapter uses learning rate 5×10^{-4}.

The exact scene executor The scene belief has $K = 10$ slots, each a categorical over 672 joint attribute states plus absent, independent relation Bernoullis $r[j, k, \text{rel}]$, and per-slot part laws. Object sets are per-slot membership masses. A `filter` multiplies by an indicator over states (coarse shapes are unions of fine ones). A `unique` opens one context per candidate slot j, conditioning slot j on membership and the others on non-membership, with legal weight equal to the mass of exactly one member; nothing is divided. `same_attr` adds the anchor's attribute value to the context so that the anchor's own state and the compared slots are conditioned consistently. `count` is an unnormalized Poisson-binomial dynamic program over per-slot masses; `exist` is one minus the all-absent product; comparisons of two counts use a joint two-count dynamic program; part functions condition on the owning slot's part law. Every output term is a sum over contexts of a product of per-slot sums in which each declared simplex appears once, so the output is the multilinear extension of the discrete executor. Behavioral tests check vertex agreement with the discrete executor on real questions, agreement with a float64 brute-force enumerator on small scenes for one program per family, and the pushforward property; the mean-field executor passes the vertex test and fails the pushforward test. The mean-field executor propagates per-slot membership marginals, normalizes `unique` by the total, and counts by the Poisson-binomial over marginals, following the quasi-symbolic execution of NS-CL (Mao et al., 2019).

Figure 12 shows the distribution tensors and count contraction. Exactness here is conditional on a retained program: selecting a top-eight program mixture remains an approximation over program uncertainty.

Readouts Soft selects the answer with largest legal mass. Commit selects the most probable values for presence, attributes, relations, parts and the program, then runs the same executor with these one-hot arguments. Failed uniqueness conditions count as errors. The small-scene test contains 5,507 count and 10,535 non-count questions. The large-scene test contains 1,008 count questions in the familiar range, 2,913 above that range, and 6,434 non-count questions. Slices are formed only after predictions are frozen.

Equal-data comparison All three methods use 8,192 retained training presentations (Table 7). SFT receives reference arguments; SPACE and GRPO receive answer supervision. SPACE addition-

ally uses the public lexical and structural priors above. Its adapter design was checked on development scenes with three to six objects and on questions generated for development scenes with eight to ten objects. SFT and GRPO set their learning rates on development scenes with three to six objects only. No method used test scenes or selected a checkpoint, and all use the fixed 8,192-presentation mark. The skeleton mask holds for every training program on its own. Because only SPACE was checked on larger development scenes, its larger-scene margins may be optimistic. On the larger scenes, the paired Soft-minus-GRPO difference is 6.90 percentage points (95% scene-bootstrap interval [6.01, 7.77]), Soft-minus-SFT is -1.07 points ($[-2.10, -0.08]$) and Soft-minus-Commit is 7.75 points ([6.99, 8.52]). On the small scenes, Soft-minus-SFT is -10.82 points ($[-11.59, -10.08]$) and Soft-minus-GRPO is 5.32 points ([4.65, 6.01]). Intervals use 2,000 paired resamples of the 1,540 small or 1,086 large scenes. Retaining distributions therefore improves the paired readout, SPACE generalizes better than GRPO, and SFT keeps a small OOD accuracy advantage in aggregate. The Gemma arm copies the Qwen recipe with read layer 24 (Table 6).

Where SFT's aggregate lead comes from Table 19 slices the same frozen larger-scene predictions by what the question asks: relation chains are the deep count and deep query templates, and part attributes are the part query, count, equality, comparison and existence templates. Slices are formed after prediction, and test labels enter only the correctness flag. SFT's argument target serializes the program and the ground-truth part attributes of every object matched to the public listing by shape and box overlap, so SFT is the only method with part-attribute supervision. It leads on every part-attribute template and on no other. SPACE leads on count answers above the training range, on familiar counts in the larger scenes, and on relation-chain templates, the axes along which the compiled program composes readings. Argument SFT runs the same compiled executor on the program it writes, so the executor performs the count. SFT writes an invalid call on 16% of larger-scene questions, and on 31 to 37% of the relation-chain templates; all count as errors.

Table 19: Larger-scene test questions by slice at 8,192 training presentations, Qwen3.5-4B, one seed. The difference is SPACE Soft minus argument SFT with a 95% paired scene-bootstrap interval over 1,086 scenes.

slice	n	SFT	SPACE Soft	Soft − SFT
all questions	10,355	0.676	0.666	-1.1 [-2.1, -0.1]
count answers 4 to 10	2,913	0.540	0.615	$+7.4$ [$+5.9$, $+9.0$]
count answers 0 to 3	1,008	0.617	0.677	$+6.0$ [$+3.3$, $+8.6$]
relation chains	3,576	0.551	0.711	$+16.0$ [$+14.7$, $+17.3$]
part attributes	6,779	0.743	0.642	-10.1 [-11.3, -8.8]

Table 7 and Figure 4 evaluate probabilistic answer quality with multiclass Brier on 128 shared larger-scene questions, retaining failed calls. This score measures both calibration and discrimination. The accuracy evaluations retain their original pools. A longer SPACE run reaches 0.709 on the small scenes and 0.673 on the larger scenes at 32,768 presentations, outside the matched comparison.

H GSM8K

Data The common pool contains 3,396 original GSM8K training questions with verified operation and operand-reference targets; all methods receive the same 8,192 samples in the same order, with no generated variants or inherited task-supervised checkpoint. Reference programs supervise SFT only; SPACE and GRPO receive final answers. Selecting for target availability restricts the training population, while the official test retains all 1,319 questions, including unsupported inputs.

Program and adapter The arithmetic operations keep a distribution over exact fixed-point values and sum the probability of every assignment that reaches the same value (Figure 13). The program has ten instruction steps with pointers to text numbers, fixed constants and previous results. Four latent tokens per step collect operation and routing information through backbone attention. The executor carries bounded candidate value distributions and sums equal answers, so this multistep top-K computation approximates the full program marginal. Commit selects the most probable operation and operand pointer values and uses these one-hot inputs in the same arithmetic kernels.

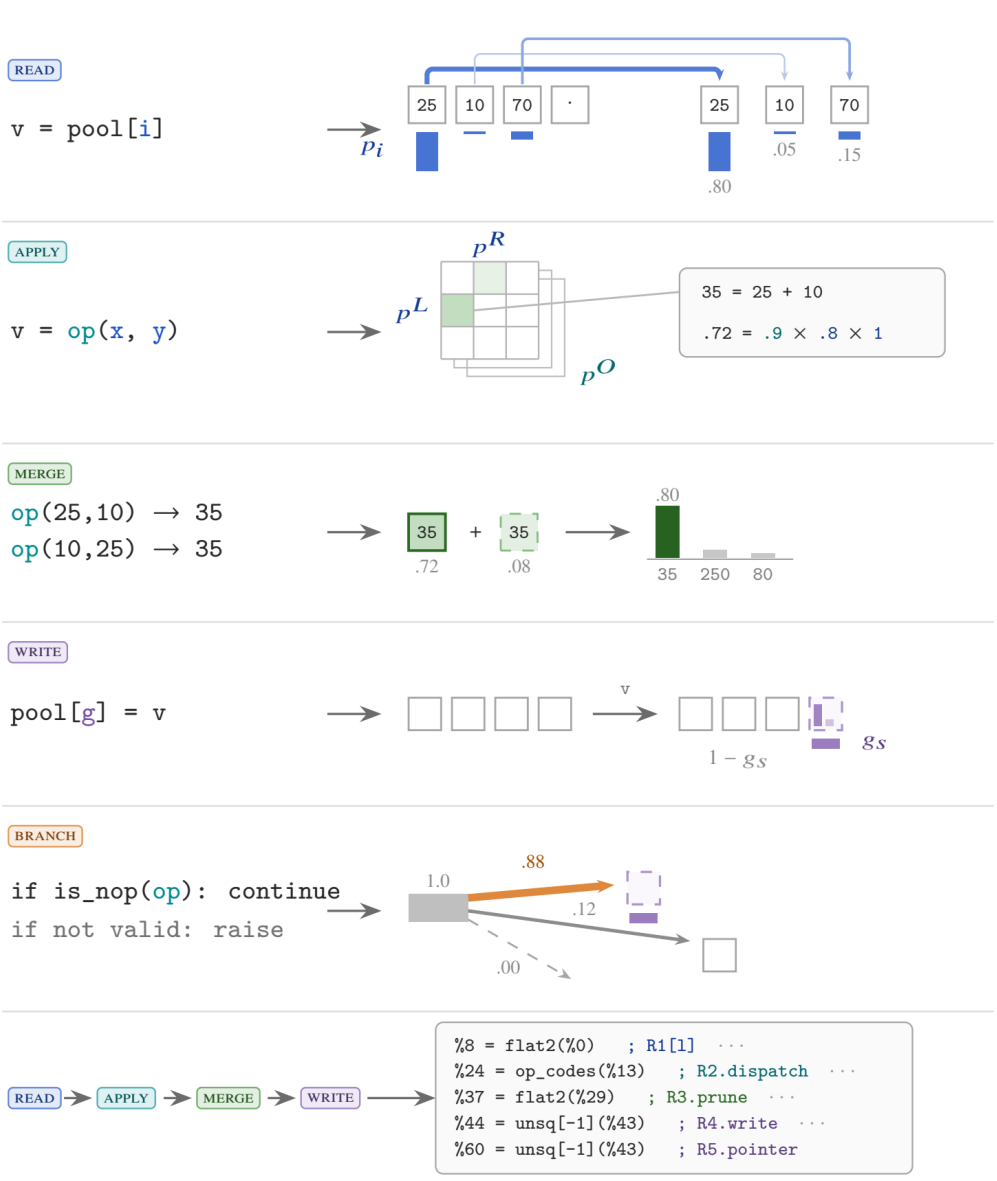

Figure 13: Lowering symbolic operations to tensor operations on the GSM8K arithmetic program. READ indexes a value pool with a soft pointer p_i, producing a distribution over values. APPLY multiplies operation and operand beliefs to weight each result. MERGE sums probability on equal values across assignments. WRITE places the result distribution into the pool at a soft position. BRANCH routes probability mass between continuing, skipping (NOP) and halting (invalid). The bottom row shows the compiled ONNX trace: each named stage maps to a block of fixed tensor operations with no learned weights.

Text baseline The `calculate` argument is one arithmetic expression over the literal numbers of the question (Table 4). SFT targets are the reference programs rewritten as such expressions; every target returns the training answer on the compiled program.

Results At 8,192 samples, GRPO exceeds SFT by 21.68 percentage points on Qwen (question-paired 95% interval $[19.11, 24.26]$) and by 17.36 points on Gemma ($[14.78, 19.94]$); SPACE Soft trails SFT by 60.50 points ($[57.77, 63.15]$) and 58.30 points ($[55.50, 61.03]$). At the clock budget set

by SPACE's full training time, the ordering on Gemma reverses (SFT 0.604, GRPO 0.119), because after 80 training questions the GRPO policy still answers in text instead of calling the tool on most questions. Qwen Soft improves accuracy over Commit by 1.14 points ($[0.15, 2.12]$), whereas the Gemma difference is unresolved (-0.53 $[-1.67, 0.53]$ points). Table 7 evaluates complete answer distributions using multiclass Brier on 128 shared test questions, retaining failures.

Wiring diagnostics A separate GSM-Symbolic diagnostic, with an earlier adapter recipe, compares number reading, operation choices and operand routing. The number reader's top value agrees with the compiled extractor on 0.91–1.00 of operands across question sizes. Probability on answer-consistent wiring is 0.338 for three numbers, 0.108 for four, 0.015 for five and 0.042 for six; operation-choice mass does not show this decline. Four-number training-template accuracy reaches 0.307 while held-out-template accuracy is 0.035. In that diagnostic, number reading holds while the probability of answer-consistent wiring is low beyond three numbers.

I SATBench

Data An input-only audit found 110 normalized condition sets shared between the source and held-out data, covering 206 training rows; those rows were removed before reserving development and removing further repeated conditions, leaving 13,031 training rows disjoint from development and test. Of these, 12,913 (Qwen) and 12,917 (Gemma) fit the reader's tokenizer and span limits and form the training order. The audit compares condition text and IDs with saved hashes. Every method uses the first 3,200 presentations of the frozen training order and is evaluated on all 2,100 test questions, which are balanced between SAT and UNSAT.

Compiled satisfiability search The adapter assigns a distribution to each literal slot within each condition span, including activity, sign and entity coordinates (Table 6). Compiled token operations detect conditions; predicted nonempty clauses are packed before padding to avoid introducing false empty clauses. The DPLL network has capacity for 130 variables and 130 clauses, nine literals per clause and 31 bounded search steps (Figure 14). An unresolved search or a formula beyond those bounds counts as an error.

The retained formula mixture Soft normalizes weights within the top-32 formula beam. This approximates the full formula marginal, and probability mass outside the beam is recorded separately. A retained hypothesis that exceeds the solver budget makes the mixture unresolved. Training minimizes the negative log probability of the final verdict through this same mixture.

Text baselines The text baselines write one `cnf` call with one clause per condition and named literals (Table 4), and the compiled solver executes the parsed clauses. Argument SFT reaches 0.981 on Qwen and 0.970 on Gemma, and text GRPO 0.940 and 0.747.

Answer-only training failure SPACE trains fresh task adapters and reader LoRAs from SAT/UNSAT answers only, without formula supervision. After 400 updates, Soft and Commit each score 0.4981 on both backbones. Table 7 reports multiclass Brier on 128 shared questions, including failures; its evaluation pool is distinct from this full-pool accuracy result. Both readers predict SAT on 2,092 questions; eight input-preparation failures remain errors. The recorded task-loss gradient is zero on every update: all retained formula hypotheses return the same verdict, making their normalized verdict mixture locally constant. A toy check with opposite-verdict hypotheses confirms that the loss can carry gradients when the retained support distinguishes the answers. This diagnoses a failure of the tested truncated-mixture training procedure.

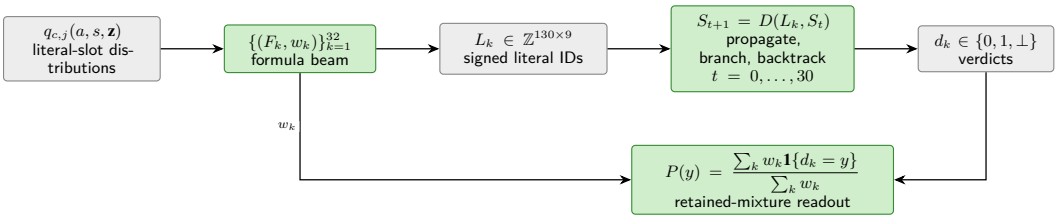

Figure 14: SAT tensor program. Literal-slot distributions yield weighted formula hypotheses, packed clause tensors enter bounded DPLL state updates, and verdicts are aggregated with the retained formula weights. The symbol $\perp$ denotes an unresolved search.

J Result-token readback and language retention

Training We freeze each Table 7 SPACE Soft reader and pretrained backbone, training only a learned result bridge and new LoRAs above the read layer. CLUTRR on both backbones and GSM8K Qwen continue matching-reader speakers; the other fits start fresh. Both CUB speakers use the final trained readers (seed 0, epoch 40), with their reader LoRAs frozen. The continued CLUTRR and GSM8K fits inherit 128 response-text updates (batch two); CLUTRR Qwen also inherits 512 answer-balanced updates (batch four, 50% independently chosen results), and GSM8K Qwen inherits 1,536 independent numeric-result updates (batch eight). All remaining readback updates use results chosen independently of the prompt, with effective batch eight (four for SATBench). Categorical prompts are answer-balanced; 50% of synthetic laws have close competing alternatives, increased to 80% in both Super-CLEVR continuations. The remaining laws are smoothed one-hot codes (exact one-hot in the CUB continuations). GSM8K instead uses exact signed numbers encoded by the program's numeric features; the final Gemma continuation broadens the sampled exponent range. Each Gemma lineage triggers one 200-update recovery block when development perplexity exceeds 30, alternating 100 pretraining-text updates with 100 updates of its task/result recipe; no Qwen lineage triggers recovery. The table gives cumulative readback plus mixed updates, excluding rejected branches.

Dataset	Qwen	Gemma-E4B
CLUTRR	2,688 + 0	2,176 + 200
CUB-200	6,144 + 0	6,144 + 200
Super-CLEVR	10,240 + 0	8,192 + 200
GSM8K	3,712 + 0	6,144 + 200
SATBench	1,024 + 0	1,024 + 200

We select maximum emitted-result adherence on speaker-validation pools of 512 examples for CLUTRR and Super-CLEVR, 599 for CUB-200, 446 for GSM8K, and 128 for SATBench, retaining the first candidate in a tie. No test score selects a speaker. At inference the speaker receives only the compiled program's own same-forward result; missing results are handled by the adherence definition below. These speaker updates are outside the equal-data budget of the fixed program readers. Appendix A.3 describes the result token itself.

Perplexity protocol We score the first 128 WikiText-103 (Merity et al., 2017) raw development paragraphs whose stripped text contains at least 200 characters, in source order, independently truncated to 512 tokens. Each paragraph begins a fresh context; BOS is explicitly prepended when defined by the tokenizer, and padding is excluded. Perplexity is $\exp(\sum_t -\log p(x_t \mid x_{<t})/N)$ over all scored next tokens, not an average of per-batch perplexities. The same raw paragraphs are used for every backbone, whose token counts differ. The test measurement uses the corresponding first 128 paragraphs of the separate test split. Mixed replay uses training paragraphs only, excluding exact token-sequence matches to either held-out sample. Because the mixed-recovery updates train on WikiText-103 training paragraphs, the same source as the scored paragraphs, post-recovery Gemma E4B test perplexities (13.02–15.92) fall below the base model's 102.12; they show that next-token prediction on this text is restored, not that general language ability is retained. Qwen checkpoints received no mixed-recovery updates, so their test perplexities (12.80–15.06, against 12.75 for the base model) reflect the interface alone, which for CUB-200 includes the trained reader LoRA (Table 21).

General-text perplexity exercises ordinary continuation through the full backbone, without requesting a task result. We separately retain qualitative training-split generations with the result token enabled and the program restricted to the public prompt; these are demonstrations rather than a new task metric. Perplexity alone does not establish that freely generated answers agree with the compiled result or measure general instruction-following quality; Table 20 measures the agreement directly.

Virtual-token adherence We generate greedily on each Table 7 checkpoint's test pool after separately fitting its speaker, with the reader fixed. We use the full test pools: 826 CLUTRR OOD stories, 5,794 CUB images, 10,355 Super-CLEVR OOD questions, 1,319 GSM8K questions and 2,100 SAT-Bench questions. Adherence counts generated answers that match the same-forward program result among examples that emit a result token; no-token examples are counted separately. Direct accuracy scores generated text against the reference on all examples, including no-token examples. Numeric answers are compared as exact decimal values. All ten task–backbone pairings have a row. Reader

Table 20: SPACE Soft result-token readback on the Table 7 test pools. Adherence conditions on an emitted result token; direct generated-text accuracy uses all n examples. Brackets are 95% public-group bootstrap intervals.

Task	Backbone	Adherence [95% CI]	No token	Direct accuracy [95% CI]	n
superclevr	gemma	0.9899 [0.9878, 0.9919]	565	0.6474 [0.6366, 0.6583]	10355
cub	qwen	0.9971 [0.9955, 0.9983]	0	0.8652 [0.8561, 0.8737]	5794
superclevr	qwen	0.9930 [0.9913, 0.9947]	468	0.6668 [0.6555, 0.6778]	10355
cub	gemma	0.9905 [0.9879, 0.9929]	0	0.8174 [0.8076, 0.8276]	5794
clutrr	qwen	0.9952 [0.9903, 1.0000]	0	0.7809 [0.7518, 0.8087]	826
clutrr	gemma	0.9939 [0.9879, 0.9988]	0	0.7361 [0.7046, 0.7664]	826
gsm8k	qwen	0.9955 [0.9917, 0.9985]	0	0.0250 [0.0167, 0.0341]	1319
gsm8k	gemma	0.9985 [0.9962, 1.0000]	0	0.0250 [0.0174, 0.0341]	1319
sat	qwen	1.0000 [1.0000, 1.0000]	8	0.4981 [0.4767, 0.5186]	2100
sat	gemma	1.0000 [1.0000, 1.0000]	8	0.4981 [0.4767, 0.5190]	2100

hashes match Table 7, but CUB program predictions differ from archived outputs on 6 of 5,794 Qwen images and 19 of 5,794 Gemma images. On the 599-image validation pool, matching the batch size and disabling TF32 matrix multiplication removes these answer differences for both backbones, but small confidence differences remain. Replay differences also occur on CLUTRR and GSM8K, and their causes remain unresolved. Full-pool generations keep their original settings. Direct accuracy scores the frozen generated text independently.

Table 21: WikiText-103 development and test perplexity for zero-shot backbones and selected SPACE speakers. Each split uses 128 raw paragraphs; brackets are 95% paragraph-bootstrap intervals. Lower is better.

Dataset	Model	Zero-shot dev	SPACE dev	Zero-shot test	SPACE test
CLUTRR	Qwen	10.96 [10.13, 11.90]	11.16 [10.32, 12.11]	12.75 [11.53, 14.11]	13.02 [11.78, 14.39]
CLUTRR	Gemma-E4B	83.55 [74.25, 94.50]	11.06 [10.21, 12.06]	102.12 [88.07, 119.21]	13.02 [11.67, 14.57]
CUB-200	Qwen	10.96 [10.13, 11.90]	12.53 [11.57, 13.61]	12.75 [11.53, 14.11]	14.27 [12.89, 15.80]
CUB-200	Gemma-E4B	83.55 [74.25, 94.50]	11.48 [10.60, 12.50]	102.12 [88.07, 119.21]	13.52 [12.13, 15.10]
Super-CLEVR	Qwen	10.96 [10.13, 11.90]	12.70 [11.69, 13.84]	12.75 [11.53, 14.11]	15.01 [13.56, 16.63]
Super-CLEVR	Gemma-E4B	83.55 [74.25, 94.50]	13.14 [12.10, 14.34]	102.12 [88.07, 119.21]	15.92 [14.35, 17.72]
GSM8K	Qwen	10.96 [10.13, 11.90]	12.53 [11.57, 13.62]	12.75 [11.53, 14.11]	15.06 [13.63, 16.62]
GSM8K	Gemma-E4B	83.55 [74.25, 94.50]	11.81 [10.89, 12.87]	102.12 [88.07, 119.21]	13.86 [12.43, 15.51]
SATBench	Qwen	10.96 [10.13, 11.90]	11.00 [10.17, 11.94]	12.75 [11.53, 14.11]	12.80 [11.58, 14.17]
SATBench	Gemma-E4B	83.55 [74.25, 94.50]	11.33 [10.45, 12.35]	102.12 [88.07, 119.21]	13.36 [12.01, 14.93]