When Output Budgets Masquerade as Representation Failures in Program-Based Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Comparisons between structured output formats usually give every format the same generation-token cap. When two formats serialize the same computation at different lengths, an equal cap is not an equal opportunity to finish. We show how to identify this confound inside one fine-tuned program representation with paired cross-budget trajectories, and what it hides. In a program-based mathematical-reasoning benchmark whose 512-token evaluation appeared to show flat JSON programs collapsing on five-call problems while compositional plans did not, we re-decode the same three flat checkpoints at 512 and 1536 tokens, with and without JSON-schema constraints, in 24 arms with 19,752 predictions. On the five-call slice the 512-token cap ends 65.7% of skeleton-disjoint and 68.5% of augmented outputs by length; at 1536 tokens no output is cut, parse rates rise from 35.2% and 31.7% to 100%, execution by 60.0 and 63.2 percentage points, and accuracy by 19.0 and 22.8 points. Three tests locate the mechanism: truncating the saved 1536-token trajectories at 512 tokens reproduces the short-cap five-call accuracy exactly and 98.6% of row-level outcomes; exact-prefix pairs carry 247 of the 253 net correct gains; and offline estimates for 640, 768, and 1024 tokens, registered before decoding, match 97.9% of decoded correctness outcomes. Tripling the cap adds 14% generated tokens on the five-call slices. Validity recovers in every held-out five-call skeleton and on all 368 length-finished four- and six-call rows, while the accuracy gain is carried by one skeleton: truncation had turned wrong computations into validity errors. The same censoring appears in every seed on all 1,319 GSM8K test problems, including 206 whose audited reference computation has no training structure, and in a second model family, Llama-3.1-8B-Instruct, with offline truncation predicting at least 98.0% of row outcomes. Executable and semantic audits find 43 test rows whose labels contradict the text and 110 whose quantities contradict a premise of the question; on the rows that pass both, the five-call accuracy gains are 27.8 and 43.7 points. A semantics-preserving inliner shows that the historical converter changed the returned value of 6.0% of training targets. Retraining Qwen3.5-9B on matched targets in both formats and decoding each to saturation leaves a compositional five-call advantage of 10.5 and 9.2 points over the inliner's flat serialization. It is positive in every seed, under exact matching, on average over audited rows, and with any skeleton or source left out; on the GSM8K test split the advantage is 4.1 points. Format comparisons therefore need per-format saturation checks before any representation-level interpretation.

1 Introduction

Program-based reasoning asks a model to emit an executable program (Gao et al., 2023; Chen et al., 2023; Gou et al., 2024; Li et al., 2024), so its evaluation couples two processes: the model must choose a correct computation, and the decoder must let the whole program reach a valid end. A common maximum-output-token setting looks like a controlled inference budget, but formats can need very different numbers of tokens for the same computation: a verbose format may hit the ceiling while a compact one stops naturally, and if truncated outputs are scored as invalid, the validity gap is read as a representational weakness. This problem arises when comparing a flat inlined program

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.

format in JavaScript Object Notation (JSON) with a compact compositional plan that calls verified arithmetic subroutines. Under a common 512-token cap, flat programs appeared to collapse at five calls; this apparent gap makes it necessary to distinguish budget censoring from computational error before comparing representations.

Studies of output budgets compare aggregate accuracy, preferred models or prompts, and rankings across caps (Sun et al., 2025; Hochlehnert et al., 2025; Guedes de Souza & Panisson, 2026), debate how far output limits explain a reported reasoning collapse (Shojaee et al., 2025; Lawsen, 2025; Dellibarda Varela et al., 2025), and, concurrently, relate a prompted-JSON penalty to the cap (Fan, 2026). We ask, for a *learned* output representation, whether a validity gap is censoring or error, and what the censoring conceals. Our instrument differs in three ways: it tests continuation on paired cross-budget trajectories, the compared formats are trained rather than prompted, and a corrected conversion with a conditional equivalence proof controls target semantics and exposes non-equivalent historical targets.

Contributions.

1. **A trajectory-level test of budget censoring.** Within one checkpoint, raising the cap from 512 to 1536 tokens removes every five-call length finish and raises execution by 60.0–63.4 seed-mean points. If the cap only cuts the continuation, truncating saturated trajectories offline must reproduce the censored arm; in §4.2 it reproduces the five-call accuracy exactly and 98.6% of row outcomes, exact prefixes supply 247 of 253 net correct gains, and forecasts registered before decoding at 640–1024 tokens match 97.9% of correctness outcomes.

2. **A structure-level separation of validity from correctness.** In §4.3, execution recovers in all six held-out five-call skeletons and on all 368 length-finished four- and six-call rows of nine further skeletons, while the accuracy gain is carried by one skeleton. Censoring recurs on 1,319 new GSM8K problems, 206 of them with new structures.

3. **A semantic-equivalence control.** A flat inliner with a conditional proof that it returns the compositional value and type, applied to the 4,795 training plans, shows that the historical converter changed 286 returned values, gives the flat step count, and shows before training that no corrected five-call target fits a 512-token cap. With matched targets and per-format saturation, compositional plans keep a 9.2–10.5-point five-call advantage over the inliner's flat serialization in every seed, on average over clean rows, and with any skeleton or source left out.

4. **Audited data and well-defined diagnostics.** Executable and semantic audits separate metadata mismatches, wording ambiguity, and premise violations from model errors; per-slot rates and isolated gold repairs replace a slot-count ratio and a gold-selected union.

2 Related Work

Formats and constrained decoding. Format restrictions can lower reasoning performance (Tam et al., 2024), though prompt and parser choices explain part of that gap (Kurt, 2024; Lee et al., 2026), and constrained decoding helps base but often hurts instruction-tuned models (Schall & de Melo, 2025). Su et al. (2026) compare code- and comment-based reasoning under proportional budgets. Constrained decoders guarantee well-formed outputs (Geng et al., 2023; Willard & Louf, 2023; Dong et al., 2025; Park et al., 2024) and are benchmarked on JSON schemas (Geng et al., 2025); validity can rise while accuracy falls (Ray, 2026), and TruncProof closes JSON within a hard token limit (Kato & Tarashima, 2026). Our formats are trained, not prompted; Appendix F extends this.

Program representations and libraries. Program-aided reasoning executes generated programs (Gao et al., 2023; Chen et al., 2023; Gou et al., 2024; Li et al., 2024; Stein et al., 2025), and library learning shortens them by naming reusable operations (Ellis et al., 2021; Grand et al., 2024; Stengel-Eskin et al., 2024; Wang et al., 2024); modularity alone need not help (Kang et al., 2024). When compared representations differ in length, a common cap is a format-dependent censoring threshold that held-out-structure splits (Keysers et al., 2020) do not control.

3 SETUP

Representations and execution. A compositional *DataflowPlan* is a typed directed acyclic graph of calls to 31 verified arithmetic primitives; each argument binds to a problem quantity, a previous call output, or a constant, and a final reference selects the answer. The flat format expands the computation into explicit typed input slots and compute and output steps; Figure 2 shows both for one problem. Both are executed deterministically and scored against the gold answer with 1% relative tolerance; exact matching moves only two augmented means, by 0.3 points, as Appendix A reports. *Parse rate* is the fraction of outputs accepted as complete JSON programs, *execution rate* the fraction that also execute without error, and *accuracy* the fraction that execute to the gold answer. A *length finish* is a decoder stop caused by the token cap rather than an end-of-sequence token. We index difficulty by the number of calls in the gold plan; this *call count* differs from the longest dependency path when calls are independent, which §4.3 examines.

Data and checkpoints. The data start from 565 plans for training problems of Grade School Math 8K (GSM8K) (Cobbe et al., 2021), proposed by Qwen3.5-27B and kept only if a deterministic executor reproduces the reference answer. Quantity perturbation yields 4,795 training examples. Splitting precedes augmentation: 68 held-out plan skeletons, that is, sequences of primitives with their wiring pattern, expand to a skeleton-disjoint test set of 507 examples, 35 of which have five calls; a high-call-count augmented set has 1,139 examples, 341 with five calls, obtained by further quantity perturbation of held-out problems from 23 of the 68 held-out skeletons. No test skeleton, taken as an ordered sequence, and no number-masked problem text occurs in training; up to reordering independent calls, one three-call held-out skeleton with 12 of 507 rows equals a training skeleton, as Appendix C details. Both five-call slices derive from the same nine source problems in six skeletons; the augmented slice holds their perturbed variants, and all 35 skeleton-disjoint five-call texts recur in 41 augmented rows, so the two slices are not independent replications. We fine-tune Qwen3.5-9B (Qwen Team, 2026) on flat targets with low-rank adaptation (LoRA) (Hu et al., 2022) of rank 32 and α=64 for three epochs, with seeds 42, 123, and 456.

Data consistency audit. The scorer reads quantities from the problem text, while the plans were verified on stored quantity metadata. We therefore execute every gold plan on the text-extracted quantities and call a row *consistent* only if the result matches the stored answer to a relative tolerance of 10^{-6}, not merely within the 1% scoring tolerance, and every quantity the plan uses has the same value in the text as in the metadata. This fails for 32 of 4,795 training rows, 3 of 507 skeleton-disjoint rows, and 40 of 1,139 augmented rows; 38 test rows fail by answer and 5 by inputs only, and all failing test rows but one are five-call rows, 41 of S2 and one of S6. In all 75 failing rows the text holds the stored values in permuted positions, because perturbation replaced the first remaining occurrence of each old number string, so the stored labels belong to a different problem than the one shown; Appendix D gives an example. A separate semantic review of the nine five-call source problems, whose classification rule and labels were fixed before any model-output count on them, finds one unit ambiguity, an S2 source that prices earrings per pair but counts sales in earrings, and 47 perturbed S1 and S4 rows whose stored answer is negative although the question presupposes a positive amount. A later premise screen of reference data adds unit bounds, such as at most 24 hours in a day, and flags 110 rows. We keep the registered labels in Tables 1–10 and report affected estimates also on consistent rows and on the *clean* rows passing all checks, 24 and 157 five-call rows per seed.

The re-decoding grid. For each seed and test set we decode the same merged checkpoint in four cells, $(B, C) \in \{512, 1536\} \times \{\text{unconstrained}, \text{JSON-schema}\}$ of cap and decoding mode, with greedy decoding and a 2816-token context; schema cells use the decoder's native structured-output backend. Checkpoint, prompt, inputs, seed, binding policy, and tolerance are identical across cells; the cap is the only configured difference within a budget pair. This gives 24 arms, 19,752 predictions, and 9,876 paired rows. All predictions keep their generated text, token count, and finish reason.

Statistical units. We report seed means with sample standard deviations over three training seeds, per-seed exact McNemar tests for paired correctness flips, and, because perturbed rows share skeletons, skeleton-level effects, skeleton-cluster bootstrap intervals with 10,000 resamples, and leave-one-skeleton-out ranges. Row-level tests describe the fixed rows, and generalization to new structures

Table 1: Five-call results for the same checkpoints in %, as mean ± sample standard deviation over three seeds. Within a budget pair only the output cap differs. **Len**: fraction terminated by the cap.

Test	Decoding	Cap	Accuracy	Parse	Execute	Len
Skeleton	Unconstrained	512	16.19±3.30	35.24±3.30	31.43±5.71	65.71±2.86
Skeleton	Unconstrained	1536	35.24±5.95	100.00±0.00	91.43±0.00	0.00±0.00
Skeleton	Constrained	512	16.19±3.30	34.29±2.86	31.43±5.71	65.71±2.86
Skeleton	Constrained	1536	37.14±2.86	100.00±0.00	91.43±2.86	0.00±0.00
Augmented	Unconstrained	512	15.25±3.06	31.67±2.80	30.21±3.74	68.52±2.49
Augmented	Unconstrained	1536	38.03±0.94	100.00±0.00	93.45±1.32	0.00±0.00
Augmented	Constrained	512	15.05±3.27	31.67±2.51	30.21±3.83	68.52±2.24
Augmented	Constrained	1536	37.63±1.35	100.00±0.00	93.65±2.06	0.00±0.00

is judged at the skeleton and source level, whose intervals hold the trained checkpoints fixed; seeds show training-run variation, and registered criteria are decision rules, not significance tests.

Evidence tiers. Tier T1 rules were fixed before decoding: forecasts and repeats E1–E2, GSM8K panel E5, retrained comparison E3 and its GSM8K run E7, and second model family E6. Tier T2 rules were fixed before any output of the run was available: panel budget curve E5b, three operational terms of E3, and two E7 additions. Tier T3 covers analyses of saved outputs and tier T4 post-output audits that read only reference data: the consistency audit, semantic review, premise screen, and subsets E5-SD-C through E5-SD-CA4 of Appendix B. Apart from E3 and E6, every flat checkpoint is historical, trained on historical targets that are 94.0% semantically exact by §5; §6 uses the historical compositional checkpoints.

4 Results

4.1 The output budget controls five-call validity

Table 1 gives the primary results. Under unconstrained decoding, the 512-token cap leaves 35.24% of skeleton and 31.67% of augmented five-call outputs parseable. At 1536 tokens every output stops naturally and parses; execution rises by 60.00 and 63.25 points and accuracy by 19.05 and 22.78 points. The effect holds in every seed: on the 341-row augmented slice the gain is 18.18–25.22 points with 1–4 rows flipping from correct to incorrect and 66–89 in the other direction, with exact McNemar $p \leq 1.7 \times 10^{-15}$. On the 35-row skeleton slice gains are 14.29–25.71 points and no row flips from correct to incorrect in any seed. Constrained arms show the same pattern. Appendix A lists every seed-level metric and paired test. The label defect of §3 hides part of the gain. On the 33 and 301 consistent five-call rows per seed, unconstrained accuracy rises from 17.2% to 37.4% and from 17.3% to 41.4%, by 20.2 and 24.1 points, at least 15.2 in every seed, and on the 42 five-call rows that fail the audit the 1536-token flat programs return the value of the gold plan executed on the text quantities in 122 of 126 seed-rows, against none at 512 tokens. On the clean rows accuracy rises from 16.7% to 44.4% and from 12.7% to 56.5%, by 27.8 and 43.7 points, at least 20.8 in every seed. Validity and accuracy gains differ in structural support: skeleton S2 supplies 20 of 20 and 225 of 233 net additional correct seed-rows, but 41 of 63 and 423 of 647 net additional executing ones.

4.2 Mechanism: truncated trajectories reproduce the short-budget arm

If the cap acted only by cutting the continuation the model would otherwise produce, then cutting the saved 1536-token outputs at 512 tokens would reproduce the 512-token arm. It nearly does. Re-tokenizing each long-budget response, keeping its first 512 generated tokens, and rescoring with the same evaluator predicts seed-mean five-call accuracy of 16.19% on the skeleton set and 15.25% on the augmented set, identical to the observed values, and parse and length-finish rates within 0.3 points of the observed ones, as the first rows of Table 3 show. Offline prediction matches observed correctness on 1,112 and execution on 1,109 of 1,128 unconstrained five-call rows, 98.6% and 98.3%.

Table 2 locates the change. Of the 770 rows that end by length at 512 tokens, 713 are exact prefixes of their long-budget continuation, and Appendix A shows one of them; they supply 247 of the 253

Table 2: Decomposition of the 512→1536 change on unconstrained five-call rows, seeds and test sets pooled, as 512/1536 counts. "Prefix": the 512-token output exactly prefixes the 1536-token one.

Finish at 512; prefix relation	Rows	Parsed	Executed	Correct	Net
Length at 512; 512 output is a prefix	713	3/713	1/655	0/247	+247
Length at 512; not a prefix	57	0/57	0/52	0/10	+10
Natural stop at 512; prefix	322	322/322	309/309	164/164	+0
Natural stop at 512; not a prefix	36	36/36	32/36	9/5	−4
All five-call rows	1128	361/1128	342/1052	173/426	+253

Table 3: Five-call length finish, parse, and accuracy in %, as unconstrained seed means, by cap. "Offline" rows truncate the saved 1536-token trajectories and rescore; they were registered before the 640–1024 decoding runs. "Decoded" rows are actual runs.

	Skeleton-disjoint			Augmented		
Cap	Len	Parse	Acc.	Len	Parse	Acc.
Offline 512	65.7	35.2	16.2	68.3	32.0	15.2
Offline 640	11.4	88.6	29.5	12.3	87.7	30.6
Offline 768	0.0	100.0	35.2	0.3	99.9	38.0
Offline 1024	0.0	100.0	35.2	0.0	100.0	38.0
Decoded 512	65.7	35.2	16.2	68.5	31.7	15.2
Decoded 640	12.4	87.6	30.5	12.5	87.5	30.3
Decoded 768	0.0	100.0	36.2	0.3	99.9	38.2
Decoded 1024	0.0	100.0	36.2	0.0	100.0	37.9
Decoded 1536	0.0	100.0	35.2	0.0	100.0	38.0

net additional correct answers. The 322 rows that stop naturally at 512 and are exact prefixes have identical aggregate outcomes at both budgets. The constrained arms give the same decomposition: 714 exact-prefix length finishes supply 249 net correct answers. The 93 non-prefix pairs carry a net change of +6, so the budget effect is dominated by continuation. Run-to-run variation is small by comparison: two fresh 1536-token repeats of seed 42 change five-call accuracy by at most 0.9 points, 1.5 under the schema, against a budget effect of 19–23 points.

How large a cap is enough? Offline truncation of the saved trajectories also estimates the budget curve; we registered these estimates before decoding at 640, 768, and 1024 tokens. Table 3 compares them with the decoded arms, and Table 14 gives every criterion. A 640-token cap truncates 12.4% and 12.5% of five-call outputs against registered 11.4% and 12.3%. Caps of 768 and 1024 tokens leave at most 0.3%, and row by row the decoded arms match the offline estimates on 99.8% of finish reasons and 97.9% of correctness outcomes over 3,384 seed-rows. The output tail, up to 844 tokens, exceeds every five-call flat training target, whose maximum is 621.

Removing the censoring costs few generated tokens. Across all 24 arms the 512-token grid generates 4,072,393 tokens and the 1536-token grid 4,293,062, a 5.4% increase, because most outputs stop naturally well before either cap; on the five-call slices, where the cap binds, the increase is 13.7% on the skeleton set and 14.4% on the augmented set. At 1536 tokens, five-call flat programs have median length 563 and 588 tokens and maxima 755 and 844, as Figure 1a shows. The targets predict this: at five calls the mean training target has 389 tokens as a compositional plan and 552 as a flat program, and 88.9% of flat targets exceed 512 tokens while no compositional target does, so a common 512-token cap binds for one format only.

4.3 Structure-level analysis: validity recovers in every evaluated skeleton

Quantity perturbation multiplies rows but not program structures, so Figure 1b and Table 4 repeat the analysis per skeleton. Execution rises in all six five-call skeletons on the augmented set and in the five not already at 100% on the skeleton set. The skeleton-cluster bootstrap interval for the execution gain is $[22.2, 78.3]$ points on the skeleton set and $[26.3, 82.1]$ on the augmented set, and dropping

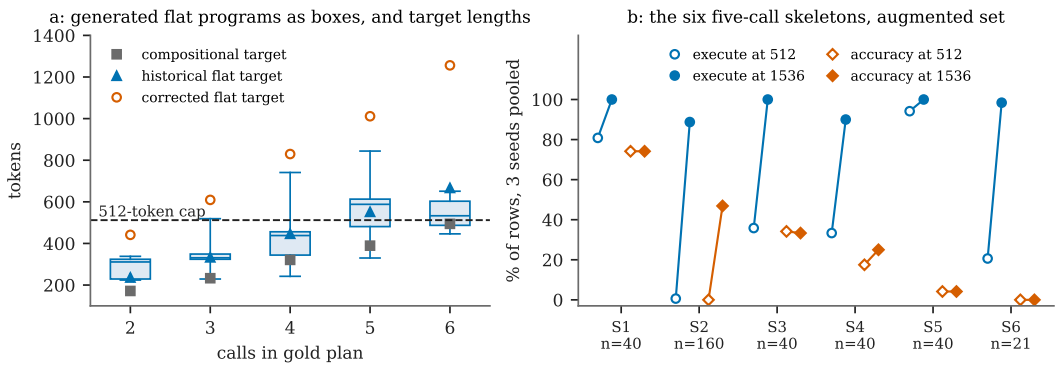


Figure 1: Panel a: natural lengths of generated flat programs at the 1536-token cap, unconstrained, over three seeds and both test sets, with whiskers from minimum to maximum, and mean training-target lengths for three serializations, by gold call count. Panel b: five-call execution and accuracy at 512 tokens as open and at 1536 tokens as filled markers for each held-out skeleton on the augmented set, pooled over three seeds, with the rows per seed given as n.

Table 4: Five-call execution and accuracy in %, from 512 to 1536 tokens, unconstrained and pooled over seeds, for each held-out skeleton. n: rows per seed. S2 combines four independent two-argument calls with one four-argument call; Appendix C defines all six skeletons.

	Skeleton-disjoint set			Augmented set		
Skeleton	n	Execute	Accuracy	n	Execute	Accuracy
S1	4	83.3→100.0	75.0→75.0	40	80.8→100.0	74.2→74.2
S2	16	0.0→85.4	0.0→41.7	160	0.6→88.8	0.0→46.9
S3	4	41.7→100.0	41.7→41.7	40	35.8→100.0	34.2→33.3
S4	4	25.0→83.3	25.0→25.0	40	33.3→90.0	17.5→25.0
S5	4	100.0→100.0	0.0→0.0	40	94.2→100.0	4.2→4.2
S6	3	33.3→100.0	0.0→0.0	21	20.6→98.4	0.0→0.0

any single skeleton leaves a gain of at least 38.6 and 41.3 points; weighting the six skeletons equally gives 47.6 and 52.0 points. By source problem, execution rises in all nine augmented sources and in the eight skeleton-set sources below 100%; weighting sources equally gives 60.2 and 64.0 points and leaving one out 54.8–67.7 and 58.4–70.9, as Table 27 lists.

Accuracy separates the structures. The largest gain is in skeleton S2, which forms 16 of 35 and 160 of 341 five-call rows and rises from 0.0% to 41.7% and 46.9%, and to 57.9% on the consistent augmented rows; S4 adds 7.5 points on the augmented set. The row-level accuracy gains of Table 1 thus describe this test distribution and are carried by S2, with 20 of 20 and 225 of 233 net correct seed-rows, all from its three sources whose wording has one reading; on its clean rows S2 reaches 66.7% and 86.4% at 1536 tokens. On the augmented set, S3 and S6 go from 35.8% and 20.6% execution to 100.0% and 98.4%, and S5 executes almost always at both budgets; for these three skeletons the long budget separates validity from correctness, as the completed programs keep accuracy at its 512-token level of about 34%, 0%, and 4%. Truncation had converted wrong computations into validity errors, and removing it reveals which structures the flat model actually solves.

Beyond the five-call slice. Pooling all call counts, length finishes at 512 tokens come from 16 skeletons, ten of them outside the five-call slice. All 368 length-finished four- and six-call seed-rows from nine skeletons execute at 1536 tokens and 180 are correct, so the separation of validity from correctness extends beyond the five-call skeletons. Call count, serialized length, and dependency depth are confounded here: S2 has only two dependency levels, and path length two supplies 305 of the net 396 additional correct augmented rows, as Appendix C shows.

New problems and new structures. On all 1,319 GSM8K test problems, which share no source text with the training data, a rule fixed before decoding and stated in Appendix H holds in every

seed: 13.3–15.5% of outputs end by length at 512 tokens and at most one row at 1536, and of the 559 seed-rows cut at 512, 557 parse and 523 execute at 1536, against 2 at 512. A model-free audit keeps the problems whose reference computation, rebuilt from the annotated GSM8K solution and checked by hand after decoding without model outputs, matches no benchmark plan structure. On these 206 problems each seed has 37–43 cut rows, all of which parse at 1536 and 35–40 execute, none at 512. Across all 1,319 rows, truncating the 1536-token outputs offline predicts the decoded 384-, 512-, and 768-token arms' finishes, execution, and correctness on 98.8–99.9% of rows in every seed and cap.

A second model family. Llama-3.1-8B-Instruct, fine-tuned by the same recipe on the same flat targets with three seeds, with rule and analysis fixed before any output as Appendix H records, shows the same censoring on the skeleton-disjoint, augmented, GSM8K, and structural panels: 6.9–37.5% of its 512-token outputs end by length, and of the 1,897 seed-rows cut there, 1,850 parse and 1,750 execute at 1536 tokens, against 6 at 512, 1,688 as exact continuations. Offline truncation of its 1536-token IDs predicts the 512-token arm's finishes, execution, and correctness on at least 98.0% of rows.

4.4 Consequence for the historical comparison

The original 512-token study reported five-call compositional versus flat execution of 95.2% versus 28.6% and accuracy of 25.7% versus 15.2% on the skeleton-disjoint set. The compositional arm is nearly saturated at 512 tokens, with 35 of 1,128 five-call outputs open at the cap, and the two formats' checkpoints come from different training runs on targets that §5 shows to be non-equivalent, so the historical gap is a fixed-budget system comparison, not an intrinsic flat failure.

A matched, saturated comparison. A registered retraining, E3 in Appendix H, removes both confounds: fresh compositional and corrected-flat checkpoints, three seeds each, are trained on the same 4,763 consistent rows with the same recipe and 894 optimizer steps, with no training sequence truncated, and decoded at 512 to 2048 tokens. The corrected flat targets are longer, so at 512 tokens 100% and 98.7% of five-call flat outputs end by length and the common-cap gap is 28.6 and 25.2 accuracy points. Each format saturates at its own cap, 512 and 1024 tokens for compositional plans and 1536 for flat programs, where compositional plans stay more accurate by 10.5 and 9.2 points, positive in every seed at 2.9–17.1 and 8.2–11.1 points and in the three skeletons that carry the difference, which meets the registered decision rule. No single unit carries the advantage: no skeleton has a negative pooled difference, leaving out any one skeleton gives 7.5–17.5 and 6.1–15.8 points and any one source problem 7.5–14.0 and 6.1–11.2, and a source-cluster bootstrap gives $[0.9, 20.2]$ and $[2.2, 17.1]$. As seed means, the difference is $+9.7$ and $+8.5$ points on the clean rows and $+10.5$ and $+10.3$ under exact matching, as Table 20 lists. On the 1,319 GSM8K test problems, where both formats saturate, accuracy is 20.7% versus 16.6%, a difference of 4.1 points that is positive in every seed at 2.7–5.8, also under exact matching, with problem-resampling interval $[2.5, 5.8]$ in Table 23. The comparison is between compositional plans and our flat serialization, which spends 2.2 times the plans' training tokens, under one recipe; within it, censoring inflated the five-call gap about 2.7-fold.

5 A Semantic Control for Format Comparisons

A representation comparison also needs targets that compute the same function in both formats.

Selected output, not last call. If a plan has independent calls $c_0 = \mathrm{ADD}(1, 2)$ and $c_1 = \mathrm{MUL}(3, 4)$ with final reference c_0, its answer is 3, but an inliner that returns the last call's output, as the historical converter does, yields 12. Argument casts and a primitive that emits a value and then overwrites its local variable also defeat textual substitution.

Construction and conditional equivalence. We consider deterministic straight-line plans whose calls have distinct identifiers and resolvable references to quantities, constants, or earlier call outputs; each primitive has typed slots, an ordered list of typed assignments over pure arithmetic expressions, and an explicit output. The inliner accepts a plan only inside this domain: finite scalar values, integers preserved by the executor's integer cast, quantity identifiers that are valid non-keyword names and shadow no built-in, explicit backward references, and expressions without lambdas, comprehensions,

or attribute access; anything else is rejected, not approximated. For each call in order, the construction materializes the same slot casts as the compositional executor, inlines every assignment to a fresh target after renaming local names in the expression syntax tree, leaving string literals untouched, records the fresh target of the call's output instruction as the call value, and finally emits the plan's explicit final reference, or the last call only when no final reference exists. Writing $E(P, q)$ and $F(P, q)$ for compositional and flat execution on quantities q, we have $E(P, q) = F(P, q)$, including the scalar type, whenever compositional execution succeeds and the flat step budget covers the expansion. The proof in Appendix G is a double induction over calls and, within a call, over assignments: predecessor outputs agree by the outer hypothesis, identical casts give identical initial locals, renaming reads exactly the current local values, and fresh targets preserve emitted values under later reassignment. If primitive j has a_j arguments and s_j steps and c literal assignments resolve constants, the flat program has $1 + c + \sum_j (a_j + s_j)$ steps. This counts our construction, and the equivalence holds for the given quantities q; whether q matches the problem text is audited in §3. All returned values and types also agree on the 293 constructed edge cases of Appendix E.

Audit of the actual training targets. Applied to the 4,795 training plans with the 31-primitive library, the corrected inliner reproduces the compositional result for every plan and rejects none. In 297 plans, or 6.2%, the final reference is not the last call; for 286 of them, 6.0% of all targets, the historical converter returns a different value, and 265 of those no longer match the gold answer; every difference arises from the final-reference defect. Evaluation scores against gold answers, so test conversion does not enter any reported accuracy. The historical flat checkpoints were thus trained on 94.0% semantically exact targets; this leaves the budget intervention, whose two arms share one checkpoint, unaffected, but no historical flat–compositional gap compares equivalent targets.

The corrected targets also make the length issue sharper. Mean training-target length at five calls is 389 tokens for compositional plans, 552 for the historical flat programs, and 1,011 for our corrected flat construction, with a maximum of 1,158 and every one above 512 tokens, as Figure 1a shows; at three calls corrected flat targets average 609 tokens. These lengths show before training that a cap cannot hold the targets, but a sufficient cap must be read off the outputs, whose tail can exceed them.

6 Compositional Error Diagnostics with Well-Defined Units

The historical analysis cited a quantity-binding "error rate" rising from 0.42 at two calls to 0.79 at five and a gold repair raising five-call augmented accuracy from 26.9% to 47.9%. We recompute both from the raw outputs of the three historical compositional checkpoints at the 512-token cap with the executor that reproduces the stored correctness of all 4,846 parsed outputs among 4,938.

Units. The historical statistic divides mismatched argument slots at position-aligned calls by the larger plan's call count. It is not a probability, since one call can contribute several mismatches and the ratio can exceed one, and it compares quantity references by identifier, pooling reference-kind changes with wrong choices. For plans with the gold call count, let M be the calls aligned by the call permutation that maximizes operator and argument agreement and S_i the argument slots of call i that are value references in both plans, with $b_{is} = 1$ if the bound values differ. We report

$$R_{\text{slot}} = \frac{\sum_{i \in M} \sum_{s \in S_i} b_{is}}{\sum_{i \in M} |S_i|}, \qquad R_{\text{call}} = \frac{1}{|M|} \sum_{i \in M} \mathbf{1}\Big\{ \sum_{s \in S_i} b_{is} > 0 \Big\},$$

together with the wiring error rate over call-output slots, comparing call references after mapping through the alignment, the operator-identifier mismatch rate per call, and coverage, the fraction of rows with the gold call count.

Table 12 shows that from two to five calls quantity-value errors rise with call count, from 0.096 of value slots at two calls to 0.292 at five on the skeleton-disjoint set and 0.336 on the augmented five-call slice, and 0.298 and 0.311 on the consistent rows, at less than half the magnitude the historical statistic suggested. With the arguments of symmetric primitives compared as multisets the rates are 0.078 and 0.232 at two and five calls on the skeleton-disjoint set and 0.270 at five calls on the augmented set, still rising with call count. Over two to five calls wiring errors are rare, at most 0.051 per call-output slot, operator identifiers differ from gold in 21–28% of calls from three calls on, and coverage falls to 55–61% at five calls. Six-call rows differ: only 2 of 24 and 71 of 240 have the gold call count, and on

Table 5: Repaired accuracy in % after isolated gold repairs of the compositional outputs, on the rows that pass the consistency audit of §3: all four-call rows and 99 of 105 and 903 of 1,023 five-call rows. N: pooled seed-rows; Elig.: percentage of rows eligible for repair. "All four": the four repairs jointly; "Gold plan": the gold plan on eligible rows. All rows: Table 13.

Slice	N	Elig.	Base	Quantity	Operator	All four	Gold plan
Skeleton, 4 calls	216	49.5	38.9	42.6	43.1	50.5	67.6
Skeleton, 5 calls	99	52.5	27.3	42.4	38.4	55.6	65.7
Augmented, 4 calls	2154	51.6	35.3	38.4	45.7	53.3	67.2
Augmented, 5 calls	903	55.4	29.6	41.7	36.7	50.2	67.4

the latter wiring errors reach 0.185 and operator mismatches 0.364. Mismatched structure is thus itself a major error class that a per-call rate does not capture.

Isolated repairs. Let B_i indicate baseline correctness and O_{ik} correctness after repair k. On rows with the gold call count we separately copy the gold quantity values, the gold wiring, the gold operators, or the gold final reference into the aligned prediction and re-execute; other rows keep their baseline outcome. A dataflow plan does not depend on its listing order, so repairs that copy call references execute the edited plan in a dependency-respecting order, and an edit that creates a cycle counts as a failed repair; at five calls, 13 repaired plans needed reordering and 2 had no legal order. We report the unconditional repaired accuracy $N^{-1}\sum_i O_{ik}$ over all N rows, not the union $N^{-1}\sum_i \max(B_i, O_{ik})$, which picks the answer by gold correctness and cannot fall below baseline.

Table 5 isolates the historical mixed repair. At five calls the quantity repair alone raises accuracy from 29.6% to 41.7% on the augmented set, improving 121 rows and degrading 11, and from 27.3% to 42.4% on the skeleton-disjoint set, improving 15 and degrading none. It is the largest single repair in every seed, at 39.2–44.5% on the augmented set against 29.2–42.5% for operator repair, which adds 7.1 and 11.1 points; at four calls operator repair is the larger effect, adding 10.4 points against 3.1 on the augmented set. On all rows the quantity repair reaches 47.5%, close to the historical 47.9%, but 103 of its 224 augmented improvements come from the 40 rows that fail the audit, where it moves text-correct bindings onto the stored values; the audited effect is about 60% of the unaudited one.

The gap between all four repairs at 50.2% and the gold plan at 67.4% consists of 152 rows with reference-kind errors and 4 with missing slots only; every other eligible row is repaired whenever the gold plan is correct. Only 3.3% of five-call augmented compositional outputs are open at the cap, so these diagnostics describe complete programs.

7 Discussion and Conclusion

Recommendations. Before reading a format validity gap, report length-finish rates by format and difficulty, check per-format saturation with a larger cap or offline truncation, count generated tokens on the binding slice, analyze each program structure, and execute gold programs on the scored inputs. Finish reasons and a budget sweep show where a cap binds; paired trajectories and structure-level results show what the censored programs would have computed.

Scope. The intervention concerns flat checkpoints of Qwen3.5-9B in every analysis and of Llama-3.1-8B-Instruct in the validity and trajectory tests, on one GSM8K-derived benchmark whose five-call slices come from nine source problems, and on the GSM8K test split. The format accuracy comparison uses Qwen3.5-9B, our inliner's flat serialization, and one recipe, and is claimed for the two test sets and the full GSM8K split, not for new-structure subsets. Compositional targets also copy each quantity's value and text span, which flat targets bind by name, so the advantage belongs to the two serialization systems, not to compositional structure alone.

Conclusion. A shared cap can make a longer serialization look like an invalid representation: raising the cap removes five-call truncation at 14% more generated tokens, continuation explains nearly all of the change, and matched, saturated targets leave a 9–10-point compositional advantage over our flat serialization.

AI USE STATEMENT

In this work, we used generative artificial intelligence (AI) tools to design and improve experiments and to implement methods. Formulating survey or interview questions and transcribing recordings are not applicable to this work. Additionally, we used generative AI tools to edit the paper to improve readability and to create or edit software code. We have reviewed all AI-assisted work: AI-assisted code was tested before use, and every reported number is computed from recorded experimental outputs by analysis scripts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects and no personal data. All experiments use publicly released language models and the public GSM8K benchmark; the derived training and test sets consist of grade-school arithmetic word problems derived from it. The consistency audit, the semantic review, and the premise screen identify benchmark rows whose stored labels do not match their text or whose quantities contradict a premise of the question; we report them as row-level annotations with the reasoning behind each class and do not alter the released benchmark. The work studies how evaluation settings affect measured validity and accuracy of program-generating models; we do not foresee harmful or dual-use applications beyond those of the underlying public models, and we have no conflicts of interest to declare. Our findings concern evaluation methodology, and we recommend per-format saturation checks so that comparisons between representations are not biased by output limits.

REPRODUCIBILITY STATEMENT

Section 3 specifies the representations, data construction, consistency audit, decoding grid, statistical units, and the tier at which each analysis rule was fixed. Appendix A gives the seed-level results of the re-decoding grid, Appendix B describes the GSM8K test panel, Appendix C the skeleton and source definitions, Appendix D the audit rules and row-level review classes, and Appendix E the training, decoding, and scoring protocol, including the retrained checkpoints. The inliner and its conditional equivalence proof are in Section 5 and Appendix G, and Appendix H lists every registered criterion and seed-level decision of the later runs, together with the stage at which each rule was fixed; these are internal registrations recorded by the authors. The anonymous supplementary material contains the data, the saved predictions of every decoding run, including those of the retrained comparison and the second model family with their generated token IDs, the training manifests and logs of the retrained checkpoints, the code for training, decoding, scoring, and analysis, and scripts that regenerate the tables and figures from the saved outputs.

REFERENCES

Aditya Basarkar, Benyamin Tabarsi, Tiffany Barnes, and Dongkuan Xu. Enhancing mathematical problem solving in LLMs through execution-driven reasoning augmentation. *arXiv preprint arXiv:2602.03950*, 2026. URL https://arxiv.org/abs/2602.03950.

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*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

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

Iñaki Dellibarda Varela, Pablo Romero-Sorozabal, Eduardo Rocon, and Manuel Cebrian. Rethinking the illusion of thinking. In Max Bramer and Frederic Stahl (eds.), *Artificial Intelligence XLII: 45th SGAI International Conference on Artificial Intelligence (SGAI-AI 2025)*, volume 16301 of *Lecture Notes in Computer Science*, pp. 116–129. Springer, Cham, 2025. doi: 10.1007/978-3-032-11402-0_9. URL https://doi.org/10.1007/978-3-032-11402-0_9.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/5c20ca4b0b20b0bd2f1d839dc605e70f-Abstract-Conference.html.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*, pp. 835–850, 2021. doi: 10.1145/3453483.3454080. URL https://doi.org/10.1145/3453483.3454080.

Hengxin Fan. Capacity, not format: Rethinking structured reasoning failures. *arXiv preprint arXiv:2606.09410*, 2026. URL https://arxiv.org/abs/2606.09410. First posted 8 June 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, 2023. doi: 10.18653/v1/2023.emnlp-main.674. URL https://aclanthology.org/2023.emnlp-main.674/.

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025. URL https://arxiv.org/abs/2501.10868.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ep0TtjVoap.

Gabriel Grand, Lionel Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.19791.

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

Rodrigo Guedes de Souza and Alison R. Panisson. Who thinks best depends on how long you let them: Budget-dependent rankings in LLM evaluation. *arXiv preprint arXiv:2608.12150*, 2026. URL https://arxiv.org/abs/2608.12150. First posted 12 August 2026.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=90UrTTxp5O.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan 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.

Deokyeong Kang, KiJung Seo, and Taeuk Kim. Revisiting the impact of pursuing modularity for code generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 11561–11571, 2024. doi: 10.18653/v1/2024.findings-emnlp.676. URL https://aclanthology.org/2024.findings-emnlp.676/.

Yoshio Kato and Shuhei Tarashima. TruncProof: A guardrail for LLM-based JSON generation under token-length constraints. In *International Joint Conference on Neural Networks (IJCNN)*, 2026. URL https://arxiv.org/abs/2605.13076. arXiv:2605.13076, first posted 13 May 2026.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygcCnNKwr.

Will Kurt. Say what you mean: A response to 'let me speak freely'. .txt engineering blog, 20 November, 2024. URL https://blog.dottxt.ai/say-what-you-mean.html. Non-peer-reviewed technical response.

A. Lawsen. Comment on The Illusion of Thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.09250*, 2025. URL https://arxiv.org/abs/2506.09250.

Ivan Yee Lee, Loris D'Antoni, and Taylor Berg-Kirkpatrick. The format tax. *arXiv preprint arXiv:2604.03616*, 2026. URL https://arxiv.org/abs/2604.03616.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28259–28277, 2024. URL https://proceedings.mlr.press/v235/li24ar.html.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. ARISE: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *arXiv preprint arXiv:2603.16060*, 2026. URL https://arxiv.org/abs/2603.16060.

Ivan Matveev. Token-oriented object notation vs JSON: A benchmark of plain and constrained decoding generation. *arXiv preprint arXiv:2603.03306*, 2026. URL https://arxiv.org/abs/2603.03306.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24547–24568, 2024. doi: 10.52202/079017-0774.

Qwen Team. Qwen3.5: Towards native multimodal agents. Qwen Blog, 2026. URL https://qwen.ai/blog?id=qwen3.5. Accessed: 2026-05-07.

Jaideep Ray. The constraint tax: Measuring validity-correctness tradeoffs in structured outputs for small language models. *arXiv preprint arXiv:2605.26128*, 2026. URL https://arxiv.org/abs/2605.26128. First posted 20 May 2026.

Maximilian Schall and Gerard de Melo. The hidden cost of structure: How constrained decoding affects language model performance. In Galia Angelova, Maria Kunilovskaya, Marie Escribe, and Ruslan Mitkov (eds.), *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI Era*, pp. 1074–1084, Varna, Bulgaria, September 2025. INCOMA Ltd., Shoumen, Bulgaria. URL https://aclanthology.org/2025.ranlp-1.124/.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *Advances in Neural Information Processing Systems*, volume 38, pp. 119605–119646, 2025. doi: 10.52202/085713-3603.

Adam Stein, Neelay Velingker, Mayur Naik, and Eric Wong. Once upon an input: Reasoning via per-instance program synthesis. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=XsQNqRdcdh.

Elias Stengel-Eskin, Archiki Prasad, and Mohit Bansal. ReGAL: Refactoring programs to discover generalizable abstractions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46605–46624, 2024. URL https://proceedings.mlr.press/v235/stengel-eskin24a.html.

Ian Su, Gaurav Purushothaman, Jey Narayan, Ruhika Goel, Kevin Zhu, Sunishchal Dev, Yash More, and Maheep Chaudhary. Broken chains: The cost of incomplete reasoning in LLMs. *arXiv preprint arXiv:2602.14444*, 2026. URL https://arxiv.org/abs/2602.14444.

Yi Sun, Han Wang, Jiaqiang Li, Jiacheng Liu, Xiangyu Li, Hao Wen, Yizhen Yuan, Huiwen Zheng, Yan Liang, Yuanchun Li, and Yunxin Liu. An empirical study of LLM reasoning ability under strict output length constraint. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7652–7671, 2025. doi: 10.18653/v1/2025.emnlp-main.389. URL https://aclanthology.org/2025.emnlp-main.389/.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? A study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1218–1236, 2024. doi: 10.18653/v1/2024.emnlp-industry.91. URL https://aclanthology.org/2024.emnlp-industry.91/.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51177–51191, 2024. URL https://proceedings.mlr.press/v235/wang24az.html.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023. URL https://arxiv.org/abs/2307.09702.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. ReasonFlux: Hierarchical LLM reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*, 2025. URL https://arxiv.org/abs/2502.06772.

A SEED-LEVEL RESULTS

Tables 6–8 give every seed-level five-call metric and paired test of the re-decoding grid; Table 7 gives the schema arms. Exact numeric matching, defined in Appendix H, matches integer-valued gold answers up to floating-point round-off and other answers to a relative tolerance of 10^{-6}. With it instead of the 1% tolerance, every skeleton-disjoint cell and every 512-token cell is unchanged; only the augmented 1536-token means move, from 38.03% to 37.73% under unconstrained decoding and from 37.63% to 37.34% under constrained decoding.

Table 6: Unconstrained five-call results by seed in %, recomputed from the saved predictions.

Test	Seed	Cap	Accuracy	Parse	Execute	Length finish
Skeleton	42	512	20.00	37.14	31.43	65.71
Skeleton	42	1536	37.14	100.00	91.43	0.00
Skeleton	123	512	14.29	37.14	37.14	62.86
Skeleton	123	1536	40.00	100.00	91.43	0.00
Skeleton	456	512	14.29	31.43	25.71	68.57
Skeleton	456	1536	28.57	100.00	91.43	0.00
Augmented	42	512	18.77	34.60	33.43	65.98
Augmented	42	1536	36.95	100.00	94.72	0.00
Augmented	123	512	13.20	31.38	31.09	68.62
Augmented	123	1536	38.42	100.00	92.08	0.00
Augmented	456	512	13.78	29.03	26.10	70.97
Augmented	456	1536	38.71	100.00	93.55	0.00

Table 7: JSON-schema-constrained five-call results by seed in %, recomputed from the saved predictions.

Test	Seed	Cap	Accuracy	Parse	Execute	Length finish
Skeleton	42	512	20.00	31.43	31.43	68.57
Skeleton	42	1536	37.14	100.00	94.29	0.00
Skeleton	123	512	14.29	37.14	37.14	62.86
Skeleton	123	1536	40.00	100.00	91.43	0.00
Skeleton	456	512	14.29	34.29	25.71	65.71
Skeleton	456	1536	34.29	100.00	88.57	0.00
Augmented	42	512	18.77	34.02	32.84	66.57
Augmented	42	1536	36.07	100.00	95.60	0.00
Augmented	123	512	12.61	31.96	31.96	68.04
Augmented	123	1536	38.42	100.00	91.50	0.00
Augmented	456	512	13.78	29.03	25.81	70.97
Augmented	456	1536	38.42	100.00	93.84	0.00

A censored row. For the skeleton-disjoint problem, quoted verbatim from the benchmark including its number agreement, "A slice of toast costs £2 and eggs cost £4 each. Dale had 1 slices of toast and 3 eggs. Andrew had 3 slice of toast and 10 eggs. How much did their breakfast cost?", whose gold answer is 60, seed 42 generates at 1536 tokens a 563-token flat program with ten steps that computes

$$(q_2 + q_4) \cdot q_0 + (q_3 + q_5) \cdot q_1$$

and returns 60. The 512-token response is its exact prefix and ends inside the ninth step, the step that adds the outputs of the third and fourth calls, while it is still listing that step's inputs. The entire correct computation has been written; only the end of that step, the closing of the step list, and the output instruction are missing, 51 tokens short of the end. Under the historical scoring this row counts as an invalid program.

Historical ensemble and routing. The historical ensemble paired compact compositional outputs with flat outputs truncated in two thirds of five-call rows, and its routing rule chose the format by call

Table 8: Five-call paired 1536-minus-512 effects. "512 only" and "1536 only" count discordant correctness outcomes; ΔAcc is in percentage points; p is the p-value of the exact two-sided McNemar test within a seed, a statement about the fixed evaluation rows.

Test	Decoding/seed	ΔAcc	512 only	1536 only	p
Skeleton	Unconstrained/42	+17.14	0	6	0.03125
Skeleton	Unconstrained/123	+25.71	0	9	0.00391
Skeleton	Unconstrained/456	+14.29	0	5	0.06250
Skeleton	Constrained/42	+17.14	0	6	0.03125
Skeleton	Constrained/123	+25.71	0	9	0.00391
Skeleton	Constrained/456	+20.00	0	7	0.01563
Augmented	Unconstrained/42	+18.18	4	66	1.65×10^{-15}
Augmented	Unconstrained/123	+25.22	3	89	5.24×10^{-23}
Augmented	Unconstrained/456	+24.93	1	86	1.14×10^{-24}
Augmented	Constrained/42	+17.30	4	63	1.11×10^{-14}
Augmented	Constrained/123	+25.81	0	88	6.46×10^{-27}
Augmented	Constrained/456	+24.63	0	84	1.03×10^{-25}

count at exactly the point where flat outputs cross the cap: at 1536 tokens, 0% of two-call, 0.1% of three-call, about 10% of four-call, and 66–68% of five-call flat outputs exceed 512 tokens. Both rules therefore coincide with the censoring boundary, which locates their historical gains at that boundary.

B GSM8K TEST PANEL

Panel E5. The panel contains all 1,319 GSM8K test problems, with the final answer marked at the end of each reference solution as the answer. GSM8K's training and test splits are disjoint and every problem of our benchmark derives from the training split; no number-masked panel text occurs in the training data or in either historical test set. The three flat checkpoints are decoded greedily and unconstrained at 512 and 1536 tokens with the prompt, greedy sampling, and scorer of the main experiments, giving 7,914 predictions. There are no gold plans, so the panel has no call-count strata. The decision rule, fixed before any output of this panel existed, and every seed-level decision are given in Appendix H.

Structural subsets. A model-free audit rebuilds each problem's reference computation from the calculator annotations of its GSM8K solution: a number equal to an earlier annotation's value is wired to it, otherwise it is a problem quantity if it occurs in the text, otherwise a constant, and the computation is the expression tree of the last annotation that reaches the answer. Every plan of our benchmark, 6,441 in the training and both test sets, is composed through the library into one expression tree. The canonical structure anonymizes leaves, flattens $+/-$ and $\times/\div$ chains into signed sums and products, and sorts children; plans with the same skeleton always share it, so a problem whose canonical structure differs from that of every benchmark plan has a computation that no training or test skeleton expresses. Seven nested versions of the subset exist, each frozen before its analysis:

- E5-SD, 464 rows, registered before the panel was decoded: every constant must be a number word of the text or a listed unit factor;
- E5-SD-R, 257 rows, registered before any panel output: the structure must be new under every admissible reading of every annotated number;
- E5-SD-C, 227 rows: signed intermediate values are linked, and five mechanical checks require that the annotations cover the whole written solution;
- E5-SD-CA, 221 rows: every E5-SD-C reference solution is read by hand against its problem, excluding rows whose reconstruction omits a required step, wires a quantity to the wrong source, or uses an unexplained constant;
- E5-SD-CA2, 215 rows: a declared rule for omitted multiplications by an explicit count of one is applied to all rows, and two rows whose reference solution does not solve the stated problem are removed;

- E5-SD-CA3, 209 rows: the same rule is re-applied to every row with an explicit "one" or "1" in its text, removing six further rows, and one operation stratum is corrected after a check of every row whose stratum depends on the reading of an annotated number;
- E5-SD-CA4, 206 rows: the same rule is applied to explicit durations of one, such as "how much does she make in 1 week", removing three rows.

E5-SD-C to CA4 were defined after the panel had been decoded and aggregate results on the earlier subsets had been read, with the decision rule unchanged; they read only problem texts, reference solutions, and benchmark plans, never a model output, and only remove rows. E5-SD-CA4, the subset used in the main text, has 3, 69, and 134 rows with 1–2, 3–4, and $\geq$5 arithmetic operations of its intended computation. As a consistency check rather than a measure of audit accuracy, on training rows whose text is a verbatim GSM8K training question with complete annotations, the annotation-derived and plan-derived structures agree on 326 of 383 rows. Table 9 gives the endpoints of every subset.

Table 9: Descriptive endpoints on the GSM8K test panel in %, as unconstrained seed means from 512 to 1536 tokens.

Panel	N	Length finish	Parse	Execute	Accuracy
E5	1,319	14.1→0.05	85.9→99.9	81.9→95.1	26.9→28.4
E5-SD	464	23.5→0.0	76.6→100.0	72.5→94.4	17.0→19.5
E5-SD-R	257	23.6→0.0	76.4→100.0	71.7→93.5	16.5→18.2
E5-SD-C	227	21.0→0.0	79.0→100.0	73.7→93.5	16.7→18.2
E5-SD-CA	221	20.5→0.0	79.5→100.0	74.1→93.4	17.2→18.7
E5-SD-CA2	215	19.5→0.0	80.5→100.0	74.9→93.2	17.4→18.9
E5-SD-CA3	209	19.6→0.0	80.4→100.0	74.6→93.0	16.9→18.5
E5-SD-CA4	206	19.6→0.0	80.4→100.0	74.6→92.9	17.2→18.8

Mechanism on the panel. Of the 559 E5 seed-rows cut at 512 tokens, 499 are exact prefixes of their 1536-token response, and 108 of 121 on E5-SD-CA4. The trajectory prediction of Section 4.2, specified for this panel as E5b before any of its outputs was available, meets its registered criteria against the decoded 384-, 512-, and 768-token arms in every seed; its rule and per-seed agreements are in Appendix H.

C SKELETON DEFINITIONS

Figure 2 shows the two formats for one test problem. Skeletons are derived from the gold plan as the ordered list of primitives with each argument's reference kind: a quantity Q, a constant C, or the output c_k of an earlier call k, counting calls from zero. Below, each call is written as the library primitive P_j with library index j, followed by its arguments in slot order, and the final reference follows a double arrow. The six five-call skeletons of Table 4 are

S1 $\mathrm{P}_0(Q,Q) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_0(c_0,Q) \to \mathrm{P}_0(c_1,Q) \to \mathrm{P}_2(c_2,c_3) \Rightarrow c_4$
S2 $\mathrm{P}_0(Q,Q) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_{10}(c_0,c_1,c_2,c_3) \Rightarrow c_4$
S3 $\mathrm{P}_9(Q) \to \mathrm{P}_9(c_0) \to \mathrm{P}_9(c_1) \to \mathrm{P}_9(c_2) \to \mathrm{P}_{28}(Q,c_0,c_1,c_2,c_3) \Rightarrow c_4$
S4 $\mathrm{P}_0(Q,Q) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_1(c_0,c_1) \to \mathrm{P}_2(Q,c_2) \to \mathrm{P}_3(c_3,Q) \Rightarrow c_4$
S5 $\mathrm{P}_7(Q,Q) \to \mathrm{P}_2(Q,c_0) \to \mathrm{P}_0(Q,Q) \to \mathrm{P}_0(c_1,Q) \to \mathrm{P}_2(c_2,c_3) \Rightarrow c_4$
S6 $\mathrm{P}_0(Q,Q) \to \mathrm{P}_2(c_0,Q) \to \mathrm{P}_{18}(c_1,Q,Q) \to \mathrm{P}_0(c_2,Q) \to \mathrm{P}_2(c_3,Q) \Rightarrow c_4$

Ordered versus graph disjointness. "Skeleton-disjoint" refers to this ordered definition. Comparing skeletons instead as labeled dependency graphs, that is, allowing independent calls to be listed in another order, the 272 training skeletons form 268 classes, and one of the 68 held-out skeletons, a three-call structure covering 12 of the 507 skeleton-disjoint rows, is isomorphic to a training skeleton after exchanging its two independent calls. None of the 23 augmented-set skeletons and none of the six five-call skeletons is isomorphic to a training skeleton, so the five-call and augmented results are unaffected; the 36 unconstrained seed-rows of the overlapping skeleton execute at both caps.

Problem from the skeleton-disjoint test set, with two calls and gold answer 1065: "In a factory that employed 852 people, 25% more workers have just been hired. How many employees are there now in the factory?"

Compositional plan, with two calls to verified primitives: P007 computes $ab/100$ and P001 computes $a + b$.


```
{"calls":[{"call_id":"c0","sub_id":"P007",
   "bindings":{"a":{"source":"quantity","qid":"q0"},
               "b":{"source":"quantity","qid":"q2"}}},
  {"call_id":"c1","sub_id":"P001",
   "bindings":{"a":{"source":"quantity","qid":"q0"},
               "b":{"source":"call_output","call_id":"c0"}}}],
 "final":{"source":"call_output","call_id":"c1"}}
```


Flat program, the same computation with the primitive bodies inlined and the call boundaries erased.


```
{"program_id":"flat_inlined",
 "slots":[{"name":"q0"},{"name":"q2"}],
 "steps":[
  {"op":"compute","target":"_s0_result","expr":"q0 * q2 / 100"},
  {"op":"compute","target":"_s0___output__","expr":"_s0_result"},
  {"op":"compute","target":"_s1_result","expr":"q0 + _s0___output__"},
  {"op":"output","target":"_s1___output__","expr":"_s1_result"}]}
```


Figure 2: Gold targets in both formats for one skeleton-disjoint test problem; quantity values, spans, types, and input lists are omitted. The flat program writes every primitive body and its output copy, so its serialization grows faster with the number of calls.

Table 10: Unconstrained accuracy and execution in %, from 512 to 1536 tokens with three seeds pooled, by the longest dependency path of the gold plan. Sk.: number of distinct skeletons.

Test	Path	Rows	Sk.	Accuracy	Execute
Skeleton	2	660	21	71.7→76.1	91.2→98.9
Skeleton	3	720	35	69.9→69.4	98.9→100.0
Skeleton	4	108	9	38.0→38.0	81.5→98.1
Skeleton	5	21	2	23.8→23.8	38.1→100.0
Skeleton	6	12	1	33.3→66.7	33.3→100.0
Augmented	2	1200	5	24.7→50.1	51.5→95.5
Augmented	3	834	6	68.8→74.0	92.1→100.0
Augmented	4	1080	9	40.1→41.0	81.6→98.5
Augmented	5	183	2	22.4→21.9	30.6→99.5
Augmented	6	120	1	32.5→65.0	32.5→100.0

Call count, length, and depth. S2 has only two dependency levels, because its first four calls are independent. By longest dependency path, Table 10 shows that path two supplies 305 of the net 396 additional correct augmented rows, with accuracy rising from 24.7% to 50.1% on its 1,200 rows. The 512-token cap penalized serialized length, which grows with calls and arguments; call count, length, and depth are confounded, and path lengths five and six contain one or two skeletons each.

Sampling structure of the five-call slices. Identifying source problems by their text with all numbers masked, the 35 skeleton-disjoint five-call rows come from nine source problems: four for S2 with 16 rows and one for each other skeleton. The 341 augmented five-call rows are quantity perturbations of the same nine problems, 160 rows for S2 and 40 or 21 for the others; all 35 distinct skeleton-disjoint five-call texts recur among them in 41 rows. Over the full test sets, the augmented set covers 23 of the 68 held-out skeletons and 30 source problems, and the two sets share 114 distinct problem texts; no test skeleton and no number-masked test problem occurs among the 272 training skeletons and 449 distinct number-masked training texts. Row-level statistics therefore describe quantity variation within a small set of source problems, the two test sets are not independent replications, and Section 4.3 reports skeleton-level effects, bootstrap intervals, and leave-one-skeleton-out ranges.

D DATA CONSISTENCY AUDIT AND REPAIR OPERATOR

Consistency check. For every row we execute the gold plan twice: on the stored quantity metadata, which reproduces every stored answer with no failure in 6,441 rows, and on the quantities that the evaluation scorer extracts from the problem text by position. A row is consistent if the second execution matches the stored answer to within $10^{-6}\max(1, |a|)$ for stored answer a, and every quantity the plan uses has equal text and metadata values; tolerances of 10^{-9} and 10^{-12} select the same rows. The answer condition fails in 12 training, 2 skeleton-disjoint, and 36 augmented rows; two of the augmented rows, one S2 and one S6, differ from the stored answer by only 0.2% and 0.4% and would pass the 1% scoring tolerance. The input condition alone fails in 20 further training rows, one three-call skeleton-disjoint row, and 4 augmented rows, where the permuted values happen to leave the answer unchanged but the gold bindings no longer match the text. In total 32 of 4,795 training rows fail, of which 9 have three calls, 6 four, 11 five, and 6 six, together with 3 of 507 skeleton-disjoint rows and 40 of 1,139 augmented rows. The perturbation step rewrote the problem text by replacing, for each changed quantity, the first remaining occurrence of its old number string. When the same string occurs more than once, the new value can land at another quantity's position. In all 75 failing rows the text contains exactly the stored values, in permuted positions. For example, one skeleton-disjoint row, also present in the augmented set, describes 4 boxes of flour at \$1, 2 trays of eggs at \$11, 8 liters of milk at \$2, and 5 boxes of baking soda at \$5, so the answer is

$$4 \cdot 1 + 2 \cdot 11 + 8 \cdot 2 + 5 \cdot 5 = 67;$$

the metadata stores 5 trays and a baking-soda price of \$2, and the stored answer is 85. All three compositional checkpoints return 67 on this row. Over the 38 five-call rows that fail the answer condition the compositional outputs return the value of the gold plan on the text quantities in 6 of 6 skeleton and 66 of 108 augmented seed-rows, and the unconstrained 1536-token flat outputs in 6 of 6 and 104 of 108, against none at 512 tokens. We do not rewrite the benchmark; we report estimates on the registered labels and on the consistent rows.

Semantic review of the five-call sources. The consistency check compares numbers; it cannot detect a problem whose wording admits two answers. We therefore read each of the nine five-call source problems, with its gold plan and the stored answers of all its perturbed rows, and classified each row as M, a metadata mismatch that fails the consistency check; U, a wording or unit ambiguity, where the text admits at least two reasonable readings with different answers or the gold plan adopts a reading the text does not state; P, a premise violation, where the stored answer is negative for a question asking for an amount of money, a price, or a time; or clean. The rule and the row labels were written down before any model-output count on this partition was computed; the aggregate five-call results of Tables 1 and 4 had been read. Eight of the nine sources have one reading, which the gold plan computes: S1 on a seasonal revenue difference, three S2 sources on baking ingredients, breakfast, and school shirts, S3 on doubling phone time, S4 on a clearance purchase, S5 on discounted textbooks, and S6 on media profit. The fourth S2 source prices earrings "for a pair" but reports sales as "20 earrings", and its gold plan multiplies the count by the per-pair price; its 4 and 40 rows form class U. Perturbation made the stored answer negative in 11 of the 44 S1 rows, where low-season sales exceed high-season sales, and in 36 of the 44 S4 rows, with a negative price per top; these 2 and 45 rows of the two test sets form class P. The rule for these rows: registered labels stay in every table, and estimates are also given on the rows that pass the review, 27 and 216 per seed.

Premise screen. The answer-sign rule checks only the final answer. A premise screen fixed after the outputs had been analyzed, and reading only problem texts, gold plans, and stored quantities, applies unit bounds to every row of the nine sources: no negative count, price, amount of money, or time among the inputs and the gold plan's intermediate values; at most 24 hours, or 1,440 minutes, in a day, on every day of a described schedule; at most 7 days in a week; and a price cut below 100%. It flags every P row and 63 further rows, class P2, counted per test set: 1 and 20 S1 rows that sell for more than 24 hours a day, 1 and 14 S3 rows whose doubled phone time exceeds a day by day five, 1 and 16 S5 rows with a textbook discount of 100% or more, and 0 and 10 S6 rows with more than 7 selling days a week. The *clean* rows are those that pass the consistency check, the review, and the screen, 24 and 157 per seed; Table 20 also reports estimates on the rows that pass the review alone. Table 11 separates these classes at 1536 tokens. A clean row whose program executes to a non-gold value disagrees with the reference computation of a problem whose premises pass these checks; the screen does not certify every premise, so we do not attribute each such row to a particular model error.

Such rows concentrate in S5, with 69 of 72 augmented seed-rows, S6 with 32 of 33, and S3 with 50 of 78, and are rarer in S4 with 10 of 18, S2 with 31 of 243, and S1 with 6 of 27.

Table 11: Five-call outcomes at 1536 tokens by review class, unconstrained, with seed-rows pooled over three seeds. M: metadata mismatch; U: wording or unit ambiguity; P: negative answer; P2: other premise violation found by the premise screen. Wrong: the program executes to a non-gold value; No exec.: parses without executing or does not parse.

	Skeleton-disjoint				Augmented			
Class	Rows	Correct	Wrong	No exec.	Rows	Correct	Wrong	No exec.
M	6	0	6	0	120	15	105	0
U	12	0	5	7	120	0	68	52
P	6	2	4	0	135	50	77	8
P2	9	3	6	0	177	58	119	0
Clean	72	32	38	2	471	266	198	7

Repair operator and qualification cases. Copying gold wiring under a non-identity alignment can make a call refer to a call listed after it; no eligible unrepaired prediction contains such a forward reference. Quantity, operator, and final-reference repairs keep the prediction's listing order. Wiring and joint repairs copy gold call references through the alignment and then execute the plan in topological order, breaking ties by listing position; an edit that creates a cycle or an unresolved reference counts as a failed repair. Five constructed qualification cases check that the gold plan executes to its answer; that a prediction equal to the gold plan but listed in a different order is repaired by the joint repair through reordering; that a value error is fixed by the quantity repair and not by the wiring repair, which leaves the plan in order; that a wiring edit that creates a cycle is classified as having no legal order and is not executed; and that a correct prediction stays correct under every repair. All five pass. On the real outputs, every eligible row without reference-kind or missing-slot differences whose gold plan is correct is repaired by the joint repair, with no exception over all call counts.

Table 12: Compositional binding diagnostics recomputed from raw outputs, three seeds pooled over all rows. "Hist.": historical slot-count ratio. R_{slot}, R_{call}: quantity-value error rates; $R_{\mathrm{slot}}^{\mathrm{sym}}$: R_{slot} with the arguments of symmetric primitives compared as multisets, as registered in Appendix H; Wire: wiring error per call-output slot; Op: operator-identifier mismatch per call; Coverage: rows with the gold call count. Thirteen of the 31 primitives fall into six classes of functionally equivalent primitives, with equal outputs on 200 random draws; counting such identifiers as matching lowers Op to 0.183 and 0.224 at three and four calls on the skeleton-disjoint set and to 0.260 at four calls on the augmented set, and leaves every other row unchanged.

Slice	Hist.	R_{slot}	$R_{\mathrm{slot}}^{\mathrm{sym}}$	R_{call}	Wire	Op	Coverage
Skeleton, 2 calls	0.42	0.096	0.078	0.125	0.000	0.000	40/96
Skeleton, 3 calls	0.57	0.147	0.139	0.130	0.051	0.208	969/1080
Skeleton, 4 calls	0.76	0.178	0.154	0.150	0.038	0.245	107/216
Skeleton, 5 calls	0.79	0.292	0.232	0.297	0.025	0.255	58/105
Skeleton, 6 calls	0.61	0.000	0.000	0.000	0.000	0.167	2/24
Augmented, 4 calls	0.71	0.166	0.147	0.135	0.012	0.284	1112/2154
Augmented, 5 calls	0.82	0.336	0.270	0.331	0.045	0.259	620/1023
Augmented, 6 calls	0.72	0.000	0.000	0.000	0.185	0.364	71/240

Table 13: Five-call repaired accuracy in % on all rows, including the 42 five-call rows that fail the consistency audit, with the legal-order operator; columns as in Table 5. The historical mixed repair reported 47.9% on the augmented and 45.7% on the skeleton-disjoint set.

Slice	N	Elig.	Base	Quantity	Operator	All four	Gold plan
Skeleton, 5 calls, all rows	105	55.2	25.7	45.7	36.2	58.1	67.6
Augmented, 5 calls, all rows	1023	60.6	26.9	47.5	33.0	55.1	71.3

E EVALUATION PROTOCOL

Prompt and decoding. Each prompt consists of the word "Problem:" followed by the problem text, a blank line, and the instruction "Generate an executable JSON program:", placed in the model's chat template with thinking disabled. Decoding is greedy at temperature 0 with the sampling seed equal to the training seed, one sequence per prompt, maximum model length 2816 tokens, and bfloat16 weights of the LoRA adapter merged into the base model. The constrained arm passes the JSON schema described below to the decoder's native structured-output interface; the schema is static and contains no problem-specific information.

Checkpoints. The three flat checkpoints are LoRA adapters of Qwen3.5-9B with rank 32, α=64, and dropout 0.05 on the query, key, value, output, gate, up, and down projections, trained for three epochs on the 4,795 historical flat targets with the prompt above and the loss on the full chat-formatted text: AdamW, learning rate 10^{-4} with cosine decay and 3% warm-up, batch 4 with gradient accumulation 4, bfloat16, gradient checkpointing, and seeds 42, 123, and 456; the training script is in the supplement. Every decoding run in this paper uses the adapter of a seed merged into the base weights; the 512- and 1536-token arms of a seed use the same merged checkpoint.

Retrained checkpoints, E3. The compositional and corrected-flat checkpoints use the same recipe on the same 4,763 rows: 894 optimizer steps per seed, no training sequence above the 2,048-token limit, the longest having 696 and 1,547 tokens, and 1.60 and 3.53 million full-sequence tokens, of which 1.21 and 3.16 million are target tokens, with mean targets of 255 and 663 tokens. The logged training loss falls from 1.17–1.20 to 0.047–0.084 for the compositional and from 0.57–0.58 to 0.021–0.039 for the corrected-flat format; per-token losses of the two formats are not directly comparable. The supplement holds the per-seed training manifests and logs and all 48 prediction files with generated token IDs.

Schema. The schema describes an object with a required program identifier string, an array of input slots, and an array of steps. Each input slot has a required name string, a required data type chosen among integer, float, string, Boolean, integer list, and float list, and an optional description. Each step has a required operation chosen among assign, compute, compare, aggregate, condition, and output, a required target name and expression, both strings, and an optional input list and target data type.

Scoring. The first brace-delimited span is parsed as JSON. Problem quantities are extracted by a fixed regular expression in order of appearance and named $q_0, q_1, \ldots$; each flat input slot binds the quantity with its own name if present and otherwise the quantity at the slot's position. The program executes in a restricted arithmetic interpreter; a result is correct if it matches the gold answer within 1% relative tolerance. The same scorer re-scores the saved responses identically; all 1,646 rows of one seed were checked.

Token accounting. Prompts are short, with median 72 and 88 tokens on the two test sets and at most 144 under the chat template, so prompt plus longest response uses at most 988 of the 2,816-token context. Saved token counts include the end-of-sequence token for natural stops. Offline truncation re-tokenizes each saved 1536-token response with the model tokenizer, keeps the first 512 generated tokens when the saved count exceeds 512, decodes, and rescores. Re-tokenized text need not reproduce the generated token IDs exactly, and offline curves are conditional on the saved trajectories; that the offline 512-token cut reproduces the decoded arm's five-call accuracy exactly and 98.6% of its row-level correctness outcomes in §4.2 is the check that this approximation is adequate

at that cap, and the decoded 640-, 768-, and 1024-token arms give the same check at the other caps with 97.9% correctness agreement. The supplement contains the token IDs of all 4,938 re-tokenized 1536-token unconstrained responses together with a verification script; it decodes every ID sequence back to the saved text, all 4,938 of them, and reproduces the 512-token agreement on 1,112 and 1,109 of 1,128 rows and every offline row of Table 3. For the original 512/1536 grid these re-tokenized IDs stand in for the generated ones; every later decoding run, namely E1, E2, E5, and E5b, saves the generated token IDs directly. For the GSM8K panel the supplement contains the evaluation procedure used for all its caps from 384 to 2048 tokens, which writes the generated token IDs of every row, and a script that rebuilds every row of Table 17 and its nine decisions from the saved 1536-token IDs and the decoded 384-, 512-, and 768-token arms.

Inliner qualification cases. The corrected inliner of Section 5 is checked against the compositional executor on 293 constructed cases: a full small arithmetic grid with last, non-last, and implicit final choices, typed chained calls, quantity and constant final references, emit-then-reassign, slot shadowing, and a string literal containing a local name. All returned values and types agree; the cases ship with the supplementary code.

F Extended related work

This appendix expands Section 2. Works first posted after April 2026 are marked as concurrent.

Output budgets as an evaluation variable. Sun et al. (2025) evaluate 30 models under strict output-length limits and find that the preferred model size and prompt style change with the budget; Hochlehnert et al. (2025) show that lowering the output-token limit lowers mathematical reasoning scores and can induce premature stopping; and, concurrently, Guedes de Souza & Panisson (2026) report ranking reversals and budget-dependent complementarity between models across caps from 64 to 4,096 tokens. In the debate on the reported collapse of reasoning models (Shojaee et al., 2025), Lawsen (2025) argues that some failures reflect output limits and that a compact generating-function answer avoids them, while Dellibarda Varela et al. (2025) find genuine limits once output constraints are relaxed. These works measure the effect of a cap with aggregate scores per budget or per problem size. This paper tests, within one checkpoint, whether a validity gap between two trained program formats is censoring, using finish reasons, exact-prefix relations, and offline truncation of saturated trajectories, and then resolves the result by held-out program structure. In our setting both effects appear at once, separated by program structure: removing the cap restores validity everywhere but correctness only where the model's complete programs are right, as §4.3 shows.

Format restrictions and their confounds. Tam et al. (2024) report that format restrictions such as JSON lower reasoning performance; a response by Kurt (2024) attributes much of that gap to mismatched prompts and parsing rather than to structure. Lee et al. (2026) find that format instructions alone cause most of the loss for open-weight models and note that the loss concentrates where structured output compresses the model's natural length; Schall & de Melo (2025) show that constrained decoding helps base models but often hurts instruction-tuned ones on generation tasks. Concurrently, Fan (2026) separates prompt length, format, and truncation for prompted JSON versus chain-of-thought answers from aggregate accuracy and error counts among truncated responses. Su et al. (2026) impose budgets as fractions of each reasoning format's unconstrained length and find that code-based reasoning degrades more gracefully than comment-based reasoning. Our setting differs in three ways that matter for identification: the formats are learned output representations of fine-tuned checkpoints rather than prompted styles, we give a conversion under which the two formats provably compute the same function and measure in §5 how far the historical training targets departed from it, and the censoring claim is tested on paired rows through finish reasons, exact prefixes, and offline truncation.

Constrained and budget-aware decoding. Grammar- and schema-constrained decoders guarantee well-formed outputs (Geng et al., 2023; Willard & Louf, 2023; Dong et al., 2025). Grammar-aligned decoding corrects the distributional distortion that such masking introduces (Park et al., 2024), and JSONSchemaBench separates efficiency, schema coverage, and output quality across engines (Geng et al., 2025). Validity and correctness can move in opposite directions: concurrently, Ray (2026) reports that in small models hard schema enforcement raises validity to 100% while lowering answer

accuracy. The concurrent TruncProof uses an LL(1) parser, which reads left to right with one token of lookahead, to estimate how many tokens are needed to close a valid JSON output and constrains decoding so that it finishes within a hard token limit (Kato & Tarashima, 2026). Matveev (2026) compares JSON with a more compact serialization under plain and constrained decoding and counts prompt and completion tokens. Our schema arm is a static JSON schema that does not force closure before the cap; budget-aware methods target syntactic closure, whereas our analysis separates a closed program from a correct computation.

Program representations, libraries, and length. Program-aided reasoning delegates computation to an interpreter (Gao et al., 2023; Chen et al., 2023; Gou et al., 2024; Li et al., 2024), and recent systems add per-instance synthesis, execution feedback, or reusable templates and skills (Stein et al., 2025; Basarkar et al., 2026; Yang et al., 2025; Li et al., 2026). Library learning names reusable operations and thereby shortens programs (Ellis et al., 2021; Grand et al., 2024); ReGAL reports higher accuracy and TroVE reports simpler solutions with higher accuracy when models call learned functions (Stengel-Eskin et al., 2024; Wang et al., 2024), whereas Kang et al. (2024) find that modularity itself is not a core factor in code-generation performance. Whenever the compared representations differ in length, a common cap is a format-dependent censoring threshold. We show on one controlled case that the threshold alone can produce an apparent representation failure, and we give checks that make such comparisons interpretable: finish reasons, per-format saturation, and a semantics-preserving conversion. Held-out-structure splits (Keysers et al., 2020) control train–test overlap but not decoder censoring, and quantity perturbation multiplies rows rather than structures, so we report skeleton-level uncertainty.

What the trajectory-level instrument adds. This paper contributes an instrument and its result on one trained representation, each part checkable from the released predictions:

1. a row-level trajectory audit, in which offline truncation of saturated trajectories reproduces the censored arm's five-call accuracy exactly and 98.6% of its row-level correctness outcomes, and exact-prefix pairs carry 247 of 253 net gains, so that censoring is examined per row rather than inferred from an aggregate change;
2. prospective checks, in which offline estimates for three intermediate caps were registered before decoding and the decoded arms agree with them on 97.9% of row-level correctness outcomes, with every registered criterion reported in Table 14;
3. a structural decomposition, in which validity recovers in every evaluated skeleton while correctness recovers only where the complete programs are right, so the restored programs expose wrong computations that truncation had recorded as validity errors; and
4. a target-semantics control, in which a semantics-preserving inliner shows that the compared training targets did not compute the same function, since 6.0% of historical flat targets returned a different value, and gives each format's reference length before any decoding.

The scope of these results is two fine-tuned model families, Qwen3.5-9B for every analysis and Llama-3.1-8B-Instruct for the validity and trajectory tests, on one GSM8K-derived benchmark and the GSM8K test split of Appendix B.

G CONDITIONAL EQUIVALENCE OF THE INLINER

This appendix spells out, step by step, the argument summarized in Section 5; the domain, the construction, and the notation are those of that section.

Claim. Let P be a plan accepted by the inliner and q the given quantities. Writing $E(P, q)$ and $F(P, q)$ for compositional and flat execution on q,

$$E(P, q) = F(P, q),$$

including the scalar type, whenever compositional execution succeeds and the flat step budget covers the expansion.

Proof. The proof is a double induction: an outer induction over the calls of P in order and, within a call, an inner induction over its assignments.

Step 1: outer hypothesis. Assume that, after the flat steps generated for all earlier calls, the fresh target recorded as the value of each earlier call holds that call's compositional output. For the first call there is no earlier call, and the hypothesis holds trivially.

Step 2: initial locals. The construction materializes the same slot casts as the compositional executor. The arguments that refer to earlier calls agree by Step 1, and quantities and constants are identical, so identical casts give identical initial local values.

Step 3: inner induction over assignments. Inner hypothesis: before the k-th assignment, every renamed local name holds the corresponding compositional local value. Each assignment of the primitive is inlined to a fresh target after local names in its expression syntax tree are renamed; string literals are untouched. By the inner hypothesis the renamed names read exactly the current local values, so the inlined assignment evaluates to the same value as the compositional assignment. Because every target is fresh, a value that has been emitted is not changed by a later reassignment of the same local name.

Step 4: call value. By Step 3, the fresh target of the call's output instruction holds the compositional output of the call. The construction records this target as the call value, which establishes the outer hypothesis for the next call.

Step 5: final reference. The flat program finally emits the plan's explicit final reference, or the last call only when no final reference exists. By Step 4 applied to every call, the emitted value, and its scalar type, equal those of $E(P, q)$. □

Step count. If primitive j has a_j arguments and s_j steps and c literal assignments resolve constants, the flat program has

$$1 + c + \sum_j (a_j + s_j)$$

steps. This counts our construction; it is not a lower bound for every equivalent flat serialization.

Scope of the claim. The equivalence holds for the given quantities q. It does not certify that q matches the problem text; the audit of §3 and Appendix D checks this separately.

H REGISTERED CONFIRMATORY CHECKS

"Registered" means that the decision rules were fixed in writing before the corresponding decoding or, where stated below for E5b, E4, three operational terms of E3, and two additions to E7, before any of the corresponding outputs was available or read; this is an internal registration, not an external pre-registration. This section lists every registered criterion and every seed-level decision. A criterion marked as met is a pre-specified decision on the fixed evaluation sets, not a calibrated significance test; inference about new structures and training runs rests on the per-seed values and on the skeleton-, source-, and problem-level intervals reported with each run.

Decoding runs E1 and E2. Before any new decoding we registered two runs: E1, two fresh repeats of seed 42 at 1536 tokens in both decoding modes on both test sets, and E2, the three checkpoints at 640, 768, and 1024 tokens unconstrained and at 768 tokens under the schema, with the offline estimates of Table 3 computed beforehand. Both use the unchanged checkpoints, prompt, decoding arguments, and scorer, and additionally save the generated token IDs. Table 14 lists every registered criterion; the registered numerical forecasts concern the unconstrained arms, and the schema arm at 768 tokens is compared, mode-matched, with the schema run at 1536 tokens and the schema E1 range in rows E2a^s. Table 15 gives all four registered five-call endpoints of every decoded E2 arm, including the schema arm, whose length finish is 0.0% and 0.2%, parse 100.0% and 99.8%, execution 90.5% and 93.1%, and accuracy 36.2% and 37.7% on the skeleton-disjoint and augmented sets. In 16 of the 32 seed-42 unconstrained five-call pairs whose 512-token response is not a prefix of the saved 1536-token response, the fresh E1 repeat has the saved 512-token response as an exact prefix.

Table 14: Registered E1/E2 criteria on five-call rows of the skeleton-disjoint and augmented sets. E2 values are means over the three checkpoints; E1 values are the largest difference in points between any two of the saved run and the two repeats of seed 42, unconstrained / schema. The E1 range used for E2a is the unconstrained E1 value of the same test set. In the Met column, slash-separated decisions are ordered skeleton-disjoint / augmented.

Run	Quantity	Criterion	Skeleton	Augmented	Met
E1	Five-call accuracy difference between 1536-token runs of seed 42	< 5	0.0 / 0.0	0.9 / 1.5	yes
E2a	Length finish at 768 / 1024 in %	≤ 1	0.0 / 0.0	0.3 / 0.0	yes
E2a	Accuracy at 768 / 1024 minus 1536 in points	within E1 range 0.0 / 0.9	$+1.0$ / $+1.0$	$+0.2$ / -0.1	no / yes
E2b	Length finish at 640 in %, registered 11.4 / 12.3	within 5	12.4	12.5	yes
E2a^s	Schema: length finish at 768 in %	≤ 1	0.0	0.2	yes
E2a^s	Schema: accuracy at 768 minus 1536 in points	within E1 range 0.0 / 1.5	-1.0	$+0.1$	no / yes

Table 15: All registered five-call endpoints of the decoded E2 arms in %, as seed means over the three checkpoints: length finish, parse, execution, and accuracy. The corresponding 1536-token values are in Table 1.

		Skeleton-disjoint				Augmented			
Decoding	Cap	Len	Parse	Exec.	Acc.	Len	Parse	Exec.	Acc.
Unconstrained	640	12.4	87.6	79.0	30.5	12.5	87.5	81.2	30.3
Unconstrained	768	0.0	100.0	90.5	36.2	0.3	99.9	93.5	38.2
Unconstrained	1024	0.0	100.0	90.5	36.2	0.0	100.0	93.0	37.9
Schema	768	0.0	100.0	90.5	36.2	0.2	99.8	93.1	37.7

Descriptive additions. The structure attribution of §4.1 and §4.3, the semantic review of Appendix D, the sensitivity analyses of E3, and the source-level and symmetry-normalized analyses at the end of this section were fixed in writing after the aggregate results had been read and before their own counts were computed; apart from their stated criteria, they carry no decision rule. The premise screen of Appendix D was fixed after the E3 sensitivity analyses had been read; it reads only reference data and defines the clean rows of Table 20 without a decision rule.

GSM8K test panel, E5. The rule was registered before any output of the panel of Appendix B existed. Let L_s be the rows that end by length at 512 tokens in seed s. The rule holds for a seed if

- at most 1% of rows end by length at 1536, and
- at least 90% of L_s parse at 1536 and the number of executing rows in L_s rises from 512 to 1536;

it is confirmed if it holds in every seed. Each structural subset is analyzed with the unchanged rule, which applies to a seed when L_s has at least 30 rows. Table 16 lists every decision.

Offline prediction on the panel, E5b. The E5b rule was written while the E5 arms were being decoded and before any panel output was available; the 384- and 768-token arms were decoded afterwards. For a cap c, a row is predicted to end by length if its 1536-token response has more than c generated tokens, and its outcomes are those of its first c generated token IDs, rescored; otherwise the 1536-token outcomes are kept. The rule requires, per seed and cap,

- a predicted length-finish rate within 2 points of the decoded one, and
- row agreement of at least 98% on length finishes and of at least 95% on execution and on correctness.

Table 16: Every seed-level decision of the registered rule on the GSM8K test panel E5 and its structural subsets. $|L_s|$: rows ending by length at 512 tokens; "Len 1536": rows of the whole panel ending by length at 1536; parse and execution are counted on L_s.

Panel	N	Seed	$\lvert L_s\rvert$	Len 1536	Parse 1536	Exec. 512→1536	Met
E5	1,319	42	176	1	175	0→162	yes
E5	1,319	123	179	0	179	2→168	yes
E5	1,319	456	204	1	203	0→193	yes
E5-SD	464	42	103	0	103	0→94	yes
E5-SD	464	123	105	0	105	1→98	yes
E5-SD	464	456	119	0	119	0→113	yes
E5-SD-R	257	42	59	0	59	0→53	yes
E5-SD-R	257	123	58	0	58	0→54	yes
E5-SD-R	257	456	65	0	65	0→60	yes
E5-SD-C	227	42	49	0	49	0→45	yes
E5-SD-C	227	123	44	0	44	0→42	yes
E5-SD-C	227	456	50	0	50	0→47	yes
E5-SD-CA	221	42	46	0	46	0→42	yes
E5-SD-CA	221	123	42	0	42	0→40	yes
E5-SD-CA	221	456	48	0	48	0→45	yes
E5-SD-CA2	215	42	44	0	44	0→40	yes
E5-SD-CA2	215	123	38	0	38	0→36	yes
E5-SD-CA2	215	456	44	0	44	0→41	yes
E5-SD-CA3	209	42	43	0	43	0→39	yes
E5-SD-CA3	209	123	37	0	37	0→35	yes
E5-SD-CA3	209	456	43	0	43	0→40	yes
E5-SD-CA4	206	42	41	0	41	0→37	yes
E5-SD-CA4	206	123	37	0	37	0→35	yes
E5-SD-CA4	206	456	43	0	43	0→40	yes

The same three checkpoints were decoded at 384 and 768 tokens with the scorer and decoding arguments of E5, giving 7,914 further predictions; the 512-token arm is that of E5. Against the decoded arms, the rule holds in all nine seed-by-cap decisions of Table 17: every row agreement is between 98.8% and 99.9%, and every predicted length-finish rate is within 0.5 points of the decoded one. Applying the same comparison, unchanged and descriptively, to the audited structural subsets E5-SD to E5-SD-CA4 gives agreements of at least 98.0% at every cap. The decoded panel follows the budget curve that the 1536-token trajectories predict; as seed means over 384, 512, 768, and 1536 tokens, length finishes go 44.7→14.1→0.8→0.05%, execution 52.9→81.9→94.6→95.1%, and accuracy 20.2→26.9→28.2→28.4%.

Table 17: Registered offline prediction E5b of the decoded 384-, 512-, and 768-token arms on all 1,319 rows of the GSM8K test panel from the 1536-token outputs, with agreements in %. Pred./dec.: predicted and decoded numbers of length finishes; $|\Delta|$: absolute difference between the predicted and decoded length-finish rates in percentage points, pp; agr.: row agreement between the offline prediction and the decoded arm.

Cap	Seed	Len. finishes pred./dec.	$\lvert\Delta\rvert$, pp	Length agr.	Exec. agr.	Correct agr.	Met
384	42	585 / 584	0.08	99.2	99.4	99.5	yes
384	123	570 / 571	0.08	99.3	99.3	99.8	yes
384	456	611 / 613	0.15	98.8	98.9	99.2	yes
512	42	182 / 176	0.45	98.9	99.0	99.4	yes
512	123	178 / 179	0.08	99.5	99.4	99.5	yes
512	456	201 / 204	0.23	99.3	99.2	99.5	yes
768	42	10 / 11	0.08	99.8	99.7	99.5	yes
768	123	9 / 8	0.08	99.9	99.9	99.5	yes
768	456	12 / 11	0.08	99.9	99.8	99.5	yes

Retrained format comparison, E3. Registered with E1/E2, before any decoding: fresh compositional checkpoints, with DataflowPlan targets and the historical compositional prompt, and corrected-flat checkpoints, with targets from the inliner of Section 5, three seeds each, trained by the recipe of Appendix E on the same 4,763 training rows that pass the consistency audit, and decoded greedily and unconstrained at 512, 1024, 1536, and 2048 tokens on both test sets in 48 arms. The primary endpoint is the five-call accuracy difference, compositional minus corrected flat, with each format at its saturation cap. A representation difference is claimed only if the seed-mean difference exceeds the E1 noise band and has the same sign in at least two of the three non-degenerate skeletons that carry it. Three terms of this rule were made operational after the run had started and before any of its outputs was read: the saturation cap is the smallest cap whose seed-mean five-call length-finish rate is at most 1%; the noise band is the E1 value of the same test set, 0.0 and 0.9 points in Table 14; and the carrying skeletons are the at most three skeletons, among those where either format has saturated accuracy strictly between 0 and 100%, with the largest product of rows and difference. Table 18 gives every five-call endpoint and Table 19 every decision. The noise band measures run-to-run decoding variation of one checkpoint, so it is a threshold rather than an error-rate-calibrated test; variation across training seeds, skeletons, and source problems is reported separately in Tables 20 and 21. Descriptively, the compositional outputs are stable under a larger cap: of the compositional rows that stop naturally at 512 tokens, 1,511 of 1,519 and 3,316 of 3,338 seed-rows are prefixes of their 1024-token response. Over all call counts, accuracy at these saturation caps is 50.0% for compositional and 45.9% for corrected-flat checkpoints on the skeleton-disjoint set and 30.9% and 22.5% on the augmented set.

Table 18: Retrained format comparison E3: five-call length finish, execution, and accuracy in %, as seed means by format and cap.

		Skeleton-disjoint			Augmented		
Format	Cap	Len	Exec.	Acc.	Len	Exec.	Acc.
Compositional	512	0.0	93.3	28.6	1.1	91.2	25.2
Compositional	1024	0.0	93.3	28.6	0.1	91.9	25.0
Compositional	1536	0.0	93.3	28.6	0.1	91.9	25.0
Compositional	2048	0.0	93.3	28.6	0.1	91.9	25.0
Corrected flat	512	100.0	0.0	0.0	98.7	1.3	0.0
Corrected flat	1024	43.8	56.2	7.6	40.7	59.1	8.3
Corrected flat	1536	1.0	99.0	18.1	0.7	98.9	15.8
Corrected flat	2048	0.0	100.0	18.1	0.0	99.6	15.8

Table 19: Registered decisions of the retrained comparison E3: five-call accuracy difference in points, compositional minus corrected flat, each at its saturation cap. Caps: compositional / corrected flat; seeds 42 / 123 / 456; carrying skeletons with their pooled differences.

Test set	Caps	Mean	Per seed	Band	Carrying skeletons	Met
Skeleton, all	512/1536	+10.5	+2.9/+11.4/+17.1	0.0	S5 +33.3, S6 +33.3, S4 +16.7	yes
Skeleton, consistent	512/1536	+11.1	+3.0/+12.1/+18.2	0.0	S5 +33.3, S6 +33.3, S4 +16.7	yes
Augmented, all	1024/1536	+9.2	+11.1/+8.2/+8.2	0.9	S5 +32.5, S4 +17.5, S1 +16.7	yes
Augmented, consistent	1024/1536	+10.7	+13.3/+9.3/+9.6	0.9	S5 +32.5, S4 +17.5, S1 +16.7	yes

Sensitivity of the retrained comparison. Four descriptive analyses of the saved E3 outputs were registered after the results above had been read and before any of their own quantities was computed; two carry a stated criterion. F1 is the difference on the rows that pass the semantic review of Appendix D, at the saturation caps and at the common 512-token cap; its criterion is that the saturated difference on these rows is positive on both test sets. F2 covers structure- and training-level units at the saturation caps: per-skeleton and per-source differences with seeds pooled, balanced means, leave-one-out ranges of the pooled difference, per-seed differences, and a bootstrap that resamples the nine source problems with 10,000 draws and a percentile interval; its criterion is that every leave-one-skeleton-out, leave-one-source-out, and seed difference is positive on both test sets. F3 gives all rows by gold call count. F4 rescores every E3 prediction with exact numeric agreement

and re-applies the registered rule; on every executed row the stored 1%-tolerance flag equals a fresh rescoring, in all 31,916 cases. Tables 20–22 report every quantity. Exact matching here, on the grid of Appendix A, and in E7 compares integer-valued gold answers up to floating-point round-off, $|r - a| < 10^{-6}$ for executed result r and gold answer a, and other answers to a relative tolerance of 10^{-6}. This integer rule was registered after a first exact rescoring that used a relative tolerance of 10^{-6} for every answer; it changes 8 of the 39,504 E3 exact flags, from two four-call augmented problems with answers above 5×10^7 in one checkpoint at all four caps, none of the 19,752 grid flags, and none of the 7,914 E7 flags, and leaves every reported exact quantity unchanged. Both formats were trained on 4,763 rows for 894 optimizer steps with no sequence above the 2,048-token training limit, the longest having 696 and 1,547 tokens; as Appendix E reports, the corrected flat sequences hold 3.53 million tokens against 1.60 million for the compositional ones.

Table 20: Sensitivity of the retrained comparison E3: five-call accuracy difference in points, compositional minus corrected flat, at the saturation caps unless stated; parentheses give the per-seed differences for seeds 42/123/456. Review rows pass the consistency check and the semantic review, 27 and 216 per seed; clean rows also pass the premise screen of Appendix D, 24 and 157 per seed. Bootstrap: resampling the nine source problems.

Quantity	Skeleton-disjoint	Augmented
All five-call rows, registered rule	+10.5 (+2.9/+11.4/+17.1)	+9.2 (+11.1/+8.2/+8.2)
Review rows, F1	+9.9 (0.0/+11.1/+18.5)	+10.0 (+12.5/+10.6/+6.9)
Clean rows	+9.7 (−4.2/+12.5/+20.8)	+8.5 (+1.9/+14.0/+9.6)
Exact matching	+10.5 (+8.6/+11.4/+11.4)	+10.3 (+13.8/+8.2/+8.8)
Review rows, common 512 cap	+28.4	+23.5
Clean rows, common 512 cap	+30.6	+26.1
Exact matching, common 512 cap	+26.7	+24.5
Skeleton-balanced mean	+15.6	+13.0
Leave one skeleton out	7.5–17.5	6.1–15.8
Source-balanced mean	+11.1	+9.2
Leave one source out	7.5–14.0	6.1–11.2
Source-cluster bootstrap, 95%	[0.9, 20.2]	[2.2, 17.1]
Seed × skeleton cells +/0/−	7/10/1	9/7/2
Registered rule on exact matching	met	met
F1: review-row difference > 0 on both sets	met	
F2: all leave-one-out and seed values > 0	met	

Table 21: E3 five-call accuracy difference in points at the saturation caps, with seed-rows pooled over three seeds, by skeleton in the upper block and by source problem in the lower block, with source labels as in Table 27.

	Skeleton-disjoint		Augmented	
Unit	Seed-rows	Difference	Seed-rows	Difference
S1	12	+8.3	120	+16.7
S2	48	+2.1	480	+1.7
S3	12	0.0	120	0.0
S4	12	+16.7	120	+17.5
S5	12	+33.3	120	+32.5
S6	9	+33.3	63	+9.5
S1	12	+8.3	120	+16.7
S2a	12	−16.7	120	−5.8
S2b	12	0.0	120	+2.5
S2c	12	+8.3	120	+5.0
S2d	12	+16.7	120	+5.0
S3	12	0.0	120	0.0
S4	12	+16.7	120	+17.5
S5	12	+33.3	120	+32.5
S6	9	+33.3	63	+9.5

Table 22: E3 accuracy by gold call count on all rows in %, as seed means with each format at its saturation cap. Comp.: compositional; Flat-C: corrected flat.

	Skeleton-disjoint				Augmented			
Calls	Rows	Comp.	Flat-C	Diff.	Rows	Comp.	Flat-C	Diff.
2	32	72.9	51.0	+21.9	–	–	–	–
3	360	53.6	53.1	+0.6	–	–	–	–
4	72	37.5	26.4	+11.1	718	37.1	28.0	+9.1
5	35	28.6	18.1	+10.5	341	25.0	15.8	+9.2
$\geq$6	8	4.2	0.0	+4.2	80	0.8	1.7	−0.8
All	507	50.0	45.9	+4.1	1139	30.9	22.5	+8.4

Retrained checkpoints on the GSM8K test split, E7. Registered before any E7 decoding: the six E3 checkpoints are decoded greedily and unconstrained on the 1,319 GSM8K test problems of Appendix B, the compositional ones at 1024 and the corrected-flat ones at 2048 tokens, each scored by its E3 scorer. Panels: all problems, and the structural subset E5-SD-CA4 with its strata. A difference is claimed on a panel only where both formats are saturated, that is, where each format's seed-mean length-finish rate is at most 1%. Criterion E7c, per panel: the saturated seed-mean accuracy difference, compositional minus corrected flat, is positive and positive in at least two of the three seeds. While the run was decoding and before any of its outputs was read, we added exact matching as in F4, re-applying E7c, and a paired interval that resamples problems with 10,000 draws and a 95% percentile interval, where the seed-mean difference is the mean over problems of the per-problem seed-mean difference. On every executed E7 row the stored 1%-tolerance flag equals a fresh rescoring, in all 7,238 cases. Table 23 gives the seed means, differences, intervals, and decisions, and Table 24 every registered endpoint per format, panel, and seed.

Table 23: Retrained checkpoints on the GSM8K test split, E7: execution and accuracy in %, as seed means for compositional / corrected flat, and the accuracy difference in points, compositional minus corrected flat, with the per-seed differences for seeds 42/123/456 in parentheses. At most 0.1% of either format's outputs end by length on any panel. Exact: difference under exact matching, with the number of positive seeds. Interval: problem-resampling 95% interval; under exact matching it is $[2.5, 5.8]$ and $[-4.0, 2.3]$. Met: E7c, 1% tolerance / exact. A dash marks a quantity not computed for the operation-count strata.

Panel	N	Exec.	Acc.	Difference	Exact	Interval	Met
GSM8K test	1,319	87.6 / 95.3	20.7 / 16.6	+4.1 (+4.0/+5.8/+2.7)	+4.1 (3/3)	[2.5, 5.8]	yes / yes
E5-SD-CA4	206	84.1 / 94.7	8.4 / 9.5	−1.1 (−2.4/+1.0/−1.9)	−1.0 (1/3)	[−4.2, 1.9]	no / no
1–2 operations	3	88.9 / 100.0	22.2 / 22.2	0.0 (+33.3/0.0/−33.3)	–	–	–
3–4 operations	69	88.9 / 90.8	14.0 / 12.6	+1.4 (−1.4/+2.9/+2.9)	–	–	–
$\geq$5 operations	134	81.6 / 96.5	5.2 / 7.7	−2.5 (−3.7/0.0/−3.7)	–	–	–

Second model family, E6. Registered before any E6 run or output existed: Llama-3.1-8B-Instruct (Grattafiori et al., 2024) is fine-tuned with LoRA by the recipe of Appendix E for three epochs and 900 optimizer steps with seeds 42, 123, and 456 on the same 4,795 historical flat targets, merged, and decoded greedily and unconstrained at 512 and 1536 tokens by the evaluator of the GSM8K panel on the 507-row skeleton-disjoint, 1,139-row augmented, and 1,319-row GSM8K panels, whose E5-SD-CA4 rows form the structural panel. With L_s the rows that end by length at 512 tokens, the rule holds for a seed and panel if at most 1% of the panel ends by length at 1536; L_s is nonempty, and the structural panel is scored when L_s has at least 30 rows; at least 90% of L_s parse at 1536; the number of executing rows in L_s rises; and the offline prediction of the 512-token arm from the 1536-token ids, with the E5b rule and the Llama tokenizer, is within 2 points of the decoded length-finish rate and agrees with at least 98% of length-finish states and 95% of execution and correctness outcomes. Table 25 lists all twelve seed-by-panel decisions; the earlier structural subsets E5-SD-CA2 and CA3 give the same decisions. The rule concerns validity and trajectories; accuracy is descriptive and, as seed means from 512 to 1536 tokens, moves from 50.5% to 51.5% on the skeleton-disjoint panel, from 24.8% to 28.5% on the augmented panel, and

Table 24: Retrained checkpoints on the GSM8K test split, E7: every registered endpoint in % per panel with its size N, seed, and format. Len: length finish; parse and execution as in Section 3; accuracy uses 1% tolerance; Exact: accuracy under exact matching, computed for the two panels; a dash marks a quantity not computed for the operation-count strata.

		Compositional, cap 1024					Corrected flat, cap 2048				
Panel, N	Seed	Len	Parse	Exec.	Acc.	Exact	Len	Parse	Exec.	Acc.	Exact
GSM8K test, 1,319	42	0.0	97.7	85.9	21.5	21.5	0.1	99.8	94.8	17.5	17.4
	123	0.1	98.9	86.8	20.9	20.8	0.0	100.0	95.5	15.2	15.2
	456	0.0	98.7	90.1	19.6	19.6	0.1	99.9	95.6	17.0	17.0
E5-SD-CA4, 206	42	0.0	98.1	83.0	8.7	8.7	0.0	99.5	93.2	11.2	10.7
	123	0.0	98.1	80.1	8.7	8.7	0.0	100.0	95.6	7.8	7.8
	456	0.0	98.5	89.3	7.8	7.8	0.0	100.0	95.1	9.7	9.7
1–2 ops., 3	42	0.0	100.0	100.0	33.3	–	0.0	100.0	100.0	0.0	–
	123	0.0	100.0	66.7	33.3	–	0.0	100.0	100.0	33.3	–
	456	0.0	100.0	100.0	0.0	–	0.0	100.0	100.0	33.3	–
3–4 ops., 69	42	0.0	100.0	87.0	13.0	–	0.0	98.6	88.4	14.5	–
	123	0.0	98.6	87.0	13.0	–	0.0	100.0	92.8	10.1	–
	456	0.0	98.6	92.8	15.9	–	0.0	100.0	91.3	13.0	–
$\geq$5 ops., 134	42	0.0	97.0	80.6	6.0	–	0.0	100.0	95.5	9.7	–
	123	0.0	97.8	76.9	6.0	–	0.0	100.0	97.0	6.0	–
	456	0.0	98.5	87.3	3.7	–	0.0	100.0	97.0	7.5	–

from 17.3% to 18.0% on GSM8K. The per-row E6 predictions, with generated token IDs, are in the supplement.

Table 25: Registered decisions of the second model family E6, per panel and seed. $|L_s|$: rows ending by length at 512 tokens; Len 1536: % of the panel ending by length at 1536; parse and execution are counted on L_s; $|\Delta|$: absolute difference in percentage points between predicted and decoded 512-token length-finish rates; agreement: offline versus decoded 512-token arm in %, length finish / execution / correctness.

Panel	N	Seed	$\lvert L_s\rvert$	Len 1536	Parse	Exec. 512→1536	$\lvert\Delta\rvert$	Agreement	Met
Skeleton-disjoint	507	42	43	0.00	43	0→43	0.39	99.6 / 99.6 / 99.2	yes
Skeleton-disjoint	507	123	43	0.59	40	0→40	0.00	100.0 / 100.0 / 100.0	yes
Skeleton-disjoint	507	456	35	0.00	35	0→27	0.20	99.8 / 99.8 / 99.2	yes
Augmented	1,139	42	384	0.00	384	0→384	0.09	98.2 / 98.2 / 99.5	yes
Augmented	1,139	123	427	2.46	399	0→398	0.53	98.9 / 98.9 / 99.5	no
Augmented	1,139	456	306	0.00	306	0→249	0.18	98.8 / 98.7 / 98.9	yes
GSM8K	1,319	42	193	0.00	193	2→186	0.08	99.2 / 98.7 / 99.5	yes
GSM8K	1,319	123	183	0.08	182	3→172	0.15	99.2 / 98.9 / 99.3	yes
GSM8K	1,319	456	167	0.30	153	0→144	0.45	98.3 / 98.0 / 99.5	yes
E5-SD-CA4	206	42	44	0.00	44	0→42	0.49	99.5 / 99.0 / 99.5	yes
E5-SD-CA4	206	123	37	0.00	37	1→34	1.46	98.5 / 98.5 / 98.5	yes
E5-SD-CA4	206	456	35	0.00	34	0→31	0.00	98.1 / 98.5 / 99.0	yes

Synthetic panel, E4. The registration also contains a validity-only run on 1,200 synthetic problems, 200 per call count from two to seven, whose texts do not name the operations, so their answers are not determined by the text; caps of 512, 1536, and 2048 tokens give 10,800 predictions. Its rule requires, in every seed and at every call count where at least 10% of 512-token outputs end by length, that execution rises and at most 1% of outputs end by length at 1536. Qualifying call counts are chosen per seed; this reading of the rule was fixed after the E4 decoding had started and before any of its predictions was read. Table 26 lists every seed-level decision; the main text does not use this panel. Counts at 2048 tokens equal those at 1536.

Table 26: Registered decisions of the synthetic panel E4, per seed and qualifying call count with 200 rows each; counts at 512→1536 tokens.

Seed	Calls	Length finishes	Executed	Met
42	7	48→4	154→189	no
123	4	23→0	182→199	yes
123	5	35→0	168→199	yes
123	6	30→0	170→199	yes
123	7	73→0	127→196	yes
456	6	30→0	170→198	yes
456	7	59→0	140→199	yes

Source-level effects and symmetry-normalized binding. Two descriptive analyses of existing outputs were registered after the aggregate, per-skeleton, binding, and repair results had been read and before their own quantities were computed. For B1, using the source assignment of the semantic review, Table 27 gives five-call execution and accuracy at 512 and 1536 tokens for each of the nine source problems, the source-balanced execution gain, which is the unweighted mean of the per-source gains, and the range of the pooled execution gain when one source is left out. For B2, two argument slots of a library primitive are interchangeable if swapping their values leaves the output unchanged on 200 random positive draws with a fixed random seed and at least 50 draws with a defined output; $R_{\mathrm{slot}}^{\mathrm{sym}}$ compares the values of interchangeable slots as multisets and is otherwise R_{slot}, on the same aligned calls and denominators. Fourteen of the 31 primitives have interchangeable slots. The registered descriptive criterion, that $R_{\mathrm{slot}}^{\mathrm{sym}}$ at five calls exceeds its value at two calls on the skeleton-disjoint set and at four calls on the augmented set, is met with 0.232 versus 0.078 and 0.270 versus 0.147 in Table 12.

Table 27: Five-call execution and accuracy by source problem for B1, in %, from 512 to 1536 tokens, unconstrained, with seed-rows pooled over three seeds. S2a–S2d are the four S2 sources on baking ingredients, breakfast, earrings, and school shirts; S2c is the source with a unit ambiguity, and S2a holds 38 of the 42 five-call rows that fail the consistency audit. Source-balanced execution gain: 60.2 points on the skeleton-disjoint and 64.0 on the augmented set; leave-one-source-out range of the pooled gain: 54.8–67.7 and 58.4–70.9 points.

	Skeleton-disjoint			Augmented		
Source	Seed-rows	Execute	Accuracy	Seed-rows	Execute	Accuracy
S1	12	83.3→100.0	75.0→75.0	120	80.8→100.0	74.2→74.2
S2a	12	0.0→100.0	0.0→50.0	120	0.0→100.0	0.0→17.5
S2b	12	0.0→100.0	0.0→75.0	120	0.0→100.0	0.0→93.3
S2c	12	0.0→41.7	0.0→0.0	120	2.5→56.7	0.0→0.0
S2d	12	0.0→100.0	0.0→41.7	120	0.0→98.3	0.0→76.7
S3	12	41.7→100.0	41.7→41.7	120	35.8→100.0	34.2→33.3
S4	12	25.0→83.3	25.0→25.0	120	33.3→90.0	17.5→25.0
S5	12	100.0→100.0	0.0→0.0	120	94.2→100.0	4.2→4.2
S6	9	33.3→100.0	0.0→0.0	63	20.6→98.4	0.0→0.0