The Communication Bottleneck: A Round-Trip Study of Tree-Structured Expression Serialization in Language Models

Xavier Suau Apple
Alex Ferrando Apple
Luca Zappella Apple
Samy Bengio Apple

Abstract

When language models reason in chain-of-thought or exchange free-text intermediates, they serialize structured information into natural language. How much tree-structured compositional content survives this bottleneck? We propose a round-trip protocol that answers this question empirically for tree-structured expressions. A generator converts a procedurally generated arithmetic expression into a word problem, a separate extractor recovers the expression from the word problem alone, and symbolic equivalence provides an exact oracle. Evaluating all pairwise combinations of sixteen models yields a communication matrix whose marginals separate generation quality from extraction quality. Three main findings emerge. First, the channel is lossy and asymmetric: swapping which model generates and which extracts shifts accuracy by up to 60.4 points, and the best pair reaches 92.9% by combining different models on each end rather than the same model on both. Second, at least 73.6% of round-trip failures originate at generation, and difficulty is driven by tree structure (operator count, depth, right-branching) rather than model family. Third, the channel is trainable: $\sim$3600 fine-tuning examples that share the evaluation's operators and tree shapes lift every open-weight model above untrained Gemini-3.1-Pro, an upper bound under matched semantics. A disjoint-domain regime with new operators and vocabulary also raises every open-weight model, confirming the gain is not an artifact of matched semantics, though a gap to the frontier remains. Together these results identify tree-structured expression serialization as a primary limiting factor when models communicate hierarchical structure through natural language.

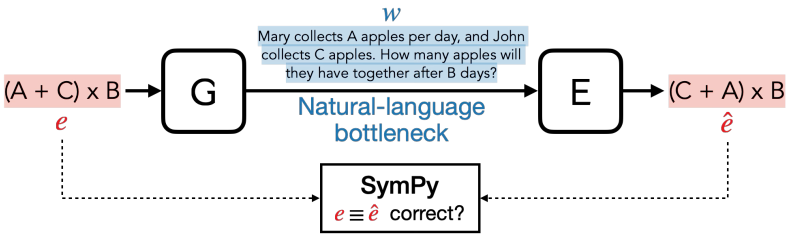

Figure 1: **The round-trip protocol.** A generator G serializes an arithmetic expression e (a tree) into a word problem w that passes through a natural-language bottleneck. A separate extractor E recovers $\hat{e}$ from w alone, and SymPy serves as an exact oracle for symbolic equivalence $e \equiv \hat{e}$. Any compositional detail the generator fails to encode is irreversibly lost, regardless of E.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Language models routinely serialize structured information into natural language (NL): chain-of-thought reasoning linearizes intermediate computations into prose [Wei et al., 2022], model outputs are consumed by downstream models in pipelines [Wu et al., 2024, Hong et al., 2024], and tool-use descriptions encode structured calls as text [Schick et al., 2023]. If this serialization is lossy [Herbst et al., 2025], any structure that the encoding step fails to express unambiguously is irreversibly lost, regardless of the downstream consumer's capability. How much compositional structure actually survives the natural-language bottleneck? We study this for hierarchical structure using tree-structured expressions built from arithmetic and logical operators, which admit an exact correctness oracle via symbolic equivalence.

Answering this question requires measuring communication *between* models, but existing benchmarks are not designed for it. Whether evaluating math [Cobbe et al., 2021, Hendrycks et al., 2021b, Glazer et al., 2025], reasoning [Rein et al., 2023, Phan et al., 2026], or code [Jimenez et al., 2024, Jain et al., 2024], they test single models on fixed problem sets. The closest prior art is Allamanis et al. [2024], who round-trip code through natural language but deliberately keep encoder and decoder fixed to the same model to sidestep what they term the 'communication chasm' between distinct models. We instead treat that chasm as the object of study: separating generator from extractor across an $N \times N$ grid of models yields a measurement instrument that exposes role asymmetry, fault attribution, and structural predictors of failure, all invisible under same-model analysis.

One might ask why study natural language at all. While structured formats like JSON, code or S-expressions natively preserve compositional hierarchy and should be utilized whenever feasible (Section 6), they do not reflect how models typically operate. Natural language remains the *de facto* channel in all the settings listed above, despite evidence that alternative formats can improve both reasoning and communication [Chen et al., 2024b]. Measuring NL compositional fidelity requires a domain with an exact correctness oracle, independent control over structural complexity, and freedom from contamination [Zhang et al., 2024]. Arithmetic satisfies all three: (i) symbolic equivalence gives a ground-truth oracle, (ii) expression trees expose operator count, nesting depth, and branching topology as controllable parameters, and (iii) instances can be procedurally generated.

To quantify the channel's capacity for compositional structure, we propose a **round-trip protocol** (Figure 1). A generator G encodes an arithmetic expression e, sampled from an enumerated space of tree skeletons, into a word problem w. A separate extractor E recovers $\hat{e}$ from w alone, verified by symbolic equivalence. A rule-based guard ensures w contains no mathematical notation, forcing structure through a linguistic channel (Section 2.1). Evaluating N models in both roles yields an $N \times N$ *communication matrix* whose marginals decompose performance into generation and extraction quality (Section 2.3). We replicate the protocol on propositional logic as a robustness check (Appendix R).

We evaluate 16 models: 12 open-weight (0.6B–32B) across five families and 4 frontier, on 2450 expressions with operator counts 2–8 and depths 2–6. Our contributions are:

- **A round-trip protocol for communication fidelity.** A generator encodes a procedurally generated expression as a word problem, an extractor recovers it, and symbolic equivalence serves as an exact oracle[1]. Evaluating N models in both roles yields an $N \times N$ communication matrix that decomposes pairwise performance into generation and extraction quality (Section 2).
- **Generation is the bottleneck, not extraction.** At least 73.6% of round-trip failures originate at the generation step. Role assignment alone shifts accuracy by up to 60.4 pp, and the highest fidelity is achieved by a cross-model pair, not by self-communication (Sections 3.2 and 3.3).
- **Failure is governed by tree structure, not model family.** Operator count, nesting depth, and a linearization complexity measure $\ell(T)$ dominate failure prediction. Right-branching trees yield lower accuracy than left-branching ones of identical size, and $\ell(T)$ captures this asymmetry directly (Section 3.3). A propositional-logic replication preserves the main rankings and fault pattern (Appendix R).

[1] Code at `https://github.com/apple-aiml-research/ml-roundtrip`

- **The bottleneck is compositional serialization, not domain knowledge.** Fine-tuning on a novel operator vocabulary with no arithmetic overlap lifts every open-weight generator, and in-domain fine-tuning on approximately 3600 examples pushes all of them above the strongest untrained frontier generator. The bottleneck is trainable under matched semantics and partially trainable across domains (Section 4).

2 Protocol Design

2.1 Expression Suite

Standard math-word-problem datasets such as GSM8K offer no control over structural complexity: expressions are hand-authored with uncontrolled nesting depth, operator count, and tree shape. We instead generate expressions procedurally from abstract syntax tree *skeletons*. Given a target operator count k and depth d, we recursively enumerate every binary-tree skeleton matching the pair (k, d). Each skeleton is then instantiated by assigning operators uniformly from $\{+, -, \times, \div\}$ to internal nodes and filling leaves with named variables from $\{A, B, \ldots, H\}$ (probability 0.7) or integer constants in $[1, 10]$ (probability 0.3). Locally degenerate subexpressions are rejected at instantiation: same-leaf patterns under a non-commutative operator ($X - X$, X/X) and multiplicative identities ($X \times 1$, $1 \times X$, $X/1$). The operator set, variable alphabet, and leaf-type probabilities are free parameters of the procedure. The values above are reasonable defaults, not tuned choices. Since the skeleton fixes the topology before instantiation, every resulting expression has the target k and d by construction (Figure 2), enabling the factorial analyses of depth, operator count, and tree shape reported in Section 3.

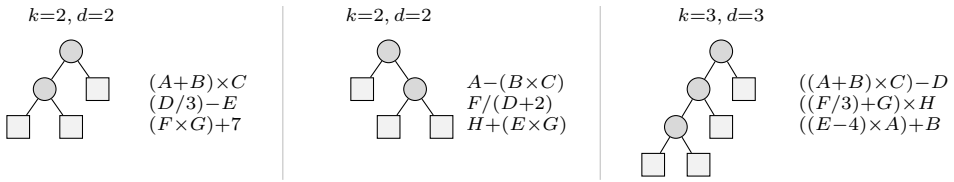

Figure 2: **Expression generation via skeleton enumeration.** Circles denote operator nodes, squares denote leaf positions. We fill each operator node with a random element of $\{+, -, \times, \div\}$ and each leaf with a variable or constant, producing distinct expressions that share the same structure. The first two skeletons share $(k{=}2, d{=}2)$ but differ in branching direction; the third has $(k{=}3, d{=}3)$.

2.2 Round-Trip Protocol

The round-trip $e \xrightarrow{\mathsf{G}} w \xrightarrow{\mathsf{E}} \hat{e}$ (Figure 1) proceeds as follows. A **generator** G, prompted with $\pi_{\mathsf{G}}^{\gamma}$ containing γ in-context (e, w) exemplars, produces a short word problem $w = \mathsf{G}(e; \pi_{\mathsf{G}}^{\gamma})$ using variable placeholders and no mathematical symbols. A rule-based *leakage guard* ($\mathrm{Leak} : w \mapsto \{0, 1\}$) rejects outputs that leak mathematical syntax, ensuring structural information passes through a genuinely linguistic channel. Outputs missing required placeholders or exhibiting degenerate repetition are counted separately as generator faults (Appendix C). Guard false negatives bias round-trip accuracy upward, so the reported figures are conservative estimates of channel fidelity. An **extractor** E, prompted with $\pi_{\mathsf{E}}^{\varepsilon}$ containing ε exemplars, recovers $\hat{e} = \mathsf{E}(w; \pi_{\mathsf{E}}^{\varepsilon})$, verified by symbolic equivalence $\texttt{sympy.simplify}(\hat{e} - e) = 0$ [Meurer et al., 2017], which accepts valid rearrangements (*e.g.,* $A + B$ vs. $B + A$) but catches all semantic errors. Any detail the generator fails to express unambiguously is irreversibly lost, regardless of the extractor. Prompts are detailed in Appendix A, worked examples in Appendix K. Shot exemplars are drawn from a pool disjoint from the evaluation set, and the shot counts γ, ε are fixed in Section 3.1. Because our contribution is the protocol itself, $\pi_{\mathsf{G}}^{\gamma}$ and $\pi_{\mathsf{E}}^{\varepsilon}$ are parameters rather than fixed choices, and prompt tuning is orthogonal to the measurements we report.

2.3 Communication Matrix

Given N models, each serving as both G and E, the protocol produces an $N \times N$ communication matrix $\boldsymbol{Q}$ whose entry q_{ij} is the round-trip accuracy when extractor i reads generator j's word problems.

For each $e \in D$, generator j produces $w = \mathsf{G}_j(e)$. If w fails the leakage guard ($\mathrm{Leak}(w) = 0$), the round trip scores zero, otherwise extractor i recovers $\hat{e} = \mathsf{E}_i(w) = \mathsf{E}_i\big(\mathsf{G}_j(e)\big)$:

$$q_{ij} \;=\; \frac{1}{|D|} \sum_{e \in D} \mathrm{Leak}\big(\mathsf{G}_j(e)\big) \;\cdot\; \mathbb{1}\big[\,\mathsf{E}_i\big(\mathsf{G}_j(e)\big) \equiv e\,\big]. \tag{1}$$

The column average $q_j^{\mathsf{G}} = \frac{1}{N}\sum_i q_{ij}$ measures the **generation quality** of model j: how decodable its word problems are across extractors. The row average $q_i^{\mathsf{E}} = \frac{1}{N}\sum_j q_{ij}$ measures the **extraction quality** of model i: how reliably it recovers expressions across generators. The diagonal entry q_{ii} is the **self-communication** score. The denominator is the full suite $|D|$, not each generator's guard-passing subset, so guard failures count as communication failures, eliminating the selection bias from evaluating generators only on expressions they can encode. A consequence is that q_i^{E} is panel-relative: guard failures enter as zeros in every extractor row, so absolute values reflect the generator mix. This balanced design, in which every G/E pair is evaluated on the same expression set, ensures that marginal averages are fair comparisons. A latent-ability analysis confirming that raw marginals track maximum-likelihood skill estimates is reported in Appendix N.

2.4 Fault Attribution

When a round trip fails ($\hat{e} \not\equiv e$), the error may originate at the generation or extraction step. Because the expression actually encoded by w is unobserved, we estimate it via multi-extractor consensus. For a word problem $w = \mathsf{G}_j(e)$, we define the *extractor vote* for any candidate expression e' as

$$P(e' \mid w) \;=\; \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\big[\mathsf{E}_i(w) \equiv e'\big]. \tag{2}$$

If $P(e \mid w) > 0$, at least one extractor recovers the target, proving the word problem faithful, and any remaining failures are considered **extractor faults**. If $P(e \mid w) = 0$, no extractor succeeds and we classify the failure as a **generator fault**. The severity of the generator fault is gauged by the **extraction agreement** $p^*(w) = \max_{e'} P(e' \mid w)$, *i.e.,* the fraction of extractors that converge on the most common output. High p^* means most extractors recover the same wrong expression, indicating that the generator produced a coherent word problem that systematically describes a different expression than e. Low p^* means extractors disagree, indicating that the word problem is garbled or ambiguous. Attribution results are reported in Section 3.3.

A generation counts as faithful if *any* of the N=16 extractors succeeds, so enlarging the panel can only shift failures from the generator-fault bucket to the extractor-fault bucket. The reported generator-fault rate is therefore conservative with respect to the bottleneck claim: every smaller sub-panel we tested reports a higher rate, up to 91.1%. Misattribution is negligible at this N (Section M.1).

3 Experimental Results

3.1 Setup

We evaluate $N = 16$ models across three groups. Ten open-weight auto-regressive: Qwen3 [Yang et al., 2025] (0.6B, 1.7B, 4B, 8B, 14B, 32B), Gemma-3 [Gemma Team, 2025] (4B, 12B, 27B), and Phi-4 [Abdin et al., 2024]. Two diffusion LMs, Dream-v0-7B [Ye et al., 2025a] and LLaDA-8B [Nie et al., 2025], to test whether parallel denoising yields a different serialization profile. Four frontier models via API: Claude Haiku 4.5 [Anthropic, 2025b], GPT-5 [Singh et al., 2025], Gemini-3-Flash [Google DeepMind, 2025a], and Gemini-3.1-Pro [Google DeepMind, 2025b]. All use greedy decoding (temperature 0, thinking disabled), Appendix E confirms $\mathrm{temp} = 0$ yields the best accuracy. The expression suite contains $|D| = 2450$ expressions, 5 instantiations per skeleton with up to 50 skeletons per (k, d) cell, for $k \in [2, 8],\ d \in [2, 6]$, with uniform random subsample with fixed seed when a cell exceeds 50 (not all pairs are feasible, see Appendix B). Guard pass rates are reported alongside all results in Table 1.

Shot configuration. We ablate $(\gamma, \varepsilon) \in \{0, 3\}^2$ and report Table 1 at the trade-off optimum $(0, 3)$: generator shots (γ=3) do not help the strongest auto-regressive models (Qwen3 $\geq$4B, Phi-4), while extractor shots uniformly improve extraction over ε=0. Frontier models were evaluated only at γ=0 for tractability. Full per-condition grids are in Appendix P.

Table 1: **The 16 × 16 communication matrix on all generated expressions (guard failures counted as errors).** Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. The rightmost column and bottom row show extraction quality (row average) and generation quality (column average). Generator and extractor are run with prompts π^0_{G} and $\pi^{\varepsilon}_{\mathsf{E}}$ respectively (configuration $\gamma{=}0, \varepsilon{=}3$); see Appendix P for the full ablation. The $\pm$ values denote standard deviation across the 16 pairings in each row or column. The Leak % row reports each model's leakage-guard pass rate as a generator. Thin rules separate open-weight auto-regressive, diffusion, and frontier models.

		GENERATORS															
		Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
	Leak pass %	20	60	94	98	99	99	98	96	96	97	64	95	100	100	100	100	
EXTRACTORS	Qwen3-0.6B	0.0	1.2	3.2	4.5	3.4	2.2	0.1	0.4	0.3	4.7	0.5	4.0	7.2	1.9	1.9	1.6	$2.3_{\pm 2}$
	Qwen3-1.7B	0.0	1.8	7.7	11.2	9.9	7.9	0.5	2.0	1.3	14.4	1.0	9.2	24.9	11.3	11.8	13.6	$8.0_{\pm 7}$
	Qwen3-4B	0.0	2.8	13.6	20.3	21.3	19.0	0.7	3.4	3.1	26.3	2.1	15.8	49.1	39.0	34.6	46.0	$18.6_{\pm 16}$
	Qwen3-8B	0.1	2.7	14.2	19.8	21.7	20.7	0.8	3.1	3.8	27.6	2.0	16.5	49.0	44.2	39.1	52.4	$19.9_{\pm 17}$
	Qwen3-14B	0.1	2.9	16.1	23.3	24.9	23.6	0.8	4.0	4.3	32.0	2.4	19.1	57.1	57.1	50.6	65.1	$24.0_{\pm 22}$
	Qwen3-32B	0.1	3.1	16.2	23.3	25.7	24.8	0.9	3.9	4.4	31.3	2.9	19.3	57.2	64.1	51.7	70.2	$25.0_{\pm 23}$
	Gemma-3-4B	0.1	2.1	10.2	13.5	13.3	10.8	0.6	2.9	1.8	17.8	1.4	11.7	29.5	17.4	17.8	19.8	$10.7_{\pm 8}$
	Gemma-3-12B	0.0	2.4	13.3	18.4	19.4	18.0	0.7	3.2	3.1	24.5	2.3	15.5	45.3	38.0	32.7	44.3	$17.6_{\pm 15}$
	Gemma-3-27B	0.1	2.8	15.9	21.5	24.1	23.1	1.1	3.8	4.3	29.8	2.4	17.6	54.5	56.3	46.3	64.8	$23.0_{\pm 21}$
	Phi-4	0.0	2.9	17.1	23.3	26.7	24.5	0.9	4.2	4.5	32.8	2.6	19.9	58.3	65.0	52.3	72.8	$25.5_{\pm 24}$
	Dream-7B	0.0	1.8	7.5	11.9	10.6	10.4	0.4	2.2	1.8	13.1	1.1	9.1	22.7	18.2	19.4	31.3	$10.1_{\pm 9}$
	LLaDA-8B	0.0	2.3	12.4	17.1	17.6	15.9	0.5	3.2	2.8	21.5	2.4	15.1	38.9	37.7	32.2	39.4	$16.2_{\pm 14}$
	C-Haiku-4.5	0.0	3.1	16.5	23.3	26.7	25.7	0.9	3.8	4.6	33.0	3.0	19.0	59.6	79.1	56.3	82.5	$27.3_{\pm 27}$
	GPT-5	0.1	3.3	17.6	24.9	28.1	24.0	0.9	4.6	4.4	34.7	2.9	21.9	64.7	91.3	64.5	88.5	$29.8_{\pm 30}$
	G-3-Flash	0.1	3.3	16.8	24.4	27.2	25.6	1.0	3.9	4.4	34.2	2.9	20.4	62.5	89.3	60.6	88.9	$29.1_{\pm 29}$
	G-3.1-Pro	0.0	2.8	16.7	24.1	27.1	25.6	0.8	4.1	4.4	35.5	3.0	20.8	62.9	92.9	62.4	88.7	$29.5_{\pm 30}$
	Gen. quality	$0.1_{\pm 0}$	$2.6_{\pm 1}$	$13.4_{\pm 4}$	$19.0_{\pm 6}$	$20.5_{\pm 7}$	$18.8_{\pm 7}$	$0.7_{\pm 0}$	$3.3_{\pm 1}$	$3.3_{\pm 1}$	$25.8_{\pm 9}$	$2.2_{\pm 1}$	$15.9_{\pm 5}$	$46.5_{\pm 17}$	$50.2_{\pm 28}$	$39.6_{\pm 19}$	$54.4_{\pm 27}$	

3.2 Characterizing the Natural-Language Channel

The full communication matrix (Table 1; per-depth breakdowns in Appendix F) reveals a channel whose capacity rises with model capability and depends on how models are paired across roles. Across the 12 open-weight models, mean round-trip accuracy (q_{ij}, Equation (1)) is 9.4%, rising to 17.0% at $\geq$8B. Frontier models raise the ceiling: the 4×4 frontier submatrix averages 74.7%, and the best cell reaches 92.9% (Gemini-3.1-Pro reading GPT-5), above the best self-communication score (max 91.3%, GPT-5) and 4.2 pp above Gemini-3.1-Pro reading from itself. The best cell is itself a role-asymmetry result: the strongest channel pairs the strongest generator with the strongest extractor, not the same model on both sides. Accuracy still degrades with complexity at every tier. Frontier generators drop from 88% at $k{=}3$ to 71% at $k{=}8$, and open-weight $\geq$8B generators from 64% at $k{=}2$ to 13% at $k{=}8$. Diffusion parallel denoising does not escape the bottleneck. LLaDA-8B (gen/extr $15.9/16.2$) and Dream-v0-7B ($2.2/10.1$) both sit within the autoregressive envelope.

Generation and extraction are distinct skills. q^{G} and q^{E} are not directly comparable in absolute terms because each marginal is upper-bounded by the panel's quality on the opposite role, but within-cohort ranks are comparable. Figure 3-left plots each model's rank as G against its rank as E. Had both roles reflected one ability, models would sit on the diagonal, but they disperse instead. GPT-5 is the top extractor ($q^{\mathsf{E}} = 29.8\%$, rank 1) yet ranks second as generator ($q^{\mathsf{G}} = 50.2\%$); Gemini-3.1-Pro leads generation ($q^{\mathsf{G}} = 54.4\%$, rank 1) but is only the second-best extractor ($q^{\mathsf{E}} = 29.5\%$). The dispersion is largest for Gemma-3-27B (rank 11 generator at 3.3%, rank 8 extractor at 23.0%): a model whose narration loses structure more than its parsing does.

Role assignment matters. Swapping which model generates and which extracts shifts accuracy by up to 60.4 pp (Gemma-3-27B with Gemini-3.1-Pro, larger than any row or column spread in Table 1), where Gemma's stronger extraction (rank 8) and weak generation (rank 11) make role assignment

decisive. The effect also appears within the frontier panel: GPT-5 generating with Gemini-3.1-Pro extracting yields 92.9%, the reverse 88.5% (4.4 pp). Pairing the strongest model on both sides is suboptimal when the panel has a model with complementary strengths.

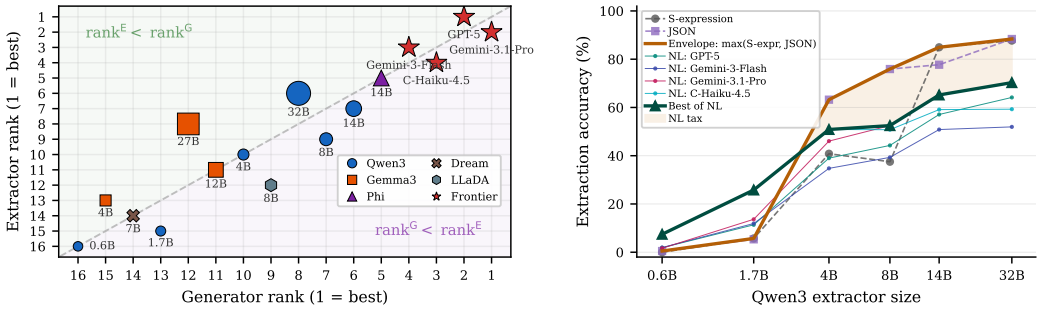

Figure 3: **(Left) Role asymmetry.** Generator rank vs. extractor rank (1 = best). Gemini-3.1-Pro leads generation but is second as extractor, GPT-5 is the top extractor but second generator. Swapping roles shifts round-trip accuracy by up to 60.4 pp. **(Right) Channel asymmetry.** Qwen3 extraction on JSON/S-expressions (structured, unambiguous) vs. Frontier-generated word problems. The shaded area is the *NL tax*: capacity large extractors have but no NL generator in our panel reaches.

The NL channel is quantifiably lossy. To separate channel loss from extractor weakness, we need a reference point where the encoding is lossless. We implement it by replacing the generator with deterministic S-expression and JSON-AST serializations and measure the NL tax $= \max(q^{\mathsf{E}}_{\text{S-expr}}, q^{\mathsf{E}}_{\text{JSON}}) - q^{\mathsf{E}}_{\text{NL}}$, the extraction gap between a lossless encoding and natural language. Figure 3-right shows this for the Qwen3 family: at 14B–32B, lossless extraction reaches 85–88% while NL extraction from Frontier models plateaus at 51–70%, an 18–20 pp tax that grows with complexity (Appendix Q). The gap is relative to the panel's best generator and would narrow with a stronger one. Below 4B, models lack fluency in structured notation and extract similarly from prose, so the structured-format advantage requires extractor familiarity.

3.3 Why Communication Fails

What predicts failure? We train a gradient-boosted classifier on all $(e, \mathsf{G}, \mathsf{E})$ triples from the 12 open-weight models and compute SHAP attributions [Lundberg and Lee, 2017, Lundberg et al., 2019] (Figure 4, left; details in Appendix G). Frontier models are excluded because their unknown parameter counts would confound the model-size features. Model size dominates: generator and extractor sizes together account for 57.0% of the attribution. R_{math} (19.5%, ↑), the fraction of words in the word problem belonging to a curated arithmetic lexicon (Appendix D), ranks second overall, ahead of extractor size, and is comparable in magnitude to the combined structural features (21.6%), indicating that pseudo-mathematical phrasing produces more decodable problems. Among structural features, depth (9.2%), operator count (7.8%), and *linearization complexity* $\ell(T)$ (4.6%) contribute in that order. We define $\ell(T)$ as the number of internal nodes of T whose right child is itself internal, counted tree-wide ($\ell{=}0$ for a left-caterpillar of any depth, $\ell{=}d{-}1$ for a right-caterpillar of depth d), reminiscent of the minimum register count in expression evaluation [Sethi and Ullman, 1970]. Controlling for depth, $\ell(T)$ predicts difficulty monotonically, with a mean within-depth Spearman correlation of $\overline{\rho}_{|d} = -0.48$ (Figure 4, center). Right-branching skeletons are therefore harder than left-branching ones of matched size. Per-cell breakdowns appear in Appendix H. Family match contributes 1.9%, providing no evidence of architecture-specific dialects.

Where does failure occur? Applying the consensus-based attribution (Section 2.4, with the full breakdown in Appendix M) to all failed round trips, we find that at least 73.6% are generator faults (no extractor recovers the target) and at most 26.4% are extractor faults. Wrong operator count is the dominant error type, accounting for 71.8% of all errors (Appendix I, with qualitative failure traces in Appendix L). The extractor-fault rate scales steeply with generator capability, from under 4% for sub-2B open-weight generators to 60–97% for frontier models (Figure 4, right). The 73.6% figure is robust to panel composition: across nine extractor-panel subsets the rate ranges 73.6–91.1%, with the full 16-model panel yielding the lowest observation (Section M.1). No panel we tested reports a generator-fault share below 73.6%.

Feature	Attribution
Generator size	39.1% ↑
R_{math}	19.5% ↑
Extractor size	17.9% ↑
Depth	9.2% ↓
OpCount	7.8% ↓
$\ell(T)$	4.6% ↓
Family match	1.9% ↑

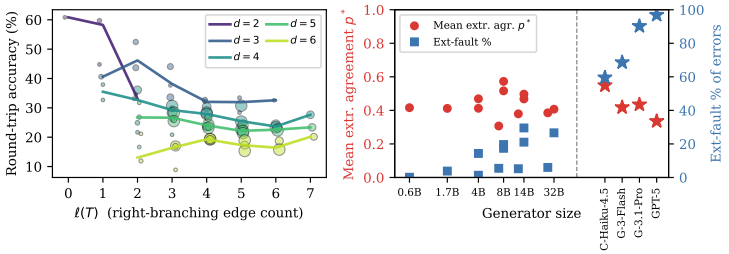

Figure 4: **Why communication fails.** Over depths 2–6. **(Left)** Global SHAP attributions, computed on the 12 open-weight models only (frontier parameter counts are unknown and would confound the size features): generator size 39.1%, R_{math} 19.5%, and extractor size 17.9% dominate. **(Center)** Mean within-depth Spearman correlation between linearization complexity $\ell(T)$ and accuracy is $\overline{\rho_{|d}} = -0.48$: right-branching skeletons are harder at matched depth. **(Right)** Extractor-fault rate rises with generator size, computed on all 16 models (12 open-weight plus Haiku, GPT-5 and the two Geminis), from $<4\%$ for the smallest open-weight generators to 60–97% for the frontier models.

3.4 External Validity

Do round-trip scores recapitulate existing benchmarks or capture a distinct capability? We correlate per-model round-trip scores with published results on 9 benchmarks across four categories (at least seven verified scores each). Extraction correlates strongly with general knowledge (MMLU $\rho = 0.98$), code (MBPP $\rho = 0.93$, LiveCodeBench $\rho = 0.93$), and reasoning (GPQA-Diamond $\rho = 0.95$), but generation correlates only moderately ($\rho = 0.47$–0.89). This gap is consistent across all four categories, confirming that serializing compositional structure into prose is partially distinct from the comprehension skills standard benchmarks test. Details in Appendix J.

Cross-domain replication. We re-run the protocol on propositional logic on a reduced 14×14 roster (five connectives, depths 2–5, logical equivalence as oracle; frontier Gemini omitted to limit compute; Appendix R). Absolute accuracy drops (mean cell 8.7% vs. 14.9% on the shared 14-model block), but model rankings mostly carry over: Spearman $\rho = 0.85$ for generation and 0.69 for extraction, with GPT-5 and Claude Haiku 4.5 again on top. The main exception is Phi-4, which falls from extraction rank 3 to 14, suggesting its arithmetic extraction relies on format-specific parsing. Rankings are stable overall but not uniform, reinforcing a core takeaway: the best model and role assignment depend on the domain, not just on overall capability.

4 Improving Communication Through Fine-Tuning

Generation is the bottleneck: at least 73.6% of round-trip failures originate there (Section 3.3). If the bottleneck is trainable rather than architectural, what kind of training lifts it and how far does it transfer? Two fine-tuning regimes test this, both sharing the round-trip format.

Arithmetic fine-tuning probes how far structural fluency carries across unseen variables and constants. Training pairs ($e \to w$ and $w \to e$) use the $\{+, -, \times, \div\}$ operators and the evaluation tree topology, with disjoint letters $\{Q \ldots Z\}$ and constants $[11 \ldots 20]$.

Assembly fine-tuning probes whether the linearization skill transfers across domains. Assembly has mixed-arity operators (three unary, six binary, four ternary), breaking the binary-only isomorphism, and coloured-object leaves (13 colours $\times$ 13 objects) disjoint from any arithmetic vocabulary. Each operator maps to multiple paraphrase templates in a NL procedural style. Every (expression, NL) pair yields two training examples ($e \to w$ and $w \to e$). Full specification in Appendix O.

Training and evaluation. Both regimes use QLoRA [Dettmers et al., 2023] (rank 32, α=64, 4-bit NF4, 3 epochs, AdamW [Loshchilov and Hutter, 2019], lr 2×10^{-4}, cosine schedule) on $\sim$3600 training messages across the 10 open-weight autoregressive models. Adapters are merged before evaluation.[2] We report each condition at its strongest shot configuration on the full grid (Appendix P): base and arithmetic FT use $\pi^0_{\mathsf{G}}, \pi^3_{\mathsf{E}}$, assembly FT uses $\pi^3_{\mathsf{G}}, \pi^3_{\mathsf{E}}$ because assembly-FT generators benefit from three exemplars but base and arithmetic-FT generators do not.

[2] A single hyperparameter setting is used across models. Per-model tuning is orthogonal to the structural-transfer question.

4.1 Fine-tuning Results

Table 2: **Fine-tuning comparison across 10 open-weight models.** Three conditions are evaluated at their strongest shot configuration (Appendix P): Base (γ=0, ε=3), assembly FT (γ=3, ε=3), arithmetic FT (γ=0, ε=3). Leak failures count as errors; all marginals are averaged over the 10-model extractor panel. **Bold** = best per row; underline = second-best.

	Leak pass %			q^{G}			q^{E}			Self		
Model	Base	FT_{asm}	FT_{arith}	Base	FT_{asm}	FT_{arith}	Base	FT_{asm}	FT_{arith}	Base	FT_{asm}	FT_{arith}
Qwen3-0.6B	20	52	**100**	0.1	0.1	**51.9**	3.0	3.7	**14.1**	0.1	0.0	**61.8**
Qwen3-1.7B	59	66	**100**	2.6	6.5	**57.2**	6.6	8.4	**34.8**	2.1	3.8	**70.4**
Qwen3-4B	94	82	**100**	14.0	17.3	**55.3**	12.4	20.9	**64.4**	15.3	21.5	**79.4**
Qwen3-8B	98	94	**100**	20.5	32.6	**59.5**	12.9	21.1	**66.2**	22.7	37.3	**83.0**
Qwen3-14B	99	98	**100**	21.4	26.2	**55.9**	14.5	24.2	**72.7**	27.3	36.7	**82.1**
Qwen3-32B	98	98	**100**	19.4	27.6	**58.0**	14.7	25.3	**73.6**	26.5	41.6	**85.6**
Gemma-3-4B	98	75	**100**	0.8	5.5	**55.4**	8.4	11.9	**37.1**	0.6	4.6	**48.4**
Gemma-3-12B	96	79	**100**	3.9	16.7	**57.8**	11.7	19.2	**59.8**	4.2	21.1	**81.2**
Gemma-3-27B	96	93	**100**	3.8	20.4	**58.4**	14.2	24.2	**70.6**	5.1	27.2	**84.6**
Phi-4	97	97	**100**	26.9	32.7	**58.2**	15.1	26.6	**74.4**	35.7	59.3	**84.0**

Arithmetic fine-tuning closes the gap to the frontier. Arithmetic FT raises generation quality from a baseline range of 0.1–26.9% to 51.9–59.5% across all 10 open-weight models (Table 2), and guard pass rates reach 100%, eliminating leakage entirely. Self-communication reaches 79–86% above 4B and 62% even for Qwen3-0.6B. The bottleneck quantified in Section 3 is therefore not architectural: training on the same operator set with disjoint variables and constants is enough to push every panel model above the strongest untrained frontier generator (Gemini-3.1-Pro, $q^{\mathsf{G}} = 45.1\%$). Because training and evaluation expressions share operator semantics and tree topology, this is best read as an upper bound on what the channel admits under matched semantics, not a generalization claim.

Structural fine-tuning lifts all generators, weakest most. Assembly FT raises generation quality for every open-weight model, by $+3.3$ to $+16.6$ pp, with the largest gains on the weakest generators (Gemma-3-12B $+12.8$, Gemma-3-27B $+16.6$) and smaller but consistently positive gains on the strongest (Qwen3-8B $+12.1$, Phi-4 $+5.8$). Extraction quality improves uniformly, $+0.7$ to $+11.5$ pp, with the strongest extractors gaining most. Self-communication moves with both, reaching 59.3% for Phi-4 and 41.6% for Qwen3-32B. Guard pass rate falls on average from 85.5% (base) to 83.4%. The drop is concentrated in Gemma-3-4B (-22.9 pp) and Gemma-3-12B (-17.5), while Phi-4 and the larger Qwen3 models are essentially unchanged (Appendix O).

Cross-domain gains are real but a gap to the frontier persists. Because the 10 base extractors never see the FT data, assembly-FT gains reflect more decodable problems rather than a shared artifact. Even so, the strongest assembly-FT generator (Phi-4, 32.7%) stays 12.4 pp below untrained Gemini-3.1-Pro (45.1%). Arithmetic FT closes that gap: every open-weight model exceeds Gemini-3.1-Pro, but only by sharing operator semantics and tree topology with evaluation, so it is best read as an upper bound.

5 Related Work

Round-trip and consistency evaluation. Round-trip protocols are established quality proxies, going back to back-translation [Sennrich et al., 2016]. Closest to us, Allamanis et al. [2024] formalize round-trip correctness for code LLMs, using the *same* model on both ends to avoid a "communication chasm". Maveli et al. [2026] round-trip code through compression–decompression bijections, finding persistent gaps across prompting, fine-tuning, and self-reflection, and Kong et al. [2025] extend the framing as a training reward for molecule–text LLMs. Both use a same-model round-trip. Wigler et al. [2026] use a cross-model NL round-trip like ours to recover personality profiles from life stories, but transmit a trait vector at fixed complexity. We study Allamanis' chasm directly for compositional content: an $N \times N$ generator–extractor matrix, symbolic equivalence as the oracle, and controlled complexity, with failures attributed via row and column marginals. Unlike self-consistency [Wang et al., 2023], which aggregates reasoning paths from one model, we test whether

a second reader recovers the sender's structure through the NL channel, complementing work on internal faithfulness [Lanham et al., 2023, Turpin et al., 2023] with an externally verifiable signal.

Multi-agent LLM systems. MetaGPT [Hong et al., 2024], CAMEL [Li et al., 2023], AutoGen [Wu et al., 2024], and Mixture-of-Agents [Wang et al., 2025] route coordination through NL but do not measure information loss at that interface, even when debate improves factuality [Du et al., 2024]. Zhou et al. [2025] argue NL is fundamentally misaligned with LLM representations. He et al. [2026] formalize this as an information bottleneck, and Tran and Kiela [2026] show that multi-agent decomposition inherently loses information through the NL channel. Our communication matrix provides direct empirical support, and the 60.4 pp role-swap effect has immediate implications for pipeline design. Cross-model [Yin et al., 2023] and heterogeneous-LLM [Ye et al., 2025b] pipelines improve over single-model baselines but do not measure text-interface loss, while LatentMAS [Zou et al., 2025] sidesteps it by collaborating in a shared latent space.

Structured channels for LLMs. Early frameworks bridged LLMs with structured interfaces via SQL-like operations [Jiang et al., 2023], but recent work identifies a persistent serialization gap: linearizing structured data into prose breaks order invariance and makes reasoning brittle to arbitrary formatting choices [Herbst et al., 2025]. Remediation moves beyond text-wrapping through hypergraph encodings [Huang et al., 2025] and table-native architectures [Li et al., 2025a], yet frontier models still show format-dependent accuracy on structured outputs [Elnashar et al., 2025]. Our NL tax quantifies this loss in a controlled symbolic setting. Recent proposals replace NL with dense vectors [Wu and Wang, 2025] or a layered telecom-style protocol [Li et al., 2025b], and our NL tax is the floor such alternatives would have to beat.

Compositional generalization. SCAN [Lake and Baroni, 2018] and CFQ [Keysers et al., 2020] evaluate compositional *parsing* (text-to-structure). Transformer performance decays with depth [Dziri et al., 2023], shows a "compositionality gap" [Press et al., 2023], and collapses at high complexity [Thomm et al., 2024, Mirzadeh et al., 2024, Shojaee et al., 2025]. The same gradients appear in math reasoning [Stolfo et al., 2023, Opedal et al., 2025], evaluated as solving rather than transmission. We invert the direction to *generation* (structure-to-text), connecting these failures to surface realization [Reiter and Dale, 2000, Zhang et al., 2006] via linearization complexity $\ell(T)$. Recent work uses equation-to-word-problem generation for data augmentation [Chen et al., 2024a], but does not ask whether the text preserves enough structure for a second model to recover the original.

6 Discussion

The bottleneck is compositional serialization, not domain knowledge. Swapping arithmetic for logical operators changes absolute difficulty but preserves rankings and the generator-dominant fault pattern (Appendix R). Fine-tuning on a novel operator vocabulary transfers broadly, lifting every open-weight generator but still leaving a gap to the frontier (Section 4.1). Under matched operator semantics, $\sim$3600 examples lift every open-weight model above untrained Gemini-3.1-Pro, so the limit is trainable rather than architectural. What stays hard is flattening a hierarchical expression into words another model can re-parse, a skill chain-of-thought and multi-agent pipelines typically rely on.

Practical recommendations. Role assignment matters: swapping generator and extractor shifts accuracy by up to 60.4 pp, and the best pair (92.9%) mixes two different models, so match models to roles by their per-role rank rather than overall strength. Prefer structured intermediates when the consumer can parse them [Gao et al., 2023, Chen et al., 2023] or when an interoperability standard such as MCP [Anthropic, 2025c] or A2A [Google LLC and Linux Foundation Contributors, 2025] already provides the schema, the NL tax reaches 18–20 pp at 14B–32B (Figure 3-right).

Limitations. The protocol covers arithmetic with 2–8 binary operators and a logic replication (Appendix R) which both admit a canonical form, so any two correct answers are provably equivalent. Our protocol requires this kind of oracle, so it does not directly transfer to domains where correctness admits many answers with no canonical form to check against, only validators specific to each instance, like planning, spatial reasoning, and program synthesis. Evaluation is monolingual (English) with one prompt pair and no tuning, so scores are a reproducible lower bound. Thinking modes are disabled to isolate serialization from test-time reasoning and keep the panel comparable.

7 Conclusion

We introduced a round-trip protocol that measures how faithfully language models transmit compositional structure through natural language. What fails is predictable: operator count and tree shape dominate, while model family does not. A logic replication keeps most of the ranking structure, so the failure pattern is not unique to arithmetic even though absolute difficulty changes. What fails is also partially fixable: arithmetic fine-tuning saturates the channel for every tested model, and structural gains transfer to non-fine-tuned extractors. Natural language is a lossy channel for compositional structure, one whose capacity scales with model capability but remains structurally bounded. As language models increasingly communicate with each other, understanding the capacity and failure modes of this channel becomes a first-order concern for system design.

References

M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

M. Allamanis, S. Panthaplackel, and P. Yin. Unsupervised evaluation of code LLMs with round-trip correctness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 1050–1066. PMLR, 2024. URL https://proceedings.mlr.press/v235/allamanis24a.html.

Anthropic. Claude 3.7 sonnet and claude code, February 2025a. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Anthropic. Claude haiku 4.5, 2025b. URL https://www.anthropic.com/claude/haiku.

Anthropic. Model Context Protocol Specification. https://modelcontextprotocol.io, 2025c. Accessed: 2026-07-26.

Anthropic. Introducing claude opus 4.6. https://www.anthropic.com/news/claude-opus-4-6, 2026a.

Anthropic. Introducing claude opus 4.7. https://www.anthropic.com/news/claude-opus-4-7, 2026b.

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

N. Chen, N. Wu, J. Chang, and J. Li. ControlMath: Controllable data generation promotes math generalist models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024a. URL https://arxiv.org/abs/2409.15376.

W. Chen, X. Ma, X. Wang, and W. W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *TMLR*, 2023.

W. Chen, C. Yuan, J. Yuan, Y. Su, C. Qian, C. Yang, R. Xie, Z. Liu, and M. Sun. Beyond natural language: Llms leveraging alternative formats for enhanced reasoning and communication. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10626–10641, 2024b.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, S. Welleck, P. West, C. Bhagavatula, R. Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

A. Elnashar, J. White, and D. C. Schmidt. Enhancing structured data generation with gpt-4o evaluating prompt efficiency across prompt styles. *Frontiers in Artificial Intelligence*, Volume 8 - 2025, 2025. ISSN 2624-8212. doi: 10.3389/frai.2025.1558938. URL https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1558938.

L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pages 10764–10799. PMLR, 2023.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

E. Glazer, E. Erdil, T. Besiroglu, D. Chicharro, E. Chen, A. Gunning, C. F. Olsson, J.-S. Denain, A. Ho, E. de Oliveira Santos, O. Järviniemi, M. Barnett, R. Sandler, M. Vrzala, J. Sevilla, Q. Ren, E. Pratt, L. Levine, G. Barkley, N. Stewart, B. Grechuk, T. Grechuk, S. V. Enugandla, and M. Wildon. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai, 2025. URL https://arxiv.org/abs/2411.04872.

Google DeepMind. Gemini 3 flash model card, 2025a. URL https://deepmind.google/models/model-cards/.

Google DeepMind. Gemini 3 pro model card, 2025b. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

Google LLC and Linux Foundation Contributors. Agent2Agent (A2A) Protocol Specification. https://github.com/google-a2a/A2A, 2025. Apache 2.0 License. Accessed: 2026-07-03.

S. He, A. Narayan, I. S. Khare, S. Linderman, C. Ré, and D. Biderman. An information theoretic perspective on agentic system design. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2512.21720.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2009.03300.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

D. Herbst, L. Karbevska, D. Kumar, A. Ahuja, F. G. Nasrabadi, and F. Frasca. Lost in serialization: Invariance and generalization of llm graph reasoners. *AAAI Workshop on Graphs and more Complex Structures For Learning and Reasoning (GCLR)*, 2025. URL https://arxiv.org/abs/2511.10234.

S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

S. Huang, H. Li, Y. Gu, X. Hu, Q. Li, and G. Xu. Hyperg: Hypergraph-enhanced llms for structured knowledge. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1218–1228, 2025.

N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, 2023.

C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

D. Keysers, N. Schärli, N. Scales, H. Buisman, D. Furrer, S. Kashubin, N. Momchev, D. Sinopalnikov, L. Stafiniak, T. Tihon, D. Tsarkov, X. Wang, M. van Zee, and O. Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2020.

L. Kong, X. Wang, Y. Chen, and M. Zhang. Round-trip reinforcement learning: Self-consistent training for better chemical LLMs. *arXiv preprint arXiv:2510.01527*, 2025. URL https://arxiv.org/abs/2510.01527.

B. Lake and M. Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

T. Lanham, A. Chen, A. Radhakrishnan, B. Steiner, C. Denison, D. Hernandez, D. Li, E. Durmus, E. Hubinger, J. Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

L. Li, C. Ye, W. Ye, Y. Sun, Z. Jiang, H. Wang, J. Tian, Y. Zhang, N. Wang, X. Fu, et al. Table as a modality for large language models. *Advances in Neural Information Processing Systems*, 2025a.

X. Li, M. Liu, and C. Yuen. LLM agent communication protocol (LACP) requires urgent standardization: A telecom-inspired protocol is necessary. *arXiv preprint arXiv:2510.13821*, 2025b. URL https://arxiv.org/abs/2510.13821. NeurIPS 2025 AI4NextG Workshop.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. Explainable ai for trees: From local explanations to global understanding. *arXiv preprint arXiv:1905.04610*, 2019.

N. Maveli, A. Vergari, and S. B. Cohen. Can LLMs compress (and decompress)? evaluating code understanding and execution via invertibility. *arXiv preprint arXiv:2601.13398*, 2026. URL https://arxiv.org/abs/2601.13398.

A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, Jan. 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

S. Nie et al. LLaDA: Large language diffusion models. *39th Conference on Neural Information Processing Systems*, 2025.

A. Opedal, H. Shirakami, B. Schölkopf, A. Saparov, and M. Sachan. Mathgap: Out-of-distribution evaluation on problems with arbitrarily complex proofs. *International Conference on Learning Representations*, 2025.

D. Paperno. On learning interpreted languages with recurrent models. *Computational Linguistics*, 48 (2):471–482, 2022. URL https://aclanthology.org/2022.cl-2.7/.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

L. Phan, A. Gatti, N. Li, A. Khoja, R. Kim, R. Ren, J. Hausenloy, O. Zhang, M. Mazeika, D. Hendrycks, et al. A benchmark of expert-level academic questions to assess ai capabilities. *Nature*, 649(8099):1139–1146, 2026.

O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

G. Rasch. Probabilistic models for some intelligence and attainment tests. *The SAGE Encyclopedia of Research Design*, 1980. URL https://api.semanticscholar.org/CorpusID:61203382.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, 2000.

T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 86–96, 2016.

R. Sethi and J. D. Ullman. The generation of optimal code for arithmetic expressions. *J. ACM*, 17: 715–728, 1970. URL https://api.semanticscholar.org/CorpusID:10845848.

F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03057.

P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.06941.

A. Singh, A. Fry, A. Perelman, A. Tart, A. Ganesh, A. El-Kishky, A. McLaughlin, A. Low, A. Ostrow, A. Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. URL https://arxiv.org/abs/2601.03267.

A. Stolfo, Z. Jin, K. Shridhar, B. Schölkopf, and M. Sachan. A causal framework to quantify the robustness of mathematical reasoning with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 545–561, 2023.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023. URL https://arxiv.org/abs/2210.09261.

J. Thomm, G. Camposampiero, A. Terzic, M. Hersche, B. Schölkopf, and A. Rahimi. Limits of transformer language models on learning to compose algorithms. *Advances in Neural Information Processing Systems*, 37:7631–7674, 2024.

D. Tran and D. Kiela. Single-agent LLMs outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*, 2026. URL https://arxiv.org/abs/2604.02460.

M. Turpin, J. Michael, E. Perez, and S. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou. Mixture-of-agents enhances large language model capabilities. *International Conference on Learning Representations*, 2025.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.01574.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

B. Wigler, M. Tsfasman, and T. M. Hrkalovic. Stories of your life as others: A round-trip evaluation of llm-generated life stories conditioned on rich psychometric profiles. *arXiv preprint arXiv:2604.06071*, 2026.

Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

S. Wu and Y. Wang. Dense communication between language models. *arXiv preprint arXiv:2505.12741*, 2025. URL https://arxiv.org/abs/2505.12741.

A. Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

J. Ye, Z. Xie, L. Zheng, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025a.

R. Ye, X. Liu, Q. Wu, X. Pang, Z. Yin, L. Bai, and S. Chen. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*, 2025b. URL https://arxiv.org/abs/2505.16997.

Z. Yin, Q. Sun, C. Chang, Q. Guo, J. Dai, X. Huang, and X. Qiu. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://arxiv.org/abs/2312.01823.

H. Zhang, L. Huang, D. Gildea, and K. Knight. Synchronous binarization for machine translation. In R. C. Moore, J. Bilmes, J. Chu-Carroll, and M. Sanderson, editors, *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York City, USA, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/N06-1033/.

H. Zhang, J. Da, D. Lee, V. Robinson, C. Wu, W. Song, T. Zhao, P. Raja, C. Zhuang, D. Slack, et al. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836, 2024.

P. Zhou, Y. Feng, H. Julaiti, and Z. Yang. Why do ai agents communicate in human language? *arXiv preprint arXiv:2506.02739*, 2025.

J. Zou, X. Yang, R. Qiu, G. Li, K. Tieu, P. Lu, K. Shen, H. Tong, Y. Choi, J. He, J. Zou, M. Wang, and L. Yang. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025. URL https://arxiv.org/abs/2511.20639.

A Prompt Templates

Generator system prompt. The following system prompt is used for the generation phase, with `{domain}` and `{variables}` filled per expression:

```
Convert a mathematical expression into a short word problem.

Rules:
1. Set the word problem in the domain of: {domain}.
2. Use EXACTLY these variable placeholders in curly braces: {variables}.
3. If the expression contains integer constants, use them as literal
   numbers in the word problem (e.g. the constant 8 becomes "8" in
   the text).
4. Each operation in the expression must correspond to a clear action
   in the story.
5. Do NOT include any mathematical symbols (+, -, *, /, parentheses)
   in your response.
6. Keep it to 2-3 sentences.
7. Output ONLY the word problem - no explanation, no title.
```

Generator in-context exemplars. When the generator runs at γ=3, three exemplars are appended to its prompt to form π^3_{G}. The exemplars are drawn from a fixed stratified pool (five entries spanning $k \in \{2, 3, 4, 6, 8\}$ and $d \in \{2, 3, 4, 5\}$, disjoint from the evaluation set); the first three entries are used at γ=3:

```
Expression: ( A + B ) * C  [domain: baking]
Word problem: A bakery made {A} loaves in the morning and {B} in the
afternoon. Each loaf was packed into a box with {C} rolls. How many
rolls were packed in total?

Expression: ( C - ( A + B + 2 ) )  [domain: real estate]
Word problem: A developer had {C} plots of land. They sold {A} plots
in the spring, {B} plots in the summer, and set aside 2 for a park.
How many plots remained?

Expression: ( ( E + D ) - A ) - ( A / C )  [domain: mountaineering]
Word problem: A team of climbers reached a peak with {E} supplies and
found {D} more along the way. They used {A} for shelter and rationed
{A} units of fuel over {C} days, consuming one day's ration. How many
supplies remained?
```

Extractor system prompt. The following system prompt is used for the extraction phase:

```
You are given a word problem where some numbers are replaced by
variable placeholders and others appear as literal integers.
Identify the mathematical expression that represents the computation
described.

Rules:
1. Use ONLY the variable names listed and any integer constants from
   the text.
2. Use parentheses to make the order of operations explicit.
3. Use plain ASCII operators: +, -, *, /. Do NOT use LaTeX, \frac,
   \times, or any markup.
4. Output ONLY the expression - no explanation, no prose.
```

Extractor in-context exemplars. When the extractor runs at ε=3, three exemplars are appended to form π^3_{E}, drawn from a stratified pool paired with the generator exemplars above but reformatted as `Variables / Problem / Expression` triples:

```
Variables: A, B, C
Problem: A bakery made {A} loaves in the morning and {B} in the
afternoon. Each loaf was packed into a box with {C} rolls. How many
rolls were packed in total?
```

```
Expression: ( A + B ) * C

Variables: A, B, C
Problem: A developer had {C} plots of land. They sold {A} plots
in the spring, {B} plots in the summer, and set aside 2 for a park.
How many plots remained?
Expression: C - ( A + B + 2 )

Variables: A, C, D, E
Problem: A team of climbers reached a peak with {E} supplies and
found {D} more along the way. They used {A} for shelter and rationed
{A} units of fuel over {C} days, consuming one day's ration. How many
supplies remained?
Expression: ( ( E + D ) - A ) - ( A / C )
```

The main matrix evaluates every pairing at $(\gamma, \varepsilon) = (0, 3)$; see Appendix P for the full ablation over $(\gamma, \varepsilon) \in \{0, 3\}^2$.

B Expression Suite Details

Table 3 shows the number of expressions generated for each feasible combination of operator count k and depth d. Each cell is populated by enumerating all skeletons for that (k, d) pair (up to 50) and sampling 5 instantiations per skeleton, following the skeleton enumeration procedure described in Section 2.1. Not all (k, d) pairs are feasible: k operators require at least $\lceil \log_2(k+1) \rceil$ levels of nesting, so shallow depths cannot accommodate high operator counts. All 2450 expressions are used in the main evaluation; each column of the communication matrix is evaluated on the subset for which that generator passed the leakage guard.

Table 3: Number of expressions per operator count k and depth d. Cells marked "—" correspond to infeasible or empty (k, d) pairs.

	d=2	d=3	d=4	d=5	d=6	Total
k=2	10	—	—	—	—	10
k=3	5	20	—	—	—	25
k=4	—	30	40	—	—	70
k=5	—	30	100	80	—	210
k=6	—	20	200	250	160	630
k=7	—	5	250	250	250	755
k=8	—	—	250	250	250	750
Total	15	105	840	830	660	2450

C Leakage Guard and Generator Faults

The *leakage guard* is a rule-based filter that rejects any generated word problem in which mathematical syntax has leaked into the prose. Two additional checks catch further generator faults (missing placeholders and degenerate repetition); these are counted as generator errors in the reported accuracies but are not reported as leakage.

1. Leakage check: operator adjacency rejection. Four regular-expression patterns reject any word problem in which a mathematical operator appears adjacent to a variable letter or digit:

- `[A-Z0-9]\s*[+*/]` (variable or digit followed by operator)
- `[+*/]\s*[A-Z0-9]` (operator followed by variable or digit)
- `\(\s*[A-Z]` (open parenthesis followed by variable)
- `[A-Z]\s*\)` (variable followed by close parenthesis)

Before pattern matching, all placeholder tokens `{X}` are replaced with the letter `X`, so the check operates on the surface form that a human reader would see. These patterns catch expressions like "A + B" or "(A * C)" that leak the formal structure into the word problem.

2. Placeholder completeness (generator fault). Every variable in the original expression must appear as a curly-brace placeholder (e.g. `{A}`, `{B}`) in the generated text. A missing placeholder means the model dropped a variable, making faithful extraction impossible. This is a generator fault rather than a leakage failure.

3. Repetition loop detection (generator fault). The guard checks for degenerate outputs containing three or more consecutive repetitions of any word-level n-gram (for $n \in \{2, 3, 4, 5\}$). Such repetition indicates the model has entered a generation loop rather than producing a coherent word problem. This is also a generator fault rather than a leakage failure.

A word problem must pass all three checks to enter the extraction phase. The $\mathrm{Leak}\ \%$ rows in the communication matrices report pass rates on the leakage check only; the other two fault types are counted as errors in the reported accuracies but are not shown as leakage errors.

A tunable guard. The guard is a deliberately simple, rule-based filter that downstream users can tighten or relax for their own setting. Practitioners building round-trip evaluations on new domains can reuse the three checks directly, swap in stricter syntactic patterns (e.g., additional operator glyphs or domain-specific markers), or replace the rule-based check with a learned classifier. The bias-direction argument below holds under any stricter replacement that still counts guard rejections as round-trip failures.

Known regex gaps. Two specifications in the leakage check are deliberately permissive and admit narrow leak forms that a stricter filter would catch. The operator class `[+*/]` omits the minus sign because `-` also serves as an English hyphen, and including it would reject compound adjectives and hyphenated numerals in otherwise leak-free prose. Leak strings of the form "`A - B`" or "`5 - 3`" therefore pass. The parenthesis patterns match only letter-adjacent parentheses, so numeric expressions such as "`(5 - 3)`" also pass. Both are instances of guard false negatives: they admit word problems into the extraction phase that a stricter filter would reject. The bias-direction argument below covers this case. A stricter guard (adding `-` contextually, and digit-adjacent parentheses) would catch more leaks at the cost of some false positives on natural prose. Because guard rejections remain charged as round-trip failures, the lower-bound direction on the NL tax is preserved under any such tightening.

Bias direction under guard false negatives. The leakage guard is conservative with respect to the paper's central claims. If the guard admits a word problem that contains implicit structural hints (a guard false negative), extractors have an easier time recovering the expression, inflating round-trip accuracy. Because guard rejections are already charged as round-trip failures in our protocol (Section 2.3), a stricter guard can only move additional word problems into the failure bucket. Consequently, the NL tax (Section 3) and the generator-fault rate (Section 3.3) are lower bounds on their true values, and the direction of bias strengthens rather than undermines the conclusion that natural language is a lossy channel for compositional structure. The reported role-swap asymmetry is a difference of two accuracies and its sign is not pinned down by this argument, so we do not claim it as a lower bound.

D R_{math} Definition and Lexicon

R_{math} measures the density of mathematical vocabulary in a generated word problem. Given a word problem $w = (w_1, w_2, \ldots, w_n)$ and a curated lexicon $\mathcal{L}$, we define

$$R_{\text{math}}(w) \;=\; \frac{\sum_{i=1}^{n} \mathbb{1}[w_i \in \mathcal{L}]}{n}, \tag{3}$$

where each w_i is lowercased and stripped of punctuation before lookup. Higher R_{math} indicates the model relied more heavily on arithmetic-signalling vocabulary (e.g. "sum," "divided," "remaining") rather than domain-specific narrative.

The lexicon $\mathcal{L}$ contains 109 terms curated from 200 sampled word problems from the full expression set, organised into seven semantic categories. The complete lexicon is listed below.

Addition (11): *added, adding, adds, combine, combined, plus, sum, total, altogether, collected, gathered.*

Subtraction (28): *subtracted, subtracting, subtracts, minus, remaining, remain, remained, remains, left, leftover, removed, removing, remove, deducted, deducting, lost, loses, fewer, less, spent, spend, spending, spends, reduced, reducing, discarded, unused, withdrawn.*

Multiplication (9): *multiplied, multiplies, times, doubled, tripled, twice, per, each, every.*

Division (20): *divided, divides, dividing, division, split, shared, share, distributed, distributing, distribution, allocated, evenly, equally, equal, among, portion, portions, fraction, half, halved.*

Result / query (19): *calculate, calculating, determine, determined, find, finds, estimate, estimated, how, many, much, what, number, amount, result, average, count, counted, counts.*

Change (18): *increase, increased, decrease, decreased, additional, extra, more, gain, gained, gains, earn, earned, earnings, save, saved, cost, costing, costs.*

Scale (4): *scaled, scaling, factor, rate.*

E Temperature Sensitivity

All main experiments use greedy decoding (temperature $\tau = 0$). To verify that this choice does not critically bias results, we sweep $\tau \in \{0.0, 0.1, \ldots, 1.0\}$ on Phi-4 with 5 independent seeds per temperature point.

Figure 5 summarises the findings. Guard pass rate (left panel) declines gradually from 89.6% ($\tau = 0$) to 70.5% ($\tau = 1$), confirming that the generation prompt remains usable across the sampling range. Round-trip accuracy (centre panel) is relatively stable for $\tau \leq 0.4$ ($35.7\% \to 30.7\%$, -5 pp), then drops more steeply to 13.9% at $\tau = 1.0$. The per-depth breakdown (right panel) reveals that deeper expressions are disproportionately affected: depth-5 accuracy falls from 34.6% to 12.7% (-21.9 pp), whereas depth-2 accuracy stays in a narrow range of 53–63% across the entire sweep. Across all temperature points the inter-seed standard deviation stays below 1.5 pp for the overall round-trip metric, indicating low run-to-run variance.

These results support the use of greedy decoding in the main experiments: $\tau = 0$ maximises round-trip accuracy while eliminating seed variance, and the relative model rankings are unlikely to change at moderate temperatures given the flat region below $\tau \approx 0.4$.

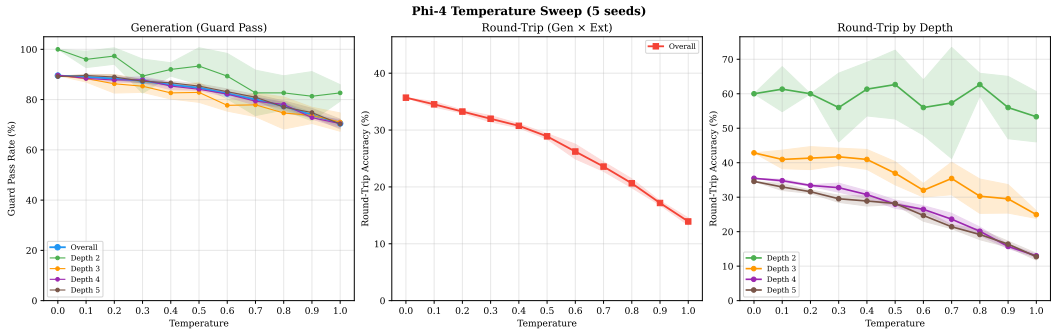

Figure 5: Phi-4 temperature sweep (5 seeds per point). Left: guard pass rate. Centre: overall round-trip accuracy. Right: round-trip accuracy by expression depth. Shaded bands show ± 1 standard deviation across seeds.

F Per-Depth Communication Matrices

Table 1 in the main text aggregates across all depths. Tables 4 to 8 show the communication matrix separately for each depth level, revealing how pairwise accuracy degrades with increasing nesting.

Table 4: Communication matrix at depth $d = 2$ (15 expressions). Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. Dotted lines separate open-weight AR, diffusion, and frontier models.

EXTRACTORS \ GENERATORS	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
Leak pass %	33	73	93	100	100	100	100	100	100	100	53	100	100	100	100	100	
Qwen3-0.6B	0.0	6.7	33.3	53.3	60.0	26.7	0.0	6.7	6.7	33.3	6.7	26.7	26.7	40.0	26.7	33.3	24.2±18
Qwen3-1.7B	0.0	20.0	20.0	60.0	33.3	20.0	6.7	20.0	20.0	53.3	0.0	13.3	46.7	60.0	46.7	60.0	30.0±21
Qwen3-4B	0.0	26.7	33.3	66.7	73.3	53.3	6.7	20.0	33.3	46.7	13.3	40.0	66.7	66.7	73.3	93.3	44.6±26
Qwen3-8B	6.7	20.0	33.3	53.3	80.0	60.0	6.7	26.7	26.7	60.0	20.0	46.7	80.0	80.0	86.7	93.3	48.8±29
Qwen3-14B	6.7	26.7	33.3	60.0	73.3	60.0	13.3	20.0	33.3	60.0	20.0	40.0	80.0	80.0	93.3	100.0	50.0±29
Qwen3-32B	6.7	26.7	33.3	60.0	73.3	53.3	13.3	20.0	33.3	60.0	26.7	46.7	86.7	80.0	93.3	100.0	50.8±29
Gemma-3-4B	6.7	20.0	20.0	53.3	60.0	46.7	6.7	33.3	26.7	40.0	20.0	20.0	53.3	66.7	73.3	93.3	40.0±24
Gemma-3-12B	0.0	33.3	33.3	60.0	73.3	60.0	13.3	26.7	26.7	53.3	13.3	46.7	80.0	66.7	100.0	93.3	48.7±29
Gemma-3-27B	6.7	26.7	33.3	60.0	73.3	53.3	6.7	26.7	33.3	60.0	13.3	46.7	80.0	93.3	86.7	100.0	50.0±30
Phi-4	0.0	26.7	26.7	46.7	73.3	60.0	6.7	20.0	33.3	60.0	6.7	46.7	80.0	80.0	100.0	100.0	47.9±32
Dream-7B	0.0	20.0	13.3	66.7	53.3	33.3	6.7	20.0	26.7	60.0	0.0	26.7	53.3	80.0	66.7	66.7	37.1±26
LLaDA-8B	0.0	20.0	26.7	60.0	86.7	60.0	0.0	33.3	40.0	60.0	26.7	53.3	73.3	93.3	86.7	93.3	50.8±30
C-Haiku-4.5	0.0	20.0	46.7	53.3	80.0	53.3	13.3	26.7	33.3	60.0	26.7	46.7	86.7	100.0	86.7	100.0	52.1±30
GPT-5	6.7	26.7	26.7	53.3	80.0	60.0	6.7	26.7	46.7	60.0	13.3	53.3	73.3	100.0	93.3	100.0	51.7±31
G-3-Flash	6.7	20.0	40.0	53.3	80.0	60.0	6.7	26.7	40.0	60.0	26.7	46.7	86.7	100.0	86.7	100.0	52.5±30
G-3.1-Pro	0.0	13.3	40.0	60.0	73.3	66.7	6.7	26.7	33.3	60.0	26.7	46.7	73.3	100.0	100.0	100.0	51.7±32
Gen. quality	2.9±3	22.1±6	30.8±8	57.5±5	70.4±13	51.7±13	7.5±4	23.8±6	30.8±9	55.4±8	16.2±9	40.4±12	70.4±16	80.4±17	81.2±20	89.2±19	

Table 5: Communication matrix at depth $d = 3$ (105 expressions). Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. Dotted lines separate open-weight AR, diffusion, and frontier models.

EXTRACTORS \ GENERATORS	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
Leak pass %	12	70	99	99	100	99	100	99	97	95	65	98	99	100	100	100	
Qwen3-0.6B	1.0	6.7	13.3	19.0	13.3	13.3	0.0	2.9	1.0	15.2	4.8	16.2	19.0	7.6	12.4	7.6	9.6±6
Qwen3-1.7B	1.0	9.5	21.0	25.7	31.4	21.9	3.8	12.4	7.6	29.5	2.9	32.4	46.7	35.2	31.4	36.2	21.8±14
Qwen3-4B	1.0	14.3	26.7	30.5	40.0	40.0	4.8	15.2	13.3	41.9	8.6	40.0	66.7	72.4	56.2	69.5	33.8±23
Qwen3-8B	1.0	13.3	28.6	33.3	39.0	41.9	5.7	13.3	11.4	40.0	10.5	43.8	74.3	73.3	61.9	81.0	35.8±25
Qwen3-14B	1.0	9.5	30.5	35.2	39.0	43.8	2.9	13.3	16.2	42.9	11.4	45.7	77.1	87.6	73.3	87.6	38.6±29
Qwen3-32B	1.0	13.3	31.4	37.1	39.0	48.6	6.7	13.3	13.3	47.6	11.4	42.9	77.1	89.5	73.3	93.3	39.9±29
Gemma-3-4B	1.0	10.5	26.7	27.6	30.5	28.6	5.7	12.4	8.6	39.0	8.6	35.2	54.3	47.6	36.2	48.6	26.3±16
Gemma-3-12B	1.0	12.4	27.6	30.5	38.1	36.2	6.7	13.3	10.5	41.0	8.6	45.7	71.4	64.8	56.2	69.5	33.3±23
Gemma-3-27B	1.0	11.4	31.4	36.2	40.0	42.9	6.7	13.3	16.2	43.8	9.5	42.9	77.1	84.8	67.6	84.8	38.1±27
Phi-4	1.0	13.3	31.4	33.3	39.0	42.9	5.7	14.3	14.3	43.8	10.5	49.5	75.2	85.7	66.7	94.3	38.8±28
Dream-7B	1.0	9.5	21.0	21.9	21.9	21.9	3.8	10.5	5.7	26.7	4.8	31.4	38.1	35.2	42.9	49.5	21.6±14
LLaDA-8B	1.0	11.4	27.6	28.6	23.8	33.3	1.9	14.3	7.6	33.3	11.4	39.0	61.0	69.5	60.0	52.4	29.8±21
C-Haiku-4.5	1.0	14.3	32.4	37.1	40.0	47.6	7.6	12.4	16.2	43.8	11.4	45.7	76.2	95.2	75.2	91.4	40.5±29
GPT-5	1.0	14.3	28.6	37.1	39.0	41.0	5.7	16.2	10.5	42.9	11.4	46.7	74.3	95.2	73.3	94.3	39.5±30
G-3-Flash	1.0	14.3	29.5	39.0	39.0	42.9	7.6	13.3	10.5	45.7	10.5	48.6	70.5	95.2	70.5	97.1	39.7±30
G-3.1-Pro	1.0	14.3	28.6	35.2	35.2	33.3	6.7	17.1	12.4	47.6	12.4	50.5	74.3	100.0	77.1	98.1	40.2±31
Gen. quality	1.0±0	12.0±2	27.3±5	31.7±6	34.3±8	36.2±10	5.1±2	13.0±3	11.0±4	39.0±8	9.3±3	41.0±8	64.6±16	71.2±26	58.4±18	72.2±26	

Table 6: Communication matrix at depth $d = 4$ (840 expressions). Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. Dotted lines separate open-weight AR, diffusion, and frontier models.

Extractors \ Generators	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
Leak pass %	19	57	94	98	99	98	98	96	95	96	64	95	100	100	100	100	
Qwen3-0.6B	0.0	1.5	4.6	5.5	4.2	2.1	0.1	0.5	0.4	5.7	0.8	5.6	8.8	2.4	2.5	2.0	$2.9_{\pm 2}$
Qwen3-1.7B	0.0	1.5	9.8	13.8	12.6	8.6	0.4	2.0	1.0	16.7	1.4	11.2	27.6	13.5	14.9	16.1	$9.4_{\pm 8}$
Qwen3-4B	0.0	2.1	15.6	26.0	23.6	20.5	0.6	3.5	2.7	29.0	2.5	18.2	56.4	43.0	39.0	55.8	$21.2_{\pm 19}$
Qwen3-8B	0.0	2.3	15.7	26.0	24.2	22.3	0.7	3.5	3.8	32.3	2.7	18.3	55.5	49.8	45.4	64.9	$22.9_{\pm 21}$
Qwen3-14B	0.0	2.9	18.5	29.6	28.6	25.4	0.6	3.8	4.2	35.7	3.0	22.5	63.6	65.0	54.6	78.0	$27.2_{\pm 25}$
Qwen3-32B	0.0	3.0	18.3	29.2	30.0	26.1	0.7	4.6	4.0	34.3	3.6	22.9	61.3	71.9	55.8	80.7	$27.9_{\pm 26}$
Gemma-3-4B	0.0	1.9	12.7	17.3	16.1	12.4	0.6	3.3	1.7	21.8	1.5	14.4	34.9	19.2	22.0	23.3	$12.7_{\pm 10}$
Gemma-3-12B	0.0	2.3	15.5	25.0	23.3	20.4	0.5	4.0	2.6	28.2	3.0	18.2	51.4	46.1	38.2	55.1	$20.9_{\pm 18}$
Gemma-3-27B	0.0	2.4	18.2	27.5	27.9	25.2	1.0	4.0	3.8	33.9	3.3	20.2	59.8	63.5	48.1	77.5	$26.0_{\pm 24}$
Phi-4	0.0	2.3	19.2	28.7	29.5	26.3	0.8	4.3	4.8	35.4	3.2	22.0	62.5	71.5	54.6	83.8	$28.1_{\pm 26}$
Dream-7B	0.0	1.3	8.9	14.5	13.5	11.4	0.2	2.0	2.0	15.5	1.2	9.9	25.6	22.3	23.0	41.0	$12.0_{\pm 11}$
LLaDA-8B	0.0	1.9	13.1	21.8	18.1	15.5	0.5	3.3	2.5	24.3	3.0	16.9	41.5	44.2	36.5	46.8	$18.1_{\pm 16}$
C-Haiku-4.5	0.0	2.6	18.1	28.7	29.5	27.4	0.7	4.0	4.2	36.0	3.6	22.1	63.1	81.9	60.6	91.9	$29.7_{\pm 29}$
GPT-5	0.0	2.6	18.6	30.4	30.7	24.4	0.5	4.5	4.2	36.7	3.2	24.5	65.4	91.3	66.8	95.7	$31.2_{\pm 31}$
G-3-Flash	0.0	2.9	18.6	30.1	31.3	26.5	0.7	4.3	3.9	36.5	3.5	22.6	65.6	90.7	63.1	96.3	$31.0_{\pm 31}$
G-3.1-Pro	0.0	2.6	17.9	30.4	30.8	26.4	0.6	4.2	4.2	37.5	3.5	23.0	65.2	93.7	65.8	96.1	$31.4_{\pm 31}$
Gen. quality	$0.0_{\pm 0}$	$2.3_{\pm 0}$	$15.2_{\pm 4}$	$24.0_{\pm 7}$	$23.4_{\pm 8}$	$20.1_{\pm 7}$	$0.6_{\pm 0}$	$3.5_{\pm 1}$	$3.1_{\pm 1}$	$28.7_{\pm 9}$	$2.7_{\pm 1}$	$18.3_{\pm 5}$	$50.5_{\pm 17}$	$54.4_{\pm 28}$	$43.2_{\pm 19}$	$62.8_{\pm 29}$	

Table 7: Communication matrix at depth $d = 5$ (830 expressions). Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. Dotted lines separate open-weight AR, diffusion, and frontier models.

Extractors \ Generators	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
Leak pass %	21	59	93	97	99	99	98	96	97	97	64	95	100	100	100	100	
Qwen3-0.6B	0.0	0.5	1.7	2.9	2.3	1.8	0.1	0.1	0.2	4.3	0.0	3.1	6.6	1.0	0.6	0.7	$1.6_{\pm 2}$
Qwen3-1.7B	0.0	1.3	5.7	10.0	7.7	7.6	0.2	1.6	1.1	15.1	1.0	8.1	25.2	8.9	9.2	12.4	$7.2_{\pm 6}$
Qwen3-4B	0.0	1.7	12.0	18.2	21.3	18.1	0.5	3.5	3.3	27.3	1.7	15.1	47.7	38.2	33.7	45.8	$18.0_{\pm 16}$
Qwen3-8B	0.0	1.6	13.7	17.8	21.9	20.4	0.5	2.9	4.0	28.4	1.1	16.3	50.0	43.1	39.3	51.4	$19.5_{\pm 18}$
Qwen3-14B	0.0	1.7	14.8	21.6	24.2	23.3	0.6	4.7	4.3	32.2	1.8	18.2	58.8	55.3	51.8	63.9	$23.6_{\pm 22}$
Qwen3-32B	0.0	1.7	14.9	21.7	25.1	24.6	0.5	3.5	4.8	31.4	2.2	18.8	58.7	62.2	51.2	70.8	$24.5_{\pm 23}$
Gemma-3-4B	0.0	1.1	8.4	12.5	12.0	10.4	0.2	2.7	1.7	16.7	1.0	10.8	28.4	17.1	16.9	17.2	$9.8_{\pm 8}$
Gemma-3-12B	0.0	1.3	11.7	15.4	17.6	17.2	0.4	2.5	3.4	25.4	1.7	14.6	43.7	36.9	32.4	41.8	$16.6_{\pm 15}$
Gemma-3-27B	0.0	1.8	14.6	19.5	23.4	22.4	0.7	4.1	4.8	30.4	1.8	16.7	54.2	56.1	48.9	63.7	$22.7_{\pm 21}$
Phi-4	0.0	1.7	16.0	22.7	26.5	24.1	0.7	4.7	4.1	34.1	2.2	19.4	59.5	62.8	55.2	72.5	$25.4_{\pm 24}$
Dream-7B	0.0	1.2	6.0	11.8	8.7	10.6	0.4	2.2	1.8	12.7	1.2	8.6	24.2	17.1	18.1	31.8	$9.8_{\pm 9}$
LLaDA-8B	0.0	1.4	11.9	16.3	18.2	17.5	0.5	2.8	2.8	22.8	1.6	14.9	40.6	35.9	32.9	37.3	$16.1_{\pm 14}$
C-Haiku-4.5	0.0	2.0	15.8	21.9	27.0	25.3	0.5	3.6	4.8	34.0	2.5	19.2	61.9	78.7	57.1	86.7	$27.6_{\pm 28}$
GPT-5	0.0	1.9	17.8	23.4	28.7	19.5	0.7	4.8	4.5	36.1	2.7	22.2	66.4	91.2	67.5	94.9	$30.1_{\pm 31}$
G-3-Flash	0.0	2.3	16.1	23.0	26.9	25.7	0.7	3.7	4.8	35.5	2.4	20.8	64.7	89.2	62.3	95.3	$29.6_{\pm 31}$
G-3.1-Pro	0.0	1.8	15.8	22.0	27.6	26.3	0.4	3.9	4.5	36.6	2.4	20.7	63.5	93.3	64.6	96.3	$30.0_{\pm 31}$
Gen. quality	$0.0_{\pm 0}$	$1.6_{\pm 0}$	$12.3_{\pm 4}$	$17.5_{\pm 6}$	$19.9_{\pm 8}$	$18.4_{\pm 7}$	$0.5_{\pm 0}$	$3.2_{\pm 1}$	$3.4_{\pm 1}$	$26.4_{\pm 9}$	$1.7_{\pm 1}$	$15.5_{\pm 5}$	$47.1_{\pm 17}$	$49.2_{\pm 29}$	$40.1_{\pm 20}$	$55.2_{\pm 30}$	

Table 8: Communication matrix at depth $d = 6$ (660 expressions). Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. Dotted lines separate open-weight AR, diffusion, and frontier models.

EXTRACTORS	GENERATORS: Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	G-3-Flash	G-3.1-Pro	Extr. quality
Leak pass %	21	62	93	97	100	99	99	95	95	97	63	95	99	100	100	100	
Qwen3-0.6B	0.0	0.6	1.1	2.0	1.1	0.3	0.2	0.0	0.0	1.7	0.0	0.8	3.5	0.6	0.6	0.6	$0.8_{\pm 1}$
Qwen3-1.7B	0.0	1.2	5.2	6.1	5.3	4.8	0.3	0.5	0.6	7.4	0.3	4.4	17.1	6.8	7.4	7.4	$4.7_{\pm 4}$
Qwen3-4B	0.0	2.6	10.6	13.0	14.1	14.2	0.3	1.1	1.2	18.6	0.8	9.4	38.3	28.9	25.6	28.9	$13.0_{\pm 12}$
Qwen3-8B	0.0	2.4	10.0	11.5	14.1	14.7	0.3	0.9	1.8	17.9	0.6	9.4	34.8	33.2	26.2	32.3	$13.1_{\pm 12}$
Qwen3-14B	0.0	2.9	12.0	14.8	17.9	17.7	0.6	1.7	2.0	24.5	0.8	11.2	42.9	43.8	39.2	45.9	$17.4_{\pm 16}$
Qwen3-32B	0.0	3.0	12.4	14.8	17.9	19.1	0.5	1.7	2.4	24.1	1.2	10.9	46.4	52.3	42.6	51.7	$18.8_{\pm 19}$
Gemma-3-4B	0.0	1.8	6.2	6.7	7.6	5.8	0.2	0.6	0.3	10.3	0.2	5.5	19.5	9.5	9.2	12.1	$6.0_{\pm 5}$
Gemma-3-12B	0.0	1.5	9.8	10.8	12.6	12.1	0.3	0.8	1.8	15.5	0.9	7.7	34.7	24.1	20.8	28.6	$11.4_{\pm 11}$
Gemma-3-27B	0.0	2.6	11.7	13.2	16.5	17.4	0.6	1.2	1.8	20.9	0.8	10.8	44.1	42.1	36.4	46.1	$16.6_{\pm 16}$
Phi-4	0.0	3.0	13.3	15.0	20.2	18.9	0.5	1.5	2.4	25.6	1.1	12.6	48.2	55.9	42.3	55.2	$19.7_{\pm 19}$
Dream-7B	0.0	1.4	5.3	5.8	6.5	6.4	0.2	0.8	0.5	7.6	0.3	4.7	14.1	10.5	11.8	14.5	$5.6_{\pm 5}$
LLaDA-8B	0.0	2.0	9.2	9.5	13.6	10.6	0.3	1.1	1.7	13.6	0.9	8.3	29.2	25.3	20.3	29.4	$10.9_{\pm 10}$
C-Haiku-4.5	0.0	2.7	12.1	15.3	19.5	20.0	0.3	1.7	2.3	25.6	1.1	10.0	48.9	72.9	46.1	63.5	$21.4_{\pm 23}$
GPT-5	0.0	3.8	14.1	17.1	21.1	25.5	0.8	2.1	2.7	28.6	1.2	13.6	60.0	90.8	55.8	70.0	$25.4_{\pm 28}$
G-3-Flash	0.0	3.2	12.7	15.8	19.4	20.6	0.5	1.5	2.6	27.1	1.2	12.1	54.1	86.5	53.2	69.7	$23.8_{\pm 26}$
G-3.1-Pro	0.0	2.3	13.8	16.2	19.2	21.4	0.6	1.7	2.7	28.9	1.2	12.9	57.0	90.3	52.0	68.0	$24.3_{\pm 27}$
Gen. quality	$0.0_{\pm 0}$	$2.3_{\pm 1}$	$10.0_{\pm 4}$	$11.7_{\pm 4}$	$14.2_{\pm 6}$	$14.3_{\pm 7}$	$0.4_{\pm 0}$	$1.2_{\pm 1}$	$1.7_{\pm 1}$	$18.6_{\pm 8}$	$0.8_{\pm 0}$	$9.0_{\pm 3}$	$37.1_{\pm 16}$	$42.1_{\pm 29}$	$30.6_{\pm 17}$	$39.0_{\pm 22}$	

G SHAP Attribution Methodology and Additional Experiments

To quantify which features drive round-trip success, we fit a gradient-boosted decision tree (GBT) classifier to the binary outcome (correct/incorrect) of each *triple* (expression, generator, extractor).

Model. We use scikit-learn's [Pedregosa et al., 2011] `GradientBoostingClassifier` with 200 trees, maximum depth 4, learning rate 0.1, and row subsampling rate 0.8. The classifier is trained on the full set of triples. A held-out 80/20 split yields a sanity-check accuracy of 88.1%, confirming the features carry substantial predictive signal. For context, the majority-class baseline (always predicting failure) achieves 87.8%, so the GBT's improvement is modest in absolute terms, and the value of the classifier lies in the SHAP decomposition rather than raw accuracy. The corresponding AUC-ROC scores are 0.812 for the high-level classifier and 0.822 for the AST classifier, consistent with a model that captures substantial but not saturating signal on the round-trip outcome.

Attribution. We compute exact SHAP values via `TreeExplainer` [Lundberg et al., 2019], which exploits the tree structure to compute Shapley values in polynomial time without sampling. For each feature f, we report *mean absolute SHAP* normalised to a percentage of the total attribution:

$$\text{Attribution}(f) \;=\; \frac{\frac{1}{n}\sum_{t=1}^{n} |\phi_f^{(t)}|}{\sum_{f'} \frac{1}{n}\sum_{t=1}^{n} |\phi_{f'}^{(t)}|} \;\times\; 100\%, \tag{4}$$

where $\phi_f^{(t)}$ is the SHAP value of feature f for triple t and the sum in the denominator runs over all features.

Direction. We additionally report the Pearson correlation between each feature's raw values and its SHAP values across triples. A positive correlation means higher feature values push the prediction toward success; a negative correlation means higher values push toward failure.

G.1 Complete Structural SHAP Attribution

The main text (Section 3.3) reports the high-level SHAP analysis with six summary features and summarizes the key findings of the structural analysis. Table 9 lists all 16 features from the AST-level gradient-boosted classifier, sorted by mean absolute SHAP value.

After controlling for model size (50.8% combined), four structural findings stand out:

- **Division is the hardest operator.** Division ($n_\div$: 6.4%) is the single largest structural predictor, followed by subtraction (n_-: 4.1%) and multiplication ($n_\times$: 3.7%); addition (n_+: 1.4%) matters least. The two non-commutative operators thus rank first and second among the four. Non-commutativity, not multiplicativity, drives operator difficulty: argument order ($a \div b \neq b \div a$, $a - b \neq b - a$) must be conveyed unambiguously in prose, whereas the commutative operators tolerate reordering. Consistent with this, the deepest tree level containing a non-commutative operator (5.5%, $\downarrow$) is the largest structural predictor after depth itself, confirming that ordering ambiguity compounds with nesting.
- **Tree shape matters beyond size and depth.** Skeleton identity (5.5%) is a strong predictor alongside depth (7.4%), indicating that expressions with different branching topologies but identical (k, d) can differ substantially in round-trip accuracy. Center of mass (3.0%, $\downarrow$) reveals the direction of this effect: right-branching trees, which require the word problem to hold an incomplete computation open while describing a nested subexpression, tend to be harder than left-branching ones.
- **Variable reuse introduces ambiguity.** Expressions where a variable appears in multiple leaves (3.5%, $\downarrow$) require coreference in the word problem, forcing the generator to signal that two quantities refer to the same variable. More unique variables (3.0%, $\uparrow$) have the opposite effect, reducing coreference demands and improving round-trip fidelity.
- **Root operator sets the frame.** Addition as root (2.2%, $\downarrow$) and subtraction as root (2.0%, $\downarrow$) are the most penalising root choices, whereas multiplicative roots ($\div$: 0.9%, $\times$: 0.7%, both $\uparrow$) provide more flexibility at the top of the tree.

Table 9: Complete SHAP feature attribution for the structural analysis. All features from the AST-level gradient-boosted classifier are shown, sorted by mean |SHAP| as a percentage of total attribution. Green (↑) indicates higher feature values increase round-trip success; red (↓) indicates higher values decrease success.

Feature	Attribution
Generator size	34.3% ↑
Extractor size	16.5% ↑
Depth	7.4% ↓
$n_\div$	6.4% ↓
Non-comm. depth	5.5% ↓
Skeleton ID	5.5% ↓
n_-	4.1% ↓
$n_\times$	3.7% ↓
Variable reuse	3.5% ↓
Center of mass	3.0% ↓
Unique variables	3.0% ↑
Root = +	2.2% ↓
Root = −	2.0% ↓
n_+	1.4% ↑
Root = ÷	0.9% ↑
Root = ×	0.7% ↑

H Skeleton Difficulty and Center of Mass

The SHAP analysis identifies skeleton identity (5.5%) and center of mass (3.0%) as significant predictors of difficulty. We now examine these effects directly by comparing skeletons that share the same operator count and depth.

Figure 6 plots the round-trip accuracy of each skeleton, grouped by (k, d). The spread within groups is substantial: at $(k{=}4, d{=}4)$, the seven observed skeletons span 22.1 percentage points, from 52.7% for the easiest to 30.6% for the hardest. At $(k{=}2, d{=}2)$, the only two skeletons differ by 1.1 percentage points (left-branching: 60.8% vs. right-branching: 59.7%). These differences are controlled for operator count and depth by construction; they reflect the effect of tree shape alone.

Center of mass as predictor. Points in Figure 6 are coloured by their normalised center of mass (CoM), ranging from blue (left-branching, CoM $= 0$) to red (right-branching, CoM $= 1$). A visual pattern emerges: within each group, bluer points tend to appear to the right (higher accuracy) and redder points to the left (lower accuracy). The within-group weighted correlation between CoM and accuracy is $r = -0.148$ ($p = 0.007$, one-tailed permutation test with 10000 shuffles, $N = 455$ skeletons), indicating that right-branching trees tend to be harder after controlling for (k, d).

Why right-branching is harder. Left-branching trees (e.g. $((A + B) \times C) - D$) correspond to a sequence of operations applied to a running result, which maps naturally onto left-to-right prose: "start with A plus B, multiply by C, subtract D." Right-branching trees (e.g. $A - (B \times (C + D))$) require the generator to introduce a nested subcomputation before the main operation, demanding forward references or subordinate clauses that are harder to produce and parse unambiguously. An analogous right-branching deficit has been reported for LSTM/GRU learners on synthetic interpreted languages [Paperno, 2022], suggesting the bias is not specific to transformer LLMs.

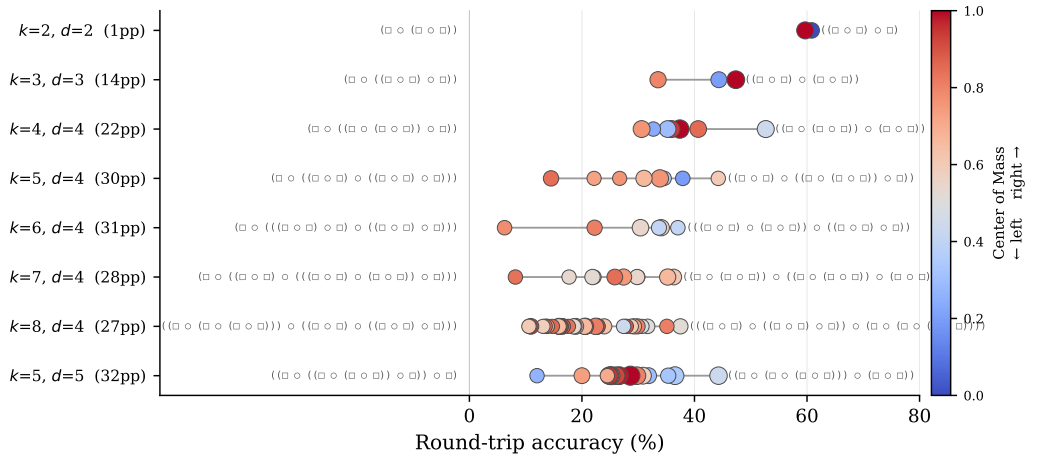

Figure 6: Skeleton difficulty within (k, d) groups. Each point is one skeleton; the x-axis shows its round-trip accuracy averaged across all models and expressions sharing that skeleton. Colour encodes center of mass (blue = left-branching, red = right-branching). The spread in parentheses on each row label shows the gap between the easiest and hardest skeleton in that group. Within-group weighted $r = -0.148$ (CoM vs. accuracy, $p = 0.007$, permutation test).

I Error Taxonomy Details

Table 10 shows the per-extractor error profile across all generators and depths. The classification algorithm processes each incorrect extraction in the following order: (1) attempt to parse $\hat{e}$ into a symbolic expression tree; if parsing fails, label $\mathrm{Err}_{\mathrm{parse}}$; (2) count the operators in both e and $\hat{e}$; if they differ, label $\mathrm{Err}_{\mathrm{arity}}$; (3) compare the unlabelled tree skeletons; if they differ, label $\mathrm{Err}_{\mathrm{struct}}$; (4) otherwise label $\mathrm{Err}_{\mathrm{content}}$ (the skeleton matches but leaf values or operator labels differ).

Table 10: Per-extractor error profile. Each row shows the percentage of that extractor's errors falling into each category, aggregated across all generators and depths. Models are grouped by family and sorted by size.

Extractor	$\mathrm{Err}_{\mathrm{parse}}$	$\mathrm{Err}_{\mathrm{arity}}$	$\mathrm{Err}_{\mathrm{struct}}$	$\mathrm{Err}_{\mathrm{content}}$
Qwen3-0.6B	0.1%	89.1%	10.3%	0.5%
Qwen3-1.7B	0.4%	74.4%	23.4%	1.8%
Qwen3-4B	1.2%	67.8%	29.3%	1.7%
Qwen3-8B	1.6%	66.2%	30.7%	1.6%
Qwen3-14B	1.8%	65.7%	31.3%	1.2%
Qwen3-32B	1.6%	64.7%	32.4%	1.2%
Gemma-3-4B	5.4%	67.9%	25.5%	1.1%
Gemma-3-12B	0.3%	70.6%	27.9%	1.3%
Gemma-3-27B	0.6%	68.0%	29.8%	1.6%
Phi-4	2.3%	67.0%	28.8%	1.9%
Dream-7B	10.5%	69.6%	18.6%	1.3%
LLaDA-8B	2.6%	77.2%	17.6%	2.5%
C-Haiku-4.5	1.9%	67.0%	29.7%	1.5%
GPT-5	2.1%	70.3%	26.2%	1.4%
Gemini-3-Flash	0.6%	73.3%	24.9%	1.2%
Gemini-3.1-Pro	1.4%	71.0%	26.3%	1.2%

Figure 7 breaks the same categories down by expression depth, showing that arity errors grow with depth while content errors shrink.

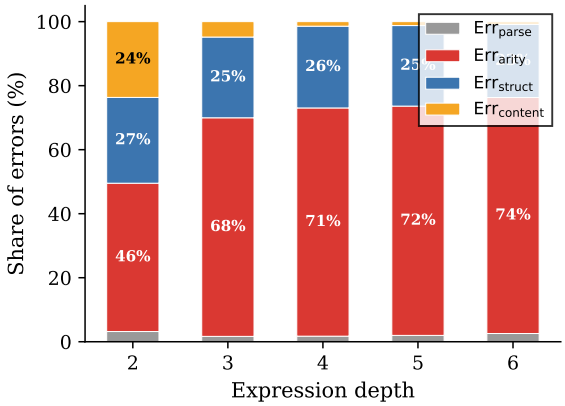

Figure 7: Distribution of error categories by expression depth. $\mathrm{Err}_{\mathrm{arity}}$ (lost or extra operators) dominates at every depth and grows from 46% at depth 2 to 74% at depth 6. $\mathrm{Err}_{\mathrm{content}}$ shrinks correspondingly, indicating that models that preserve the correct operator count and tree shape rarely misidentify leaves. $\mathrm{Err}_{\mathrm{parse}}$ errors are small (2.0% overall).

J Correlation with External Benchmarks

A compact summary appears in Section 3.4; this appendix provides the full per-benchmark breakdown.

We compute Spearman rank correlations between per-model round-trip scores and published results on 9 external benchmarks spanning four categories: math (GSM8K [Cobbe et al., 2021], MATH [Hendrycks et al., 2021b], MGSM [Shi et al., 2023]), code (MBPP [Austin et al., 2021], LiveCodeBench [Jain et al., 2024]), general knowledge (MMLU [Hendrycks et al., 2021a], MMLU-Pro [Wang et al., 2024], BBH [Suzgun et al., 2023]), and reasoning (GPQA [Rein et al., 2023]). Benchmark scores are drawn exclusively from official technical reports and vendor announcements [Yang et al., 2025, Gemma Team, 2025, Abdin et al., 2024, Anthropic, 2025a,b, Singh et al., 2025]; benchmarks with verified scores for fewer than seven models are excluded. For each benchmark we correlate its model rankings with three round-trip scores (Section 2.3): *generation quality* (column-average of the communication matrix), *extraction quality* (row-average), and *self-communication* (diagonal cell). Because frontier vendors did not publish standard-mode scores on most benchmarks, the analysis is dominated by the 10 open-weight models; each benchmark uses pairwise deletion, so the effective n varies (7–10; see Table 13). Table 11 lists the source document, evaluation protocol, and coverage for each model family. Because evaluation protocols differ across families (e.g. base-model few-shot for Qwen3 vs. 0-shot instruct for Gemma 3), the correlations should be read as rough indicators rather than precise estimates.

Table 11: Provenance of external benchmark scores. Each score traces to the listed source; **null** entries are omitted from the correlation analysis via pairwise deletion.

Model family	Source	Eval mode	Benchmarks (of 9)
Qwen3 (0.6–32B)	Yang et al. [2025]	base few-shot[†]	9/9
Gemma 3 IT (4–27B)	Gemma Team [2025]	0-shot instruct	7/9
Phi-4 (14B)	Abdin et al. [2024]	simple-evals	5/9
Claude Haiku 4.5	Anthropic [2025b]	no standard-mode scores published	
GPT-5	Singh et al. [2025]	non-reasoning	1/9

[†]MATH-500, LiveCodeBench, and GPQA-Diamond use instruct non-thinking scores where base-model scores are unavailable.

Table 13 reports the per-benchmark correlations sorted by average ρ across the three round-trip phases. Several patterns emerge:

Extraction quality is consistently well-predicted. For 7 of 9 benchmarks, extraction ρ reaches 0.83 or higher, peaking at MMLU ($\rho = 0.98$) and MGSM ($\rho = 0.96$). The models that extract expressions most accurately are, broadly, those that score highest on standard evaluations.

Generation quality is harder to predict. Generation ρ ranges from 0.47 (MMLU) to 0.89 (MGSM) and fails to reach significance for MMLU, MMLU-Pro, MATH, and GPQA-Diamond individually. The gap between extraction and generation ρ is largest for MMLU (0.98 vs. 0.47) and GPQA-Diamond (0.95 vs. 0.50), confirming that faithfully *generating* a word problem from an expression is a skill not fully predicted by standard evaluations.

Code benchmarks correlate strongly. MBPP and LiveCodeBench both reach $\rho = 0.93$ for extraction and 0.77 for generation, placing code among the best-predicted categories rather than the weakest.

Math benchmarks are the weakest category. MATH shows uniformly moderate correlations ($\rho = 0.52/0.50/0.30$ for generation/extraction/self-communication), and GSM8K follows a similar pattern (0.68/0.72/0.50). MGSM is an outlier (0.89/0.96/0.89, n=7). Standard math-solving ability overlaps only partially with the structural linearization our protocol measures.

Table 12 aggregates these findings at the category level via Fisher-z averaging of per-benchmark ρ values. Extraction correlates strongly across all four categories ($\rho = 0.82$–0.95). Generation is highest for code ($\rho = 0.77$) and math ($\rho = 0.74$) and lowest for reasoning and general knowledge ($\rho = 0.50$–0.55).

Table 12: Fisher-z averaged Spearman ρ between round-trip scores and external benchmarks, grouped by category: **Math**, **Code**, **General**, **Reasoning**.

	Code	**Reasoning**	**General**	**Math**
benchmarks	2	1	3	3
Generation Acc.	0.77	0.50	0.55	0.74
Extraction Acc.	0.93	0.95	0.94	0.82
Self-Consistency	0.77	0.90	0.86	0.64

Table 13: Spearman rank correlation (ρ) between round-trip scores and external benchmarks, sorted by average ρ. Columns are colored by category: **Math**, **Code**, **General**, **Reasoning**.

	MGSM	**LiveCodeBench**	**MBPP**	**MMLU**	**GPQA-Diamond**	**MMLU-Pro**	**BBH**	**GSM8K**	**MATH**
n	7	9	9	10	10	10	9	9	10
Generation Acc.	0.89	0.77	0.77	0.47	0.50	0.50	0.67	0.68	0.52
Extraction Acc.	0.96	0.93	0.93	0.98	0.95	0.93	0.83	0.72	0.50
Self-Consistency	0.89	0.77	0.77	0.97	0.90	0.79	0.63	0.50	0.30

K Round-Trip Examples

Table 14 presents example round-trips generated by Qwen3-32B and frontier models (as both G and E) at increasing complexity. Each row shows the skeleton (tree shape), the source expression, the generated word problem, and the expression recovered by the extractor. All five examples were successfully verified by SymPy, despite the recovered expressions differing syntactically from the originals (e.g. reordering commutative operands or distributing subtraction).

Table 14: Round-trip examples at increasing operator count k and depth d. Skeletons use the notation from Figure 2: $\square$ = leaf, $\circ$ = operator. The "Recovered" column shows the extractor's output, which SymPy confirms as symbolically equivalent to the original expression. The first four rows use Qwen3-32B as both G and E; the last two use frontier models.

(k, d)	Skeleton Expression	Word Problem	Recovered
$(2, 2)$	$(\square \circ \square) \circ \square$ $(E - H) - H$	A farmer had {E} acres of land. He used {H} acres for planting corn and another {H} acres for growing soybeans. How many acres were left unplanted?	$E - (H + H)$
$(3, 3)$	$\square \circ (\square \circ (\square \circ \square))$ $D - (1 - (D + C))$	A construction site has {D} tons of steel. Workers use 1 ton for a beam, then recover {D} tons from an old structure and add {C} tons from storage. How much steel is available now?	$D - 1 + D + C$
$(5, 4)$	$(((\square \circ \square) \circ \square) \circ \square) \circ (\square \circ \square)$ $(((F/3) + 1) - C) - (A + 9)$	A traveler drove {F} miles and then took a break after every 3 miles. They added 1 rest stop but skipped {C} due to road closures. How many rest stops were left after subtracting the {A} planned stops plus 9 extra ones?	$(F/3{+}1{-}C){-}(A{+}9)$
$(6, 5)$	$((\square \circ ((\square \circ \square) \circ \square)) \circ \square) \circ (\square \circ \square)$ $((E - ((C{\times}6){\times}B)) - 9) + (G/8)$	A research team counted {E} fish in a coral reef. They observed that {C} divers each removed 6 fish per hour for {B} hours, and 9 fish swam away. Later, {G} baby fish were evenly distributed among 8 areas. How many fish remained in the reef?	$E{-}(C{\times}6{\times}B{+}9){+}G/8$
Frontier models			
$(5, 5)$	$G{\times}(H{+}G/(C{+}H{+}C))$ G/E: Gemini-3.1-Pro	A dispatcher calculates the total fleet size by combining {C} cargo jets, {H} transport helicopters, and another {C} light aircraft. A stockpile of {G} crates of aviation fuel is divided evenly among this entire fleet. Each aircraft takes its share of crates, adds {H} emergency supply boxes to its payload, and multiplies this new load by {G} scheduled flights to find the total cargo transported.	$(H{+}(G/(C{+}H{+}C))){\times}G$
$(5, 5)$	$C{-}F{\times}(F{-}(7{+}F{+}D))$ G/E: GPT-5	In a basketball game, start with 7 bonus points, then add {F} for free throws and {D} for assists, and take this total away from {F}. Multiply that result by {F}, then take that amount away from {C} to find the team's final tally.	$C{-}((F{-}(7{+}F{+}D)){\times}F)$

L Round-Trip Failure Examples

Table 14 shows successful round-trips. Table 15 complements it with representative failures covering all four error categories defined in Appendix I, including both extractor and generator faults. All examples are drawn from strong model pairs ($\geq$14B) and frontier models to illustrate that these failures are not trivial.

Table 15: Representative round-trip failures covering all four error categories, with a mix of extractor and generator faults. For each row: the source expression, the word problem written by G, the expression recovered by E, and a diagnosis identifying whether the fault lies in generation or extraction, based on the consensus vote across the 16 extractors. Underlining highlights the locus of the error in the recovered expression.

Category	Expression & Word Problem	Recovered	Diagnosis
$\mathrm{Err}_{\mathrm{parse}}$ G: GPT-5 E: GPT-5	$((F + H) \times (10 + F/E))$ "You bake {F} trays of cookies and {H} trays of brownies, then count all the trays together. For each tray, you set the glaze amount by splitting {F} cups evenly among {E} bowls and then adding 10 more cups, and you make that amount for every tray."	`(F + H) (F/E + 10)` Juxtaposition without an explicit operator, so the expression does not parse.	**Extractor fault.** The expected expression is the product of "trays" and "glaze per tray," but the extractor drops the explicit $\times$ and emits two parenthesised groups side by side, which the parser rejects.
$\mathrm{Err}_{\mathrm{arity}}$ G: Qwen3-14B E: GPT-5	$G + 3 + 7 - 7$ "A soccer team scored {G} goals in the first half, then scored 3 more goals, and later scored 7 additional goals before giving up 7 in the second half. How many goals did they have at the end of the game?"	$(G + 3) + 7$ *[drops -7]* 2 operators instead of 3.	**Extractor fault.** The word problem clearly narrates four events, including "giving up 7," but the extractor stops after the third and omits the final subtraction, effectively simplifying $+7-7$ away.
$\mathrm{Err}_{\mathrm{arity}}$ G: Qwen3-14B E: Phi-4	$D + F \times 9 + 8$ "A designer creates {D} dresses and {F} fashion sets, each set requiring 9 dresses. After making 8 additional dresses, how many total dresses has the designer created?"	(C/F) 1 operator instead of 3, and with wrong variable (C does not appear in the problem).	**Generator fault.** The narration flattens the two additions and the multiplication into a single sentence and loses the link between "sets," "each requiring 9 dresses," and the final "$+8$" term. The extractors, including the frontier models, fail to recover the three-operation expression, and several emit a single division using a variable (C) that never appears in the problem.
$\mathrm{Err}_{\mathrm{struct}}$ G: GPT-5 E: Qwen3-14B	$E + B - (1 - 9)$ "During a forest survey, a ranger counts {E} birch saplings and then adds {B} pine saplings to the tally. From this total, they subtract the result of taking 9 away from 1 to correct an earlier note. How many saplings are recorded now?"	$(E + B) - (9 - 1)$ Same three operators ($+$, $-$, $-$), with operands of the inner subtraction swapped.	**Extractor fault.** The word problem states "taking 9 away from 1," i.e. $1 - 9$, but the extractor flips the operands to the more natural $9 - 1$, preserving the outer structure while inverting the inner subtraction.
$\mathrm{Err}_{\mathrm{content}}$ G: Qwen3-14B E: GPT-5	$B \times F + E - D$ "During a cycling race, {B} cyclists each rode {F} laps around the track. They then received an extra {E} bonus points for completing the race, but {D} points were deducted for penalties. What is the total points earned by all cyclists?"	$B \times (F + E - D)$ Same skeleton after canonicalisation, with B distributed over a parenthesised sum rather than multiplied only with F.	**Extractor fault.** The word problem distinguishes "laps per cyclist" ($B \times F$) from the additive bonus $+E$ and penalty $-D$, but the extractor binds all three addends under the multiplication, changing the value at every operand assignment. Most extractors recover the intended grouping despite the mild unit slip (laps vs. points), indicating the English is clear enough for most readers.
$\mathrm{Err}_{\mathrm{content}}$ G: Qwen3-32B E: Claude Haiku 4.5	$F + 8 + E$ "{F} flower bulbs were planted in the spring, and by summer there were 8 more blooming flowers. Then, {E} extra flowers were added to the garden bed."	$D + 8 + E$ Same skeleton, with leaf F replaced by D.	**Generator fault.** The word problem starts with "{F} flower bulbs," but the surface form places the variable in sentence-initial position with no anchoring cue, and extractors consistently read the leading token as D. The extractors exhibit the same $F \to D$ confusion: they recover the tree skeleton correctly but mis-bind the leading variable, a universal failure that points to the surface form rather than any individual extractor.

M Fault Attribution Details

This appendix provides implementation details and per-model breakdowns for the fault attribution analysis of Section 3.3, based on the extractor vote $P(e' \mid w)$ (Equation (2)) and extraction agreement p^* (Section 2.4).

Computing extractor votes

The symbolic equivalence $\equiv$ in Equation (2) is evaluated via SymPy: each predicted string is parsed into an expression and simplified; two predictions are equivalent if their simplified forms match. Predictions that fail to parse are treated as distinct. This is applied to all N extractor outputs per word problem to compute $P(e' \mid w)$ and, for non-decodable word problems, the extraction agreement p^* (Section 2.4). Simplification is parallelized across expressions using `ProcessPoolExecutor`.

Aggregate results

Table 16 reports the aggregate breakdown. Of 392276 errors, 73.6% arise from non-decodable word problems ($P(e \mid w) = 0$), matching the headline rate reported in Section 3.3.

Table 16: Fault attribution summary for all incorrect round-trips.

Category	Count	% of errors
Extractor fault ($P(e \mid w) > 0$)	103668	26.4
Non-decodable ($P(e \mid w) = 0$)	288608	73.6

Per-depth breakdown

Table 17 shows the per-depth breakdown. At depth 2, 37.9% of errors are extractor faults (decodable word problems); as depth increases, the extractor-fault rate falls to 24.7% at depth 6. Deeper word problems are longer and more ambiguous, causing extractor outputs to diverge.

Table 17: Fault attribution by expression depth. Ext-fault % is the share of errors from decodable word problems.

Depth	Errors	Ext-fault %
2	1622	37.9
3	13926	29.2
4	128749	26.8
5	133516	27.1
6	114463	24.7

Per-generator breakdown

Table 18 breaks down the two fault-attribution metrics by generator model, ordered by parameter count.

Small models produce systematic errors. Qwen3-0.6B and Gemma-3-4B have the lowest extractor-fault rates (<2%). Their word problems are wrong in a consistent, predictable way: nearly all extractors decode the same incorrect expression, and the word problems are almost never decodable. In contrast, GPT-5 shows the opposite pattern: 97% of its errors are extractor faults, meaning the word problem is faithful but extractors fail to recover the target expression.

Large models shift errors to the extractor. The frontier generators (Haiku-4.5, Gemini-3-Flash, Gemini-3.1-Pro, GPT-5) have extractor-fault rates of 60–97%: their word problems are far more often decodable, so when failures occur they are predominantly attributable to extractor limitations.

Table 18: Fault attribution by generator model. Larger generators produce more decodable word problems (higher ext-fault %).

Generator	Errors	Ext-fault %
Qwen3-0.6B	6889	0.1
Qwen3-1.7B	19630	3.8
Qwen3-4B	28240	14.3
Gemma-3-4B	35235	1.3
Dream-7B	18753	5.5
LLaDA-8B	23910	19.6
Qwen3-8B	27813	17.4
Gemma-3-12B	34419	5.2
Qwen3-14B	28676	21.0
Phi-4	24817	29.5
Gemma-3-27B	33939	6.0
Qwen3-32B	29524	26.6
Haiku-4.5	19624	59.6
Gemini-3-Flash	23456	68.6
Gemini-3.1-Pro	17838	90.2
GPT-5	19513	96.9

Takeaway

Fault attribution reinforces a central finding of this paper: generation is harder than extraction. Over 73% of errors arise from non-decodable word problems, with the rate reaching essentially 100% for the smallest generator and dropping to 3% for GPT-5. This asymmetry is consistent with the generation-extraction gap observed in Section 3.2 and the IRT generation-clarity coefficients in Appendix N. Improving round-trip fidelity will require advances primarily on the encoding side, particularly for deep expressions where generators most frequently distort the intended arithmetic structure.

M.1 Misattribution Sensitivity

The consensus-based attribution (Section 2.4) classifies a word problem as a generator fault when no extractor in the panel recovers the target expression. A faithful word problem will be misclassified whenever every extractor independently fails on it. We bound this probability as follows.

Let f_i denote the success rate of extractor i on confirmed-faithful word problems (those where at least one extractor succeeds). Assuming conditional independence across extractors, the probability that a faithful word problem is misclassified as a generator fault is

$$P(\text{misattribution}) \;=\; \prod_{i=1}^{N}(1 - f_i). \tag{5}$$

Table 19 reports f_i for all 16 extractors, estimated from 14232 confirmed-faithful word problems. Values range from 0.064 (Qwen3-0.6B) to 0.820 (GPT-5), with a mean of $\bar{f} = 0.55$. Using the per-extractor values in Equation (5) gives $P(\text{misattribution}) = 4.5 \times 10^{-7}$, yielding fewer than 1 expected false attribution among the 18038 word problems classified as generator faults by the consensus rule.

The independence assumption is optimistic: extractors may share failure modes on the same hard word problems. However, the panel spans six architectural families and four frontier APIs, limiting correlated failures. Even under moderate positive correlation, the false-attribution rate remains negligible given the diversity of the panel.

Table 19: Per-extractor success rate f_i on 14232 confirmed-faithful word problems (those where at least one extractor recovers the target). Models sorted by f_i.

Extractor	Correct / 14232	f_i
Qwen3-0.6B	913	0.064
Qwen3-1.7B	3154	0.222
Dream-7B	3959	0.278
Gemma-3-4B	4179	0.294
LLaDA-8B	6349	0.446
Gemma-3-12B	6890	0.484
Qwen3-4B	7281	0.512
Qwen3-8B	7782	0.547
Gemma-3-27B	9028	0.634
Qwen3-14B	9393	0.660
Qwen3-32B	9783	0.687
Phi-4	9992	0.702
Haiku-4.5	10708	0.752
Gemini-3-Flash	11404	0.801
Gemini-3.1-Pro	11557	0.812
GPT-5	11672	0.820
Mean		0.545

Panel-composition sensitivity

The consensus-based attribution depends on the extractor panel: a generation is classified as faithful whenever any single extractor recovers the target, so the generator-fault rate is a function of panel composition. Table 20 reports the rate under nine panel subsets, all reusing the same extraction data. Across these subsets the rate ranges 73.6–91.1%, with the full 16-model panel yielding the lowest value: every restricted panel classifies more failures as generator faults, because smaller panels afford fewer chances for "at least one" extractor to succeed. Within same-size panels ($|P|$=3), the Gemma-3-only panel gives 85.8% and the other-open-weight panel gives 82.6%, indicating that stronger extractors are if anything more likely to collectively fail on genuinely bad word problems. The diversity-controlled "one per family" panel (77.3%) is close to Qwen3-only (79.3%) and above the full open-weight panel (74.4%), offering no evidence that correlated failures within a single family inflate the headline figure.

Table 20: Generator-fault rate under different extractor panel subsets (incl-guard convention). The full-panel rate of 73.6% is the lowest observation; every restricted panel yields a higher rate.

Panel	$\|P\|$	Errors	GenF %
Full (all models)	16	392276	73.6
Open-weight only	12	308537	74.4
Qwen3 only	6	155314	79.3
Gemma-3 only	3	76713	85.8
Other (Phi/Dream/LLaDA)	3	76510	82.6
Frontier only	4	83739	91.1
Strong ($\geq$14B + frontier)	8	174623	84.8
Weak ($\leq$4B)	4	113553	83.7
One per family (median)	6	146976	77.3

N Latent-Ability Model (IRT)

As a validation of the additive decomposition in Section 2.3, we fit a Rasch model [Rasch, 1980] to the binary per-example outcomes in the communication matrix. The fit uses all 16 models (12 open-weight and 4 frontier). For each triple (extractor i, generator j, expression k), we model the probability of a correct round-trip as

$$\log \frac{P(\text{correct})}{1 - P(\text{correct})} \;=\; \mu + \theta_i + \beta_j - \delta_k, \tag{6}$$

where θ_i is the extraction skill of model i, β_j is the generation clarity of model j, and δ_k is the difficulty of expression k. Equivalently, the predicted success probability is

$$P(\text{correct}) \;=\; \frac{1}{1+\exp\bigl[-(\mu+\theta_i+\beta_j-\delta_k)\bigr]}. \tag{7}$$

For example, given the estimated global baseline $\mu = -1.90$, pairing Phi-4 ($\theta = +0.53$, $\beta = +1.07$) on a median-difficulty expression ($\delta = 0.1$) gives $P = \sigma(-1.90+0.53+1.07-0.1) \approx 40.1\%$, while a hard expression ($\delta = 1.2$, $\sim$90th percentile) yields $P = \sigma(-1.90+0.53+1.07-1.2) \approx 18.2\%$. Parameters are estimated by L_2-regularised logistic regression (regularisation strength $C = 10$).

Table 21 reports the estimated coefficients for each model alongside its raw marginal accuracy from the communication matrix. Spearman rank correlations between the IRT coefficients and the matrix marginals in Table 1 are $\rho = 1.00$ for extraction and $\rho = 0.96$ for generation, confirming that the simple averages faithfully summarise latent ability. Among the three parameter groups, generation clarity has the largest variance ($\sigma_\beta = 1.93$, compared with $\sigma_\delta = 1.00$ for expression difficulty and $\sigma_\theta = 0.98$ for extraction), indicating that variation in how well models encode structure exceeds variation in expression difficulty or extraction skill. Including all three parameter groups reduces the log-loss from 0.44 (model-only, $\theta + \beta$) to 0.38 (full model with δ), confirming that expression difficulty adds predictive signal beyond the model coefficients.

Table 21: Rasch model coefficients and matrix marginal accuracies. θ: extraction skill; β: generation clarity; Raw Ext. / Gen.: row / column averages of the 16×16 communication matrix in Table 1 (guard failures counted as errors). Models are sorted by family and size.

Model	θ	Raw Ext. (%)	β	Raw Gen. (%)
Qwen3-0.6B	-2.89	2.3	-4.23	0.1
Qwen3-1.7B	-1.43	8.0	-1.61	2.6
Qwen3-4B	-0.08	18.6	+0.02	13.4
Qwen3-8B	+0.11	19.9	+0.53	19.0
Qwen3-14B	+0.42	24.0	+0.55	20.5
Qwen3-32B	+0.49	25.0	+0.44	18.8
Gemma-3-4B	-0.92	10.7	-3.20	0.7
Gemma-3-12B	-0.14	17.6	-1.42	3.3
Gemma-3-27B	+0.36	23.0	-1.74	3.3
Phi-4	+0.53	25.5	+1.07	25.8
Dream-7B	-1.07	10.1	-1.60	2.2
LLaDA-8B	-0.34	16.2	+0.66	15.9
Haiku-4.5	+0.67	27.3	+2.10	46.5
GPT-5	+0.81	29.8	+2.22	50.2
Gemini-3-Flash	+0.77	29.1	+1.74	39.6
Gemini-3.1-Pro	+0.82	29.5	+2.59	54.4
Spearman ρ		1.00		0.96

O Fine-Tuning: Assembly Domain Details

This appendix provides the full domain specification for both the nested assembly fine-tuning data, along with training examples and qualitative analysis supporting the fine-tuning experiment in Section 4.

O.1 Assembly Domain Specification

Expressions are random trees over eleven assembly operators of mixed arity: three unary (`label`, `inspect`, `polish`), six binary (three commutative: `weld`, `glue`, `rivet`; three non-commutative: `mount`, `pour`, `load`), and four ternary (two commutative: `mix`, `assort`; two non-commutative: `layer`, `thread`). Operators are applied to coloured-object leaves drawn from 13 colours $\times$ 13 objects. Tree depth is sampled uniformly from 2 to 5, matching the benchmark range. Each operator maps to multiple paraphrase templates describing physical assembly steps in a *procedural* style (ordinal step markers). Every (expression, NL) pair yields two training examples ($e \to w$ and $w \to e$).

We generate 500 pairs per depth with a 90/10 train/test split, producing approximately 3600 training messages.

O.2 Training Examples

Table 22 shows three nested assembly expressions rendered in procedural style, illustrating the mixed arity (unary `polish`, binary `mount`, ternary `assort`/`load`/`mix`) and non-commutative argument ordering.

Table 22: **Sample assembly expressions** (depth 3) with procedural NL renderings.

Expression	Natural-language rendering
`assort(label(copper gem), gold ring, yellow card)`	First, mark the copper gem with a tag to get the first component. Second, assort the first component, the gold ring, and the yellow card into one set as the final product.
`mount(rivet(bronze ball, copper coin), black cube)`	First, clinch the bronze ball to the copper coin with a pop rivet to get the first component. Second, position the first component on the face of the black cube as the final product.
`polish(layer(silver ring, gold rod, blue cube))`	First, deposit the silver ring at the bottom, the gold rod in the centre, and the blue cube at the top to get the first component. Second, buff the first component to a shine as the final product.

O.3 Procedural assembly FT: failure taxonomy

The 2.1 pp guard-pass drop reported for assembly FT (Table 2) is concentrated in two models, not shared across the panel. Table 23 gives the per-model Leak pass rate (operator adjacency, the only category counted in the guard column) for the base and assembly FT conditions. Gemma-3-4B and Gemma-3-12B account for nearly all of the aggregate drop (-22.9 and -17.5 pp). Phi-4 and Qwen3-14B/32B stay within ± 0.7 pp of their base, and Qwen3-0.6B/1.7B improve ($+32.6$ and $+7.0$ pp) because their base outputs leak heavily and FT reduces that tendency. We do not have a mechanistic account for the Gemma-specific drop. The *missing-variable* fault (a generator error under our reporting rule, not a guard failure) rises by $+7.0$ pp on average under FT, dominated again by Qwen3-0.6B ($+42.2$) and Qwen3-1.7B ($+17.8$), where the smallest generators drop placeholders more often after training.

Table 23: Leak **pass rate under base and procedural assembly FT** (operator adjacency, the only failure category counted in the Leak column of Table 2). Values are the share of 1790 problems that clear the operator-leak check. Two Gemma models concentrate the aggregate drop.

Model	Base %	FT_{asm} %	Δ pp
Qwen3-0.6B	19.6	52.2	$+32.6$
Qwen3-1.7B	58.9	65.9	$+7.0$
Qwen3-4B	93.9	82.2	-11.7
Qwen3-8B	97.5	94.0	-3.5
Qwen3-14B	99.2	98.5	-0.7
Qwen3-32B	98.4	97.7	-0.7
Gemma-3-4B	98.0	75.1	$\mathbf{-22.9}$
Gemma-3-12B	96.2	78.7	$\mathbf{-17.5}$
Gemma-3-27B	96.1	92.7	-3.4
Phi-4	96.6	96.5	-0.1
Panel mean	85.4	83.3	-2.1

O.4 Qualitative Analysis

Table 24 illustrates how assembly fine-tuning change the style of generated word problems for Phi-4.

Table 24: Word problems generated by Phi-4 (base and assembly fine-tuned FT) for three expressions. Assembly fine-tuning shifts output toward explicit arithmetic descriptions.

Expression: `( C - ( A + B + 2 ) )` [depth 3]	
Base	A real estate agent is calculating the remaining budget for a property renovation. The total budget is {C}, and the costs include {A} for materials, {B} for labor, and an additional 2 for miscellaneous expenses. The agent needs to determine how much budget is left after these expenses.
FT	A developer had {C} plots of land. They sold {A} plots in the spring, {B} plots in the summer, and set aside 2 for a park. How many plots remained?
Expression: `( B - C * ( A + G ) )` [depth 3]	
Base	A delivery truck starts with {B} gallons of fuel. It uses {C} gallons for each mile driven, and it travels a total of {A} miles plus an additional {G} miles for detours. How many gallons of fuel does the truck have left after completing the trip?
FT	A bus company had {B} buses in service. They assigned {A} buses to the north route and {G} buses to the south route, then multiplied the total by {C} to get the number of buses used for express service. How many buses remained in service?
Expression: `( G * ( D + C ) / H )` [depth 3]	
Base	A musician is composing a piece and wants to determine the total number of beats in a section. They have {G} measures, each containing {D} beats plus an additional {C} beats for a special rhythm. To find the average number of beats per measure, they divide the total beats by {H}.
FT	First, add {D} and {C} together to get the first component. Second, multiply {G} by the first component to get the second component. Third, divide the second component by {H} to get the third component. Fourth, the third component is the final product.

The base model produces fluent narratives whose structure often diverges from the target expression: in the first example, `C - (A + B + 2)` becomes "remaining budget" after listed expenses, flattening the subtraction-of-a-sum into a single subtractive relation. The assembly-fine-tuned model makes operations explicit and sequential: narrative outputs preserve operator order ("sold ... in the spring, {B} plots in the summer, and set aside 2 for a park"), and the third example collapses to a pure step-marker form ("First, add ... Second, multiply ... Third, divide ..."). This structural fidelity comes at a stylistic cost: two Gemma models contribute most of the guard-rate drop reported in Table 23.

P Shot-Configuration Grid

Figures 8 to 10 report generation quality (q^{G}) and extraction quality (q^{E}) across all evaluated (γ, ε) shot configurations for each training condition. The base figure justifies the $(\gamma{=}0, \varepsilon{=}3)$ configuration used in Table 1; the FT figures justify the per-condition selections used in Table 2. Bold marks the per-row maximum in each panel. All values (%) are averaged over the 10-model open-weight extractor panel with guard failures counted as errors.

(a) Generation quality (q^{G})

	$\gamma = 0$		$\gamma = 3$	
Model	ε=0	ε=3	ε=0	ε=3
Qwen3-0.6B	0.1	0.1	**0.3**	**0.3**
Qwen3-1.7B	**2.6**	**2.6**	0.3	0.3
Qwen3-4B	13.7	**14.0**	2.9	3.1
Qwen3-8B	19.7	**20.5**	5.3	5.6
Qwen3-14B	20.7	**21.4**	10.6	11.3
Qwen3-32B	18.3	**19.4**	10.0	10.7
Gemma-3-4B	0.8	0.8	1.0	**1.1**
Gemma-3-12B	3.8	**3.9**	2.6	2.5
Gemma-3-27B	3.6	3.8	**4.1**	**4.1**
Phi-4	25.9	**26.9**	10.4	10.9

(b) Extraction quality (q^{E})

	$\gamma = 0$		$\gamma = 3$	
Model	ε=0	ε=3	ε=0	ε=3
Qwen3-0.6B	2.6	**3.0**	0.7	0.7
Qwen3-1.7B	3.1	**6.6**	1.2	2.1
Qwen3-4B	**12.5**	**12.4**	5.4	5.4
Qwen3-8B	12.8	**12.9**	5.5	6.0
Qwen3-14B	14.3	**14.5**	6.5	7.1
Qwen3-32B	14.0	**14.7**	6.7	7.0
Gemma-3-4B	**8.6**	8.4	3.1	3.3
Gemma-3-12B	**11.9**	11.7	5.2	5.3
Gemma-3-27B	14.0	**14.2**	6.4	6.5
Phi-4	**15.3**	15.1	6.7	6.7

Figure 8: **Base condition: shot-configuration grid.**

(a) Generation quality (q^{G})

	$\gamma = 0$		$\gamma = 3$	
Model	ε=0	ε=3	ε=0	ε=3
Qwen3-0.6B	0.0	0.0	**0.3**	0.1
Qwen3-1.7B	0.2	0.1	2.6	**6.5**
Qwen3-4B	**20.8**	11.2	12.1	17.3
Qwen3-8B	22.3	22.8	23.3	**32.6**
Qwen3-14B	20.3	24.8	**27.6**	26.2
Qwen3-32B	23.7	21.3	20.5	**27.6**
Gemma-3-4B	6.0	**7.4**	3.4	5.5
Gemma-3-12B	7.9	11.5	12.0	**16.7**
Gemma-3-27B	23.2	23.1	**25.8**	20.4
Phi-4	**36.9**	35.2	34.1	32.7

(b) Extraction quality (q^{E})

	$\gamma = 0$		$\gamma = 3$	
Model	ε=0	ε=3	ε=0	ε=3
Qwen3-0.6B	3.3	**3.9**	2.5	3.7
Qwen3-1.7B	4.1	7.3	3.4	**8.4**
Qwen3-4B	18.2	17.7	18.3	**20.9**
Qwen3-8B	19.6	18.3	17.6	**21.1**
Qwen3-14B	21.9	20.7	21.8	**24.2**
Qwen3-32B	22.8	22.6	22.5	**25.3**
Gemma-3-4B	9.1	9.5	10.0	**11.9**
Gemma-3-12B	17.3	14.5	17.3	**19.2**
Gemma-3-27B	20.0	20.5	22.8	**24.2**
Phi-4	24.9	22.6	25.6	**26.6**

Figure 9: **Assembly FT (procedural) condition: shot-configuration grid.** Panel means are $16.13/15.75/16.17/18.54$ for $(\gamma, \varepsilon) \in \{(0,0), (0,3), (3,0), (3,3)\}$, so the panel optimum is (γ=3, ε=3). Per-row optima vary: Phi-4 and Qwen3-4B peak at $(0,0)$, Qwen3-14B and Gemma-3-27B at $(3,0)$.

(a) Generation quality (q^{G})

	$\gamma = 0$	
Model	ε=0	ε=3
Qwen3-0.6B	47.7	**51.9**
Qwen3-1.7B	57.0	**57.2**
Qwen3-4B	53.2	**55.3**
Qwen3-8B	56.4	**59.5**
Qwen3-14B	54.8	**55.9**
Qwen3-32B	56.0	**58.0**
Gemma-3-4B	53.9	**55.4**
Gemma-3-12B	56.5	**57.8**
Gemma-3-27B	55.0	**58.4**
Phi-4	56.6	**58.2**

(b) Extraction quality (q^{E})

	$\gamma = 0$	
Model	ε=0	ε=3
Qwen3-0.6B	13.7	**14.1**
Qwen3-1.7B	17.2	**34.8**
Qwen3-4B	62.8	**64.4**
Qwen3-8B	64.7	**66.2**
Qwen3-14B	71.0	**72.7**
Qwen3-32B	70.0	**73.6**
Gemma-3-4B	**39.2**	37.1
Gemma-3-12B	**62.2**	59.8
Gemma-3-27B	**72.2**	70.6
Phi-4	74.0	**74.4**

Figure 10: **Arithmetic FT condition: shot-configuration grid.** Arithmetic FT was evaluated with γ=0 only (in-domain generators do not benefit from exemplars).

Q Lossless-Channel Extraction Baselines

To quantify the cost of using natural language as the communication channel, we replace the NL word problem with an unambiguous structured representation and measure how accurately the extractor recovers the original infix expression. We evaluate two formats:

S-expression (prefix notation). Each expression is deterministically converted to prefix notation, *e.g.,* $(A + B) \times C$ becomes `(* (+ A B) C)`.

JSON abstract syntax tree. Each expression is converted to a nested JSON dictionary, *e.g.,* $(A + B) \times C$ becomes `{"op": "*", "left": {"op": "+", "left": "A", "right": "B"}, "right": "C"}`.

Both formats encode tree structure without ambiguity, so extraction accuracy on either task upper-bounds what any faithful NL encoding could achieve for a given extractor.

The S-expression vs. NL comparison scales with operator count for the Qwen3 family: for the strongest extractors (14B–32B), S-expression accuracy stays high even at k=8 (78–82%), whereas NL extraction falls off sharply as expressions grow more complex. The NL tax is thus negligible at low complexity and widens at high complexity, confirming that the natural-language bottleneck specifically targets structurally complex expressions; Figure 3-right quantifies this gap against the best available lossless format across extractor sizes.

Format dependence at mid-scale. Comparing the two lossless channels across model sizes reveals that the structured-format ceiling is not format-invariant. JSON extraction substantially exceeds S-expression extraction for mid-range models (Qwen3-4B: 63.2% vs. 40.8%; Qwen3-8B: 75.9% vs. 37.5%), while both converge at 32B ($\approx 88\%$). This gap likely reflects pretraining exposure: models encounter far more JSON than S-expressions in their training corpora. The NL tax should therefore be measured against the best available lossless format ($\max$(S-expr, JSON)), as reported in Figure 3-right.

R Propositional-Logic Replication

To test whether the communication bottleneck is specific to arithmetic or reflects a more general structural limitation, we replicate the protocol on propositional logic on a reduced 14×14 roster (the arithmetic frontier additions Gemini-3-Flash and Gemini-3.1-Pro are omitted to limit compute). Formulas are built from five connectives (AND, OR, NOT, IMPLIES, XOR) over propositional variables P–W, at depths 2–5. Correctness is judged by *logical equivalence* via SymPy, so surface-form variation (e.g. commutativity of AND) does not penalise the extractor. A domain-adapted leakage guard forbids logical operator words and symbols in the generated scenario, mirroring the arithmetic guard.

Protocol. The generator receives a formula and a randomly sampled real-world domain (access control, safety regulations, game rules, etc.) and must produce a 2–3 sentence scenario using proposition placeholders $\{P\}, \{Q\}, \ldots$ without any logical vocabulary. The extractor receives the scenario and must recover the original formula using only the connectives AND, OR, NOT, IMPLIES, XOR. The same 14 models serve as both generators and extractors.

Example (depth 3).

```
Formula: ( ( P AND Q ) IMPLIES R )
Domain: safety regulations

Scenario: When equipment {P} is running and the temperature
reading {Q} exceeds the threshold, the automatic shutdown
protocol {R} activates.
```

Results. Table 25 shows the full logic communication matrix. Overall accuracy is lower than in arithmetic (mean cell accuracy 8.7% vs. 14.9% on the shared 14-model block), consistent with the additional ambiguity introduced by logical connectives that lack the concrete semantics of arithmetic operators. Despite this difficulty shift, the relative ordering of models is largely preserved.

Figure 11 compares arithmetic and logic ranks for both generation and extraction quality. Generation ranks are well preserved (Spearman $\rho = 0.85$, Kendall $\tau_b = 0.69$): GPT-5 and Claude Haiku 4.5

lead in both domains. Extraction ranks show comparable pairwise agreement ($\tau_b = 0.67$), though the Spearman correlation is lower ($\rho = 0.69$) due to a single large displacement: Phi-4 drops from extraction rank 3 in arithmetic to rank 14 in logic, showing a strong imbalance in terms of extraction abilities (ranked 3rd on arithmetic). This suggests that its strong arithmetic extraction reflects format-specific parsing fluency rather than general structural competence.

Excluding Phi-4, extraction ρ rises above generation ρ, indicating that for all other models the extraction ranking transfers at least as reliably as the generation ranking.

Table 25: **The 14 × 14 communication matrix on all generated expressions (guard failures counted as errors).** Each cell shows round-trip accuracy (%). Rows are extractors, columns are generators. The rightmost column and bottom row show extraction quality (row average) and generation quality (column average). Generator and extractor are run with prompts π^0_{G} and $\pi^{\varepsilon}_{\mathsf{E}}$ respectively (configuration γ=0, ε=3); see Appendix P for the full ablation. The ± values denote standard deviation across the 14 pairings in each row or column. The Leak % row reports each model's leakage-guard pass rate as a generator. Thin rules separate open-weight auto-regressive, diffusion, and frontier models.

EXTRACTORS \ GENERATORS	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B	Gemma-3-4B	Gemma-3-12B	Gemma-3-27B	Phi-4	Dream-7B	LLaDA-8B	C-Haiku-4.5	GPT-5	Extr. quality
Leak pass %	92	96	95	97	98	99	91	95	98	99	87	87	98	100	
Qwen3-0.6B	1.4	2.7	3.5	1.9	3.8	3.9	1.5	2.2	3.2	5.7	2.7	2.0	5.6	5.8	$3.3_{\pm 1}$
Qwen3-1.7B	2.7	4.3	5.9	5.5	8.4	7.4	2.1	3.4	5.2	12.0	4.4	4.0	14.2	19.6	$7.1_{\pm 5}$
Qwen3-4B	3.0	4.3	7.9	5.5	9.5	8.9	2.2	4.0	6.2	14.6	5.6	3.7	18.8	34.5	$9.2_{\pm 8}$
Qwen3-8B	2.6	4.8	8.2	6.2	10.5	10.9	2.1	4.2	7.7	15.7	5.0	4.2	21.5	40.8	$10.3_{\pm 10}$
Qwen3-14B	2.6	4.8	8.8	6.9	11.3	11.6	2.2	4.7	7.8	17.1	4.6	3.6	22.8	37.8	$10.5_{\pm 9}$
Qwen3-32B	2.6	4.8	9.4	8.0	11.7	13.4	2.0	5.6	8.1	19.0	5.4	4.5	26.2	54.8	$12.5_{\pm 13}$
Gemma-3-4B	2.2	3.8	6.2	4.4	7.4	6.8	2.1	3.0	4.6	11.6	4.0	3.2	14.3	15.9	$6.4_{\pm 4}$
Gemma-3-12B	2.5	4.7	8.2	6.3	10.7	10.9	2.1	4.7	7.8	15.6	5.3	4.4	20.1	35.2	$9.9_{\pm 9}$
Gemma-3-27B	2.7	4.5	8.7	7.4	11.4	11.2	2.1	4.5	7.3	17.4	4.9	4.5	22.0	38.9	$10.5_{\pm 10}$
Phi-4	1.7	2.4	1.1	0.8	1.1	2.1	0.6	0.4	0.5	2.1	0.7	0.5	3.6	4.1	$1.5_{\pm 1}$
Dream-7B	1.8	3.0	2.9	2.5	4.2	3.4	1.5	1.6	2.3	6.2	2.2	2.1	7.4	9.1	$3.6_{\pm 2}$
LLaDA-8B	2.4	3.7	6.6	6.5	9.0	9.1	2.3	4.1	6.9	14.4	4.3	4.2	15.6	27.6	$8.3_{\pm 7}$
C-Haiku-4.5	2.6	5.1	9.7	8.3	12.8	13.8	1.9	4.6	9.8	19.2	5.8	5.2	28.0	57.3	$13.1_{\pm 14}$
GPT-5	2.8	4.9	9.9	8.5	14.0	14.5	2.3	4.6	8.2	21.1	5.5	5.0	28.3	80.4	$15.0_{\pm 19}$
Gen. quality	$2.4_{\pm 0}$	$4.1_{\pm 1}$	$6.9_{\pm 3}$	$5.6_{\pm 2}$	$9.0_{\pm 4}$	$9.1_{\pm 4}$	$1.9_{\pm 0}$	$3.7_{\pm 1}$	$6.1_{\pm 3}$	$13.7_{\pm 5}$	$4.3_{\pm 1}$	$3.6_{\pm 1}$	$17.8_{\pm 8}$	$33.0_{\pm 21}$	

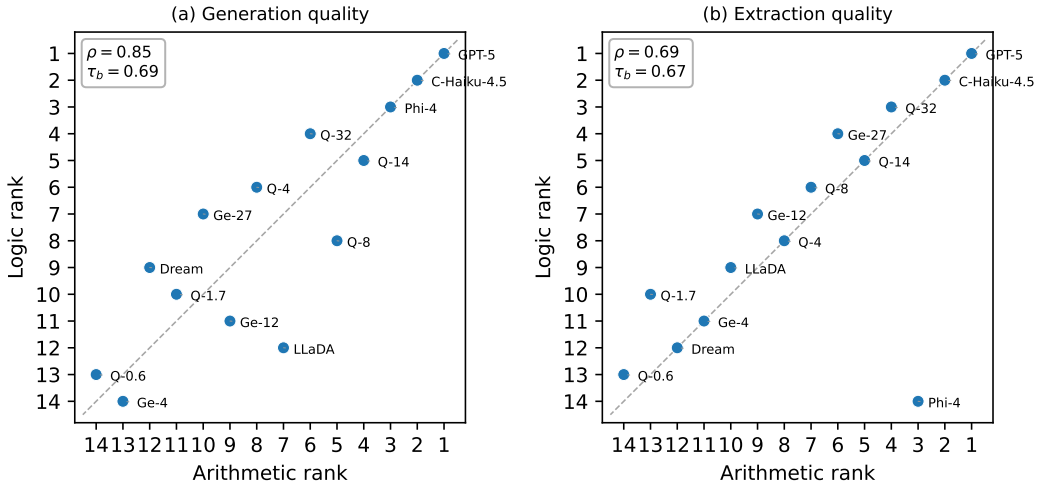

Figure 11: Arithmetic vs. logic rank correlation for generation quality (left) and extraction quality (right). Each point is one model; the dashed diagonal marks perfect agreement. Spearman ρ and Kendall τ_b are shown in each panel. Phi-4 (bottom-right in panel b) is the sole large outlier, dropping from extraction rank 3 to rank 14.

S Compute Resources

All GPU experiments were run on NVIDIA A100 80GB nodes. Frontier models were accessed over API, we report call counts (one request per generated or extracted example).

Table 26: Compute consumed by experiments reported in this paper. GPU hours are wall-clock on NVIDIA A100 80GB nodes, summed across generator and extractor roles. API calls count requests to frontier models (one request per generated or extracted example) and are listed separately because they do not consume our GPU budget.

Experiment	A100 80GB (h)	API calls	Section
Communication matrix (arithmetic)	182.5	85,644	Section 3.2
Communication matrix (logic)	119.2	72,714	Section 3.4
Lossless baseline (S-expression)	0.6	–	Section 3.2
Lossless baseline (JSON AST)	0.5	–	Section 3.2
Fine-tuning (arithmetic)	68.5	–	Section 4.1
Fine-tuning (assembly)	57.3	–	Section 4.1
Temperature sweep (Phi-4)	13.3	–	Appendix E
Total	**441.9**	**158,358**	

T Author Notes

T.1 Broader Impact

This is a measurement study with no new model, human-subjects data, or deployed system, so direct societal impact is limited. Positively, the round-trip protocol gives practitioners a principled way to audit model-to-model and chain-of-thought pipelines for silent structural errors before deployment. Negatively, our fine-tuning results show that modest data suffices to improve structured-to-text serialization, which could marginally aid production of persuasive structured content, a risk we judge small relative to existing generation capabilities and which is mitigated by open release of our code and data.

T.2 Use of LLMs

Claude Opus 4.6 [Anthropic, 2026a] and 4.7 [Anthropic, 2026b] were used under human supervision to assist with code, drafting, and discussion of research directions. All scientific claims, experimental designs, and final phrasings were reviewed and validated by the authors, who take full responsibility for the content.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract's three findings (lossy asymmetric channel with up to 60.4 pp role-swap gap and 92.9% best pair, $\geq$73.6% of failures at generation with structure-driven difficulty, and $\sim$3600 fine-tuning examples lifting open-weight models above the strongest non-fine-tuned frontier generator) are supported by Sections 3.2, 3.3 and 4 respectively. Scope (16 models, arithmetic plus logic replication) is stated in the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are openly discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not contain formal theorems or theoretical results. The main numbered equations in Sections 2.3 and 2.4 are derived from basic algebra and do not require formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all details about inference and prompts in Appendix A, as well as detailed information about the fine-tuning procedure in Appendix O and the S-expr/JSON lossless experiments in Appendix Q. Details for other side experiments are provided throughout the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We release the code at `https://github.com/apple-aiml-research/ml-roundtrip`. The data is procedurally generated with the code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details are provided in Section 3.1, and fine-tuning params in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, std intervals are provided whenever available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources (NVIDIA A100 80GB GPU hours and API hours) are reported per experiment in Table 26 (Appendix S). Preliminary and failed runs are not included in those totals, so the full project consumed additional compute not reported here.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work has no conflict with the given code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We include a Broader Impact section in appendix (Section T.1), discussing the very limited negative impact of our work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work poses no risk related to data or model misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all models used in this work. All open-source models have Apache2.0 or MIT licenses. We additionally have direct permission from Anthropic and OpenAI to use their respective models.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets. Upon acceptance, code will be released, which will allow to procedurally generate new arithmetic expressions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No human subjects are used in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We describe the use of LLMs in Section T.2.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.