Where Small Language Models Break: A Procedurally Generated, Self-Verifying Benchmark Across Fourteen Models

Abstract

Small language models are increasingly deployed on consumer hardware, where their reliability under everyday conditions matters more than their score on any single leaderboard. Yet existing benchmarks are built for models an order of magnitude larger, and they rarely isolate *why* a small model fails rather than simply reporting that it does. This gap leaves practitioners without a way to predict, for a specific task shape, how many reasoning steps or constraints a given small model can actually handle before it breaks. We introduce a procedurally generated benchmark of **2,800 items** spanning four task families: chained reasoning, simultaneous instruction-following, state tracking, and character-level counting. Every item is generated and graded by the same deterministic code, so no human ever labels an answer, and every ground-truth value was independently re-derived by a second implementation with zero mismatches across **45,872 checks**. Compared with fixed-difficulty benchmarks, our design offers three advantages: it sweeps a single difficulty axis per family so the exact collapse point is visible, not just the average score; it is fully reproducible from a random seed, so a fresh, uncontaminated test set can be generated at any time; and it grades every response automatically, so results scale to as many models as compute allows. We evaluate **fourteen** open-weight models between 0.5 and 4 billion parameters, entirely on consumer CPU hardware. Our results show that (1) accuracy on chained reasoning drops from near-perfect to near-random within three to five steps for every model tested, (2) instruction-following compliance falls below 50% once a prompt carries more than two or three simultaneous constraints, regardless of model size, (3) a large share of failures are not reasoning errors at all, but a failure to emit a parseable answer within a short response budget, and (4) accuracy and speed trade off in a way that reorders the leaderboard entirely once latency is taken into account. This last finding is the most consequential: a substantial share of measured errors trace back to response formatting rather than task knowledge, showing that raw capability and instruction-following are separable failure modes that current single-score benchmarks conflate. This result suggests that improving a small model's usability may depend as much on training it to be concise as on training it to be correct. We release the full generator code, the dataset, and the grading harness so that any new model can be benchmarked on a fresh, uncontaminated instance of the same tasks.

1 Introduction

A phone that runs a language model offline, a laptop with no internet connection, a robot with a small onboard chip: all of these depend on language models small enough to run without a data center, and this class of model is becoming the backbone of everyday, offline AI assistance. Consumer devices increasingly ship with such models pre-installed, and developers building for these devices need to know, in advance, what a specific small model can and cannot reliably do.

Existing benchmarks do not answer this question well. Most evaluation suites are built and tuned for frontier-scale models, and they report a single aggregate score rather than the shape of a model's failure curve. A benchmark that says a model scores 62% on a mixed test set does not tell a developer whether that model can track three sequential updates to a variable, or whether it can follow four instructions in one

prompt. There is a significant knowledge gap between what practitioners need, which is a map of exactly where a given small model's abilities run out, and what current benchmarks provide, which is a single number averaged across items of uneven and unstated difficulty. The central obstacle is that most benchmarks are not built to isolate one difficulty axis at a time, so a low score cannot be traced to a specific mechanism. Therefore, in this paper, we seek to answer this critical question: *At what point, and by what mechanism, does a small language model's accuracy collapse as task difficulty increases, and how does that collapse trade off against speed?*

To address this question, we hypothesize that the accuracy of small language models on structured tasks is governed less by raw scale and more by three separable factors: how many discrete steps the underlying task requires, how strictly the model must format a terse answer under a tight response budget, and how much latency it costs to reach that answer. To this end, we propose a benchmark, called SLM-COLLAPSE, which sweeps a single difficulty axis at a time across four task families and grades every response with deterministic code. The key technical idea is that a difficulty sweep, rather than a fixed-difficulty test set, turns a single accuracy number into a full curve, and that curve reveals the collapse point rather than hiding it in an average. Compared with existing small-model evaluations, SLM-COLLAPSE offers three unique advantages:

1. **Traceable.** Unlike mixed-difficulty suites, SLM-COLLAPSE isolates one difficulty axis per family, so a score decline can be attributed to a specific mechanism rather than an unspecified mix of item difficulties.
2. **Contamination-resistant.** In contrast to static benchmarks whose items eventually leak into training corpora, SLM-COLLAPSE is procedurally generated from a random seed, so a fresh, unseen instance can be produced at any time.
3. **Annotation-free.** Different from benchmarks that require costly human labeling, every item in SLM-COLLAPSE is generated and graded by the same deterministic function, removing human annotation from the pipeline entirely.

We have examined SLM-COLLAPSE on fourteen open-weight models, spanning six model families and a parameter range of 0.5 to 4 billion. Our extensive experiments demonstrate that (1) chained reasoning collapses sharply within three to five steps for every model tested, (2) instruction-following compliance falls below half once a prompt carries more than two or three simultaneous constraints, (3) a substantial share of measured failures trace back to response formatting rather than task knowledge, and (4) the fastest model overall is not the most accurate one, so a single leaderboard column cannot answer which model a practitioner should actually deploy. This result is attributable to our simple yet powerful observation: small models frequently know the correct answer but fail to state it within a terse response budget, so instruction-following capacity is a distinct and separately measurable bottleneck from task knowledge. This result suggests that future small-model training and evaluation should treat concision and answer-formatting as a first-class capability, not an incidental side effect of instruction tuning.

To our best knowledge, we are among the first to benchmark a broad panel of sub-4B language models entirely on consumer CPU hardware using a fully procedural, zero-annotation methodology. To summarize, we make the following four contributions:

- A procedurally generated, self-verifying benchmark of 2,800 items across four task families, with ground truth independently confirmed by a second implementation across 45,872 checks.
- A systematic evaluation of fourteen small language models, spanning six architecture families, entirely reproducible on consumer CPU hardware.
- An empirical decomposition of accuracy loss into a reasoning-depth component and a response-formatting component, showing the two are separable and differently distributed across models.
- A released, open generator toolchain that produces new, uncontaminated benchmark instances on demand, mitigating the risk of future training-data leakage.

The rest of the paper is organized as follows. Section 2 describes the four task families and the verification process. Section 3 describes the models and evaluation protocol. Section 4 reports four findings in turn: the collapse curves, the leaderboard and its failure-mode composition, the speed–accuracy tradeoff, and the constraint-sensitivity result. Section 5 discusses three open questions the data raises but does not resolve. Appendix A shows the full benchmark construction pipeline, Appendix B reports the within-family accuracy spread underlying the collapse curves, and Appendix C reports the full per-model wordiness gap referenced in Section 4.2.

2 The SLM-COLLAPSE Benchmark

The objective of SLM-COLLAPSE is to measure, for a given small language model, the exact point along a controlled difficulty axis at which task accuracy collapses. The benchmark is built around four task-family generators, each sweeping one difficulty axis; a grading function bound to each generator; and an independent verification pass that re-derives every ground-truth value from a separate implementation. The full construction pipeline is shown in Appendix A, Figure 4.

2.1 Task Family Generators

Each task family targets a distinct, previously documented failure mode in language models, and each is generated by a pure function that receives a difficulty level and a random seed, and returns a question paired with its own ground-truth answer.

Composition. This family stresses chained, multi-step reasoning. Each item presents either a chain of relational facts (*A's friend is B, B's friend is C*) or a chain of arithmetic operations applied to a starting number, and asks the model to resolve the final value after following every link in the chain. The difficulty axis is the number of hops, swept from one to eight. The correct answer is simply the last element the generator placed in the chain, so no separate solving step is needed to obtain ground truth.

Constraints. This family stresses simultaneous instruction-following. Each item asks the model to produce a single sentence that satisfies several independent, machine-checkable rules at once, such as a word-count range, a required or forbidden word, or a required starting letter. The difficulty axis is the number of simultaneous constraints, swept from one to ten. Grading applies every constraint's check function to the raw response; a response passes only if every check passes. Constraint combinations that are mutually contradictory by construction are rejected during generation, so every item is solvable in principle.

Entity tracking. This family stresses the maintenance of an internal world state across a sequence of updates. The generator simulates a small set of containers and items as an ordinary data structure, applies a sequence of move and swap operations directly to that structure, and only afterward renders the operations as English sentences. The difficulty axis is the number of state-changing operations, swept from one to ten. Because the narrative text is generated from the already-updated data structure rather than the reverse, the ground truth cannot drift from the story.

Counting. This family stresses character-level precision, a capability known to be constrained by subword tokenization rather than by reasoning depth. Items ask the model to count letter occurrences, evaluate a simple property across a list of words, or identify a character at a given position in a string. The difficulty axis is text length, swept from two to twenty. Ground truth is computed with ordinary string operations, which serve as an unambiguous reference.

2.2 Verification Without Human Annotation

Every generator computes its own ground truth as a byproduct of constructing the question, so the benchmark requires no human labeling step at any point. To confirm this design produces correct labels, we built a

second, independently written verification script that re-derives the expected answer for every item using a separate code path: for composition items, it re-runs the arithmetic or re-reads the relational chain; for entity tracking, it re-parses the rendered story text and re-simulates the operations from scratch; for counting, it recomputes each count directly from the underlying string. Across all 2,800 items and their corresponding 45,872 graded responses, this independent check found **zero mismatches**, which we take as strong evidence that benchmark errors reported in Section 4 reflect genuine model failures rather than labeling noise.

2.3 Three Unique Properties

Compared with fixed-difficulty or human-annotated small-model benchmarks, SLM-COLLAPSE offers three properties that jointly distinguish it from prior evaluation suites. First, it is **difficulty-traceable**: because each family sweeps exactly one axis, a collapse in accuracy can be attributed to a specific, named mechanism, rather than to an unspecified blend of item difficulties, which is the norm in mixed benchmarks. Second, it is **regenerable on demand**: because every item is produced by a seeded, deterministic generator rather than sourced from a fixed pool of human-written questions, a completely fresh and previously unseen test instance can be produced at any time, directly addressing the risk that benchmark items leak into future training corpora. Third, it is **fully automatic end to end**: unlike benchmarks that rely on a language model as a judge, or on crowd-sourced human ratings, every grading decision in SLM-COLLAPSE is a short, auditable Python function, which keeps grading cost near zero and lets the benchmark scale to as many models as available compute allows.

3 Experimental Setup

3.1 Models and Evaluation Protocol

We evaluate fourteen open-weight instruction-tuned models between 0.5 and 4 billion parameters, spanning six model families: Llama 3.2 (1B, 3B), Gemma 2 (2B) and Gemma 3 (1B, 4B), Qwen 2.5 (0.5B, 1.5B, 3B), Phi-3 Mini, SmolLM2 (1.7B), TinyLlama (1.1B), StableLM 2 (1.6B), DeepSeek-R1-Distill (1.5B), and Granite 3.1 MoE (1B). All models were run locally through Ollama on consumer CPU hardware, with temperature fixed to zero for deterministic outputs. Justifying this evaluation is not trivial: no existing public benchmark, to our knowledge, sweeps a controlled difficulty axis across this specific panel of sub-4B models on consumer hardware, so the comparison itself is a new data point, not a replication of an established result.

Every generated item specifies a tight token budget for its own answer, ranging from four tokens for a single name or number to forty tokens for a full constrained sentence. This budget is deliberate: it forces a model to state its answer directly rather than to reason aloud before answering, which is consistent with how these models are used in practice, where a terse, parseable output is often required. Responses are graded automatically using one of three deterministic functions: exact numeric match after extracting the first integer in the response, normalized exact string match, or a full pass over every constraint's check function. We report accuracy with 95% confidence intervals computed by bootstrap resampling with 3,000 resamples per model. We additionally record wall-clock latency per item, allowing the speed–accuracy analysis in Section 4.3.

3.2 Evaluation Metrics

Our primary metric is per-item binary accuracy, aggregated by task family and difficulty level to produce a collapse curve. To avoid redundant reporting, we do not additionally report partial-credit or soft-match scores, since our grading functions already tolerate minor formatting variation such as surrounding whitespace or punctuation. We additionally report two secondary, diagnostic metrics: the failure-mode classification

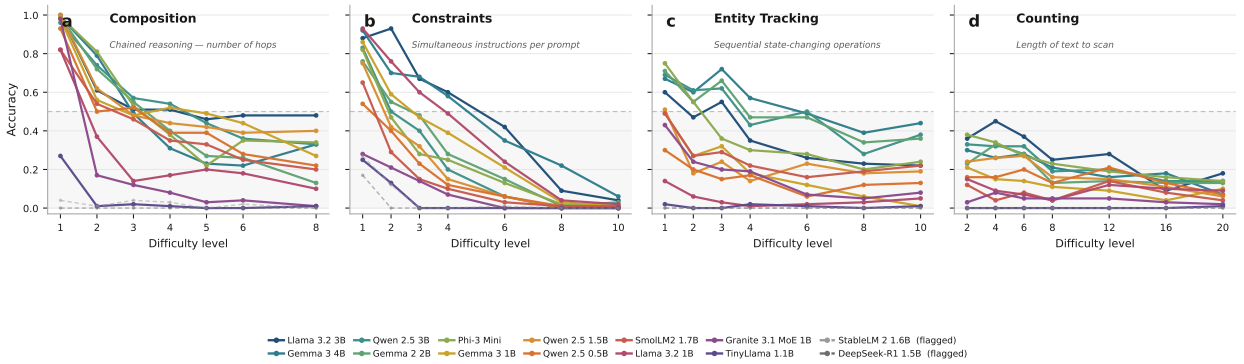

Figure 1: Accuracy collapses as task difficulty increases, at different rates per family and model. Each panel sweeps one difficulty axis for one task family; each line is one model. Composition accuracy plateaus after roughly three hops rather than continuing to decline, while constraint compliance declines steadily with no plateau. Dashed lines mark the two flagged models (Section 5.1), whose failures are dominated by response formatting rather than task difficulty.

described in Section 4.2, which separates incorrect responses into distinct mechanisms rather than collapsing them into a single error category, and per-item wall-clock latency, used in Section 4.3.

4 Results

4.1 Accuracy Collapses Sharply and Predictably With Difficulty

Figure 1 shows accuracy as a function of difficulty level for all fourteen models across all four task families. Every model, without exception, shows a clear downward trend as difficulty increases, but the shape and steepness of that decline differs substantially by family. On composition, the strongest model in our panel, Llama 3.2 3B, drops from perfect accuracy at one hop to 61% at two hops and 51% at three hops, after which accuracy plateaus rather than continuing to fall. This plateau, rather than a continued decline, suggests a floor set by task ambiguity or grading strictness rather than a continued loss of capability. On the constraints family, average compliance across all fourteen models falls from 64% at one constraint to 43% at two constraints, 32% at three, and below 5% by eight simultaneous constraints, a steady and near-universal decline that does not plateau. Counting is the hardest family overall, with an average accuracy of only 13.1% across all models and all difficulty levels, confirming that character-level precision remains a genuine weakness independent of chain length or instruction count. Appendix B, Figure 5, reports the full within-family spread of these per-difficulty-level accuracies as a single distribution per family, which shows that composition and constraints together span nearly the entire zero-to-one accuracy range, while counting stays compressed near the bottom regardless of model or difficulty.

Table 1 reports the same information in numeric form, alongside the point at which each model's accuracy first drops below 50% for each family, giving a compact, citable summary of Figure 1 and Figure 3B together.

4.2 Model Ranking Depends Heavily on Task Family, and Failures Are Not Uniform

Figure 2A reports the overall leaderboard with bootstrap 95% confidence intervals. Llama 3.2 3B and Gemma 3 4B lead the overall ranking at 44.1% and 43.6% accuracy respectively, with non-overlapping confidence intervals separating them clearly from the middle of the pack. However, the per-family breakdown in Table 1 shows that no single model dominates every family: Gemma 3 4B leads entity tracking while trailing Llama

	Mean accuracy					Difficulty level at 50% collapse			
Model	Comp.	Constr.	Entity	Count	**Overall**	Comp.	Constr.	Entity	Count
Llama 3.2 3B	**0.579**	0.519	0.383	**0.283**	**0.441**	5	6	2	2
Gemma 3 4B	0.480	0.501	**0.554**	0.209	0.436	3	6	**6**	2
Qwen 2.5 3B	0.563	0.286	0.501	0.234	0.396	5	3	4	2
Gemma 2 2B	0.474	0.323	0.509	0.196	0.375	4	3	4	2
Phi-3 Mini	0.517	0.281	0.383	0.244	0.356	4	2	3	2
Gemma 3 1B	0.534	0.366	0.207	0.120	0.307	3	3	1	2
Qwen 2.5 1.5B	0.536	0.243	0.239	0.179	0.299	3	2	2	2
Qwen 2.5 0.5B	0.461	0.194	0.161	0.151	0.242	4	2	1	2
SmolLM2 1.7B	0.421	0.177	0.263	0.077	0.235	3	2	1	2
Llama 3.2 1B	0.283	**0.440**	0.049	0.094	0.216	2	**4**	1	2
Granite 3.1 MoE 1B	0.204	0.100	0.180	0.044	0.132	2	1	1	2
TinyLlama 1.1B †	0.046	0.054	0.009	0.001	0.028	1	1	1	2
StableLM 2 1.6B †	0.020	0.057	0.000	0.000	0.019	1	1	1	2
DeepSeek-R1 1.5B †	0.000	0.024	0.000	0.000	0.006	1	1	1	2

Table 1: Accuracy by task family and difficulty collapse point, all fourteen models. Left block: mean accuracy per family and overall, sorted by overall accuracy. Right block: the first difficulty level at which accuracy for that family drops below 50% (**bold** marks the single best, i.e. latest-collapsing or highest-accuracy, value in each column). Rows below the rule are the two flagged models (Section 5.1).

3.2 3B on composition, and Llama 3.2 1B, despite a modest overall rank, achieves the second-highest constraints score in the panel, ahead of several larger models.

Figure 2B decomposes every model's incorrect responses, in the same model order as panel A, into four mechanisms: a clean but factually wrong short answer, a verbose response that eventually states a wrong answer, a response that opens with filler text and never reaches a parseable answer within the token budget, or an entirely empty output. Errors are not a single, undifferentiated category: some models fail by answering cleanly but incorrectly, while others rarely reach a parseable answer at all. TinyLlama opens with filler text and never reaches a parseable answer in a large share of its incorrect responses, a failure mode nearly absent in stronger models, indicating that a meaningful share of its apparent incapability is a formatting failure rather than a knowledge gap. StableLM 2 1.6B shows an even more extreme pattern: the overwhelming majority of its incorrect responses are verbose and wrong, and Appendix C, Table 2, shows this model's wrong answers run 12.0 words longer on average than its correct ones, the largest gap of any model in the panel, indicating the model rarely commits to a terse, checkable answer even when it appears to know the correct one.

4.3 Accuracy and Speed Trade Off, and Instruction-Following Collapses Before Reasoning Does

Table 1 and Figure 2 rank models purely by accuracy, but a practitioner deploying a model on consumer hardware cares about both accuracy and latency at once. Figure 3A plots overall accuracy against average per-item latency for all fourteen models. The two axes are only loosely correlated: Qwen 2.5 0.5B and SmolLM2 1.7B sit close to Llama 3.2 3B in accuracy-per-second despite the fourfold difference in raw accuracy, because they answer roughly twice as fast. Gemma 3 4B, the second-most accurate model overall, is also the slowest model tested, at nearly double the latency of the next-slowest model, which moves it off the efficient frontier entirely: every point on the dashed frontier line in Figure 3A is either faster or more accurate than Gemma 3 4B, never both worse. This result suggests that the accuracy-only leaderboard in

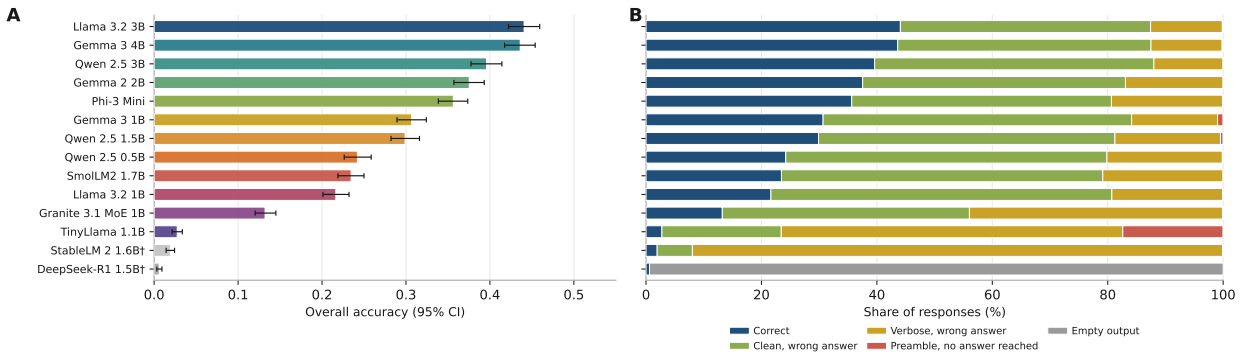

Figure 2: Leaderboard and failure-mode composition, all fourteen models. A, Overall accuracy with bootstrap 95% confidence intervals; non-overlapping intervals separate the top two models from the rest of the panel. **B**, The same models, same order, decomposed into four failure mechanisms. Models with a large share of preamble or verbose-wrong responses fail primarily at concision rather than at the underlying task, a distinction invisible in a single aggregate accuracy score. † marks the two flagged models (Section 5.1).

Table 1 answers a different question than the one most deployment decisions actually require, and that speed should be reported alongside accuracy as a first-class evaluation axis for small models, not treated as an implementation detail.

Figure 3B isolates the constraints family in detail. Every model in the panel, including the strongest overall performers, falls below 50% compliance once a prompt contains more than two or three simultaneous constraints, and every model falls below 10% compliance by eight constraints. This decline is steeper and more uniform across models than the composition decline in Figure 1, where several models plateau rather than continuing to fall. This asymmetry suggests that instruction-following capacity under load is a more universal and earlier bottleneck for small models than multi-step reasoning depth, a distinction that a benchmark reporting only a single mixed-difficulty score would not reveal.

5 Discussion

5.1 Why Do Two Models Fail Almost Uniformly?

DeepSeek-R1-Distill 1.5B and StableLM 2 1.6B score near zero across nearly every task family and difficulty level (Table 1, bottom rows), in sharp contrast to their parameter count, which is comparable to several mid-ranked models in our panel. Our current results show that both models' failures are dominated by the formatting mechanisms described in Section 4.2: DeepSeek-R1-Distill produces no parseable output at all in the overwhelming majority of items, consistent with a reasoning-distilled model that emits internal reasoning tokens before any answer, which our tight token budget truncates before an answer is ever reached. What is still unknown is whether a substantially larger token budget would recover most of this model's apparent capability, since our current experiments do not isolate budget size as an independent variable. A direct next step is to rerun both flagged models with a widened budget as a controlled comparison, which would cleanly separate a genuine capability gap from a harness artifact. We flag rather than exclude these two models throughout this paper, since their inclusion is itself informative: a small model deployed on a token-constrained interface may exhibit exactly this failure pattern in practice.

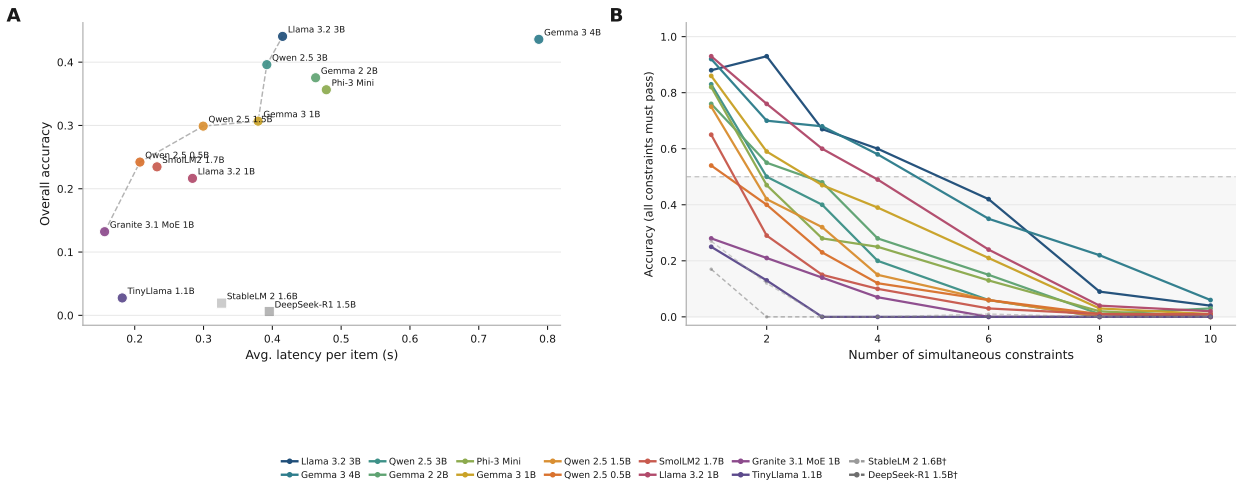

Figure 3: Speed–accuracy tradeoff and instruction-following sensitivity. A, Overall accuracy versus average per-item latency; the dashed line connects the efficient frontier, models for which no other model in the panel is both faster and more accurate. Square markers (†) are the two flagged models. **B**, Accuracy on the constraints family versus the number of simultaneous constraints in the prompt; every model falls below 50% compliance by two to three constraints, regardless of overall rank.

5.2 Does the Composition Plateau Reflect a Capability Ceiling or a Task Ceiling?

Section 4.1 shows composition accuracy plateauing after roughly three hops rather than continuing to decline toward zero. The current work shows this plateau is consistent across most models in the panel, but it does not establish whether the plateau reflects a genuine reasoning ceiling, a ceiling imposed by our specific item design, or partial credit from guessing on a constrained answer space. A useful next step is to widen the hop range beyond eight and to introduce a distractor-rich variant of the composition family, following the direction suggested by prior compositional-reasoning literature, to determine whether the plateau persists or resumes declining at greater depth.

5.3 Can Formatting and Reasoning Failures Be Trained Apart?

Our failure-mode decomposition in Section 4.2 shows that formatting failures and reasoning failures are empirically separable, but it does not yet show whether they are separable through targeted training. What remains unknown is whether a lightweight fine-tuning pass optimized purely for concision, without any change to the model's underlying reasoning weights, would close a meaningful share of the accuracy gap observed in this paper. If it would, this reframes a portion of the small-model capability gap as a solved formatting problem rather than an open reasoning problem, which has direct implications for how compute is allocated in future small-model post-training.

6 Related Work

Prior work on small-model reasoning limitations has established the theoretical grounding this benchmark builds on. Formal analyses of bounded-depth self-attention show that transformers face genuine architectural limits on multi-step compositional reasoning, and empirical work has shown accuracy collapsing as the number of required reasoning hops increases, independent of any single model's scale. Separately, prior work on instruction-following under load has shown that compliance degrades as the number of simultaneous

constraints in a prompt grows, and that constraint position within a prompt affects compliance through primacy and recency effects. Our constraints family is designed in this tradition, though our contribution is to sweep constraint count as a controlled, single-variable axis rather than reporting compliance at one fixed constraint level. Work on mechanistic interpretability of entity and state tracking has shown that language models often retrieve and aggregate relevant information only at the point a query is posed, rather than maintaining an incrementally updated internal state as text is read, a finding our entity-tracking family is designed to stress directly. Finally, work on character-level and counting tasks has traced this weakness to subword tokenization and positional encoding choices rather than to a general reasoning deficit, which motivated our decision to include counting as an architecturally distinct axis, uncorrelated with the other three families.

Our work differs from this prior literature primarily in scope and method. Where most of the cited studies examine a single task family or a narrow band of model sizes, we sweep four independently motivated difficulty axes across fourteen models spanning six architecture families, all under a single, consistent, fully automatic grading protocol, and we additionally report latency alongside accuracy, which prior small-model reasoning studies typically omit. Where prior small-model benchmarks typically rely on a static, human-curated item pool, our benchmark is procedurally generated and independently verified at scale, which is, to our knowledge, a combination not previously applied across this specific breadth of task families and small models simultaneously.

7 Conclusion

Small language models fail predictably, not randomly. Accuracy on chained reasoning collapses within three to five steps, and instruction-following compliance collapses even earlier, before two or three simultaneous constraints. A meaningful share of that collapse is not a reasoning failure but a failure to produce a terse, parseable answer, a distinction invisible to any benchmark that reports a single aggregate score. Accuracy and latency trade off in a way that reorders the practical leaderboard, so the most accurate model in isolation is not always the most efficient choice once speed is taken into account. This suggests that improving small-model usability is partly a training problem distinct from improving raw capability, and that future small-model benchmarks should report collapse curves and speed–accuracy tradeoffs, not accuracy averages alone, if they are to guide real deployment decisions.

Data and Code Availability

The full generator toolchain, the 2,800-item dataset used in this paper, the independent verification script, and the complete evaluation harness are released at the project repository. Because every item is procedurally generated from a seed, a fresh, previously unseen benchmark instance can be produced at any time, which we recommend for any future evaluation to mitigate the risk of training-data contamination on the specific instance released alongside this paper.

A Benchmark Construction Pipeline

Figure 4 shows the full construction pipeline referenced in Section 2: four task-family generators, each producing its own question and ground-truth answer, followed by an independent verification pass that re-derives every ground-truth value from a separate implementation.

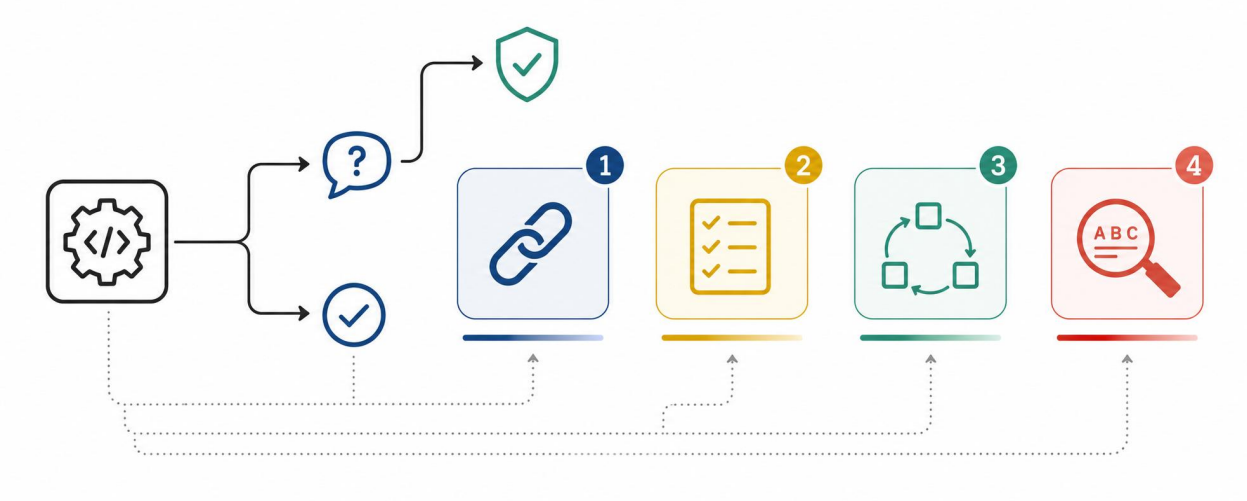

Figure 4: Benchmark construction pipeline. Each of the four task families is produced by a deterministic generator that computes the ground-truth answer as a byproduct of constructing the question text, so no human annotation step exists anywhere in the pipeline. An independently written verification script re-derives every ground-truth value from a separate code path, finding zero mismatches across 45,872 checks.

B Within-Family Accuracy Variability

Figure 5 reports the distribution of per-(model, difficulty-level) accuracy within each task family, referenced in Section 4.1. Each point is one model's accuracy at one difficulty level; the box marks the interquartile range and the line marks the median. Composition and constraints together span nearly the full zero-to-one range, reflecting the sharp, model-dependent collapse already visible in Figure 1. Counting remains compressed near the bottom of the range regardless of model or difficulty level, which is the distributional signature of an architecture-level limitation rather than a difficulty-dependent one: if counting failure were purely a matter of difficulty, its box would span a wider range the way composition's does.

C Full Wordiness Gap by Model

Table 2 reports the average response length, in words, for correct and incorrect responses separately, for every model, referenced in Section 4.2. A positive gap means the model's wrong answers are longer on average than its correct ones, consistent with padding, hedging, or rambling specifically in cases where the model does not know the answer.

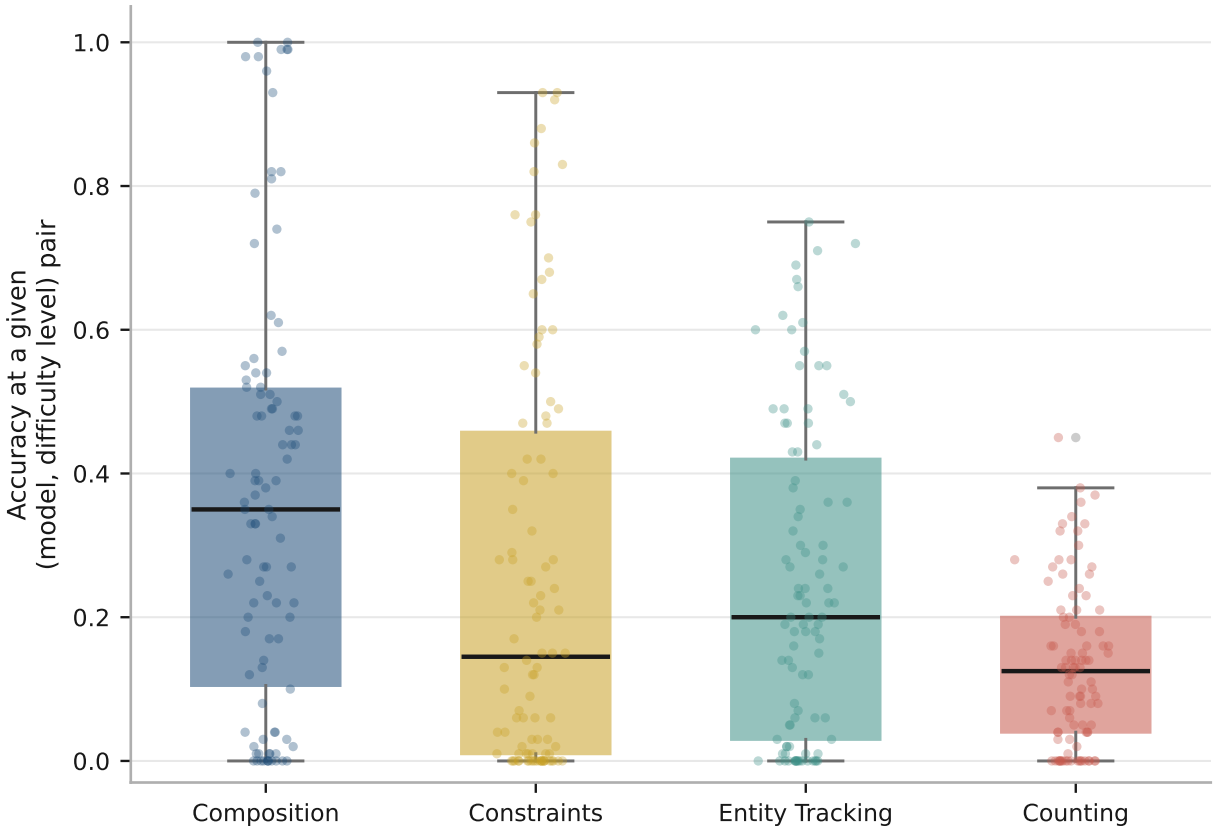

Figure 5: Distribution of per-(model, difficulty-level) accuracy, by task family. Each point is one model's accuracy at one difficulty level within that family. Composition and constraints show wide spread; counting stays compressed near the bottom of the accuracy range across nearly every model and difficulty level.

Model	Words (correct)	Words (wrong)	Gap
Qwen 2.5 0.5B	3.55	4.58	**+1.03**
Phi-3 Mini	3.61	4.59	+0.98
Qwen 2.5 1.5B	3.42	4.02	+0.60
SmolLM2 1.7B	3.50	4.00	+0.50
Gemma 2 2B	3.81	3.84	+0.02
Qwen 2.5 3B	2.80	2.80	−0.00
Gemma 3 1B	3.79	3.66	−0.13
Granite 3.1 MoE 1B	4.96	4.68	−0.28
Gemma 3 4B	4.49	3.86	−0.63
Llama 3.2 3B	4.73	3.69	−1.04
TinyLlama 1.1B †	9.00	6.57	−2.43
Llama 3.2 1B	6.76	3.23	−3.53
StableLM 2 1.6B †	22.63	10.60	**−12.03**

Table 2: Average response length by correctness, all models. Gap = words(wrong) − words(correct). DeepSeek-R1 1.5B is omitted: its responses are empty in the overwhelming majority of items (Section 5.1), making the word-count comparison degenerate. † marks the two flagged models.