LLMs Can Follow Instructions, But Not Many at Once: Multiplicative Collapse in Compositional Constraint Satisfaction

Anonymous authors
Paper under double-blind review

Abstract

Large language models follow a single explicit instruction reliably, but deployed prompts rarely contain just one. We ask what happens as instructions accumulate: how fast the chance of satisfying all of them declines, what sets the rate, and whether effort at inference time can slow it. We introduce *Constraint Saturation Evaluation* (CSE), a procedurally generated benchmark that varies the number of simultaneous constraints k from 1 to 12 and checks every constraint with a deterministic verifier, with no LLM judge at any stage. Across 19 models, 36 constraint types, and 472,019 checks, per-constraint pass rates decline gently while joint success collapses: at k=8, models pass 45% of individual constraints but satisfy all eight on only 9.6% of prompts. No pair of constraints interferes: failures of different constraints are nearly uncorrelated, so the risk of failure compounds across constraints, and no choice or arrangement of constraints avoids the collapse. The per-constraint rates themselves follow a clear hierarchy: constraints that must be tracked across the whole output lose their single-constraint accuracy about twice as fast as lexical ones, and constraints settled by a single decision degrade least. When constraints cannot all be met, models keep the ones that hold up best under load. Planning before generation does not delay the collapse; self-correction and best-of-5 sampling delay it by one or two constraints. Only four of the 19 models satisfy every constraint on a majority of prompts at k=5, and 12 lose that majority at three constraints or fewer. We will release all verifiers, probes, model outputs, and code for reproducibility.

1 Introduction

Consider the system prompt of a customer-support assistant. It asks for a reply under 120 words that addresses the customer by name, quotes the order number, promises no refund, contains no links, ends with a question, and is returned inside a JSON object. Each instruction is easy, and a current model follows any one of them almost every time. The reply is usable only if it follows all of them. In our measurements, the median model already fails on most prompts that carry three instructions of this kind. Instructions collected from 50 real agentic applications average 11.9 constraints each (Qi et al., 2025a).

How often that happens depends on how failures combine. Under one account, each instruction fails on its own: if every instruction fails 10% of the time, a reply that must satisfy ten of them succeeds with probability $0.9^{10} \approx 35\%$. Under the other, instructions *interfere*: some pairs compete for the same resources, so joint success depends on which instructions appear together and can land far above or below that product. The two accounts call for different fixes. Interference suggests choosing and ordering instructions carefully; independence says the only lever is making each instruction more reliable.

Existing benchmarks cannot tell the two apart. Most evaluate one to three constraints at a time (Zhou et al., 2023), use fixed compositions (Yao et al., 2024), or group prompts into a few coarse levels of constraint count (Guo et al., 2025; Ye et al., 2026a). FollowBench (Jiang et al., 2024) adds one constraint at a time but stops at five, and relies on an LLM judge that must itself track many

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

constraints at once, the very ability under test. What is missing is the basic curve: how joint success falls as constraints are added, and why.

We measure that curve with *Constraint Saturation Evaluation* (CSE). CSE composes prompts from 36 constraint types in 8 dimensions, ranging from avoiding a letter to producing a scene graph, at every k from 1 to 12, and checks every constraint with a deterministic verifier. We evaluate 19 models from 8 families on 4,527 probes (472,019 constraint checks) and ask four questions.

- **How fast does joint success fall?** The average per-constraint pass rate loses about 7% of its value with each added constraint and is predictable from the first eight values of k to within 0.3pp. Joint success falls much faster, from 75.6% of probes at k=1 to 9.6% at k=8. Four models keep a majority of probes fully correct at five constraints; 12 of 19 lose it by three (Section 4.1).
- **Do constraints interfere?** Barely. Pairwise failure correlations average φ=+0.085, and none is meaningfully negative. The coupling that exists comes from shared output features: when a model writes the wrong number of sentences, every sentence-based constraint fails together. Joint success therefore decays like a product of per-constraint rates; the weak positive coupling keeps it somewhat above that product, most visibly for the strongest models (Section 4.3).
- **Which constraints degrade fastest?** Structural constraints lose their single-constraint accuracy 1.97× faster than lexical ones, and constraints settled by one decision, such as including a word or emitting JSON, degrade least (Section 4.4). When constraints cannot all be met, models keep the same ones that survive load (Section 4.6).
- **Can inference-time effort help?** Planning before writing leaves the breakdown point unchanged. Self-correction and best-of-5 sampling move it by one or two constraints; best-of-5 recovers only about two-fifths of what independent retries would predict, because a model tends to fail the same prompt the same way on every draw (Section 5).

2 Related Work

Multi-constraint instruction following. IFEval (Zhou et al., 2023) pioneered deterministic verification with 25 constraint types but evaluates at $k \leq 3$. COLLIE (Yao et al., 2024) generates constraints from a formal grammar and shows that composition increases difficulty, using fixed composition levels rather than a varying k. FollowBench (Jiang et al., 2024) adds one constraint per level up to k=5 and tracks hard and soft satisfaction, the closest existing per-k analysis, but uses GPT-4 to judge open-ended constraints and does not model the shape of the decline. RECAST (Guo et al., 2025) raises constraint density to over a dozen per instance and trains on verifiable constraints, but reports only four nested tiers (5, 10, 15, and all constraints), and MulDimIF (Ye et al., 2026a) groups code-checked prompts into four levels spanning one to eight constraints. InfoBench (Qin et al., 2024) finds that models fail number and linguistic constraints preferentially, a hierarchy we extend across k, and in tool use CCTU (Ye et al., 2026b) finds that no model exceeds 20% task completion under strict multi-constraint adherence. Other work varies the form of a composition. ComplexBench (Wen et al., 2024) combines constraints through chains, selections, and nesting as well as conjunction; MulDimIF (Ye et al., 2026a) varies whether constraints are stated as examples, as a list, or woven into the request; Zeng et al. (2025) show that the order in which constraints are listed affects compliance; and Sakai et al. (2025) test whether models respect a required order of concepts.

Compositional generalization. Outside text, ConceptMix (Wu et al., 2024) reports that DALL-E 3, the strongest of the eight text-to-image models it tests, drops from 83% to 8% accuracy as the number of visual concepts added to an object grows from 1 to 7. In reasoning, Dziri et al. (2023) observe accuracy falling to near zero as composition depth grows and argue theoretically that independent per-step errors make it decay exponentially, and Press et al. (2023) measure a compositionality gap of about 40% that does not close as GPT-3 grows from 1B to 175B parameters; see also Lake & Baroni (2018). In long-horizon execution, small differences in per-step accuracy compound into large differences in the length of task a model can complete (Sinha et al., 2026), and performance drops when a task arrives over several conversational turns (Laban et al., 2026).

Improving multi-constraint following. A separate line of work trains models to follow complex instructions: curriculum tuning on constraints written by a stronger model (Sun et al., 2024), execution feedback from code verifiers (Dong et al., 2025), training data mined from real prompts or refined automatically (An et al., 2025; Liu et al., 2025; Cheng et al., 2025), constraint back-translation (Qi

et al., 2025b), and reinforcement learning with verifiable rewards (Lambert et al., 2025; Pyatkin et al., 2025), where Pyatkin et al. (2025) find that models overfit to the constraint types seen in training. At inference time, DeCRIM (Palmeira Ferraz et al., 2024) splits an instruction into its constraints, checks the response against each, and revises it to fix those that fail; attention steering (Lv et al., 2026) reduces interference from inactive sub-tasks. These methods report gains at the fixed composition levels of existing benchmarks. Section C covers these lines of work in more detail, along with mechanistic accounts and phase transitions in random constraint satisfaction.

3 Constraint Saturation Evaluation

3.1 Constraints and metrics

CSE has 36 constraint types in 8 dimensions (Table 3; full specifications in Table 7). *Structural* constraints fix counts of words, sentences, paragraphs, characters, or lines. *Ordering* constraints fix how sentences are arranged: lengths that rise, alternate, or grow like a Fibonacci sequence, or first words in alphabetical order. *Word-level* constraints restrict every word, for instance to be unique or within a length range. *Numerical* constraints require numbers (integers summing to a target, correct equations) or forbid digits. *Format* constraints require a markdown table, a bullet list, or code blocks. *Relational* constraints require a structured answer to a small reasoning problem, such as a JSON scene graph or a solved logic grid. *Lexical* constraints govern which letters and words appear (avoid a letter, include given words, spell an acrostic), and *meta* constraints refer to the response itself, such as a sentence that correctly states the response's word count. Every verifier is deterministic and returns both a pass/fail judgment and a continuous score $s \in [0, 1]$ for partial compliance: 48 words against a target of 50 scores about 0.96 but fails. The 36 types were retained from 38 candidates (Section L).

For m probes, each with k constraints and binary outcomes $s_i^j \in \{0, 1\}$, following Jiang et al. (2024) we report

$$\text{sCSR} = \frac{1}{m} \sum_{i=1}^{m} \prod_{j=1}^{k} s_i^j, \tag{1}$$

the fraction of probes on which *every* constraint holds, and

$$\text{mCSR} = \frac{1}{mk} \sum_{i=1}^{m} \sum_{j=1}^{k} s_i^j, \tag{2}$$

the average per-constraint pass rate. The two coincide at $k{=}1$ and separate as k grows. sCSR is what a deployed system needs; mCSR tells us whether the per-constraint rates, which sCSR compounds, are themselves the problem. For each constraint i we also report its mean continuous score $\bar{s}_i$ and the *comprehension–maintenance gap*

$$\Delta_i = \bar{s}_i - \text{mCSR}_i. \tag{3}$$

A large Δ means models come close to the constraint, showing that they understood it, without holding it exactly. Finally, the *breakdown point* k^* of a model is the smallest k at which its sCSR falls below 50%, and k^*_{int} locates the same crossing by linear interpolation between neighboring values of k.

3.2 Probes

Probes are generated procedurally (Algorithm 1). For each k, the composer draws a k-subset of constraint types, rejects subsets that cannot be jointly satisfied, samples parameters from fixed calibrated ranges, shuffles the order in which the constraints are presented, and assembles the prompt around a topic. Rejected draws are replaced rather than dropped, so every k receives its full quota. At $k{\leq}5$ all subsets are enumerated and shuffled; at $k{\geq}6$ they are sampled uniformly with a cap on reuse. The compatibility checker combines 23 unconditional pair blocks, such as N2 (no digits) with N1 (numbers summing to a target), with parameter-dependent arithmetic checks and propagation across constraints (Section J). Rejection rates rise from about 32% at $k{=}4$ to 98% at $k{=}12$ (Table 10), so surviving high-k probes over-represent compatible constraint types; Section 4.5 tests whether this matters. Parameter ranges are held fixed across k (e.g., word-count targets of 40–60), so each constraint asks for the same thing at every k. We fit models on $k{=}1{-}8$ and hold out $k{=}9{-}12$ for validation (Section Q).

Impossible probes. CSE also contains 444 probes whose constraints cannot all be satisfied, which reveal what a model gives up when it must give something up. They are of two kinds. The 55 *designed* probes contain a pair with a proof of unsatisfiability, such as L1 (avoid the letter 'e') with L2 (include "telephone"). The 389 *inferred* probes were marked impossible during auditing because no model in the original panel satisfied them. That is a statement about those models rather than a proof. All impossible probes are excluded from Table 1 and from every satisfiable-probe statistic (Section U).

3.3 Extraction and verification

Models see a topic and all k constraint instructions in natural language, with no decomposition hints or ordering cues. Responses are wrapped in `<answer>` tags, and only tagged content is scored. The extractor takes the *last* tag pair, so mid-response revisions count; accepts an opening tag without a closing one, so text written before truncation still counts; and falls back to the full response when tags are missing. Markdown emphasis and headings are stripped before verification, while structural formatting (bullets, tables, code blocks) is kept. A probe passes only if all k constraints pass.

Verifier audit. To check the verifiers, an LLM judge reviewed near-miss failures (continuous score ≥ 0.80) among the 223,643 failed checks of the original 15-model panel, in one audit pass per constraint category. The judge audits verifiers and never scores a response. The audit surfaced four parsing issues that cut across constraints. The extractor matched the first rather than the last answer tag when a model quoted the instruction; sentence splitting treated JSON, tables, and bullet lists as prose, producing false ordering failures; and the R3 parser read several JSON objects as one, producing 1,051 false failures (19.2% of R3 failures). These three were fixed and every response rescored. The fourth remains open: about a quarter of the sampled near-miss R2 failures were correct orderings written in a list format the parser does not read, so R2 pass rates are underestimates (Section R). A second audit, of *passing* responses, found verifier gaps that let a response satisfy a check without meeting the constraint as intended (Section A). All reported results use the closed verifiers: the two sentences of an S6 mirror pair must differ, F3 counts only non-empty code blocks, and W1 identifies words with a regular expression rather than by splitting on whitespace.

4 Experiments

4.1 Joint success collapses

How fast does joint success fall? All 19 models follow the same pattern (Figure 1a). Averaged over models, sCSR falls from 75.6% at $k{=}1$ to 9.6% at $k{=}8$ and 2.7% at $k{=}12$, while mCSR over the same range falls only to 45.3% and 33.0%. At $k{=}10$, 11 of the 19 models solve no probe at all; the five strongest still solve 8–25%, and at $k{=}12$ the best of them, GPT-6 Astra and GPT-5.5, solve 12.7%. Breakdown points range from $k^*{=}1$ (Kimi K2.6) to $k^*{=}7$ (GPT-6 Astra and GPT-5.5), with a median interpolated crossing of 2.4 constraints (Table 1).

What does a failure look like? For strong models, many failures are near misses. Of the probes Claude 5 Opus fails, 57% miss exactly one constraint, as do 44–48% of those failed by GPT-6 Astra, GPT-5.6 Sol, and GPT-5.5; the three fastest-decaying models miss exactly one on fewer than 9% of their failures. On one $k{=}12$ probe, Claude 5 Opus satisfied eleven constraints, but the last words of its five sentences began with D, R, E, D, M where they had to spell DREAM. sCSR counts that response as a failure, as a system that needs every instruction followed would.

The two metrics rank models differently. Claude 5 Opus has the highest per-constraint accuracy in the panel (82.6% mCSR, 91.9% Score) but ranks fourth on sCSR. On the 4,079 probes it shares with GPT-5.5, it satisfies 3.2pp more constraints (95% CI $[2.2, 4.2]$) and yet solves 10.8pp fewer probes ($[-12.3, -9.1]$). The reason is how each metric weights k. mCSR counts constraints, so a 12-constraint probe weighs twelve times as much as a single-constraint one; sCSR counts every probe once. The probe set is also uneven across k: 54% of the 4,083 probes have $k{\leq}4$ (29% have $k{=}3$), while 48% of the constraint checks come from probes with $k{\geq}8$. sCSR is therefore mostly a low-k measure and mCSR mostly a high-k one; the per-k curves in Table 4 avoid both weightings. Claude 5 Opus is the most reliable model at high k, where almost nobody solves whole probes (70.6% mCSR at $k{=}12$, against 52.9% for GPT-5.5), and GPT-5.5 is the most reliable at low k, where whole

Table 1: Results on the 4,083 satisfiable probes, grouped by output budget and ranked by sCSR within each group. sCSR: probes with every constraint satisfied (Equation (1)); mCSR: constraints satisfied (Equation (2)); Score: mean continuous score; k^*: breakdown point, the smallest k with sCSR below 50%; k^*_{int}: the same crossing interpolated between neighboring k (undefined for Kimi K2.6, already below 50% at k=1); Tag: answer-tag compliance. †Provider-default temperature; all other models run at temperature 0. ‡Control run of GPT-6 Astra at the smaller budget. Shading: $k^* \geq 5$ (blue), $k^* = 1$ (orange).

Model	sCSR	mCSR	Score	k^*	k^*_{int}	Tag
16,384-token budget						
GPT-6 Astra†	**64.3**	79.1	86.5	**7**	**7.0**	●
GPT-5.6 Sol†	56.3	76.8	86.3	6	5.5	●
Claude 5 Opus†	52.3	**82.6**	**91.9**	6	5.2	●
Claude 4.7 Opus†	46.2	61.5	77.0	5	4.1	◕
4,096-token budget						
GPT-6 Astra†‡	64.3	79.4	86.7	7	6.9	●
GPT-5.5†	63.1	79.4	87.1	7	6.1	●
Gemini 3.1 Pro	39.2	42.0	58.7	4	3.4	◑
Kimi K3	36.9	32.1	50.8	4	3.4	●
Claude 4.6 Opus	31.5	55.5	76.6	3	2.8	◕
Gemini 3.1 Flash-Lite	24.1	56.6	79.4	3	2.4	●
GPT-5.4 Pro	23.2	35.6	58.5	3	2.2	●
DeepSeek V4 Pro	22.4	22.4	47.1	3	2.4	◕
GPT-5.2	18.8	34.0	57.8	2	1.7	◕
Claude 4.5 Sonnet	17.3	49.4	75.8	2	1.8	●
Llama 4 Maverick	15.0	45.4	71.5	2	1.6	●
Qwen 3 235B Inst.	12.6	45.7	71.0	2	1.3	●
Llama 3.1 405B	12.4	46.4	74.6	2	1.4	●
Llama 3.1 70B	11.6	44.6	71.6	2	1.2	●
Grok 4.1	11.3	42.6	70.8	2	1.2	●
Kimi K2.6	7.2	15.5	35.3	1	–	◕

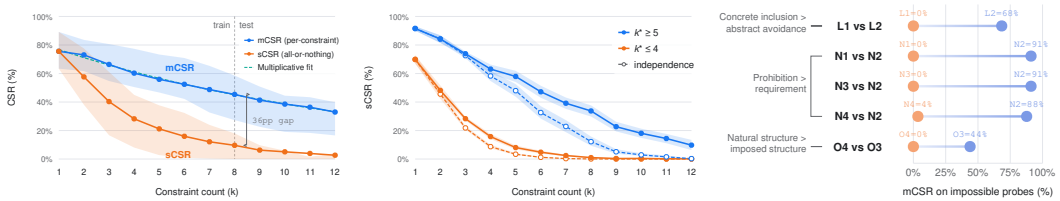
probes are won (98.4% against 87.8% at k=1; Table 4). The same effect sets the gap between mCSR and sCSR, which ranges from -4.8pp (Kimi K3, strong at k=1 but quick to decay) to $+34.0$pp (Llama 3.1 405B).

4.2 Per-constraint accuracy decays geometrically

Is the decline predictable? If each added constraint lowers the per-constraint pass rate by a constant factor, then $\text{mCSR}(k) = a \cdot r^{k-1}$. We fit this model on k=1−8 and test it on the held-out k=9−12. The aggregate fit is $76.8\% \times 0.927^{k-1}$, with a held-out mean absolute error of 0.33pp, and adding a floor term does not improve it (0.31pp). Each added constraint therefore costs about 7% of the current per-constraint pass rate, and sCSR, which compounds these rates, follows $e^{-0.306k}$ closely. Put differently, the aggregate per-constraint pass rate halves about every nine constraints ($\ln 0.5 / \ln 0.927 \approx 9.1$); at the two ends of the panel, the fitted rates imply about 57 constraints for Claude 5 Opus (r=0.988) and about two for DeepSeek V4 Pro (r=0.721). The decay factor r differs widely between models: 13 decay slowly (r>0.93), three at an intermediate rate (GPT-5.2, GPT-5.4 Pro, Gemini 3.1 Pro, $r \approx 0.86$), and three quickly (DeepSeek V4 Pro, Kimi K2.6, Kimi K3, r=0.72−0.78). The fast decayers do not reach zero: from k=8 on they hold between 11% and 17% mCSR, and five constraints (L2, L5, L6, O4, R3) supply 72% of their passed checks there. Extrapolating a single model is less reliable than extrapolating the aggregate (held-out error 6.7pp): eight models, including all three fast decayers, sit above their predicted k=12 value, while the strongest fall below it, by 19pp for GPT-5.5 and 26pp for Claude 4.7 Opus (Section G).

4.3 Constraints barely interfere

Do constraints interfere? If specific pairs of constraints interfered, their failures would be correlated: negatively if satisfying one came at the cost of the other, positively if they tended to break together. We compute the φ coefficient, the correlation between two binary variables, between the



(a) Decay with k (b) Joint success vs. independence (c) Choices under conflict

Figure 1: **(a)** Per-constraint accuracy declines gently while joint success collapses. Blue: mCSR averaged over models; orange: sCSR; teal (dashed): the multiplicative fit $76.8\% \times 0.927^{k-1}$, fit on k=1−8 and tested on k=9−12, right of the vertical line (held-out error 0.33pp). **(b)** Observed sCSR (solid) against the independence prediction (dashed), the product of each model's per-constraint pass rates at that k over the probe's constraints, averaged over the five models with $k^* \geq 5$ (blue) and the 14 with $k^* \leq 4$ (orange). The curves coincide at k=1 by construction and separate as failures cluster on the same responses. Bands: in (a), the middle half of the 19 models at each k; in (b), 95% intervals from 2,000 resamples of the probes. **(c)** What models keep when two constraints conflict (designed impossible probes; pass rates over 19 models; periwinkle: kept, peach: dropped): inclusion over avoidance (L2 68% in the probe type shown and 79–97% in the other four, L1 0–4%), prohibition over requirement (N2 88–91%; N1, N3 and N4 0–4%), and the ordering closer to natural prose over imposed structure (O3 44%, O4 0%; Section U.1).

failure indicators of every pair of constraints that co-occur often enough (601 of the 630 pairs). To keep the fact that everything fails more at high k from inducing correlation, each φ is computed within a single k and a single model, and then summarized by the median across models. The correlations are weak: the mean is φ=+0.085, 258 pairs (43%) lie within ±0.05, and none falls below −0.05 (the minimum is −0.028). No pair of constraints competes, and the positive correlations are small. There is no incompatible pair to avoid, so no selection of constraints can avoid the collapse. The coupling that exists comes from shared output features (Section N).

How close is joint success to the product? If failures were independent, a model's chance of solving a probe would equal the product of its pass rates, at that k, for the probe's constraints. Averaged over models, this prediction matches at low k (57.6% observed against 55.8% predicted at k=2) and falls increasingly short as k grows: 9.6% against 3.3% at k=8, and 2.7% against 0.1% at k=12 (Figure 1b). The shortfall is largest for the five models with $k^* \geq 5$, which at k=8 solve 33.7% of probes against 12.2% predicted; for the other 14 models the two never differ by more than 7pp. Small positive correlations compound across many pairs. A response that breaks one constraint is somewhat more likely to break others, so failures concentrate on fewer responses and more responses pass everything. In these data the product therefore acts as a lower bound on joint success rather than an exact description: it explains why joint success collapses, and the coupling slows the collapse most for the models strong enough to keep some high-k probes intact.

4.4 Which constraints fail first

Do all constraints degrade at the same rate? Near-independence says that failures combine multiplicatively, but not that every constraint loses accuracy at the same rate. For each constraint we compute its *retention*, $\text{ret}(k) = \text{mCSR}(k)/\text{mCSR}(1)$, the fraction of its single-constraint pass rate that survives at k, and fit a linear slope over k. Its *retention half-life* $t_{0.5}$ is the smallest k at which retention falls below 50% (>12 if retention stays above 50% up to k=12). Table 3 (Section E) lists, for each of the 36 constraints, the single-constraint pass rate, mCSR over all k, the comprehension–maintenance gap Δ, the slope, and $t_{0.5}$, with dimensions ordered from fastest to slowest degradation. Structural constraints degrade fastest (slope −0.068), followed by ordering (−0.063), and lexical constraints slowest (−0.035): a ratio of 1.97× (95% CI $[1.81, 2.15]$). L5 (forbidden word) and L1 (lipogram, avoiding one letter) keep 88% and 94% of their single-constraint pass rates at $k \geq 8$. L1 is the informative case, because it is hard (53% at k=1) yet its difficulty does not grow with k. Its failures happen early: averaged over models, 51% of genuine L1 failures occur in the first fifth of the answer, and this position does not shift with k (ρ=−0.07). Whether a model avoids the letter is decided by its opening vocabulary, not by how much else it is tracking (Section H).

What makes a constraint degrade quickly? The fast-degrading constraints must be tracked across the whole response: word and sentence counts, orderings across sentences, the uniqueness of every word. The slow ones are settled by a single decision: include a required word (L2), avoid a forbidden one (L5), return a JSON scene graph (R1). The comprehension–maintenance gap (Equation (3)) measures this difference. It is large when models come close without holding the constraint exactly, as for W1 (all words unique: 78.7% score against 31.4% pass rate), W2 (45pp), and S1 (36pp), and near zero for L2 (2pp) and R1 (0.3pp). Across the 36 constraints, a larger gap goes with faster degradation (Spearman $\rho{=}-0.48$, 95% CI $[-0.71, -0.16]$). Single-constraint difficulty correlates with the slope about as strongly ($\rho{=}+0.55$), and with 36 constraints the two predictors cannot be separated.

4.5 Is the collapse an artifact?

Rejection sampling makes surviving high-k probes over-represent compatible constraint types, so high-k probes might simply contain harder constraints. They do not. Whether a constraint's difficulty is measured by its pass rate over all probes or on single-constraint probes, the average difficulty of a probe's constraints shows no trend with k, and probes built from easy and hard constraints converge to the same sCSR by $k{\approx}8$ (9.9% against 9.3%; Section P). Presentation order is randomized, and a constraint's position in the prompt barely predicts whether it passes (mean $|\rho|{=}0.011$; Section O).

Rankings are stable where the data allow. Resampling probes, k^* stays fixed in at least 95% of resamples for 12 of the 19 models. At the top it does not: GPT-5.5 lands on $k^*{=}7$ in only 56% of resamples, and GPT-6 Astra's lead over GPT-5.5 (1.2pp sCSR, 95% CI $[0.0, 2.5]$) is a tie. GPT-6 Astra's standing does not come from its larger budget, since its 4,096-token run matches its 16,384-token run to within 0.07pp ($[-1.06, 0.89]$). Refitting the decay model on the 15 models that tag at least 90% of their answers gives $77.7\% \times 0.938^{k-1}$, with per-model r still spanning 0.78 to 0.99. Finally, closing the W1 verifier gap changes any model's sCSR by at most 1.1pp and changes no k^* or rank.

4.6 What models give up when they must

Designed contradictions. No model satisfies every constraint of any designed impossible probe, so each probe forces a choice, and the choices follow three rules (Figure 1c). Models keep a required inclusion over an avoidance: when L2 requires "telephone" and L1 forbids 'e', L2 survives 68–97% of the time depending on the probe type and L1 0–4%. They keep a prohibition over a requirement: N2 (no digits) survives 88–91% against at most 3.5% for N1, N3, or N4, which require numbers. And they keep the ordering closer to natural prose: O3 (alternating short and long sentences) survives 44% against 0% for O4 (sentence lengths that grow like a Fibonacci sequence), with 9 of 19 models satisfying neither. Each rule holds on every probe type in aggregate and is reversed by at most one model on any of them, and embedding the L1–L2 contradiction among up to six other constraints leaves it intact (L2 79–84%, L1 at most 3.5%; Section U.1).

Triage on inferred probes. Two models solve some of the 389 inferred probes outright (seven probe–model pairs on six probes, five by Claude 5 Opus and two by GPT-5.6 Sol), confirming that those probes were hard rather than impossible. On the rest, which constraints survive follows the load hierarchy. Across all 36 constraints, the rate at which a constraint is kept on inferred probes correlates with its pass rate at high k ($\rho{=}0.97$) and with its overall mCSR ($\rho{=}0.93$), much more than with its single-constraint pass rate ($\rho{=}0.60$). L5 is kept on 92% of attempts and N2 on 68%; S4, M1, S6, and M2 on 3–6%. When models cannot satisfy everything, they drop constraints in the same order in which load erodes them (Table 12).

5 Can Inference-Time Effort Delay the Collapse?

Independence makes different predictions for different interventions. If the collapse is set by per-constraint reliability rather than by disorganized generation, then asking the model to plan should not move it. Extra attempts should help, but only up to a limit: if failures on a probe were independent across draws, N attempts would succeed with probability $1-(1-p)^N$, and any correlation across draws lowers that ceiling. We test three interventions, each on three models, and report the breakdown point k^* measured on each experiment's own probe subset and budget, so baselines differ from

Table 2: Breakdown point k^* without and with each intervention. Planning is compared against a budget-matched control; self-correction and best-of-5 against a single pass. Effect: change in the per-constraint decay factor r (planning), change in sCSR (self-correction), and peak observed sCSR gain against the gain that independent draws would predict (best-of-5).

Intervention	Model	k^*	Effect
Planning	Claude 4.7 Opus	$6 \to 7$	$\Delta r = -0.017$
	GPT-5.5	$8 \to 7$	$\Delta r = +0.021$
	DeepSeek V4 Pro	$5 \to 5$	$\Delta r = +0.014$
Self-correction	Claude 4.7 Opus	$5 \to 6$	0.0pp
	GPT-5.5	$7 \to 8$	+3.7pp
	DeepSeek V4 Pro	$3 \to 4$	+11.5pp
Best-of-5	GPT-5.2	$2 \to 4$	+22 / +53pp
	Claude 4.6 Opus	$3 \to 5$	+22 / +54pp
	Llama 4 Maverick	$2 \to 3$	+24 / +51pp

Table 1 (Table 2). **Planning** requires a <scratchpad> plan before the answer and is compared, probe by probe, against a control with the same 16,384-token budget and no scaffold (50 probes per k at k=1−8, 25 at k=9−10). **Self-correction** returns the model's first answer with an instruction to check and revise it (temperature 0, 8,192 tokens; 50 probes per k at k=1−10). **Best-of-5** draws five answers at temperature 0.7 and counts a probe as solved if any draw satisfies every constraint (4,096 tokens; 40 probes per k at k=1−8).

Planning does not help; output budget does. Per-constraint decay is unchanged by the scaffold (control $93.0\% \times 0.948^{k-1}$, scaffold $90.2\% \times 0.952^{k-1}$), and the change in k^* is inconsistent in sign, with the strongest model losing a constraint. The null result does not come from models ignoring the instruction, since they produced the plan on 100%, 100%, and 56% of probes. It agrees with reports that chain-of-thought prompting does not improve, and can lower, accuracy on IFEval and ComplexBench (Li et al., 2025). The budget, by contrast, matters on its own: moving DeepSeek V4 Pro from its 4,096-token baseline to the 16,384-token control, with no scaffold, raises its decay factor from 0.730 to 0.877 and its k^* from 3 to 5.

Retries and self-correction help, up to a limit. Both move the breakdown point by one or two constraints. Self-correction helps most where per-constraint reliability is lowest, as expected if a revision pass mainly rescues probes that were close to passing. Best-of-5 falls well short of what independent draws would allow. At its largest, the observed gain in sCSR is 22–24pp, against 51–54pp if the five draws failed independently, so retries deliver about two-fifths of the possible gain. The draws are not copies of one another (4.98–4.99 distinct answers per probe), yet a model tends to fail the same probe in the same way on every draw.

Two kinds of independence. In Section 4.3, failures are nearly uncorrelated *across constraints within one response*, pair by pair. That is why joint success decays like a product. Best-of-N tests a different kind of independence, *across draws for one probe*, and there failures are strongly correlated. Which constraints fail within a response is close to random. Whether a given probe fails is largely fixed by the probe itself. The two findings point the same way: for a given model, some probes are hard as a whole. This would also explain why joint success sits above the product at high k. Put simply, extra attempts and a revision pass each buy one or two constraints, restructuring the prompt buys none, and none of them removes the collapse. We ran each intervention on three models. GPT-5.5 is missing from best-of-5 because its endpoint rejects the temperature parameter. The self-correction comparison is not fully paired because a few second-pass generations failed.

6 Discussion

A collapse without a phase transition. Random k-SAT has a sharp satisfiability threshold. As clauses are added, a large random formula stays satisfiable with high probability up to a critical ratio of clauses to variables, and beyond that ratio it is unsatisfiable with high probability (Mertens

et al., 2006; Mézard & Montanari, 2009). CSE differs in two ways. First, every satisfiable probe passed the compatibility checker, so the decline comes from the model itself. Second, the collapse is smooth and has no critical number of constraints. It follows from multiplying many rates that are each slightly below one, and the interaction we do find slows it down. This tells us where the lever is: per-constraint reliability. Failures barely depend on which constraints appear together, so choosing or ordering constraints cannot help. Decoding-time tools can take some constraints out of the product. Structured generation (Willard & Louf, 2023; Beurer-Kellner et al., 2023) enforces formats and vocabularies exactly, and constrained beam search (Lu et al., 2021; 2022) steers the output toward satisfying lexical constraints. Whether these tools move the breakdown point for the constraints that remain is untested. CSE measures unassisted generation so that such tools can be evaluated against it.

Why would tracked constraints degrade faster? Some constraints are settled by a single decision: the model meets them once and can then set them aside. Others must be tracked, and stay open until the response ends. For these, the model has to keep a running quantity, such as the number of words written so far, the length of the previous sentence, or every word already used, and check it at each step. Every added constraint of this kind is one more quantity to keep. The comprehension–maintenance gap fits this picture. Tracked constraints fail by small margins, as a drifting count would, and single-decision constraints tend to pass or fail outright. L1 fits it too. In principle, avoiding a letter must be tracked, but models appear to settle it by choosing their vocabulary at the start: its failures neither move later in the response nor become more frequent as k rises. If this account is right, giving the model an external counter or a verifier it can call should help tracked constraints most and leave single-decision constraints unchanged.

Implications for deployment. Many applications need an instruction to be followed nearly every time. Only five models satisfy a single CSE constraint on at least 90% of probes, and with two constraints only GPT-6 Astra and GPT-5.5 still clear that bar. Even GPT-6 Astra, the top model by sCSR, fails on more than half of its probes at k=7. To see what this means for a 90% joint-success target: if each constraint holds with probability 0.98, the target allows five constraints, and at 0.95 it allows two. A practitioner can use the per-k rates and decay factors in Table 4 to estimate where a given model crosses such a target. Joint success runs above the product, so this estimate is conservative. Near the breakdown point, continuous scores hide the problem. W1 averages 78.7% on the continuous score but passes only 31.4% of the time. Wherever compliance matters, it should be checked with binary verifiers.

Single-constraint accuracy does not predict robustness. Kimi K3 ranks fourth of 19 at k=1 (94.3%) and falls to 17th on mCSR. Gemini 3.1 Pro drops from fifth (90.5%) to 14th. Rankings also move the other way: Gemini 3.1 Flash-Lite rises from tenth (78.1%) to sixth, and Claude 5 Opus, only sixth at k=1 (87.8%), has the highest per-constraint accuracy at every k from 8 to 12. What separates these models is the decay factor r, and a single-constraint benchmark cannot measure it. Leaderboards that test one to three constraints at a time therefore rank models by a quantity that says little about how they behave under realistic instruction loads.

7 Conclusion

When a prompt asks for many constraints to be satisfied at once, how does a model actually perform? We measured how instruction following degrades as constraints accumulate, using deterministic verifiers across 19 models and up to 12 constraints per prompt. Following more instructions at once requires following each one more reliably, particularly the ones that must be maintained throughout a response. This matters because real prompts rarely ask for one thing: system prompts, style guides, and agent instructions stack requirements on every request. Because every check is deterministic, CSE can be rerun on a new model at the cost of generation alone, and its decay factor summarizes in one number how that model will behave as prompts grow. Three questions remain open: how other composition structures, such as sequences and conditionals, behave; what constraint load costs the primary task; and whether training on many-constraint prompts changes the decay factor itself. We discuss limitations in Section A.

AI-Assistance Disclosure

We used large language model assistants to write and debug experiment and analysis code, render figures, audit reported numbers against result files, and edit parts of the text for grammar and repetition. The authors directed the research, reviewed and revised all assisted output, and take full responsibility for the content of the paper.

References

Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, and Baobao Chang. UltraIF: Advancing instruction following from the wild. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18722–18737, 2025. doi: 10.18653/v1/2025.emnlp-main.945.

Esma Balkır, Alice Pernthaller, Marco Basaldella, José Hernández-Orallo, and Nigel Collier. Confident rankings with fewer items: Adaptive LLM evaluation with continuous scores. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. arXiv:2601.13885.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7 (PLDI):1946–1969, 2023. doi: 10.1145/3591300.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. Controllable text generation with language constraints. *arXiv preprint arXiv:2212.10466*, 2022.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. SPaR: Self-play with tree-search refinement to improve instruction-following in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Vanya Cohen, Geraud Nangue Tasse, Nakul Gopalan, Steven James, Matthew Gombolay, Ray Mooney, and Benjamin Rosman. Compositional instruction following with language models and reinforcement learning. *Transactions on Machine Learning Research*, 2024. arXiv:2501.12539.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Alessandro Favero, Antonio Sclocchi, Francesco Cagnetta, Pascal Frossard, and Matthieu Wyart. How compositional generalization and creativity improve as diffusion models are trained. In *Forty-second International Conference on Machine Learning*, 2025.

Haosheng Gan, Berk Tinaz, Mohammad Shahab Sepehri, Zalan Fabian, and Mahdi Soltanolkotabi. ConceptMix++: Leveling the playing field in text-to-image benchmarking via iterative prompt optimization. *arXiv preprint arXiv:2507.03275*, 2025.

Zhengkang Guo, Wenhao Liu, Mingchen Xie, Jingwen Xu, Zisu Huang, Muzhao Tian, Jianhan Xu, Yuanzhe Shen, Qi Qian, Muling Wu, Xiaohua Wang, Changze Lv, He-Da Wang, Hu Yao, Xiaoqing Zheng, and Xuanjing Huang. RECAST: Expanding the boundaries of LLMs' complex instruction following with multi-constraint data. *arXiv preprint arXiv:2505.19030*, 2025.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10864–10882, 2024. doi: 10.18653/v1/2024.findings-emnlp.637.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022.

Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. SugarCrepe: Fixing hackable benchmarks for vision-language compositionality. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Irene Huang, Wei Lin, M. Jehanzeb Mirza, Jacob A. Hansen, Sivan Doveh, Victor Ion Butoi, Roei Herzig, Assaf Arbelle, Hilde Kuehne, Trevor Darrell, Chuang Gan, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. ConMe: Rethinking evaluation of compositional reasoning for modern VLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2I-CompBench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579, 2025. doi: 10.1109/TPAMI.2025.3531907.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.06120.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling (COLM)*, 2025.

Baiqi Li, Zhiqiu Lin, Deepak Pathak, Jiayao Li, Yixin Fei, Kewen Wu, Tiffany Ling, Xide Xia, Pengchuan Zhang, Graham Neubig, and Deva Ramanan. GenAI-Bench: Evaluating and improving compositional text-to-visual generation. *arXiv preprint arXiv:2406.13743*, 2024.

Peiyu Li, Xiuxiu Tang, Si Chen, Ying Cheng, Ronald Metoyer, Ting Hua, and Nitesh V. Chawla. Adaptive testing for LLM evaluation: A psychometric alternative to static benchmarks. In *Forty-third International Conference on Machine Learning*, 2026. arXiv:2511.04689.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.11423.

Wei Liu, Yancheng He, Yu Li, Hui Huang, Chengwei Hu, Jiaheng Liu, Shilong Li, Wenbo Su, and Bo Zheng. AIR: Complex instruction generation via automatic iterative refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31952–31974, 2025. doi: 10.18653/v1/2025.emnlp-main.1628.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of NAACL*, 2021.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A Smith, and Yejin Choi. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of NAACL*, 2022.

Yuheng Lu, ZiMeng Bai, Caixia Yuan, Huixing Jiang, and Xiaojie Wang. Enhancing complex instruction following for large language models with mixture-of-contexts fine-tuning. *arXiv preprint arXiv:2505.11922*, 2025.

Fangrui Lv, Yulei Qin, Ruixin Hong, Jian Liang, Jinyang Wu, Ke Li, Xing Sun, and Changshui Zhang. Attend to the active: Structure-aware dynamic attention in LLMs for compositional instruction following. In *International Conference on Learning Representations (ICLR)*, 2026.

Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random K-SAT from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. LLM self-correction with DeCRIM: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7773–7812, 2024. doi: 10.18653/v1/2024.findings-emnlp.458.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023. doi: 10.18653/v1/2023.findings-emnlp.378.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. AgentIF: Benchmarking instruction following of large language models in agentic scenarios. *arXiv preprint arXiv:2505.16944*, 2025a.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. Constraint back-translation improves complex instruction following of large language models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 2388–2398, 2025b. doi: 10.1145/3746252.3761324.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. InFoBench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Huimin Ren, Yan Liang, Baiqiao Su, Chaobo Sun, Hengtong Lu, Kaike Zhang, and Chen Wei. LexInstructEval: Lexical instruction following evaluation for large language models. *arXiv preprint arXiv:2511.17561*, 2025.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. Revisiting compositional generalization capability of large language models considering instruction following ability. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31219–31238, 2025. doi: 10.18653/v1/2025.acl-long.1508.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.09677.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*, 2024.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2V-CompBench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8406–8416, 2025.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of CVPR*, 2022.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, 2024.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023. doi: 10.48550/arXiv.2307.09702.

Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. ConceptMix: A compositional image generation benchmark with controllable difficulty. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pp. 86004–86047, 2024.

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, Tao Gui, Qi Zhang, Zhongchao Shi, Jianping Fan, and Xuanjing Huang. MulDimIF: A multi-dimensional constraint framework for evaluating and improving instruction following in large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026a.

Junjie Ye, Guoqiang Zhang, Wenjie Fu, Zelin Li, Tao Gui, Qi Zhang, and Xuanjing Huang. CCTU: A benchmark for tool use under complex constraints. In *Asia-Pacific Chapter of the Association for Computational Linguistics (AACL)*, 2026b. arXiv:2603.15309.

Jie Zeng, Qianyu He, Qingyu Ren, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. Order matters: Investigate the position bias in multi-constraint instruction following. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.20270.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.