

Form Rescued Is Not Content Gained: What Constrained Decoding Does to Accuracy, Why Published Studies Disagree About Its Sign, and the Parse-Failure Accounting That Decides the Small-Model Case

Pranay Mahendrakar

SONYTECH · ORCID 0009-0003-7224-029X

pranaymahendrakar@sonytech.in · mahendrakarpranay@gmail.com · +91 6361723454

Abstract. Constrained decoding masks every token that would break a grammar or a JSON schema, so the output always parses. Whether it also changes how often the answer is right is contested: one line of work reports that format restrictions lower reasoning accuracy, a second that enforcement raises accuracy by a few points, a third that structured output has no causal effect in most settings, and a September 2026 study of small models that enforcement removes every structural failure but leaves a semantic gap that depends on scale. In this analysis the author reviews and synthesises published results to show that these findings measure different differences. The paper writes the accuracy change as an accounting identity with five terms: the effect of asking for a format at all, the rescue of answers a strict parser would have rejected, the cost of a schema that forces the answer before the reasoning, the distortion introduced by how the mask meets the tokenizer and by local renormalisation, and the reach of constraints that encode meaning rather than syntax. Each published design fixes some terms and lets others dominate. Arithmetic on the small-model study's own tables shows that under one engine every model's content-accuracy gain equals, to within rounding, the number of its tasks whose native output failed the schema, so the reported content gain is accounted for, task for task, by the outputs enforcement made valid, one of which the study itself calls hollow, and the largest residual semantic gap is a single task on which one model emitted one function call where two were required. The paper consolidates 32 published results in a table, maps eleven designs onto the five terms, states an attribution procedure, and specifies the arms that would separate a mask's effect on content from its effect on parseability.

Keywords: constrained decoding; structured output; JSON schema; function calling; grammar-constrained generation; evaluation methodology; small language models; reasoning

SONYTECH ORCID: 0009-0003-7224-029X

1. Introduction

Software that consumes a language model's output rarely wants prose. It wants a JSON object that validates against a schema, a function call whose arguments have the declared types, or a query in a formal language that a parser will accept. Two families of technique supply this. The first asks: a prompt that describes the format, perhaps with the schema pasted in, and a parser that hopes for the best. The second enforces: at each decoding step the engine computes which tokens can still lead to a valid string and sets the probability of every other token to zero. Enforcement is now a standard serving feature. It is implemented with finite-state machines over the vocabulary (Willard and Louf, 2023; Koo et al., 2024), with pushdown automata for context-free grammars (Ugare et al., 2024a; Dong et al., 2024; Park et al., 2025), and in query languages that interleave prompting with constraints (Beurer-Kellner et al., 2022). A recent benchmark of real-world schemas evaluates six such frameworks against one another (Geng et al., 2025).

The guarantee these engines give is about form. What they do to content is disputed, and the dispute has

reached the point where published results disagree about the sign. Tam et al. (2024) reported that format restrictions lower the accuracy of several models on reasoning tasks, that stricter restrictions cost more, and that classification tasks can gain. Geng et al. (2025) ran three of the same reasoning tasks under four enforcement frameworks and found that every framework scored at or above the unconstrained baseline on all three, by up to about four points on GSM8K. Yuan et al. (2025) applied causal inference to the question and concluded that, for GPT-4o, structured output had no statistically significant causal effect in 43 of 48 scenarios, while noting that coarse metrics on the same data showed positive, negative and neutral effects. Galeone et al. (2026) found that enforcement degraded task performance substantially in several settings. Matveev (2026) summarised plain JSON generation as beating JSON-mode enforcement on one-shot and final accuracy, although the study's own scenario table gives higher final accuracy under enforcement (92.8 against 88.9 percent on aligned cases), and found that enforcement took one model from zero to 75 percent.

A September 2026 study by Chavan (hereafter "the anchor study") adds a scale dimension. Five open models between 0.6 and 4 billion parameters were run on fourteen structured-output tasks under three conditions:

native greedy decoding, Outlines, and XGrammar. The study scores each output on two axes, schema validity and content accuracy. Enforcement lifts schema validity from 78.6-92.9 percent to 100 percent for every model below 100 percent, and lifts content accuracy too, but not to 1.0 for the three smallest models. The authors read the residue as a semantic gap that depends on scale: failures of type, such as a price written as a string, are rescued, while failures of instruction, such as a missing second function call, are not. Their summary is that "CD's reach ends exactly where schema conformance ends", and that "Constrained decoding rescues form; it does not rescue scale" (Chavan, 2026).

This paper asks what would have to be true for each of these findings to be a finding about constrained decoding. The author's answer is that "the effect of constrained decoding on accuracy" names at least five quantities, and that every study above measured a different sum of them.

The first is the request. Asking for JSON changes what a model generates even when nothing is enforced (Parikh, 2026; Sclar et al., 2024; He et al., 2024). A comparison between a free-text answer and an enforced JSON answer contains this term, and it is not caused by the mask.

The second is rescue. If a strict parser rejects an unconstrained output whose content was right, enforcement turns a zero into a one without changing the content. Whether that counts as a content gain depends entirely on what the scorer did with unparseable output (Galeone et al., 2026; Molfese et al., 2025).

The third is order. A schema that places the answer field before any reasoning field forces an autoregressive model to commit before it computes. Tam et al. (2024) traced their JSON-mode reasoning losses largely to this, and there is a theoretical account of why it costs (Banerjee et al., 2025; Merrill and Sabharwal, 2024).

The fourth is distortion. A mask applied token by token can disagree with the tokenizer about where a grammar symbol begins, and greedy renormalisation can steer the model into a prefix from which only improbable completions remain (Beurer-Kellner et al., 2024; Park et al., 2024; Li et al., 2025a). This term depends on the engine, not on the idea of enforcement.

The fifth is reach. A constraint can encode more than syntax: the set of valid tool names, the entities in a knowledge base, the type system of a programming language. When it does, enforcement can fix errors of meaning (Geng et al., 2023; Zhang et al., 2023; Mundler et al., 2025). Where it stops is a property of the constraint language and of the person who wrote the schema.

The paper makes six contributions. First, it states the five terms as an accounting identity and shows which

difference each published design measured (Section 3). Second, it re-reads the anchor study's tables and shows that under XGrammar the content gain of each of the five models equals the number of its schema-invalid native outputs to within rounding, so that the reported content gain is accounted for, task for task, by the outputs enforcement made valid, one of which the study itself calls hollow; the residual "semantic gap" is less than one task's worth of score in every model and is carried by a single task (Section 4, Figure 1a). Third, it collects the evidence on reach, showing that the boundary the anchor study draws is set by its schema rather than by enforcement (Section 5). Fourth, it separates order and distortion, the two costs a mask can impose, and shows that the engine alone moves GSM8K accuracy between 0.10 and 0.42 in one published study (Sections 6 and 7, Figure 1b). Fifth, it separates the request from the mask (Section 8, Figure 1c). Sixth, it consolidates 32 published results (Table 1), maps eleven designs onto the five terms (Table 2), proposes an attribution procedure (Algorithm 1), and lists five studies that would settle what remains open (Section 12).

In this analysis the author reviews and synthesises published results to show that the sign of the reported effect is a property of each study's design rather than of constrained decoding. Every result is credited to the paper that reported it. Where the paper derives a number from a source's tables, the derivation is shown and labelled as the author's.

2. What the anchor study reports

The anchor study's design is compact and its numbers can be read directly from its tables (Chavan, 2026). Five models were tested: Qwen3-0.6B, Llama-3.2-1B, Llama-3.2-3B, Phi-4-mini (3.8B) and Qwen3-4B. The fourteen tasks fall into four groups: five JSON-generation tasks, three schema-adherence tasks, three function-calling tasks and three extraction tasks. Every condition uses greedy decoding at temperature zero, so each model-task-condition cell is a single output, and the study reports 210 evaluations in total. Each task carries a JSON schema. Expected values, against which content can be scored, are listed for the extraction and function-calling tasks.

Two metrics are defined. Schema validity asks whether the parsed output validates against the task's schema under JSON Schema Draft 2020-12. Content accuracy is the fraction of field values matching the expected answers, after normalising case, whitespace and some punctuation; for function calling, a "function call score" gives half credit for selecting the right function and half for the presence of the required parameters. The paper notes that unnormalised scoring produced ten false semantic failures, all of which normalisation removed, and that thirteen real failures remained.

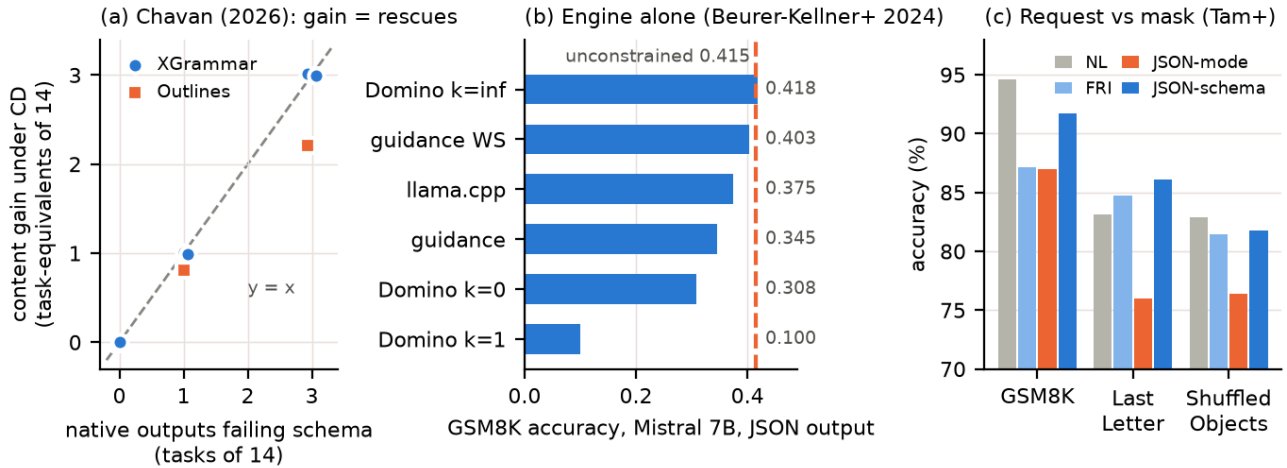


Figure 1. (a) Content-accuracy gain under constrained decoding, in task-equivalents (content accuracy times 14), against the number of each model’s 14 native outputs that failed schema validation, for the five models of Chavan (2026) under XGrammar and Outlines; the values are the author’s arithmetic on that study’s Tables 5 and 6, assuming content accuracy is a 14-task mean (Section 4.1; the anchor study’s per-task Figure 1 confirms it). (b) GSM8K accuracy of Mistral 7B with JSON-formatted reasoning under different constrained-decoding engines and Domino lookahead settings, with the unconstrained score as a dashed line, from Beurer-Kellner et al. (2024), Tables 2 and 4. (c) gpt-4o-mini accuracy in free prose (NL), under a format-restricting instruction (FRI), under JSON mode without a schema, and under schema-enforced structured output, from Tam et al. (2024), Table 2.

Table 5 of the anchor study reports native schema validity of 78.6 percent for both Llama models, 92.9 percent for Qwen3-0.6B and Phi-4-mini, and 100 percent for Qwen3-4B; both engines bring every model to 100 percent. With fourteen tasks per model, 78.6 percent is eleven valid outputs and 92.9 percent is thirteen. The native structural failures therefore number three, three, one, zero and one across the five models: eight single outputs in total.

Table 6 reports content accuracy. Native, Outlines and XGrammar values are 0.771, 0.929 and 0.986 for Llama-3.2-1B; 0.777, 0.991 and 0.991 for Llama-3.2-3B; 0.871, 0.929 and 0.943 for Qwen3-0.6B; 1.000, 0.999 and 1.000 for Qwen3-4B; and 0.929, 0.999 and 1.000 for Phi-4-mini.

Three task-level observations carry the study’s interpretation. On the receipt-extraction task, Qwen3-0.6B and Phi-4-mini wrote prices as strings or with a currency symbol; enforcement forced numeric values and content accuracy on that task rose from 0.000 to 1.000. On a function-calling task that required two calls, a web search followed by an email, Qwen3-0.6B emitted only the search under every condition and scored 0.200 throughout; the schema’s minItems of 1 allowed a single call. On the same task, Llama-3.2-1B under XGrammar scored 1.000 although the email body consisted of the list markers "1. 2. 3. 4. 5." and nothing else, which the study calls a "hollow rescue". The study also asks whether a constrained 1B model can match an unconstrained 3B model, and reports that it exceeds it (0.986 against 0.777), while the same constraint lifts the 3B model to 0.991.

The study names its limitations: dense models only, a hand-designed task suite, and coarse field-level matching. Its practical recommendation is XGrammar with a 3-4B model.

3. Five terms in one comparison

Every study in Section 1 compares two numbers and attributes the difference to a format or to its enforcement. This section writes down what that difference contains. The notation is the author’s.

Fix a model, a task set and a schema G . Consider three arms. In arm N the model answers in free natural language and is asked for no format. In arm Q the prompt requests the format, and may include the schema, but nothing is enforced. In arm C the prompt is the same as in Q and an engine masks every token that would violate G . Three further arms, defined in Algorithm 1 (Section 11), repair C one respect at a time: arm Co rewrites the schema so that a free-text reasoning field precedes every answer field, or applies the mask only after an unconstrained draft (Reddy et al., 2026); arm Ca also replaces the engine with one that aligns subword tokens with grammar symbols or reweights for global constraint satisfaction (Beurer-Kellner et al., 2024; Park et al., 2024); arm Ck also extends the schema with the restrictions the instruction implies.

Score each arm in two ways. A lenient scorer L extracts the answer however it was written, as an LLM-based extractor or a repair pass does (Tam et al., 2024; Yu et al., 2024). A strict scorer S gives credit only if the output validates against G and its content is right. For any arm, $S \leq L$. Define the parse loss of an arm as the difference $L - S$. For an arm whose outputs always validate, the parse

loss is zero, provided the engine's guarantee holds and the two scorers agree about the content of a valid output.

With these definitions two identities follow by rearrangement.

For a comparison of enforced against requested output, both scored strictly,

$$S(C) - S(Q) = [L(Q) - S(Q)] + [L(C) - L(Q)].$$

For a comparison of enforced output against free text, both scored leniently,

$$L(C) - L(N) = [L(Q) - L(N)] + [L(C) - L(Q)].$$

The bracketed terms are the five quantities of Section 1, grouped.

Rescue, P = L(Q) - S(Q). The content the requested arm already had but the strict scorer refused. It is zero by construction when the scorer is lenient, and it is the whole of a gain when the mask changes nothing else.

Request, R = L(N) - L(Q). What asking for the format costs, or gains, before any token is masked. It is present only in comparisons against free text.

Mask effect, M = L(C) - L(Q). The only term that is a property of enforcement as such. It splits further into three parts, each identifiable with one more arm.

Order, O. The loss from a schema that makes the answer precede the computation that would produce it. Identified by comparing C with the same mask under a reasoning-first schema or a draft-conditioned pass.

Distortion, D. The loss from how the mask is implemented: tokenization misalignment, which can prevent the model from emitting the subword it would have chosen, and myopic renormalisation, which can commit the model to a prefix from which only improbable continuations remain. Identified by comparing engines on the same schema.

Reach, K. The gain from constraints that encode meaning: a closed set of valid identifiers, a type system, a minimum count that the instruction implies. Identified by comparing a syntax-only schema with one that encodes the task's semantic restrictions.

With $O = L(Co) - L(C)$ and $D = L(Ca) - L(Co)$, the mask effect satisfies $M = [L(Ca) - L(Q)] - O - D$: what an order-safe, aligned mask for the same schema does, minus the two costs the schema and the engine in use impose. Reach, $K = L(Ck) - L(Ca)$, is the further gain available from a schema that says more; it enters M only to the extent that the schema in use already encodes it. Two consequences matter for reading the literature. A strictly scored comparison of enforced against requested output (the first identity) contains P, which can be large, and no R. A leniently scored comparison of enforced output against free text (the second identity) contains R,

which can be large, and no P. The two designs can therefore report opposite signs for the same engine on the same model without either being wrong, and neither sign is the mask effect.

The identities are bookkeeping, not theory. They say nothing about how large each term is. Sections 4 to 8 collect what the literature reports about each.

4. Rescue, and what the scorer did with unparseable output

4.1 The anchor study's content gain is its rescue count

The anchor study's two axes are presented as separating structure from semantics. Its own tables show that, for this task suite, they do not.

The first step is to establish how content accuracy is aggregated. The paper lists expected values only for the six extraction and function-calling tasks, so one might expect content accuracy to be a mean over those six. It cannot be. Qwen3-0.6B under XGrammar is reported at 0.943, and the paper states that this model scored 0.200 on the two-call task under every condition. A six-task mean containing a 0.200 cannot exceed $(5 + 0.2)/6 = 0.867$. A fourteen-task mean with that single partial score and every other task correct is $(13 + 0.2)/14 = 0.943$, which matches the table to three decimals. The same reading reproduces the model's native value: with the receipt task at 0.000 and the two-call task at 0.200, $(12 + 0.2)/14 = 0.871$, as reported. The author therefore reads content accuracy as a mean over all fourteen tasks. The anchor study's per-task figure confirms this reading: it gives a content score for each of the fourteen tasks, and the means of its cells reproduce Table 6 (Chavan, 2026, Figure 1). How the eight tasks without listed expected values were scored is not stated; the figure shows them at 1.00 in every cell except four native outputs that score 0.00, consistent with their scoring 1 for a valid output.

On that reading, multiplying each content value by 14 gives a score in task-equivalents, and the gain from native decoding to enforcement can be compared with the number of native outputs that failed the schema. Under XGrammar the gains are 3.01 for Llama-3.2-1B, 3.00 for Llama-3.2-3B, 1.01 for Qwen3-0.6B, 0.00 for Qwen3-4B and 0.99 for Phi-4-mini. The native schema failures, from Table 5, are 3, 3, 1, 0 and 1. Every model sits on the line $y = x$ to within the rounding of three-decimal reporting (Figure 1a). Under Outlines, three models sit on the line and two fall below it: Llama-3.2-1B gains 2.21 against three rescues, and Qwen3-0.6B gains 0.81 against one. The per-task figure locates both shortfalls. Llama-3.2-1B's two-call output, which scored 0.00 natively, reaches only 0.20 under Outlines against 1.00 under XGrammar. Qwen3-0.6B's business-card extraction, correct natively and under XGrammar, falls to 0.80 under Outlines (Chavan, 2026, Figure 1).

This arithmetic is the author's, derived from the anchor study's Tables 5 and 6. It supports a specific inference, stated at the level the arithmetic allows: under XGrammar, the content gain the study reports is accounted for by the outputs that enforcement made valid, and no additional content was gained or lost elsewhere in the suite. The inference does not require knowing the scorer's internals. It requires only that a schema-invalid native output contributed nothing, or almost nothing, to content accuracy, which is what the receipt task shows directly: the native values were the right prices in the wrong type, and they scored 0.000.

Equal counts are what a pure-rescue reading requires; they do not prove it. One of Llama-3.2-1B's three rescued outputs is the two-call task, which scores 0.00 natively, 0.20 under Outlines and 1.00 under XGrammar, and whose XGrammar output is the one the study calls a hollow rescue, "list structure without content" (Chavan, 2026). On that task the engines changed what was written, not only its form, and the full XGrammar credit is the function call score's, not a measure of a good email.

Two of the anchor study's claims change status under this reading. The claim that constrained decoding "eliminates all structural failures" stands. The claim that it "improves accuracy", in the sense of content, is a restatement of the first claim on a second axis: the content axis inherits the structural failures because a type error is scored as a content error. The receipt task is labelled in the study's own failure taxonomy as "type coercion", a structural archetype; its rescue appears in the content column because the content scorer is strict about type.

The comparison between a constrained 1B and an unconstrained 3B model changes status as well. Llama-3.2-3B had three schema-invalid native outputs, so its native content of 0.777 includes three zeros that enforcement removes. The constrained 1B model's 0.986 against the unconstrained 3B model's 0.777 is a comparison of a rescued model with an unrescued one. Under a lenient scorer the comparison would be between roughly 13.8 and an unknown number no lower than 10.9 task-equivalents, and the published data cannot say which is larger.

4.2 The residual gap rests on one task

The anchor study's second finding is that a semantic gap survives enforcement and depends on scale. In task-equivalents, the residual shortfall from a perfect 14 under XGrammar is 0.20 for Llama-3.2-1B, 0.13 for Llama-3.2-3B, 0.80 for Qwen3-0.6B, and zero for the two largest models. The Qwen3-0.6B residual is the two-call task. The Llama residuals are single partial scores that do not move with the decoder: Llama-3.2-1B scores 0.80 on business-card extraction and Llama-3.2-3B scores 0.88 on the database-query function call, natively and under both engines (Chavan, 2026, Figure 1). They are content

misses that enforcement neither caused nor cured. With one greedy output per cell, the "scale-dependent semantic gap" is a single task on which the smallest model emitted one call where two were needed, plus fractional misses in two other models. The anchor study did test sampling in one respect: a temperature probe of "3 temperatures $\times$ 3 samples per condition" found the native failures "systematic—stable across sampling conditions—not artifacts of greedy decoding" (Chavan, 2026, Section 4.1), but that probe concerns the baseline's structural failures, not whether the content residual under enforcement varies with scale. The ordering is consistent with scale, and it is a reasonable hypothesis, but five points with one decisive task do not establish a scaling relation.

The function call score sharpens the point. Half its weight is on choosing the right function and half on the presence of required parameters; the values of those parameters are not checked. The hollow rescue, a perfect score for an email body of five empty list markers, shows that this metric is itself partly structural. The study observes this and names it. What follows from it is that the content axis is not an independent semantic measurement for the function-calling tasks either.

4.3 The same pattern elsewhere

The anchor study is not unusual. Where the baseline is scored strictly and fails to parse often, enforcement usually shows a large gain, and the gain is mostly rescue.

Galeone et al. (2026) report the cleanest case of the scorer deciding the result. Three 7-9B models prompted with no system prompt reached 77-85 percent task accuracy on GSM8K, and 0 percent "output accuracy", the joint event of a correct answer and valid JSON, because none of them produced JSON. With a minimal format prompt, Gemma 2-9B wrapped every response in a markdown fence, so its output accuracy was 0 percent while a fallback parser recovered 88.4 percent task accuracy from the same outputs. GPT-4o did the same under the same prompt, at 0 percent output accuracy and 95.7 percent recovered task accuracy. The same outputs score 0 or 88 depending on whether a fence is stripped before parsing.

Zhang et al. (2023) report that Mistral-7B had a syntax-error rate above 90 percent with some unseen tools and scored 0 percent; an engine that enforced tool syntax raised it to 52 percent. Matveev (2026) reports that Qwen2.5-Coder-7B scored 0 percent on plain JSON generation and 75 percent under JSON-mode enforcement, which masks for JSON syntax through response_format json_object and passes no schema, while also consuming roughly an eighth of the tokens. In each case the gain is concentrated where the baseline's parse loss is large.

The converse also holds. Where the baseline already parses, the strictly scored gain shrinks to the mask effect,

and the mask effect is small. On the same ordering benchmark, GPT-OSS 120B reached 100 percent schema validity with and without enforcement, and its semantic success was 83.0 percent prompted and 81.3 percent enforced, a difference of -1.7 points with a 95 percent interval of -5.0 to +1.7 (Li, 2026). Qwen3-30B-A3B, also at 100 percent validity in both modes, moved from 31.3 to 30.7 percent. Qwen3-4B in the anchor study, valid on every task natively, gained nothing.

One published case runs against the pattern. On the same ordering benchmark, Llama-3.1-8B's schema validity rose from 68.7 to 100 percent under provider schema mode, so nearly a third of its prompted orders failed the schema, yet its semantic success moved only from 31.7 to 36.0 percent, a gain of 4.3 points with a 95 percent interval of -0.7 to +9.3 (Li, 2026, Tables 1 and 5). Unsafe acceptances fell from 13.3 to 8.3 percent, which was significant. A strictly scored gain is $P + M$, so unless the mask effect was substantially negative, P here was a small fraction of the 31.3 points of invalid output: most of Llama's schema-invalid orders were wrong in content as well as in form. The share of invalid outputs is an upper bound on P , not an estimate of it.

4.4 Lenient scoring removes the term, and the sign can flip

Tam et al. (2024) designed the rescue term out. Their scorer was an LLM prompted to extract the final answer, which they describe as a perfect text parser; they chose the extractor model by comparing candidates against a reference. Their results therefore contain little P for most models, but not none. They report near-zero parse failures for Gemini 1.5 Flash and GPT-3.5-Turbo and judge parse errors "not the primary factor", yet their Table 3 gives parse-error rates of 60.07 percent for Claude 3 Haiku in JSON on GSM8K and 86.66 percent in YAML on DDXPlus, and they name Claude 3 Haiku and LLaMA 3 8B as the models with the most parse errors. With that qualification, on reasoning tasks the remaining terms were mostly negative. Molfese et al. (2025) show why the choice of extractor matters in evaluation generally. On multiple-choice benchmarks, regular-expression extraction agreed with human judgement 97.9 percent of the time when the prompt constrained the answer format and 84.3 percent when the model was allowed to reason first; they describe the resulting trade-off between constraining the format for easy extraction and leaving the model free to reason. Yu et al. (2024) report that regular-expression extraction is a frequent source of evaluation error.

Geng et al. (2025) compared four engines with an "LM only" baseline on Last Letter, Shuffled Objects and GSM8K, using Llama-3.1-8B-Instruct and a two-field schema with reasoning before the answer. Every engine scored at or above the baseline on every task: on GSM8K, 80.1 percent for LM only against 81.6 to 83.8 percent for the engines. Their quality section does not

report how an LM-only output that failed to parse was scored, or how many did. Without that number the result cannot be split between rescue and mask effect. The authors attribute the best engine's roughly three-point margin to its token-healing implementation, which is a statement about the distortion term, not about rescue.

The result is a simple rule for reading this literature. A strictly scored gain from enforcement is an upper bound on the mask effect, loose by the baseline's parse loss. A leniently scored comparison against free text mixes the mask effect with the request effect. Neither, alone, measures what enforcement does to content.

5. Reach: where a constraint stops is a choice

The anchor study's boundary, "CD's reach ends exactly where schema conformance ends", is true by definition: a mask cannot enforce what the grammar does not express. The substantive question is where schema conformance can be made to end. The literature shows that this boundary moves with the constraint language and with the schema's author, and that it has been moved well past syntax.

5.1 Constraints that encode meaning

Geng et al. (2023) framed closed information extraction, entity disambiguation and constituency parsing as grammars, including grammars that depend on the input. For closed information extraction, where subjects and objects must be among 2.7 million Wikidata entities and relations among 888 Wikidata relations, few-shot LLaMA-33B scored a triplet F1 of 17.5 unconstrained and 36.0 constrained. For entity disambiguation with input-dependent grammars, LLaMA-7B's average accuracy rose from 37.4 to 77.2. These gains are not syntactic. The grammar contains the set of admissible answers, so an invented entity name cannot be written.

Zhang et al. (2023) enforced not only call syntax but the set of available tool names and their arguments, which removed the "tool name error" category entirely. Mundler et al. (2025) built prefix automata for a type system and report that enforcing well-typedness cut compilation errors by more than half and raised functional correctness in synthesis, translation and repair, for models above 30 billion parameters as well as small ones. Poesia et al. (2022) enforced scope, typing and contextual rules for SQL, Vega-Lite and SMCaFlow programs. Scholak et al. (2021) report that incremental parsing during decoding turned fine-tuned T5 text-to-SQL models of passable quality into state-of-the-art ones. Fu et al. (2024) used constrained decoding to enforce security properties in generated code and report it more effective than prefix tuning, without a specialised training set. Ugare et al. (2024b) added backtracking driven by semantic checks and report better accuracy for generated SQL and Vega-Lite. Wang et al. (2023) go a step further and have the model first predict a specialised grammar for the particular output, then

generate under it, so that the constraint itself is chosen per input.

Each of these is a case where a constraint expresses part of the task's meaning, and enforcing it fixes errors the anchor study would classify as semantic.

5.2 What the anchor's schema chose not to say

The two-call task is the anchor study's central example of a failure enforcement cannot reach. Its schema had `minItems` of 1 on the array of calls. The instruction required two calls, to two named functions. A schema that set `minItems` to 2, or that required one element whose name was the search function and one whose name was the email function, would have made the single-call output impossible.

Whether such a schema would have produced a correct second call is a separate question, and the hollow rescue suggests caution: Llama-3.2-1B, given room for the email, filled the body with list markers. Enforcing that an item exists does not make its content good. But the claim at issue is about where enforcement's reach ends, and in this task it ended where the schema's author stopped writing, not where token masking stops working. The author's reading is that the anchor study's taxonomy distinguishes failures its schemas could express from failures they did not, which is a property of fourteen hand-written schemas.

5.3 Where encoding meaning genuinely runs out

The opposite case is also published, and it is the strongest support for the anchor study's conclusion as a practical matter. Li (2026) built a restaurant-ordering benchmark whose schema forbids extra properties and bounds quantities, then scored whether the emitted order was the transaction the customer meant. Enforcement produced 100 percent schema-valid objects for every model, and semantic success of 81.3 percent for the strongest and 2.0 percent for the weakest; 41.7 percent of all Gemma-2-2B's orders were unsafe acceptances, including an allergen conflict. Rules such as "this customer is allergic to peanuts, so satay noodles must be refused" are expressible as constraints only if the schema is rewritten for each customer's stated restrictions, which is an input-dependent grammar of the kind Geng et al. (2023) used, built from a reading of the utterance that is itself the hard part.

Basu et al. (2024) report that the best model on a benchmark of nested API-call sequences reached a full-sequence match of 28 percent. Patil et al. (2023) identify wrong arguments and hallucinated API usage as the main obstacles in API calling. Neither failure is syntactic, and a grammar would have to know the plan to prevent it. Shrimall et al. (2025) argue that schemas written for human developers are ambiguous or incomplete as instructions to a model, and report that a system which rewrites schemas for model consumption, combined with reflection-based extraction, improved extraction

accuracy by up to 64.7 percent on one web-extraction dataset. That places part of the semantic gap in the schema rather than in the model, although the reported gain is for the combined system, not for the schema rewrite alone.

The fair summary is that reach is bounded by what can be stated as a constraint before the model has understood the input. Catalogues, types and counts that the instruction fixes are within reach. Plans, intentions and conditional rules that depend on the content of the request mostly are not. The anchor study's two-call task sits on the near side of that line.

6. Order: a schema can remove the computation

The mask effect can be negative for two reasons. This section takes the first, which is about the schema and not the engine.

6.1 The answer-first schema

An autoregressive model computes as it writes. If it must write the answer before any reasoning, it must answer without the intermediate tokens that chain-of-thought prompting supplies (Wei et al., 2022; Kojima et al., 2022). Tam et al. (2024) found that under JSON mode, which guarantees valid JSON but not a particular key order, 100 percent of GPT-3.5-Turbo's responses on the last-letter task put the answer key before the reasoning key. Zero-shot chain-of-thought had become direct answering. This, not the mask as such, was the main source of their JSON-mode reasoning losses, and they say so: the order of keys and whether reasoning is decoupled from format are the factors they identify.

Yu et al. (2026) measured order directly by asking for answers before reasoning. An autoregressive Qwen model lost up to about 67 percent of its accuracy relative to the reasoning-first order, while a masked diffusion model, which can settle later tokens before earlier ones, lost at most about 14 percent. The comparison isolates order from format: no schema is involved.

6.2 The theoretical account and its fix

Banerjee et al. (2025) give an explanation. Building on the result that a constant-depth transformer answering in one step is limited to a small circuit class, while intermediate generation extends what it can compute (Merrill and Sabharwal, 2024), they show that a grammar admitting only syntactically valid final answers confines the model to that small class, so that some problems solvable with unconstrained reasoning become unsolvable under the constraint. They then show that augmenting the grammar with rules that allow free reasoning before the constrained answer restores the capability while keeping the guarantee. Their algorithm switches the mask off inside reasoning spans and on inside answer spans. On GSM-Symbolic, Qwen2.5-Math-7B scored 29 percent with the answer-only

grammar, 29 percent with unconstrained chain-of-thought, and 38 percent with the switched mask; DeepSeek-R1-Distill-Llama-8B scored 13, 21 and 31 percent respectively.

Their table also shows the rescue term at work inside the "unconstrained" arm. Across the nine models of the current version (Banerjee et al., 2025, Table 1, v4), unconstrained chain-of-thought parsed in 82 to 95 percent of cases, against 88 to 100 percent for the switched mask, so part of the switched mask's advantage over unconstrained reasoning is parse rescue. The advantage over the answer-only grammar is the order term, and parse rescue cannot explain it: for the two models above, the answer-only grammar parses at 99 and 96 percent and the switched mask at only 94 and 92, so the switched mask gains accuracy while parsing less often.

Reddy et al. (2026) generate an unconstrained draft first and apply the mask in a second pass conditioned on it. They frame the cost of hard constraints as a cumulative "projection tax" that grows when little of the model's probability mass falls on valid continuations, and argue that a draft raises that mass. Under strict structured scoring their method raised a 1B model from 15.24 to 39.04 percent on GSM8K compared with standard enforcement. Tam et al.'s NL-to-Format arm, which answers in prose and then converts, matched unrestricted prose on most models (Tam et al., 2024). Li et al. (2025b) let a large model reason freely and train a small model to restructure its answer, and report gains of 35.4 percent in format accuracy and 29.4 percent in content correctness. Geng et al. (2024) refine a black-box model's unconstrained output with a small local model that does have logit access. Wang et al. (2025a) post-process unstructured output with a fine-tuned small model and report 99.5 percent schema accuracy and 94.0 percent content similarity when it is combined with enforcement.

The common element is that reasoning happens before or outside the mask. When it does, the order term goes to zero by construction.

6.3 What this does to the comparisons

Tam et al.'s JSON-mode arm left order uncontrolled. Their gpt-4o-mini experiment with schema-enforced structured output, where a schema can fix key order, is different: on GSM8K the model scored 94.57 percent in prose and 91.71 percent under schema enforcement, on Shuffled Objects 82.85 and 81.77, and on Last Letter 83.11 and 86.07 (Tam et al., 2024). Prose still led on two of the three tasks, by 2.86 and 1.08 points, and schema enforcement led on the third. JSON mode, without schema, scored 86.95, 76.43 and 76.00. Most of the gap between prose and JSON mode closes once the schema is enforced. That is consistent with the order account, provided the enforced schema fixed the reasoning field before the answer field, as the two-field

format the authors describe would; the paper does not report key order for this arm.

Geng et al. (2025) put reasoning first in every schema. Their positive result for every engine is what the order account predicts too, once rescue is allowed for. The anchor study's tasks have no reasoning field and require no computation beyond copying and formatting, so its design sets the order term to zero by the choice of tasks. That is why it can detect no cost of enforcement: the main published mechanism of cost was absent from its suite.

7. Distortion: the engine is a variable

The second way a mask can cost accuracy is in how it is computed. Two mechanisms are documented.

7.1 Tokenization misalignment

Grammars are written over characters; models emit subword tokens. A template that inserts fixed text, or a mask that permits only tokens which do not cross a grammar-symbol boundary, can stop the model from emitting the multi-character token it would naturally use, forcing an unusual segmentation (Beurer-Kellner et al., 2024; Koo et al., 2024). Vieira et al. (2024) show that token-level models are sensitive to how a prompt string ends, such as a trailing space, and give algorithms for converting them into character-level models. Park et al. (2025) note that sound enforcement must compute how a tokenizer aligns with a grammar's terminals, and that this is expensive enough to take tens of minutes for common grammars in existing tools.

Beurer-Kellner et al. (2024) measured the consequence. On GSM8K with JSON-formatted reasoning and five-shot prompts, Mistral 7B scored 0.415 unconstrained. With the same task and output format, a template-based engine scored 0.345, a whitespace-flexible variant 0.403, and a grammar engine with online parsing 0.375. Their subword-aligned engine scored 0.418. Varying the lookahead of their own engine moved the score to 0.308 at $k = 0$ and 0.100 at $k = 1$. For Llama-2 13B the template engine fell from 0.262 to 0.152, about 11 points. The engine alone, with the model, the prompt and the output format held fixed, moved accuracy across a range of more than 30 points (Figure 1b). The authors call a method minimally invasive if every valid output the unconstrained model could produce remains producible under the constraint, and their aligned engine met or slightly exceeded the unconstrained accuracy in every reported setting.

Geng et al.'s attribution of their best engine's margin to token healing is an observation of the same term with the opposite sign: an engine that repairs token boundaries around the prompt performed best (Geng et al., 2025).

7.2 Myopic renormalisation

A mask applied one token at a time keeps any token from which some valid completion exists. It does not ask how probable those completions are. Park et al. (2024) show that this skews the output distribution away from the model's distribution conditioned on the grammar. In their illustrative example, the target distribution placed most mass on strings ending in 1, but sampling under grammar-constrained decoding produced such strings only 30 percent of the time, because a first token that admitted only one improbable completion was chosen about half the time. Their correction samples repeatedly and learns the expected future grammaticality of each prefix. Its result is instructive in two directions. It converged toward the correct conditional distribution on most benchmarks, and the authors report that it had no strong effect on solving downstream tasks.

Li et al. (2025a) give a concrete code-completion case. A model intending to write a matrix-rank function generated the prefix it would use, after which the constraint blocked the continuation carrying 99.59 percent of its probability and steered it into a different function, one the model itself rated as wrong. Their backtracking method, which retreats when most of the next-step mass is invalid, improved API completion by up to 360.87 percent over plain enforcement on a synthetic dataset and by up to 7.84 percent on HumanEval. Lipkin et al. (2025) and Loula et al. (2025) correct the same myopia with weighted rejection sampling and sequential Monte Carlo, building on Lew et al. (2023); Loula et al. report that the correction let small open models outperform models more than eight times larger, and attribute the gain to a better approximation of the constrained posterior. Melcer et al. (2024) trade some of the distortion for speed. Tuccio et al. (2026) start from the observation that enforcement degrades semantic quality when the model's preferred outputs fit the grammar poorly, train the model toward the grammar with label-free reinforcement learning, and report an average gain of 9.8 points over constrained greedy decoding. Training against a schema validator is a related route: Lu et al. (2025) report gains in both valid-JSON generation and downstream tasks, and Agarwal et al. (2025) train a 1.5B model with reasoning rewards for schema adherence. These methods change the model so that the mask has less to remove; in the terms of Section 3 they lower D by raising the mass the model already places on valid continuations, rather than by changing the engine.

Galeone et al. (2026) show a third version of the problem: enforcement applied without telling the model the schema. Their constrained arm on GSM8K used a JSON grammar with no format instructions in the prompt. Qwen 2.5-7B's output accuracy fell from 74.30 percent with a minimal format prompt to 32.83 percent with enforcement alone, and 52.4 percent of Gemma 2-9B's enforced outputs were exact duplicates. Llama

3.1-8B moved the other way, from 44.43 to 52.46 percent. A mask that the prompt has not prepared the model for has low feasible mass in the sense of Reddy et al. (2026) from the first token.

7.3 Where this leaves the distortion term

Distortion is real, can be large, and is a property of the engine and the prompt rather than of enforcement in principle. Aligned and reweighted engines report recovering unconstrained accuracy or better. Two cautions apply. First, provider-hosted modes do not disclose their implementation, so their distortion term is unknown. This applies to the JSON mode of Tam et al.'s API models, though not to their open-weight models, which ran on the open-source Text-Generation-Inference server (Tam et al., 2024); to the schema mode of Li (2026); and to the GPT-4o function calling of Yuan et al. (2025), whose study otherwise used format-restricting instructions and no structured-output mode. Second, correcting the distribution does not guarantee better answers: the correction of Park et al. (2024) changed likelihoods more than accuracy.

8. The request is not the mask

The fifth term is present whenever enforced output is compared with free prose. It is the effect of asking for structure, and it exists without any decoder intervention.

Parikh (2026) asked 44 models one-word questions with wide answer spaces, first in chat and then with the request "Reply with JSON only". The request alone, with no enforcement, raised the modal answer's share on an open prompt from 41 to 64 percent and lowered mean answer-choice surprisal from 1.80 to 1.58 bits. On the 36 of 44 models whose providers support schema enforcement, enforcing the schema compressed the answers by a further 0.03 bits (1.56 to 1.53), an increment Parikh reads only as a battery-wide mean because response_format "is a native decoder constraint for some providers and a gateway coercion for others" (Parikh, 2026). In that setting essentially the whole effect of structured output was the request.

Yuan et al. (2025) intervened separately on instruction wording and on output format and tested for causal effects with McNemar tests pooled across instruction variants. For GPT-4o they found no significant causal effect of format in 43 of 48 scenarios, against coarse comparisons that suggested effects in both directions; small open models showed effects in 7 of 24. Output produced through function calling outperformed format-restricting instructions on five of eight datasets for GPT-4o, and the reasoning model o3 was unaffected by format in their tests.

The sensitivity of models to meaning-preserving prompt formatting is well documented: Sclar et al. (2024) report differences of up to 76 accuracy points across formats for LLaMA-2-13B in few-shot settings, and He et al. (2024)

report up to 40 percent variation for GPT-3.5-Turbo on code translation across plain text, Markdown, JSON and YAML prompt templates. Tam et al.'s own Table 1 isolates a request effect inside a request: adding the schema to the JSON prompt, without enforcement, lowered Claude 3 Haiku's GSM8K score from 86.99 to 23.44 percent, and GPT-3.5-Turbo's from 74.70 to 49.25 (Tam et al., 2024). Their Table 3 reports a 60.07 percent parsing-error rate for Claude 3 Haiku in JSON on GSM8K, which suggests that this particular drop may contain parse loss despite the LLM extractor; the paper does not state which prompt variant the error table refers to.

Xia et al. (2024) report that format-following ability is independent of content quality across the models they test, and Zhou et al. (2023) and Yang et al. (2025) build benchmarks that score format adherence separately. These are measurements of the request side: how well a model complies when asked. They are a reason to expect R to vary by model, which is what Yuan et al. (2025) found.

9. Re-reading the disagreement

With the five terms separated, the published results stop contradicting one another. Table 2 lists, for eleven designs, which comparison was made and which terms the design allowed to be large.

The anchor study measured enforced against native output, strictly scored on a content axis that treats type errors as content errors, on tasks with no reasoning, with two widely used engines. Its design makes P large (eight invalid native outputs of seventy), R zero (the request is fixed) and O zero (no reasoning), and short copying tasks plausibly leave little room for distortion, although Section 4.1 found one direct case: under Outlines, Qwen3-0.6B lost 0.20 on a business-card extraction it had right natively and under XGrammar. Under XGrammar its content gain equals, to within rounding, its count of rescued outputs; one of those, the hollow two-call output, shows that the engine changed content as well as form, so the gain cannot be read as P alone.

Tam et al. (2024) measured enforced against free prose, leniently scored, with order uncontrolled in JSON mode and the schema sometimes pasted into the prompt. Their design makes P small for most models, though not for Claude 3 Haiku and LLaMA 3 8B, and lets R and O be large. Their negative result on reasoning tasks is R plus O, and where the order was fixed by a schema the gap mostly closed. Their positive result on classification, which they attribute to a narrowed answer space, is consistent with a small reach term on tasks whose answers form a closed set.

Geng et al. (2025) measured enforced against requested output, with reasoning first and an unreported parse-failure count. Their design makes R zero and O small, and their positive result is P plus whatever mask effect

remained, with the best engine's margin attributed by them to distortion handling.

Beurer-Kellner et al. (2024) held model, task and output format fixed and varied the engine, isolating D. Banerjee et al. (2025) and Reddy et al. (2026) varied the schema or the order of operations, isolating O. Yuan et al. (2025) and Parikh (2026) isolated R. Galeone et al. (2026) and Matveev (2026) measured strictly scored output accuracy with large and model-specific parse losses, and found both large gains and large losses: P for models that could not produce JSON unaided, and distortion or an unprepared mask for models that could. Matveev's mask enforced JSON syntax with no schema, so his results describe a generic-JSON mask, not a schema mask. Li (2026) held the request fixed across arms with a strong model that already parsed, and found a mask effect indistinguishable from zero.

Read this way, the sign of "the effect of constrained decoding" is a property of the design, not of the technique. A study that compares enforced output with a strictly scored, often-unparseable baseline will report a gain. A study that compares enforced output with leniently scored prose, under a schema that puts the answer first, will report a loss. A study that holds the request and the order fixed, uses an aligned engine and a model that already parses will report approximately nothing. Each is a correct report of the quantity it measured.

The author's reading of the mask effect itself, M, is that the published evidence places it near zero for capable models under aligned engines and reasoning-first schemas, negative under misaligned engines or answer-first schemas, and positive where the constraint encodes the task's admissible answers. This is a hedged inference from designs that each isolate one term, not a measurement: no study collected here runs all the arms on one model and task.

10. The strongest case for the anchor study's reading

The anchor study's conclusions deserve a fair statement before the limits of its evidence are pressed.

First, deployment scores strictly. A transaction compiler, an extraction pipeline or a tool-calling agent receives no credit for a correct price written as a string if the downstream system rejects the type. For a practitioner choosing between a 1B and a 3B model, the strictly scored number is the relevant one, and the anchor study's recommendation follows from it. Rescue is not an artefact from the user's point of view; it is the product.

Second, "schema conformance is necessary but not sufficient for semantic correctness" is correct, and is supported by every study that has looked, most sharply by the ordering benchmark (Li, 2026), the nested-call benchmark (Basu et al., 2024) and the hollow rescue

itself. A team that reads 100 percent schema validity as 100 percent correctness will be wrong. The structured-output literature has begun to build checks that sit after the mask for this reason: per-field trust scores for extracted objects (Goh and Mueller, 2026) and consistency scores across repeated generations (Wang et al., 2025b).

Third, the anchor study did not claim that enforcement changes content beyond the rescued tasks. Its taxonomy puts type coercion in the rescuable column, which is the author's point stated from the other side.

What the evidence does not establish is narrower. It does not establish that content accuracy, as computed, measures something separate from structure; under XGrammar it moves exactly with the structural repairs. It does not establish that the residual gap depends on scale; it rests on one task and fractional misses. And it does not establish that enforcement's reach ends at schema conformance in any sense stronger than the schema's own wording, since the decisive task's schema could have required the second call.

11. Consolidated evidence and an attribution procedure

Table 1 collects the published results the argument rests on, one row per result, with the source named on the row. Values are as reported by each source. The final column gives the term the author reads as dominant in each comparison, which is an interpretation and not a reported quantity.

Two rows need comment. The text-to-table study found enforcement helpful on numeric tables and harmful on short textual descriptions; its authors attribute the harm to densely packed text and long aggregation, and the author's assignment of D or O to that row is a guess at mechanism that the study did not test (Oestreich and Muller, 2025). The final row records that the 100 percent guarantee is itself conditional: on the hardest real-world schemas in JSONSchemaBench, the share of schemas for which open engines produced a valid output ranged from 0.03 to 0.41, against 0.13 for the unconstrained model, because engines failed to compile, timed out or under-constrained (Geng et al., 2025). The anchor study's fourteen hand-written schemas sit at the easy end of that range.

Table 2 maps eleven designs onto the five terms. "Large", "small" and "zero" describe what the design permits, as read by the author from each source's methods; they are not measured magnitudes.

Algorithm 1 states the attribution procedure the author proposes for any claim that constrained decoding changed accuracy. Every arm uses the same model, task set, decoding temperature and number of samples.

The procedure is ordered so that the cheapest arms come first. Arms N, Q and C are what most studies already run;

the anchor study ran Q-like native decoding and C. Re-scoring existing outputs with a lenient scorer costs no generation at all and would give P for any study that kept its raw outputs. Arms Co and Ca need a schema edit and an engine swap. Arm Ck is the only one that requires thought about the task, and it is the one that tests a claim about reach.

12. Studies that would settle the open part

Five studies, each using components that already exist, would settle what the published record leaves open.

Study 1: re-score the anchor suite. Take the anchor study's released native outputs and score them with a lenient extractor that accepts a numeric string as a number and strips a currency symbol. If the native content values rise to within a task of the enforced values, the inference of Section 4.1 is confirmed, and the content gain was rescue. If they do not, enforcement changed content on some task, and the study would identify which. This requires no new generation.

Study 2: the reach arm on the two-call task. Re-run the decisive task with a schema that requires two calls with the two named functions, and score the parameter values, not only their presence. If Qwen3-0.6B then produces a usable email, the failure was within reach and the schema had not expressed it. If it produces a hollow one, the failure is beyond reach in the sense that matters, and the anchor study's taxonomy holds for this case with a better test behind it.

Study 3: the full arm set on one small model. Run Algorithm 1 on Llama-3.2-1B and Qwen3-0.6B over a suite that includes reasoning tasks, with an aligned engine. This is the study the scale question needs. It would say whether the mask effect, as distinct from rescue, changes with scale, and whether small models pay a larger distortion cost, as Reddy et al. (2026) and Tuccio et al. (2026) suggest when they report their largest gains at small scale.

Study 4: hard schemas. Repeat the anchor study's two axes on the GitHub Hard and JSON Schema Store subsets of JSONSchemaBench. On those schemas the guarantee itself fails for most engines (Geng et al., 2025). A claim that enforcement removes "all structural failures" should be tested where engines report compile errors, timeouts and under-constraint, not only on hand-written schemas.

Study 5: reasoning models with hidden traces. Models that reason in a separate channel before any visible output may have an order term near zero regardless of schema. Yuan et al. (2025) report o3 unaffected by format in their tests. Running the order arm with and without a hidden reasoning channel on the same model family would show whether the most-cited mechanism of cost is disappearing from deployed systems.

Table 1. Published results on what enforcing an output format does to accuracy, with the comparison each made.

Source	Setting	Comparison	Reported result	Dominant term
Chavan (2026)	5 models 0.6-4B, 14 tasks	native vs XGrammar, validity	78.6-92.9% to 100%	P
Chavan (2026)	same	native vs XGrammar, content	e.g. Llama-1B 0.771 to 0.986	P (Sec. 4.1)
Chavan (2026)	Qwen3-0.6B, two-call task	native vs CD	0.200 to 0.200	K not encoded
Galeone et al. (2026)	3 models 7-9B, GSM8K	no prompt, strict JSON	task 77-85%, output 0%	P
Galeone et al. (2026)	Gemma 2-9B, GSM8K	minimal prompt, fence	output 0%, fallback 88.4%	P
Galeone et al. (2026)	Qwen 2.5-7B, GSM8K	prompt vs grammar only	74.30% to 32.83%	D / unprepared mask
Zhang et al. (2023)	Mistral-7B, unseen tools	prompt vs tool FSM	0% to 52%	P + K
Matveev (2026)	Qwen2.5-Coder-7B	plain JSON vs JSON mode	0% to 75%	P
Matveev (2026)	Hermes-4-405B	plain JSON vs JSON mode	1-shot 92.5% to 35.0%; final 92.5% to 100%	D
Li (2026)	GPT-OSS 120B, ordering	prompt vs schema mode	83.0% to 81.3% (ns)	M near 0
Li (2026)	Llama-3.1-8B, ordering	prompt vs schema mode	validity 68.7% to 100%; semantic + 4.3 (ns)	P small despite parse loss
Li (2026)	Gemma-2-2B, ordering	schema mode	100% valid, 2.0% semantic	K limit
Tam et al. (2024)	GPT-3.5-Turbo, Last Letter	JSON mode	100% answer-first	O
Tam et al. (2024)	gpt-4o-mini, GSM8K	prose vs schema-enforced	94.57% vs 91.71%	R + O small
Tam et al. (2024)	gpt-4o-mini, Last Letter	prose vs schema-enforced	83.11% vs 86.07%	M + R
Tam et al. (2024)	Claude 3 Haiku, GSM8K	JSON prompt, + schema text	86.99% to 23.44%	R (P possible)
Geng et al. (2025)	Llama-3.1-8B, GSM8K	LM only vs 4 engines	80.1% vs 81.6-83.8%	P + D
Beurer-Kellner et al. (2024)	Mistral 7B, GSM8K	engines, same format	0.100-0.418 vs 0.415	D
Beurer-Kellner et al. (2024)	Llama-2 13B, GSM8K	template engine	0.262 to 0.152	D
Banerjee et al. (2025)	Qwen2.5-Math-7B, GSM-Symb.	answer-only vs switched	29% to 38%	O
Reddy et al. (2026)	1B model, GSM8K	CD vs draft-conditioned	15.24% to 39.04%	O / feasible mass
Yu et al. (2026)	Qwen (AR), reasoning	answer-first vs CoT-first	up to ~67% relative drop	O
Park et al. (2024)	toy grammar	target vs GCD sampling	30% of strings end in 1	D (distribution)
Li et al. (2025a)	API completion	CD vs backtracking	up to +360.87%	D
Tuccio et al. (2026)	Llama 1-8B, 3 tasks	greedy GCD vs RL-adapted	+9.8 points average	D
Geng et al. (2023)	LLaMA-33B, closed IE	unconstrained vs grammar	F1 17.5 to 36.0	K
Geng et al. (2023)	LLaMA-7B, entity disamb.	unconstrained vs input grammar	37.4 to 77.2	K
Oestreich and Muller (2025)	Phi-4, Rotowire teams	unstructured vs structured	cell F1 0.675 to 0.937	P + K
Oestreich and Muller (2025)	Phi-4, E2E	unstructured vs structured	cell F1 0.818 to 0.727	D or O
Yuan et al. (2025)	GPT-4o, 8 tasks x 3 formats	causal test	no effect in 43 of 48	R near 0
Parikh (2026)	44 models, 31 prompts	chat vs JSON request vs enforced	-0.22 bits; enforced -0.03 more (36 models)	R
Geng et al. (2025)	Llama-3.2-1B, GitHub Hard	empirical schema coverage	0.03-0.41 across open engines	guarantee limit

13. What is not known

The size of the mask effect M, holding the request, the order and the engine fixed, is not known for any small model. No study collected here runs arms Q, C, Co and Ca on the same small model and tasks with both scorers.

Whether the anchor study's native outputs contain the right content in invalid form on all eight schema-invalid outputs is not known. The receipt task shows it for two of them. The anchor study's per-task figure locates the other six, all scored 0.00 natively: Llama-3.2-1B's complex-API, database-record and two-call outputs, and Llama-3.2-3B's database-record, config-file and two-call outputs (Chavan, 2026, Figure 1). It gives their scores, not their text, and Llama-3.2-1B's two-call output is the

one whose XGrammar rescue was hollow.

Whether the semantic gap the anchor study reports depends on scale is not known. The evidence is one task with one greedy output per model.

How much distortion provider-hosted structured-output modes introduce is not known, because their implementations are not published.

Whether reach can be extended to instruction-level failures such as a missing second call, in the sense of producing a correct call rather than a present one, is not known. The hollow rescue suggests that forcing presence is not enough.

Table 2. Which terms each published design allows into its reported difference.

Design	Scorer	Rescue P	Request R	Order O	Distortion D	Reach K
Chavan (2026)	strict on type	large	zero	zero (no reasoning)	small	as written
Tam et al. (2024)	LLM extractor	small; large for some	large	large in JSON mode	unknown for API models	small
Geng et al. (2025)	not stated	unknown	zero	small	varied, reported	none
Beurer-Kellner et al. (2024)	well-formed + answer	small	zero	fixed	isolated	none
Banerjee et al. (2025)	parse + answer	small	zero	isolated	small	none
Galeone et al. (2026)	strict JSON	large	varies	small	large for some	none
Li (2026)	strict, semantic	model-specific	zero	none	unknown (provider)	not encoded
Matveev (2026)	strict, repair loop	large for some	zero	none	large for some	none
Yuan et al. (2025)	task metrics	small	isolated	not reported	none; unknown for function calling	none
Parikh (2026)	exact match	zero	isolated	none	small, confounded (36 models)	none
Geng et al. (2023)	task F1	small	zero	none	small	isolated

Algorithm 1. Attributing an accuracy change to constrained decoding

Input: model, tasks T, schema G, engine E, lenient scorer L, strict scorer S	
Arms:	
N	free prose answer, no format requested
Q	format and G stated in the prompt, no mask
C	prompt of Q, mask for G with engine E
Co	as C, G rewritten so a free reasoning field precedes every answer field (or: mask applied to a second pass conditioned on Q's free draft)
Ca	as Co, engine replaced by a subword-aligned or reweighted engine
Ck	as Ca, G extended with every restriction the instruction implies (closed sets of names, counts, types, input-dependent enumerations)
Step 1	check the guarantee: $S(C) = L(C)$ on every task; if not, report the engine's coverage failures before any accuracy claim
Step 2	rescue $P = L(Q) - S(Q)$ request $R = L(N) - L(Q)$ mask effect $M = L(C) - L(Q)$ order $O = L(Co) - L(C)$ distortion $D = L(Ca) - L(Co)$ reach $K = L(Ck) - L(Ca)$
Step 3	report $S(C) - S(Q) = P + M$ and $L(C) - L(N) = M - R$, with intervals
Step 4	license "enforcement improved content" only if $M > 0$ with its interval excluding 0 license "the gain is parseability" if $S(C) - S(Q) > 0$ and $M \sim 0$ license "format hurts reasoning" for whichever of R, O, D is > 0; name the term, not the technique license "beyond reach" for a failure only if it persists in Ck

Whether the request effect R is shrinking as models are trained on structured output is not known. Yuan et al. (2025) report it near zero for GPT-4o and o3 but present for small open models, and Parikh (2026) reports a battery-wide drop in answer-choice surprisal across 44 models that is individually significant for six of them; these measure different outcomes.

14. Limitations

This analysis rests heavily on one short preprint, posted a month before this paper, and on arithmetic applied to its rounded tables. The anchor study's text does not state that content accuracy is a fourteen-task mean; the reading rests on two exact matches for Qwen3-0.6B and on the study's per-task figure, whose cell means reproduce every value in Table 6. The rescue reading for Outlines is weaker than for XGrammar: two models fall

below the line, because Outlines rescued Llama-3.2-1B's two-call output only to 0.20 and lowered Qwen3-0.6B's natively correct business-card extraction to 0.80 (Chavan, 2026, Figure 1).

The five terms are an accounting device. Their separation assumes that the strict and lenient scorers agree about the content of a valid output, and that the arms differ only in the stated respect. In practice a reasoning-first schema also changes output length, and an engine swap can change speed and therefore any time-limited result. The assignments in the final column of Table 1 and throughout Table 2 are the author's readings of each source's design and are open to dispute; they are labelled as interpretation. Several of the sources are 2026 preprints, some with very small task counts: the TOON benchmark has four cases, and the anchor study has fourteen tasks with one output each.

Figure 1 juxtaposes results from three studies with different models, tasks and scorers. It illustrates the three terms; it does not rank engines or models.

15. Conclusion

Constrained decoding does what it promises about form, on schemas its engine supports. What it does to content depends on which difference is measured. A strictly scored comparison against a baseline that often fails to parse reports rescue, and in the anchor study, under XGrammar, the content gain equals the number of outputs enforcement made valid, to within rounding. A leniently scored comparison against free prose reports the request effect and, where the schema puts the answer first, the order cost. A comparison across engines reports distortion. A comparison between a syntax-only schema and one that encodes the task's admissible answers reports reach. The mask effect itself, with those terms held fixed, has not been measured on the small models the scale question concerns.

Two corrections to the anchor study's summary follow. "CD's reach ends exactly where schema conformance ends" is true, and where schema conformance ends is decided by whoever writes the schema; on the study's decisive task it could have been written further. "Constrained decoding rescues form" is also true, and on the evidence of the study's own tables it is all that the reported content gain shows. The six arms of Algorithm 1, three of which most studies already run, and a lenient re-scoring that needs no new generation at all, would turn the question of what enforcement does to content from a matter of which design one happened to choose into a measured quantity.

References

- Agarwal, B., Joshi, I., & Rojkova, V. (2025). Think Inside the JSON: Reinforcement Strategy for Strict LLM Schema Adherence. arXiv:2502.14905. doi:10.48550/arXiv.2502.14905.
- Banerjee, D., Suresh, T., Ugare, S., Misailovic, S., & Singh, G. (2025). CRANE: Reasoning with constrained LLM generation. ICML 2025. arXiv:2502.09061. doi:10.48550/arXiv.2502.09061.
- Basu, K., Abdelaziz, I., Kate, K., Agarwal, M., Crouse, M., Rizk, Y., et al. (2024). NESTFUL: A Benchmark for Evaluating LLMs on Nested Sequences of API Calls. arXiv:2409.03797. doi:10.48550/arXiv.2409.03797.
- Beurer-Kellner, L., Fischer, M., & Vechev, M. (2022). Prompting Is Programming: A Query Language for Large Language Models. PLDI 2023. arXiv:2212.06094. doi:10.48550/arXiv.2212.06094.
- Beurer-Kellner, L., Fischer, M., & Vechev, M. (2024). Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation. arXiv:2403.06988. doi:10.48550/arXiv.2403.06988.
- Chavan, A. (2026). Constrained Decoding Eliminates Structural Failures in Small LLMs but Reveals a Scale-Dependent Semantic Gap. arXiv:2609.23742. doi:10.48550/arXiv.2609.23742.
- Dong, Y., Ruan, C. F., Cai, Y., Lai, R., Xu, Z., Zhao, Y., & Chen, T. (2024). XGrammar: Flexible and Efficient Structured Generation Engine for Large Language Models. MLSys 2025. arXiv:2411.15100. doi:10.48550/arXiv.2411.15100.
- Fu, Y., Baker, E., Ding, Y., & Chen, Y. (2024). Constrained Decoding for Secure Code Generation. arXiv:2405.00218. doi:10.48550/arXiv.2405.00218.
- Galeone, C., Park, M., Ettore, G., & Ligorio, D. (2026). When Correct Isn't Usable: Improving Structured Output Reliability in Small Language Models. arXiv:2605.02363. doi:10.48550/arXiv.2605.02363.
- Geng, S., Josifoski, M., Peyrard, M., & West, R. (2023). Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. EMNLP 2023. arXiv:2305.13971. doi:10.48550/arXiv.2305.13971.
- Geng, S., Doner, B., Wendler, C., Josifoski, M., & West, R. (2024). Sketch-Guided Constrained Decoding for Boosting Blackbox Large Language Models without Logit Access. ACL 2024. arXiv:2401.09967. doi:10.48550/arXiv.2401.09967.
- Geng, S., Cooper, H., Moskal, M., Jenkins, S., Berman, J., Ranchin, N., et al. (2025). JSONSchemaBench: A Rigorous Benchmark of Structured Outputs for Language Models. arXiv:2501.10868. doi:10.48550/arXiv.2501.10868.
- Goh, H. W., & Mueller, J. (2026). Real-Time Trustworthiness Scoring for LLM Structured Outputs and Data Extraction. arXiv:2603.18014. doi:10.48550/arXiv.2603.18014.
- He, J., Rungta, M., Koleczek, D., Sekhon, A., Wang, F. X., & Hasan, S. (2024). Does Prompt Formatting Have Any Impact on LLM Performance? arXiv:2411.10541. doi:10.48550/arXiv.2411.10541.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. NeurIPS 2022. arXiv:2205.11916. doi:10.48550/arXiv.2205.11916.
- Koo, T., Liu, F., & He, L. (2024). Automata-based constraints for language model decoding. COLM 2024. arXiv:2407.08103. doi:10.48550/arXiv.2407.08103.
- Lew, A. K., Zhi-Xuan, T., Grand, G., & Mansinghka, V. K. (2023). Sequential Monte Carlo Steering of Large Language Models using Probabilistic Programs. arXiv:2306.03081. doi:10.48550/arXiv.2306.03081.
- Li, Y. (2026). When JSON Is Not Enough: Semantic Reliability of Schema-Constrained LLM Ordering Agents. arXiv:2607.18261. doi:10.48550/arXiv.2607.18261.
- Li, Y., Li, J., Li, G., & Jin, Z. (2025a). AdapTrack: Constrained Decoding without Distorting LLM's Output Intent. ICSE 2026. arXiv:2510.17376. doi:10.48550/arXiv.2510.17376.
- Li, Y., Liao, Y., Chen, Z., Wang, Y., & Wang, Y. (2025b). DICE: Structured Reasoning in LLMs through SLM-Guided Chain-of-Thought Correction. EMNLP 2025. arXiv:2510.09211. doi:10.48550/arXiv.2510.09211.
- Lipkin, B., LeBrun, B., Vigly, J. H., Loula, J., MacIver, D. R., Du, L., et al. (2025). Fast Controlled Generation from Language Models with Adaptive Weighted Rejection Sampling. COLM 2025. arXiv:2504.05410. doi:10.48550/arXiv.2504.05410.
- Loula, J., LeBrun, B., Du, L., Lipkin, B., Pasti, C., Grand, G., et al. (2025). Syntactic and Semantic Control of Large Language Models via Sequential Monte Carlo. arXiv:2504.13139. doi:10.48550/arXiv.2504.13139.
- Lu, Y., Li, H., Cong, X., Zhang, Z., Wu, Y., Lin, Y., et al. (2025). Learning to Generate Structured Output with Schema Reinforcement Learning. arXiv:2502.18878. doi:10.48550/arXiv.2502.18878.
- Matveev, I. (2026). Token-Oriented Object Notation vs JSON: A Benchmark of Plain and Constrained Decoding Generation. arXiv:2603.03306. doi:10.48550/arXiv.2603.03306.
- Melcer, D., Gonugondla, S., Perera, P., Qian, H., Chiang, W.-H., Wang, Y., et al. (2024). Approximately Aligned Decoding. NeurIPS 2025. arXiv:2410.01103. doi:10.48550/arXiv.2410.01103.
- Merrill, W., & Sabharwal, A. (2024). The Expressive Power of Transformers with Chain of Thought. ICLR 2024. arXiv:2310.07923. doi:10.48550/arXiv.2310.07923.
- Molfese, F. M., Moroni, L., Gioffre, L., Scire, A., Conia, S., & Navigli, R. (2025). Right Answer, Wrong Score: Uncovering the Inconsistencies of LLM Evaluation in Multiple-Choice Question

- Answering. Findings of ACL 2025. arXiv:2503.14996. doi:10.48550/arXiv.2503.14996.
- Mundler, N., He, J., Wang, H., Sen, K., Song, D., & Vechev, M. (2025). Type-Constrained Code Generation with Language Models. arXiv:2504.09246. doi:10.48550/arXiv.2504.09246.
- Oestreich, J., & Muller, L. (2025). Evaluating Structured Decoding for Text-to-Table Generation: Evidence from Three Datasets. arXiv:2508.15910. doi:10.48550/arXiv.2508.15910.
- Parikh, T. (2026). Structured Output Collapses Answer Diversity Across 44 Language Models. arXiv:2607.18476. doi:10.48550/arXiv.2607.18476.
- Park, K., Wang, J., Berg-Kirkpatrick, T., Polikarpova, N., & D'Antoni, L. (2024). Grammar-Aligned Decoding. NeurIPS 2024. arXiv:2405.21047. doi:10.48550/arXiv.2405.21047.
- Park, K., Zhou, T., & D'Antoni, L. (2025). Flexible and Efficient Grammar-Constrained Decoding. arXiv:2502.05111. doi:10.48550/arXiv.2502.05111.
- Patil, S. G., Zhang, T., Wang, X., & Gonzalez, J. E. (2023). Gorilla: Large Language Model Connected with Massive APIs. arXiv:2305.15334. doi:10.48550/arXiv.2305.15334.
- Poesia, G., Polozov, O., Le, V., Tiwari, A., Soares, G., Meek, C., & Gulwani, S. (2022). Synchromesh: Reliable code generation from pre-trained language models. arXiv:2201.11227. doi:10.48550/arXiv.2201.11227.
- Reddy, A., Walker, T. T., Ide, J. S., & Bedi, A. S. (2026). The Hidden Cost of Structured Generation in LLMs: Draft-Conditioned Constrained Decoding. arXiv:2603.03305. doi:10.48550/arXiv.2603.03305.
- Scholak, T., Schucher, N., & Bahdanau, D. (2021). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. EMNLP 2021. arXiv:2109.05093. doi:10.48550/arXiv.2109.05093.
- Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. ICLR 2024. arXiv:2310.11324. doi:10.48550/arXiv.2310.11324.
- Shrimal, A., Jain, A., Chowdhury, S., & Yenigalla, P. (2025). PARSE: LLM Driven Schema Optimization for Reliable Entity Extraction. EMNLP 2025 Industry Track. arXiv:2510.08623. doi:10.48550/arXiv.2510.08623.
- Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H.-y., & Chen, Y.-N. (2024). Let Me Speak Freely? A Study on the Impact of Format Restrictions on Performance of Large Language Models. arXiv:2408.02442. doi:10.48550/arXiv.2408.02442.
- Tuccio, G., Furnari, A., Gangemi, A., & Mongiovi, M. (2026). GrammarRL: Effective Grammar-Constrained Decoding via Reinforcement Learning. arXiv:2609.39869. doi:10.48550/arXiv.2609.39869.
- Ugare, S., Suresh, T., Kang, H., Misailovic, S., & Singh, G. (2024a). SynCode: LLM Generation with Grammar Augmentation. arXiv:2403.01632. doi:10.48550/arXiv.2403.01632.
- Ugare, S., Gumaste, R., Suresh, T., Singh, G., & Misailovic, S. (2024b). IterGen: Iterative Semantic-aware Structured LLM Generation with Backtracking. ICLR 2025. arXiv:2410.07295. doi:10.48550/arXiv.2410.07295.
- Vieira, T., LeBrun, B., Giulianelli, M., Gastaldi, J. L., DuSell, B., Terilla, J., et al. (2024). From Language Models over Tokens to Language Models over Characters. ICML 2025. arXiv:2412.03719. doi:10.48550/arXiv.2412.03719.
- Wang, B., Wang, Z., Wang, X., Cao, Y., Saurous, R. A., & Kim, Y. (2023). Grammar Prompting for Domain-Specific Language Generation with Large Language Models. arXiv:2305.19234. doi:10.48550/arXiv.2305.19234.
- Wang, D. Y.-B., Shen, Z., Mishra, S. S., Xu, Z., Teng, Y., & Ding, H. (2025a). SLOT: Structuring the Output of Large Language Models. arXiv:2505.04016. doi:10.48550/arXiv.2505.04016.
- Wang, G., Yu, J., Zhang, X., Jiang, D., Song, Y., Deb, T., et al. (2025b). STED and Consistency Scoring: A Framework for Evaluating LLM Structured Output Reliability. arXiv:2512.23712. doi:10.48550/arXiv.2512.23712.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903. doi:10.48550/arXiv.2201.11903.
- Willard, B. T., & Louf, R. (2023). Efficient Guided Generation for Large Language Models. arXiv:2307.09702. doi:10.48550/arXiv.2307.09702.
- Xia, C., Xing, C., Du, J., Yang, X., Feng, Y., Xu, R., et al. (2024). FOFO: A Benchmark to Evaluate LLMs' Format-Following Capability. arXiv:2402.18667. doi:10.48550/arXiv.2402.18667.
- Yang, J., Jiang, D., He, L., Siu, S., Zhang, Y., Liao, D., et al. (2025). StructEval: Benchmarking LLMs' Capabilities to Generate Structural Outputs. arXiv:2505.20139. doi:10.48550/arXiv.2505.20139.
- Yu, L., Fu, Y., Zhang, S., Liu, H., Varma T, M., Ver Steeg, G., & Dong, Y. (2026). Thinking Out of Order: When Output Order Stops Reflecting Reasoning Order in Diffusion Language Models. arXiv:2601.22035. doi:10.48550/arXiv.2601.22035.
- Yu, Q., Zheng, Z., Song, S., Li, Z., Xiong, F., Tang, B., & Chen, D. (2024). xFinder: Large Language Models as Automated Evaluators for Reliable Evaluation. ICLR 2025. arXiv:2405.11874. doi:10.48550/arXiv.2405.11874.
- Yuan, H., Zhao, Y., Zhang, L., Luo, W., & Ma, Z. (2025). Quantifying the Impact of Structured Output Format on Large Language Models through Causal Inference. arXiv:2509.21791. doi:10.48550/arXiv.2509.21791.
- Zhang, K., Chen, H., Li, L., & Wang, W. (2023). Don't Fine-Tune, Decode: Syntax Error-Free Tool Use via Constrained Decoding. arXiv:2310.07075. doi:10.48550/arXiv.2310.07075.
- Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., et al. (2023). Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911. doi:10.48550/arXiv.2311.07911.