Schema-Key Wording as an Instruction Channel in Structured Generation under Constrained Decoding

Yifan Le
Independent Researcher
leyifan@zju.edu.cn

Abstract

Constrained decoding is widely used to make large language models produce structured outputs that satisfy schemas such as JSON. Existing work mainly treats schemas as structural constraints, overlooking that schema-key tokens also enter the autoregressive context and may guide generation. To the best of our knowledge, we present the first systematic study of schema keys as an implicit instruction channel under constrained decoding. We formulate structured generation as a multi-channel instruction problem, where task signals can be placed in prompts, schema keys, or both. We further provide a projection-aware analysis that gives a sufficient condition under which an unconstrained expected-score advantage of an instructional key is preserved after grammar projection. Experiments on GSM8K and Math500 across seven language models show that changing only schema-key wording can substantially affect accuracy, with both positive and negative effects across models. Prompt-level and schema-level instructions also interact non-additively. The evidence is substantially stronger on GSM8K than on Math500. Our findings show that schema design is not merely output formatting, but part of instruction specification in structured generation.

1 Introduction

Structured generation is a key interface between large language models (LLMs) and downstream systems. In applications such as information extraction, tool use, and workflow automation, outputs often need to satisfy predefined formats such as JSON or schema-constrained records. Constrained decoding addresses this need by restricting next-token choices according to a grammar or schema, thereby ensuring structural validity (Scholak et al., 2021; Willard and Louf, 2023).

However, structural validity is not the same as task success. Most constrained decoding work focuses on how constraints are represented and enforced, for example with context-free grammars, finite-state machines, or incremental parsers (Dong et al., 2024; Beurer-Kellner et al., 2024). Recent studies further show that hard constraints can alter model behavior and sometimes affect reasoning quality or generation distribution (Ye et al., 2025; Yun et al., 2025; Wang et al., 2026; Banerjee et al., 2025). This suggests that structured generation should be studied not only as a validity problem, but also as an instruction-design problem.

We focus on a simple but overlooked source of instruction: schema keys. A JSON key is usually treated as a field name for parsing, but under autoregressive generation it is also a generated prefix that conditions the following value. Thus, a neutral key such as `steps` and a more instructional key such as `Solve the problem step by step, showing all reasoning and intermediate steps` can expose different semantic cues while preserving the same canonical output structure. To the best of our knowledge, this is the first systematic study of schema keys as an implicit instruction channel under constrained decoding.

We formulate structured generation as a *multi-channel instruction problem*, where instruction signals may be placed in the prompt, in schema keys, or in both. We compare four controlled settings: no additional instruction, key-only instruction, prompt-only instruction, and instruction through both channels. The model, task input, output structure, decoding engine, and hyperparameters are kept fixed, allowing us to isolate the effect of instruction placement.

To motivate why schema-key wording can matter, we provide a projection-aware analysis. Constrained decoding can be viewed as projecting the model's unconstrained distribution onto the grammar-valid distribution. Under this view, we derive a sufficient condition under which an unconstrained expected-score advantage of an instruc-

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 2523–2535
November 6-10, 2026 ©2026 Association for Computational Linguistics

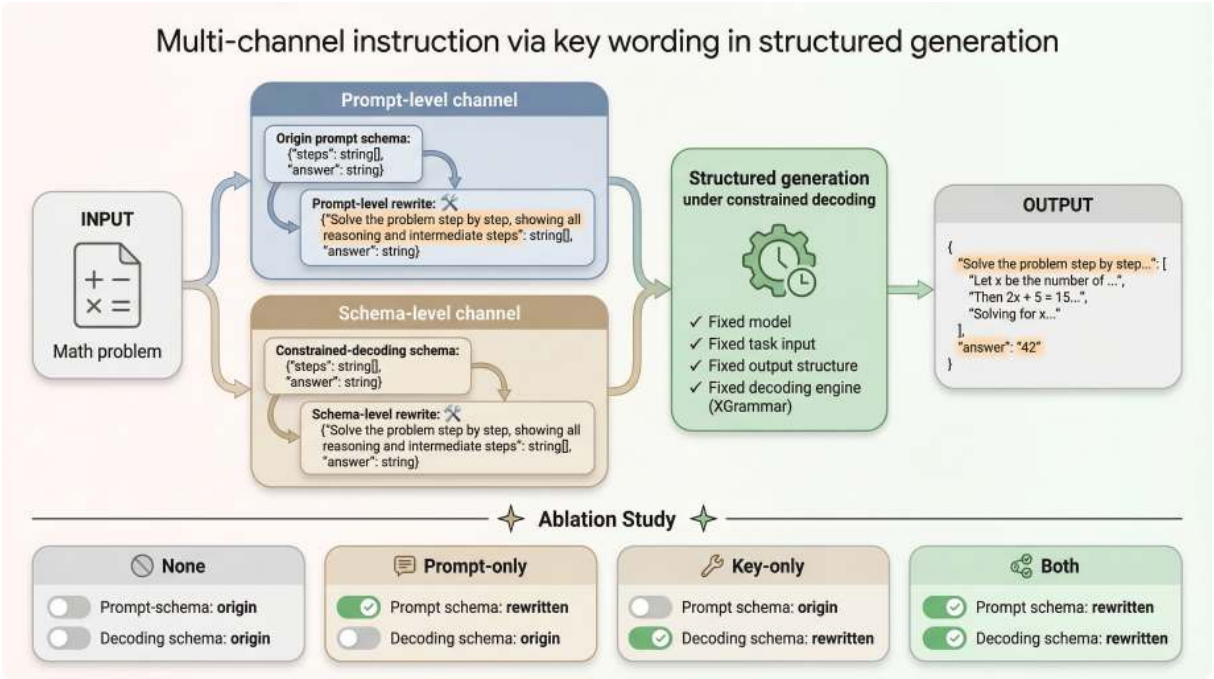


Figure 1: Overview of multi-channel instruction via key wording in structured generation. The same key wording can be injected through two locations: the schema string in the prompt and the schema used by constrained decoding. Holding the model, task input, output structure, and decoding engine fixed, we study four ablation settings: *None*, *Prompt-only*, *Key-only*, and *Both*.

tional key is preserved after grammar projection. This provides a distribution-level perspective on why schema-key effects may vary across models and settings.

Experiments on GSM8K and Math500 show that changing schema-key wording alone can affect accuracy under constrained decoding. The direction and magnitude of these effects vary substantially across individual models and tasks. We also observe non-additive interactions between prompt-level and schema-level instructions, indicating that the two channels may be complementary, redundant, or conflicting depending on the model. The evidence is particularly strong on GSM8K, whereas many Math500 effects are smaller or statistically inconclusive. These results show that schema design is not merely a formatting choice, but part of instruction specification in structured generation.

Our contributions are as follows:

1. We present the first systematic study of schema keys as an implicit instruction channel under constrained decoding.

2. We introduce a multi-channel formulation of structured generation and isolate prompt-level, schema-level, and joint instruction effects through controlled ablations.

3. We provide a projection-aware sufficient-condition analysis and empirically show heterogeneous model- and task-dependent sensitivity to schema-key and prompt-level instruction.

2 Related Work

2.1 Structured Generation with Constrained Decoding

Structured generation is a key capability for real-world deployment of LLMs. Increasingly, benchmarks evaluate not only whether a model can follow instructions, but also whether it can produce outputs that satisfy explicit structural constraints (Zhou et al., 2023; Jiang et al., 2024; Liu et al., 2024). The dominant approach for structured output generation is constrained decoding, where candidate tokens are restricted during inference according to a predefined schema, ensuring that the final output is syntactically and structurally valid.

Most existing work in this area focuses on improving the efficiency and correctness of structural constraints. Representative methods rely on context-free grammars (CFGs), pushdown automata (PDAs), finite-state machines (FSMs), or incremental parsing to constrain model outputs (Dong et al., 2024; Community, 2026; Willard and

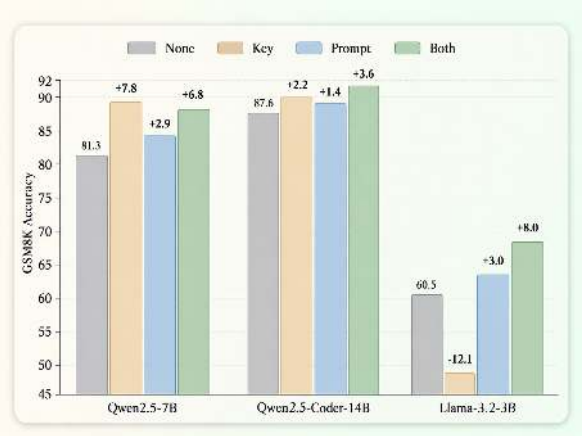


Figure 2: Performance under different instruction placements on representative models in GSM8K. Labels above the non-baseline bars indicate percentage-point changes relative to the *None* setting. The y-axis is truncated to improve visual comparability.

Louf, 2023; Park et al., 2025; Scholak et al., 2021; Beurer-Kellner et al., 2024). In addition, some work introduces extra generation stages or post-processing modules to preserve generation quality while maintaining structural validity, such as two-stage constrained decoding or dedicated models for structured transformation (Wang et al., 2025, 2026; Zheng et al., 2026; Banerjee et al., 2025).

2.2 Effects of Structural Constraints on Model Behavior

Beyond structural validity and decoding efficiency, several studies have begun to examine how format constraints affect model behavior. Prior work shows that constrained decoding may introduce distributional bias and reduce diversity, creativity, or reasoning performance (Ye et al., 2025; Yun et al., 2025; Tam et al., 2024; Schall and de Melo, 2025). At the same time, grammar constraints are not always harmful: in certain tasks, they can even improve model performance, although their effect depends heavily on model scale, task type, and constraint design (Raspanti et al., 2025). Overall, existing studies mainly analyze the side effects of constrained decoding from the perspectives of restricted search space, bias introduced by decoding constraints, or the strength of format control.

Unlike these studies, we focus on a finer-grained but largely overlooked factor: whether schema-key wording itself acts as an implicit instruction signal under constrained decoding.

3 Method

3.1 Problem Setup

We study structured generation where a model receives an input x and generates a serialized output y that must satisfy a schema s, such as a JSON schema. Constrained decoding enforces this requirement by masking invalid next tokens. At step t, let the unconstrained distribution be

$$p_t(v) = p_\theta(v \mid x, y_{<t}). \tag{1}$$

Given the valid token set $\mathcal{V}_t(s, y_{<t})$, constrained decoding uses

$$\begin{aligned} q_t(v) &= \frac{p_t(v)\mathbf{1}[v \in \mathcal{V}_t(s, y_{<t})]}{Z_t(s, y_{<t})}, \\ Z_t(s, y_{<t}) &= \sum_{u \in \mathcal{V}_t(s, y_{<t})} p_t(u). \end{aligned} \tag{2}$$

Although this guarantees syntactic validity, it does not make the schema semantically neutral. In autoregressive generation, schema keys appear before their values and thus condition subsequent value generation.

Our intervention changes only the wording of a target key. Let s_k be a schema whose target field is named by key k, and let π_k map valid outputs under s_k to the same canonical object used for evaluation. Thus, different key wordings can preserve the same field order, value types, number of fields, and canonical interpretation while changing the generated key tokens.

3.2 Projection-Aware View of Schema-Key Instructions

Following the projection view of constrained decoding (Wang et al., 2026), we characterize when an instructional key can help. Let k_0 denote the neutral key `steps`, and let k_1 denote the instructional key used in our experiments: `Solve the problem step by step, showing all reasoning and intermediate steps`. After the key prefix has appeared in the context, let $\bar{P}_{k,x}$ denote the unconstrained continuation distribution over the value field, and let $\bar{Q}_{k,x}$ denote the corresponding distribution after grammar projection. This conditional view factors out the mechanical cost of forcing the key string itself and focuses on its downstream effect on value generation.

For a bounded metric $M \in [0, 1]$, define

$$\tilde{M}_k(y) = \begin{cases} M(\pi_k(y)), & y \in \mathcal{L}(s_k), \\ 0, & y \notin \mathcal{L}(s_k), \end{cases} \tag{3}$$

Table 1: Performance under different instruction placements across models under greedy decoding. "None" corresponds to constrained decoding (CD) with neutral wording in both channels. "Key only" injects instruction through the decoder-enforced schema key, "Prompt only" injects instruction through the prompt-side schema, and "Both" applies both channels. All values are answer accuracy (%). The best result for each model and benchmark is shown in bold.

Model	Placement	GSM8K	Math500
Qwen2.5-3B	None (CD)	75.06	34.20
	Key only	**79.38**	**37.00**
	Both	73.09	36.60
	Prompt only	77.63	35.00
Qwen2.5-7B	None (CD)	81.27	39.40
	Key only	**89.08**	**42.20**
	Both	88.10	41.00
	Prompt only	84.15	41.20
DeepSeek-R1-Distill-Qwen-7B	None (CD)	63.38	41.80
	Key only	69.83	24.00
	Both	76.72	**47.40**
	Prompt only	**77.03**	43.00
Qwen2.5-Coder-14B	None (CD)	87.57	**38.40**
	Key only	89.76	34.20
	Both	**91.13**	34.60
	Prompt only	88.93	36.80
Llama-3.2-1B	None (CD)	9.40	7.00
	Key only	15.39	7.20
	Both	**23.73**	**12.60**
	Prompt only	13.65	7.00
Llama-3.2-3B	None (CD)	60.50	21.20
	Key only	48.45	19.00
	Both	**68.46**	**22.60**
	Prompt only	63.46	22.20
Llama-3.1-8B	None (CD)	79.53	30.20
	Key only	76.35	**31.60**
	Both	**82.79**	31.20
	Prompt only	82.18	29.60

and define the expected scores

$$R_{\bar{P}}(k,x) = \mathbb{E}_{y\sim\bar{P}_{k,x}}[\tilde{M}_k(y)],$$
$$R_{\bar{Q}}(k,x) = \mathbb{E}_{y\sim\bar{Q}_{k,x}}[\tilde{M}_k(y)]. \tag{4}$$

Let the value-level projection tax and its induced bound be

$$\mathcal{T}_{\text{val}}(k,x) = D_{\text{KL}}(\bar{Q}_{k,x}\|\bar{P}_{k,x}),$$
$$B_k(x) = \sqrt{\tfrac{1}{2}\mathcal{T}_{\text{val}}(k,x)}. \tag{5}$$

Under the standard shared-support and stopping convention, Pinsker's inequality gives

$$\left|R_{\bar{Q}}(k,x) - R_{\bar{P}}(k,x)\right| \le B_k(x). \tag{6}$$

Therefore, if

$$R_{\bar{P}}(k_1,x) - R_{\bar{P}}(k_0,x) > B_{k_1}(x) + B_{k_0}(x), \tag{7}$$

then $R_{\bar{Q}}(k_1,x) > R_{\bar{Q}}(k_0,x)$. This gives a sufficient condition under which an unconstrained expected-score advantage of an instructional key is preserved after grammar projection. The condition is sufficient but not necessary. The full derivation is in Appendix A.

This analysis is distributional. Since our experiments use greedy decoding, the bound should be interpreted as a diagnostic explanation rather than a deterministic guarantee for every generated trajectory.

3.3 Structured Generation as Multi-Channel Instruction

We formulate structured generation as a multi-channel instruction problem. Let c_p denote instruction signals placed in the prompt, and let c_s denote instruction signals placed in schema keys enforced by the decoder:

$$y \sim p_\theta(\cdot \mid x, c_p, c_s; s). \tag{8}$$

Table 2: Channel effects and prompt–schema interaction effects under greedy decoding. Values are percentage-point changes with unadjusted 95% paired item-level percentile bootstrap confidence intervals based on 20,000 resamples. $\Delta_{\text{key}} = R_{01} - R_{00}$, $\Delta_{\text{prompt}} = R_{10} - R_{00}$, and $\Delta_{\text{int}} = R_{11} - R_{10} - R_{01} + R_{00}$. A superscript * indicates that the confidence interval excludes zero. Effects are computed from item-level outcomes before rounding the accuracies shown in Table 1.

(a) GSM8K

Model	Δ_{key}	Δ_{prompt}	Δ_{int}
Qwen2.5-3B	$+4.32^*$ $[+2.05, +6.60]$	$+2.58^*$ $[+0.38, +4.78]$	-8.87^* $[-12.05, -5.76]$
Qwen2.5-7B	$+7.81^*$ $[+5.84, +9.78]$	$+2.88^*$ $[+0.83, +4.93]$	-3.87^* $[-6.37, -1.36]$
DeepSeek-R1-Distill-Qwen-7B	$+6.44^*$ $[+3.41, +9.55]$	$+13.65^*$ $[+10.99, +16.38]$	-6.75^* $[-10.61, -2.96]$
Qwen2.5-Coder-14B	$+2.20^*$ $[+0.61, +3.79]$	$+1.36$ $[-0.23, +2.96]$	$+0.00$ $[-1.90, +1.97]$
Llama-3.2-1B	$+5.99^*$ $[+3.64, +8.34]$	$+4.25^*$ $[+2.20, +6.29]$	$+4.09^*$ $[+0.83, +7.35]$
Llama-3.2-3B	-12.05^* $[-15.09, -9.02]$	$+2.96^*$ $[+0.30, +5.69]$	$+17.06^*$ $[+13.19, +21.00]$
Llama-3.1-8B	-3.18^* $[-5.38, -0.99]$	$+2.65^*$ $[+0.68, +4.62]$	$+3.79^*$ $[+1.06, +6.52]$

(b) Math500

Model	Δ_{key}	Δ_{prompt}	Δ_{int}
Qwen2.5-3B	$+2.80$ $[-0.80, +6.40]$	$+0.80$ $[-2.80, +4.40]$	-1.20 $[-6.20, +3.80]$
Qwen2.5-7B	$+2.80$ $[-0.60, +6.20]$	$+1.80$ $[-1.40, +5.00]$	-3.00 $[-7.40, +1.40]$
DeepSeek-R1-Distill-Qwen-7B	-17.80^* $[-22.00, -13.60]$	$+1.20$ $[-2.40, +4.60]$	$+22.20^*$ $[+16.80, +27.80]$
Qwen2.5-Coder-14B	-4.20^* $[-7.40, -1.20]$	-1.60 $[-4.40, +1.20]$	$+2.00$ $[-1.80, +6.00]$
Llama-3.2-1B	$+0.20$ $[-2.60, +3.00]$	$+0.00$ $[-2.60, +2.80]$	$+5.40^*$ $[+1.20, +9.60]$
Llama-3.2-3B	-2.20 $[-5.80, +1.40]$	$+1.00$ $[-2.20, +4.40]$	$+2.60$ $[-2.20, +7.40]$
Llama-3.1-8B	$+1.40$ $[-1.80, +4.60]$	-0.60 $[-3.80, +2.60]$	$+0.20$ $[-4.40, +5.00]$

Here, s specifies the structural constraint, while c_p and c_s specify where semantic guidance is injected.

We compare four instruction-placement settings:

- **None**: both the prompt schema and decoder-enforced schema use neutral keys.
- **Key-only**: the prompt schema uses neutral keys, while the decoder-enforced schema uses an instructional key.
- **Prompt-only**: the prompt schema describes an instructional key, while the decoder-enforced schema keeps the neutral key.
- **Both**: the instructional key is used in both the prompt schema and the decoder-enforced schema.

This separates whether the instruction is only described to the model, forced into the generated output, or supplied through both routes.

3.4 Controlled Comparison and Effect Measures

Across settings, we keep the model, dataset, constrained decoding engine, decoding hyperparameters, model parameters, field order, value types, number of fields, and canonical interpretation fixed. The only intended intervention is the placement of the instruction signal.

Let R_{ij} denote the evaluation result when $c_p = i$ and $c_s = j$, where $i, j \in \{0, 1\}$. We quantify the schema-level, prompt-level, and joint effects as

$$\begin{aligned} \Delta_{\text{key}} &= R_{01} - R_{00}, \\ \Delta_{\text{prompt}} &= R_{10} - R_{00}, \\ \Delta_{\text{both}} &= R_{11} - R_{00}. \end{aligned} \tag{9}$$

To measure whether the two channels combine additively, we define

$$\Delta_{\text{int}} = R_{11} - R_{10} - R_{01} + R_{00}. \tag{10}$$

Positive values indicate synergy, near-zero values indicate approximate additivity, and negative values indicate redundancy or conflict between prompt-level and schema-level instructions.

4 Experiments

4.1 Experimental Setup

Tasks and metrics. We evaluate instruction placement on two mathematical reasoning benchmarks, **GSM8K** (Cobbe et al., 2021) and **Math500** (Hendrycks et al., 2021). Both tasks require multi-step reasoning and a final answer, making them suitable for studying whether schema keys influence reasoning behavior under constrained decoding. We parse the generated JSON output, extract the final answer field, and report answer accuracy.

Table 3: Effect of instruction placement across channels on representative models on GSM8K under greedy decoding. For clarity, we report a subset of models illustrating heterogeneous responses to instruction placement. These examples are intended to illustrate model-level variation rather than a family-level trend. The best result for each model is shown in bold.

Model	None	Key	Prompt	Both
Qwen2.5-7B	81.27	**89.08**	84.15	88.10
Qwen2.5-Coder-14B	87.57	89.76	88.93	**91.13**
Llama-3.2-3B	60.50	48.45	63.46	**68.46**

As a decoding-regime sensitivity analysis, we additionally repeat the original stochastic decoding configuration with `temperature=0.8` for three complete runs while keeping all other inference settings fixed. The results are reported in Appendix B.

Models. To study cross-model sensitivities to instruction channels, we evaluate a diverse set of representative open-source models, covering both the Qwen family (Yang et al., 2024; Hui et al., 2024) and the LLaMA family (Grattafiori et al., 2024; Meta AI, 2024), as well as a reasoning-oriented variant based on Qwen, DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI, 2025). Specifically, we evaluate Qwen2.5-3B, Qwen2.5-7B, DeepSeek-R1-Distill-Qwen-7B, Qwen2.5-Coder-14B, Llama3.2-1B, Llama-3.2-3B, and Llama-3.1-8B. This model set covers different parameter scales, model families, and post-training styles, allowing us to examine whether schema-key effects are stable across models or tied to particular instruction-following behaviors.

Structured output and constrained decoding. All experiments use the same structured generation framework and the same constrained decoding backend, **XGrammar** (Dong et al., 2024). The target JSON output contains a reasoning field and an answer field. Across all settings, we keep the number of fields, field order, value types, parsing procedure, answer extraction rule, decoding backend, and inference configuration fixed. The controlled intervention is where the reasoning instruction is placed: in the prompt-side schema description, in the decoder-enforced schema key, or in both.

All main experiments use greedy decoding with vLLM and XGrammar. We use `temperature=0`, `n=1`, `top_p=1.0`, `top_k=0`, `min_p=0.0`, `presence_penalty=0`, `frequency_penalty=0`, `repetition_penalty=1.0`, `max_tokens=8192`, and `seed=None`. For statistical uncertainty, we report unadjusted 95% paired item-level percentile bootstrap confidence intervals based on 20,000 resamples.

Instruction-placement settings. Following Section 3, we compare four settings: **None**, **Key-only**, **Prompt-only**, and **Both**. In None, both channels use neutral schema wording. In Key-only, the instructional wording appears only in the decoder-enforced schema key. In Prompt-only, the same instruction is described in the prompt while the enforced key remains neutral. In Both, the instruction appears in both locations. This design separates whether the instruction is merely described to the model, forced into the generated context, or supplied through both channels.

4.2 Main Results under Different Instruction Placements

Table 1 reports the full results across models and benchmarks. The central observation is that schema-key wording is not a neutral formatting detail under constrained decoding. Even when the output structure and decoding engine are fixed, changing the decoder-enforced key can noticeably change task accuracy. This supports our view that schema keys can become part of the effective instruction context once they are generated autoregressively.

On GSM8K, Key-only improves Qwen2.5-7B from 81.27 to 89.08 and Qwen2.5-3B from 75.06 to 79.38. The effect is not uniformly positive: Llama-3.2-3B drops from 60.50 to 48.45 under Key-only, whereas Llama-3.2-1B improves from 9.40 to 15.39. DeepSeek-R1-Distill-Qwen-7B also improves under Key-only from 63.38 to 69.83 and reaches 77.03 under Prompt-only. These results show that schema-level instruction is a real behavioral factor, but its direction and magnitude depend on the model and task.

The Both setting further shows that prompt-level and schema-level instructions are not uniformly additive. On GSM8K, Both yields the best performance for Qwen2.5-Coder-14B, Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B, whereas Qwen2.5-7B performs best under Key-only and DeepSeek-R1-Distill-Qwen-7B performs best under Prompt-only. Thus, combining the same instruction across both channels can lead to complementarity, redundancy, or interference depending on the model.

Overall, Table 1 supports three claims: schema keys can act as instruction-bearing tokens; instruction-channel sensitivity is model- and task-dependent; and prompt-level and schema-level signals can interact rather than simply reinforce each other.

4.3 Cross-Model Sensitivity to Instruction Channels

Table 3 highlights representative GSM8K cases that make the cross-model pattern easier to inspect. Unlike Table 1, which provides the complete benchmark results, this subset isolates three distinct behaviors: Qwen2.5-7B benefits most from schema-level instruction, Qwen2.5-Coder-14B obtains its best result when both channels are used, and Llama-3.2-3B is harmed by Key-only but improves under prompt-side guidance.

These cases suggest that the effectiveness of instruction placement is not determined only by the benchmark or by the formal JSON schema. It also depends on how a model interprets text appearing in different parts of the structured generation interface. A schema key that helps one model can distract another, even when the constrained output space is identical. Figure 2 provides a visual summary of these representative patterns.

4.4 Single-Channel and Interaction Effects

To connect the empirical results with the effect definitions in Section 3, Table 2 reports both single-channel effects and interaction effects on GSM8K and Math500. Figure 3 visualizes the same quantities as a channel-sensitivity map, making the cross-model and cross-benchmark patterns easier to compare.

As shown in Figure 3, instruction-placement effects are heterogeneous across models and tasks. On GSM8K, the paired bootstrap confidence intervals exclude zero for schema-key effects in all seven models, with five positive and two negative effects. Prompt effects are positive and supported for six of seven models, while interaction effects are supported for six of seven models and can be either positive or negative. For example, Qwen2.5-7B shows a positive key effect ($\Delta_{\text{key}} = +7.81$) and a negative interaction ($\Delta_{\text{int}} = -3.87$), whereas Llama-3.2-3B shows a negative key effect (-12.05) and a large positive interaction ($+17.06$). DeepSeek-R1-Distill-Qwen-7B shows positive key and prompt effects but a negative interaction. In contrast, Math500 provides weaker evidence: all prompt main effects are statistically inconclusive, and only selected key and interaction effects have confidence intervals excluding zero.

For Qwen2.5-7B, all instructed settings improve over None, with the largest gain coming from Key-only. Qwen2.5-Coder-14B also improves under all instructed settings, but obtains its largest gain under Both, indicating a more complementary relationship between prompt-level and schema-level instruction. In contrast, Llama-3.2-3B suffers a large negative change under Key-only but improves under Prompt-only and Both. Llama-3.1-8B is harmed by Key-only while improving under Prompt-only and Both. These results explain why schema wording should be treated as a model-dependent instruction design choice rather than a purely syntactic field name.

The same table also helps interpret the interaction between channels. If prompt-level and schema-level instructions combined additively, Δ_{int} would be approximately zero. Instead, Table 2 shows both positive and negative supported interaction effects on GSM8K, indicating that the two channels can exhibit synergy, redundancy, or interference depending on the model. For Qwen2.5-7B, the interaction is negative, whereas Qwen2.5-Coder-14B is approximately additive and Llama-3.2-3B shows a strong positive interaction. These patterns support the multi-channel view introduced in Section 3: constrained decoding does not merely enforce output validity, but also changes where instruction signals enter the generation process. Prompt descriptions and schema keys can reinforce each other, compete with each other, or produce asymmetric effects depending on the model.

4.5 Summary of Findings

The experiments lead to three main findings. First, schema-key wording can substantially influence reasoning accuracy even when the output structure and constrained decoding engine are fixed. Second, the direction and magnitude of these effects vary across individual models and tasks, with substantially stronger statistical evidence on GSM8K than on Math500. Third, prompt-level and schema-level instructions can interact non-additively, with both positive and negative interaction effects observed across models. These findings suggest that schema keys should be treated as instruction-bearing design choices rather than neutral field names in structured generation.

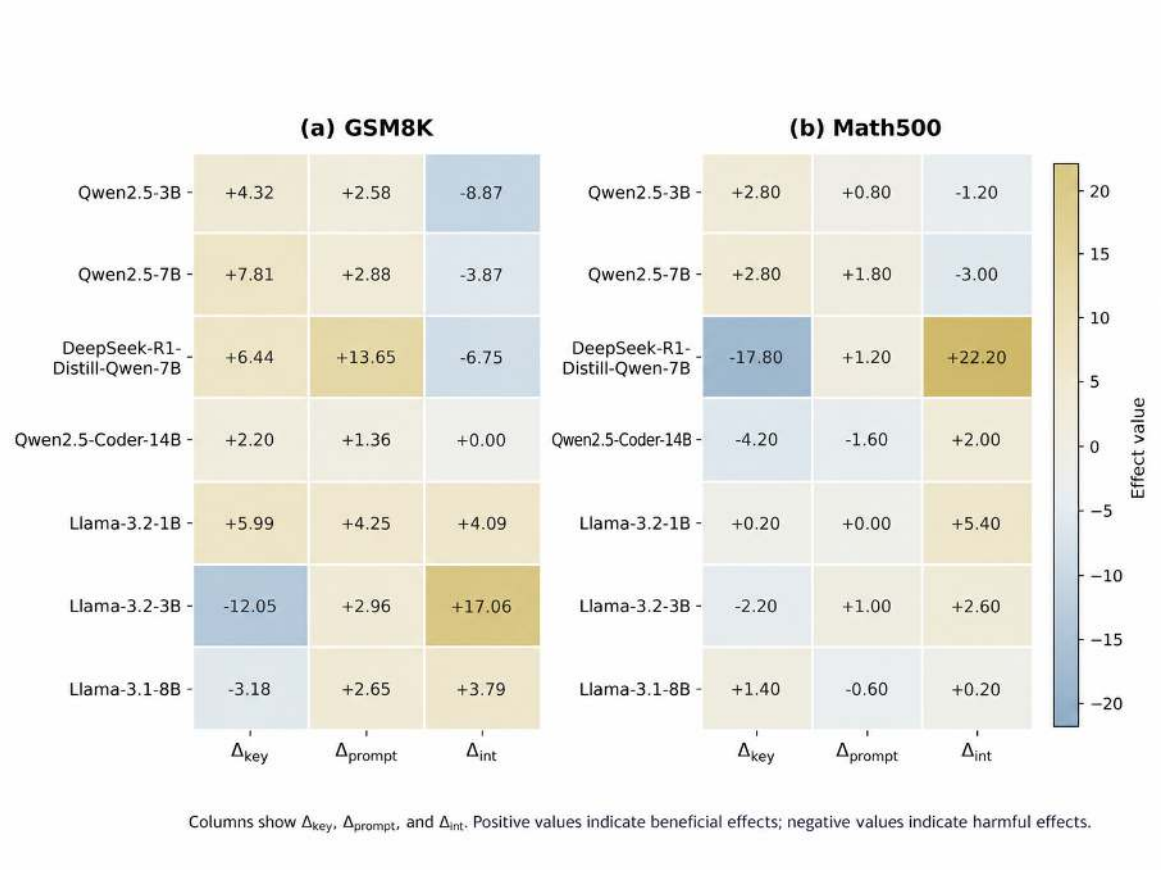


Figure 3: Channel-sensitivity map of single-channel and interaction effects on GSM8K and Math500. Rows correspond to models, and columns correspond to Δ_{key}, Δ_{prompt}, and Δ_{int}. Positive values indicate beneficial effects, while negative values indicate harmful effects. The map highlights model- and task-dependent sensitivity to instruction placement and the non-additive interaction between prompt-level and schema-level instructions.

5 Discussion and Conclusion

This work shows that structured generation under constrained decoding is not only a matter of enforcing valid output formats. Because schema keys are generated as part of the autoregressive context, their wording can act as an instruction-bearing signal rather than a neutral parsing detail. Our results demonstrate that changing the reasoning-field key can alter downstream accuracy even when the schema structure, field types, parsing rules, and decoding backend are fixed.

The multi-channel formulation helps interpret these model-dependent and non-additive effects. Prompt-side descriptions and decoder-enforced keys inject instruction signals through different routes: the prompt provides guidance before generation, while the schema key is forced into the generated prefix and directly conditions the following value. As a result, placing the same instruction in the prompt, in the schema key, or in both can lead to different outcomes. The experiments show substantial heterogeneity across individual models: schema-level instruction can help, hurt, or interact with prompt-side guidance depending on the model and task. The interaction analysis further shows that the two channels can be complementary, redundant, or even interfering, rather than simply additive.

These findings suggest a practical implication for structured generation systems: schema wording should be treated as part of the instruction design. Developers often tune prompts while leaving field names as arbitrary implementation choices, but our results show that this separation is incomplete under constrained decoding. For fair evaluation and reproducible deployment, prompts, schemas, key names, decoding backends, and parsing rules should be reported together. In conclusion, schema keys do not merely describe the shape of an answer; under constrained decoding, they can also shape how the answer is produced.

Limitations

Our study focuses on mathematical reasoning benchmarks, specifically GSM8K and Math500. These tasks are useful for studying intermediate reasoning and final-answer extraction, but they do not cover the full range of structured generation scenarios, such as information extraction, tool use, code generation, data-to-text generation, or agent workflows. Our model coverage is also limited to several open-source Qwen and Llama-family models, together with a Qwen-based distilled reasoning model, so the observed instruction-channel sensitivities may not directly transfer to other model families, larger systems, or proprietary models. Finally, we study a controlled but limited schema design space centered on reasoning-field wording. Other schema factors, including field descriptions, field order, nesting depth, optional fields, enum values, and natural-language schema documentation, may also affect model behavior and deserve further study.

Our evaluated JSON schema contains two required fields and remains structurally simple; we do not test nested schemas, optional fields, or a broader range of key paraphrases. In addition, we do not include an unconstrained-generation baseline or directly measure the terms in the projection-aware analysis. Finally, the statistical evidence is substantially stronger on GSM8K than on Math500, where many effects remain inconclusive.

Ethics Statement

This work uses publicly available benchmarks and open-source models, and does not involve private user data, human subjects, or personally identifiable information. The main ethical consideration is transparency in structured-generation evaluation. Since schema key wording can influence model performance, benchmark results under constrained decoding may depend on implementation choices that are easy to overlook. We therefore recommend that structured-generation studies report not only prompts and models, but also schemas, key names, decoding backends, and parsing rules. In deployment, schema variants should be evaluated carefully, especially in high-stakes settings where subtle changes in field names may affect correctness, robustness, or user trust.

References

Debangshu Banerjee and 1 others. 2025. CRANE: Reasoning with constrained LLM generation. *arXiv preprint arXiv:2502.09061*.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

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

MLC Community. 2026. XGrammar 2: Highly optimized structured generation engine for agentic LLMs. *arXiv preprint arXiv:2601.04426*.

DeepSeek-AI. 2025. Deepseek-r1-distill-qwen-7b. Hugging Face model card. Accessed: 2026-04-16.

Yixin Dong, Charlie F. Ruan, and 1 others. 2024. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15065*.

Aaron Grattafiori, Abhimanyu Dubey, Aakanksha Jauhri, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems*.

Binyuan Hui, Jian Yang, Zeyu Cui, and 1 others. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Yanjun Jiang, Yuxuan Wang, Xuehai Zeng, Wanjun Zhong, and 1 others. 2024. 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 (ACL)*.

Michael Xieyang Liu, Frederick Liu, Alexander J. Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J. Cai. 2024. "we need structured output": Towards user-centered constraints on large language model output. *arXiv preprint arXiv:2404.07362*.

Meta AI. 2024. Llama 3.2 model card. Hugging Face model card. Accessed: 2026-04-16.

Kyoung Park, Tianyuan Zhou, and Loris D'Antoni. 2025. Flexible and efficient grammar-constrained decoding. *arXiv preprint arXiv:2502.05111*.

Federico Raspanti, Tanir Özçelebi, and Mike J Holenderski. 2025. Grammar-constrained decoding makes large language models better logical parsers. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL) - Industry Track*.

Justin Schall and Gerard de Melo. 2025. The hidden cost of structure: How constrained decoding affects language model performance. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*.

Darren Yow-Bang Wang, Zhengyuan Shen, Soumya Smruti, and 1 others. 2025. SLOT: Structuring the output of large language models. *arXiv preprint arXiv:2505.04016*.

Shuo Wang and 1 others. 2026. Draft-conditioned constrained decoding for structured generation in LLMs. *arXiv preprint arXiv:2603.03305*.

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

An Yang, Baosong Yang, Beichen Zhang, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jiacheng Ye and 1 others. 2025. Efficient and asymptotically unbiased constrained decoding for large language models. *arXiv preprint arXiv:2504.09135*.

Longfei Yun, Chenyang An, Zilong Wang, Letian Peng, and Jingbo Shang. 2025. The price of format: Diversity collapse in LLMs. *arXiv preprint arXiv:2505.18949*.

Chujie Zheng and 1 others. 2026. Thinking before constraining: A unified decoding framework for large language models. *arXiv preprint arXiv:2601.07525*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, and 1 others. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Derivation of the Projection-Aware Schema-Key View

This appendix provides the derivation behind the projection-aware view in Section 3.2. The analysis is distributional. Our main experiments use greedy decoding, so the result should be interpreted as a theoretical diagnostic of constraint-induced distortion rather than a deterministic guarantee for every greedy trajectory.

A.1 Constrained Decoding as KL Projection

At step t, let p_t be the unconstrained next-token distribution and let $\mathcal{V}_t$ be the valid next-token set under the grammar and current prefix. The constrained distribution is

$$\begin{aligned} q_t(v) &= \frac{p_t(v)\mathbf{1}[v \in \mathcal{V}_t]}{Z_t}, \\ Z_t &= \sum_{v \in \mathcal{V}_t} p_t(v). \end{aligned} \tag{11}$$

For any distribution r supported on $\mathcal{V}_t$, we have

$$\begin{aligned} D_{\mathrm{KL}}(r\|p_t) &= \sum_{v \in \mathcal{V}_t} r(v) \log \frac{r(v)}{p_t(v)} \\ &= \sum_{v \in \mathcal{V}_t} r(v) \log \frac{r(v)}{Z_t q_t(v)} \\ &= D_{\mathrm{KL}}(r\|q_t) - \log Z_t. \end{aligned} \tag{12}$$

Since $D_{\mathrm{KL}}(r\|q_t) \geq 0$, the minimum is achieved at $r = q_t$, with value $-\log Z_t$. Thus, constrained decoding performs a reverse-KL projection of the model distribution onto the set of valid next-token distributions.

The token-level projection cost is therefore

$$D_{\mathrm{KL}}(q_t\|p_t) = -\log Z_t. \tag{13}$$

For a sequence $y = (y_1, \ldots, y_T)$ sampled under the constrained distribution Q, assuming the same stopping convention for P and Q and $Q(y) > 0 \Rightarrow P(y) > 0$, the sequence-level KL is

$$\begin{aligned} D_{\mathrm{KL}}(Q\|P) &= \mathbb{E}_{y\sim Q}\left[\log \frac{Q(y \mid x)}{P(y \mid x)}\right] \\ &= \mathbb{E}_{y\sim Q}\left[\sum_{t=1}^{T} \log \frac{q_t(y_t)}{p_t(y_t)}\right] \\ &= \mathbb{E}_{y\sim Q}\left[\sum_{t=1}^{T} -\log Z_t(G_k, h_t)\right]. \end{aligned} \tag{14}$$

We call this quantity the cumulative projection tax:

$$\mathcal{T}(G_k, x) = \mathbb{E}_{y\sim Q}\left[\sum_{t=1}^{T} -\log Z_t(G_k, h_t)\right]. \tag{15}$$

The projection tax is small when the model already assigns high probability mass to valid continuations, and large when constrained decoding must strongly override the model's preferred next-token distribution.

A.2 Schema-Key Conditioning

Let $s(k)$ denote the token sequence corresponding to schema key k. In JSON-style constrained decoding, the key tokens are generated before the corresponding value. Hence, after the field name has been emitted, the model distribution over the value is conditioned on $s(k)$:

$$p_\theta(y_{\text{val}} \mid x, h, s(k)). \tag{16}$$

This provides a direct autoregressive conditioning pathway through which schema-key wording can affect generation. An instructional key may shift the continuation distribution relative to a neutral key. A CoT-style key can move the model distribution toward reasoning-like continuations, while a neutral key may provide weaker semantic guidance.

For this reason, we define a conditional reference distribution $\bar{P}_{k,x}$ over continuations after the key prefix $s(k)$ is already present in the context. We also define $\bar{Q}_{k,x}$ as the corresponding constrained continuation distribution after the same prefix. This conditional formulation isolates the semantic effect of the key from the structural cost of forcing the key tokens themselves.

A.3 Bounding the Effect of Projection

Let $\tilde{M}_k(y) \in [0,1]$ be the bounded metric defined in Section 3.2. For any two distributions A and B over the same continuation space,

$$\begin{aligned}\left|\mathbb{E}_{y\sim A}[\tilde{M}_k(y)] - \mathbb{E}_{y\sim B}[\tilde{M}_k(y)]\right| \\ \leq \mathrm{TV}(A,B),\end{aligned} \tag{17}$$

where TV denotes total variation distance. By Pinsker's inequality,

$$\mathrm{TV}(A,B) \leq \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}(A\|B)}. \tag{18}$$

Taking $A = \bar{Q}_{k,x}$ and $B = \bar{P}_{k,x}$ gives

$$\left|R_{\bar{Q}}(k,x) - R_{\bar{P}}(k,x)\right| \leq B_k(x), \tag{19}$$

with $B_k(x)$ defined in Equation 5.

A.4 A Sufficient Condition for CoT-Key Improvement

Let k_0 be a neutral key and k_1 be a CoT-style key. From the bound above,

$$\begin{aligned}R_{\bar{Q}}(k_1,x) &\geq R_{\bar{P}}(k_1,x) - B_{k_1}(x), \\ R_{\bar{Q}}(k_0,x) &\leq R_{\bar{P}}(k_0,x) + B_{k_0}(x).\end{aligned} \tag{20}$$

Therefore,

$$\begin{aligned}&R_{\bar{Q}}(k_1,x) - R_{\bar{Q}}(k_0,x) \\ &\geq R_{\bar{P}}(k_1,x) - R_{\bar{P}}(k_0,x) \\ &\quad - B_{k_1}(x) - B_{k_0}(x).\end{aligned} \tag{21}$$

Thus, if

$$R_{\bar{P}}(k_1,x) - R_{\bar{P}}(k_0,x) > B_{k_1}(x) + B_{k_0}(x), \tag{22}$$

then

$$R_{\bar{Q}}(k_1,x) > R_{\bar{Q}}(k_0,x). \tag{23}$$

This is a sufficient condition, not a necessary one. It formalizes a condition under which an unconstrained expected-score advantage of an instructional key is preserved after grammar projection. If the model is insensitive to key wording, then

$$R_{\bar{P}}(k_1,x) - R_{\bar{P}}(k_0,x) \approx 0, \tag{24}$$

and the condition is unlikely to hold. If the CoT-style key introduces a large projection tax, the condition may also fail. This offers one possible explanation for why key-level effects can vary across models.

A.5 An Optional Activation-Based Interpretation

The sufficient condition above can be connected to reasoning-field activation. Let $\mathcal{R}_x$ denote the set of useful reasoning continuations for input x, and define

$$A_k = \Pr_{y\sim\bar{P}_{k,x}}[y \in \mathcal{R}_x]. \tag{25}$$

Let

$$\begin{aligned}\mu_k^+ &= \mathbb{E}[\tilde{M}_k(y) \mid y \in \mathcal{R}_x], \\ \mu_k^- &= \mathbb{E}[\tilde{M}_k(y) \mid y \notin \mathcal{R}_x].\end{aligned} \tag{26}$$

Then

$$R_{\bar{P}}(k,x) = A_k \mu_k^+ + (1 - A_k)\mu_k^-. \tag{27}$$

If the CoT-style key increases A_k and the conditional quality gap $\mu_k^+ - \mu_k^-$ remains positive, then the semantic gain term $R_{\bar{P}}(k_1,x) - R_{\bar{P}}(k_0,x)$ is expected to increase. This interpretation is not used as a theorem; it only explains why key wording can change the semantic benefit term in Equation 7.

B Decoding-Regime Sensitivity

As a decoding-regime sensitivity analysis, we additionally repeated the full factorial evaluation three times with `temperature`=0.8, while keeping the models, datasets, prompts, schemas, constrained-decoding backend, and all other inference settings fixed.

Tables 4 and 5 report the mean factorial effect across the three runs, the sample standard deviation across runs, and unadjusted 95% paired item-level percentile bootstrap confidence intervals computed from run-averaged item-level contrasts.

Several large GSM8K effects remain stable across runs. For example, Qwen2.5-7B retains a positive schema-key effect, while Llama-3.2-3B retains a large negative schema-key effect and a positive interaction. At the same time, selected effects are sensitive to the decoding regime, most notably for DeepSeek-R1-Distill-Qwen-7B.

Table 4: Factorial effects on GSM8K under stochastic decoding with `temperature`=0.8 across three runs. Values report mean effect ± run-level standard deviation and unadjusted 95% paired item-level bootstrap confidence intervals (in percentage points).

Model	Δ_{key}	Δ_{prompt}	Δ_{int}
Qwen2.5-3B	$+2.60 \pm 0.69$ $[+0.96, +4.25]$	-0.73 ± 2.28 $[-2.35, +0.91]$	-4.22 ± 2.60 $[-6.75, -1.72]$
Qwen2.5-7B	$+7.88 \pm 0.20$ $[+6.44, +9.33]$	$+5.23 \pm 1.40$ $[+3.74, +6.75]$	-4.40 ± 1.91 $[-6.27, -2.58]$
DeepSeek-R1-Distill-Qwen-7B	-13.44 ± 4.71 $[-15.72, -11.17]$	$+6.77 \pm 2.65$ $[+4.98, +8.54]$	$+16.96 \pm 3.60$ $[+13.98, +19.91]$
Qwen2.5-Coder-14B	$+1.87 \pm 0.37$ $[+0.30, +3.44]$	$+2.15 \pm 0.54$ $[+0.94, +3.39]$	-0.71 ± 0.49 $[-2.70, +1.26]$
Llama-3.2-1B	$+3.13 \pm 0.74$ $[+1.67, +4.65]$	$+4.12 \pm 0.46$ $[+2.91, +5.33]$	-1.47 ± 0.49 $[-3.51, +0.56]$
Llama-3.2-3B	-13.62 ± 0.64 $[-15.77, -11.52]$	$+4.09 \pm 1.34$ $[+2.35, +5.81]$	$+16.48 \pm 2.42$ $[+13.52, +19.43]$
Llama-3.1-8B	$+1.16 \pm 1.16$ $[-0.58, +2.91]$	$+4.78 \pm 1.06$ $[+3.18, +6.37]$	-1.06 ± 3.25 $[-3.29, +1.11]$

Table 5: Factorial effects on Math500 under stochastic decoding with `temperature`=0.8 across three runs. Values report mean effect ± run-level standard deviation and unadjusted 95% paired item-level bootstrap confidence intervals (in percentage points).

Model	Δ_{key}	Δ_{prompt}	Δ_{int}
Qwen2.5-3B	$+1.87 \pm 2.32$ $[-0.27, +4.07]$	$+0.67 \pm 1.42$ $[-1.27, +2.60]$	-2.07 ± 2.01 $[-5.07, +1.00]$
Qwen2.5-7B	$+1.67 \pm 1.29$ $[-0.53, +3.87]$	$+1.00 \pm 1.11$ $[-1.00, +3.00]$	-0.93 ± 1.10 $[-3.93, +2.13]$
DeepSeek-R1-Distill-Qwen-7B	-17.00 ± 1.74 $[-20.53, -13.60]$	$+2.13 \pm 1.14$ $[-0.07, +4.40]$	$+16.40 \pm 2.20$ $[+12.13, +20.73]$
Qwen2.5-Coder-14B	-3.47 ± 0.83 $[-5.53, -1.40]$	$+0.13 \pm 1.36$ $[-1.67, +2.00]$	-0.67 ± 2.50 $[-3.80, +2.33]$
Llama-3.2-1B	-1.93 ± 1.55 $[-3.40, -0.47]$	-0.27 ± 0.95 $[-1.67, +1.13]$	$+2.60 \pm 1.04$ $[+0.67, +4.60]$
Llama-3.2-3B	-5.00 ± 1.56 $[-7.20, -2.87]$	$+2.40 \pm 2.75$ $[+0.47, +4.33]$	$+4.53 \pm 2.48$ $[+1.60, +7.53]$
Llama-3.1-8B	$+0.60 \pm 2.80$ $[-1.67, +2.93]$	$+0.53 \pm 1.70$ $[-1.67, +2.73]$	-1.67 ± 2.70 $[-4.73, +1.33]$