MetaCog-Bench: Quantifying the Metacognition Gap in Edge LLM Tool Calling Under Information Insufficiency

Anonymous ACL submission

Abstract

When personal assistants powered by edge-scale language models (8B–32B) encounter under-specified user requests, such as "order me a pizza for tonight" without specifying size, toppings, or address, they overwhelmingly fabricate plausible parameter values rather than seeking clarification. We introduce **MetaCog-Bench**, a diagnostic benchmark spanning 22 API domains with 4,573 samples, featuring a five-category response taxonomy and per-condition annotations that distinguish *fabrication* from coincidental *ground-truth leakage*. Evaluating 14 model configurations across four families, we find that hallucination rates (HR) reach 32–81% and, critically, that fabrication risk varies unpredictably across domains: different models exhibit different vulnerability profiles, making single-domain evaluation insufficient. We then show that a structured audit prompt requiring per-parameter source verification collapses HR to 0.9–4.3% across *all* architectures, compressing cross-model variance by a factor of 19. This reveals a **metacognition gap**: models can reliably perform parameter verification when appropriately prompted, yet this behavior remains absent under standard instructions. Fine-tuning on MetaCog-Bench closes 30–34% of this gap on standard prompts without inference-time scaffolding, validating the benchmark as a training resource.

1 Introduction

As 8B–32B models are increasingly deployed as personal assistants performing real-world tasks through tool calling (Erdogan et al., 2024; Qin et al., 2024; Lu et al., 2025), the fidelity of parameter extraction from natural language has become a cornerstone of their utility. However, real-world prompts are often inherently ambiguous. As illustrated in Figure 1, a request to "order me a pizza for tonight" specifies neither size, toppings, nor delivery address, yet edge models confidently fabricate values like `size="large"` and `address="123 Main St"` rather than seeking clarification, revealing a fundamental gap in their grounding abilities.

Despite their prominence, current tool-calling benchmarks primarily evaluate models under *informationally complete* requests (Patil et al., 2025; Liu et al., 2025), effectively bypassing the structural gaps inherent in real-world user prompts. As illustrated in Table 1, while recent efforts like Zhang et al. (2024b) and Ross et al. (2025) have begun to address information insufficiency, they remain constrained by limited domain coverage or coarse-grained response taxonomies. Crucially, no existing framework systematically isolates **fabrication** (the invention of plausible but ungrounded values) from **ground-truth leakage** (the coincidental output of suppressed values), nor do they provide the **condition-level annotations** required for a precise diagnostic of model awareness.

To fill these voids, we introduce **MetaCog-Bench**, a systematic tool-calling benchmark spanning 22 API domains with 4,573 samples. Constructed from XLAM-60k (Zhang et al., 2025) via a three-stage pipeline, the dataset features a five-category response taxonomy and per-condition annotations that distinguish parameter *fabrication* from coincidental *ground-truth leakage*. Our evaluation of 14 model configurations spanning four architectural families reveals three key findings: (1) hallucination rates (HR) persist between 32–81% even when models generate reasoning traces; (2) the vast majority of failures are driven by the *active fabrication* of plausible parameters rather than passive errors, accounting for 65–84% of removed conditions; and (3) fabrication risk varies unpredictably across API domains, with different models exhibiting different vulnerability profiles (mean pairwise Spearman $\rho = 0.18$), meaning that single-domain evaluation systematically underestimates deployment risk.

These results raise a fundamental question: do edge models lack the requisite verification logic, or

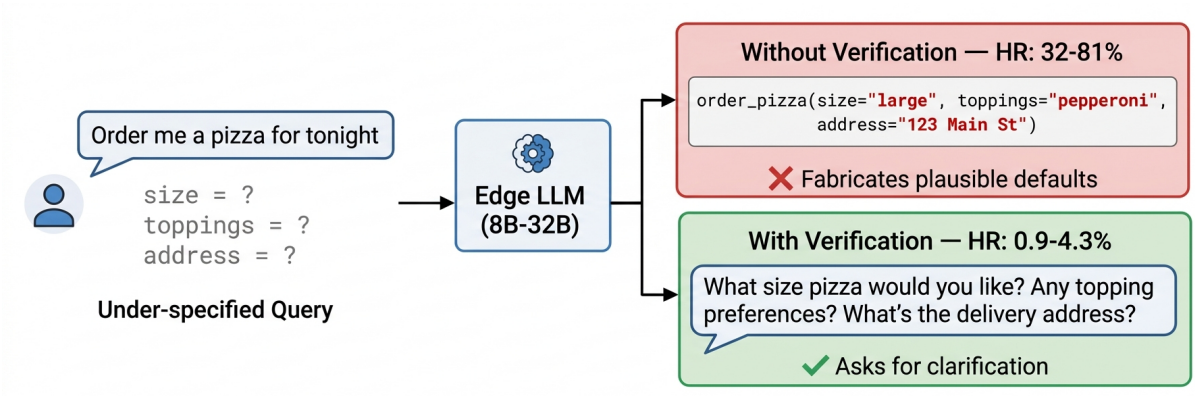

Figure 1: The **metacognition gap** in edge LLM tool calling. Given an under-specified query, edge models (8B–32B) fabricate plausible parameter values (*top path*, HR 32–81%) rather than seeking clarification (*bottom path*, HR 0.9–4.3% with structured verification). The fabricated values are format-legal and undetectable by schema validation.

Benchmark	Venue	#Samples	#Domains	Info-Insuf.	Resp. Labels	Fab./Leak.	Cond. Annot.
BFCL (Patil et al., 2025)	ICML'25	∼2K	6	✗	✗	✗	✗
ToolACE (Liu et al., 2025)	ICLR'25	26K+	390	✗	✗	✗	✗
ToolBeHonest (Zhang et al., 2024b)	EMNLP'24	700	7	Partial	Partial	✗	✗
When2Call (Ross et al., 2025)	NAACL'25	∼4.5K	–	✓	3-way	✗	✗
MetaCog-Bench (Ours)	–	**4,573**	**22**	✓	**5-way**	✓	✓

Table 1: Comparison with existing tool-calling benchmarks. **Info-Insuf.**: systematic information-insufficiency evaluation; **Resp. Labels**: fine-grained response classification; **Fab./Leak.**: per-argument fabrication vs. leakage distinction; **Cond. Annot.**: per-condition detectability and inferrability annotations.

is this capability merely dormant? To investigate, we design a diagnostic experiment using a structured *audit prompt* that requires models to trace each required parameter back to an explicit mention in the user query before making any tool call. The result is striking: HR collapses from 32–81% to 0.9–4.3% across *all* 14 configurations, compressing cross-model variance by a factor of 19. Models spanning four architectural families and baseline HR that varies by a factor of forty all converge to a narrow band below 5% when prompted to verify. We term this discrepancy the **metacognition gap**: models can achieve near-perfect verification when explicitly prompted, yet this behavior does not emerge under standard instructions. The core challenge is therefore not building new capability but eliciting existing behavior without external scaffolding. As a first step toward closing this gap, we show that SFT on MetaCog-Bench enables models to internalize verification habits, closing 30–34% of the gap on standard prompts (e.g., Qwen3-8B improves from 81.0% to 55.7%).

Our contributions are summarized as follows:

- **MetaCog-Bench**: We introduce a diagnostic benchmark of 4,573 samples spanning 22 API domains, the first tool-calling framework to employ a five-category response taxonomy and condition-level annotations that isolate *fabrication* from *ground-truth leakage*. Its cross-domain coverage reveals that fabrication risk varies unpredictably across API categories, with no single domain consistently harder across models.

- **The Metacognition Gap**: We empirically characterize the **metacognition gap** in edge-scale LLMs (8B–32B). While standard prompting yields hallucination rates (HR) of 32–81%, a structured audit prompt collapses HR to 0.9–4.3% across all tested architectures, compressing cross-model variance by 19×. This demonstrates that reliable parameter verification is achievable for all models tested, but does not emerge under standard instructions.

- **Trainability Baseline**: We demonstrate that this gap can be narrowed by internalizing verification reasoning through training. Fine-tuning on MetaCog-Bench closes 30 to 34% of the metacognition gap on standard prompts (e.g., Qwen3-8B improving from 81.0% to

55.7% HR), providing a foundation for future research into autonomous parameter verification.

2 Related Work

2.1 Tool-Calling Benchmarks and Hallucination

BFCL (Patil et al., 2025) evaluates tool-calling capability under informationally complete requests but does not address information insufficiency. ToolBeHonest (Zhang et al., 2024b) diagnoses hallucination across solvability, planning, and missing-tool dimensions on 700 samples, yet focuses on missing tools rather than missing parameters and covers only seven domains. When2Call (Ross et al., 2025) is closest to our setting, defining call/ask/refuse decision boundaries, but does not distinguish fabrication from ground-truth leakage or provide condition-level annotations. Yin et al. (2025) reveal that reasoning RL amplifies tool hallucination, reinforcing that reasoning and verification are independent capabilities. ToolACE (Liu et al., 2025) synthesizes large-scale API dialogues for SFT but likewise targets calling capability under complete information. Collectively, the gaps in cross-domain coverage, fine-grained failure classification, and condition-level diagnostics motivate MetaCog-Bench.

2.2 LLM Abstention and Metacognition

Abstention, that is, judging whether to answer or refuse, is a metacognitive capability that remains underexplored (Wen et al., 2025). R-Tuning (Zhang et al., 2024a) first demonstrates that refusal is a transferable skill trainable by aligning responses with parametric knowledge boundaries. However, Kirichenko et al. (2025) show that reasoning models suffer a 24% average decline in abstention ability, as RLVR rewards only final-answer correctness. Li et al. (2025) further confirm that identifying missing information (40–50% accuracy) is independent of reasoning capability.

These works focus on QA abstention ("does the model know the answer?"). Tool-calling parameter verification is structurally different: the issue is whether the *user* has provided all required inputs, not whether the *model* has internal knowledge. This distinction means models performing well on general abstention may still systematically fail at parameter verification, motivating a dedicated evaluation framework.

2.3 Hallucination Mitigation

On the inference side, HiTEC (Cui et al., 2025) uses per-tool error checklists that are effective but require manual design per tool and special inference-time instructions. On the training side, Relign (Xu et al., 2025) trains models via SFT and DPO to defer under insufficient conditions, reducing HR from 61.5% to 18.8%, but evaluates on a single benchmark without cross-architecture validation. Kumar et al. (2025) and Chu et al. (2025) warn that SFT teaches surface format rather than genuine behavioral change, a limitation we revisit in our trainability experiments (§4.4). MetaCog-Bench provides a cross-domain foundation enabling researchers to train and evaluate verification behavior on standard prompts across architectures, without per-tool checklists or inference-time scaffolding.

3 MetaCog-Bench

3.1 Problem Formulation

We formalize the task as follows. Given a user query q and a tool set $\mathcal{T}$ where each tool $t \in \mathcal{T}$ defines a set of required parameters $P(t) = \{p_1, \ldots, p_k\}$, the model must decide whether q provides sufficient information to invoke t. A query is **under-specified** when at least one required parameter $p_i \in P(t)$ cannot be grounded in q, that is, no substring of q maps to a valid value for p_i.

This formulation differs from QA abstention in a fundamental way. In QA tasks, the question is whether the model *internally knows* the answer. In tool calling, the question is whether the *user has explicitly provided* all required inputs. Even a model with perfect world knowledge should refuse to fill in an unspecified delivery address, because the correct value depends entirely on the user's intent, not on facts the model could retrieve.

3.2 Construction Pipeline

MetaCog-Bench is constructed from xLAM-60k (Zhang et al., 2025) through a three-stage pipeline (Figure 2).

Stage 1: Condition Extraction. GLM-5 (GLM-5-Team, 2026) extracts conditions from each query following the LHAW taxonomy (Pu et al., 2026), classifying each as Goal, Constraint, Input, or Context. API categories are annotated simultaneously (22 categories). We retain only samples with four or more conditions, because removing 1 to 3 conditions from shorter queries would leave fragments

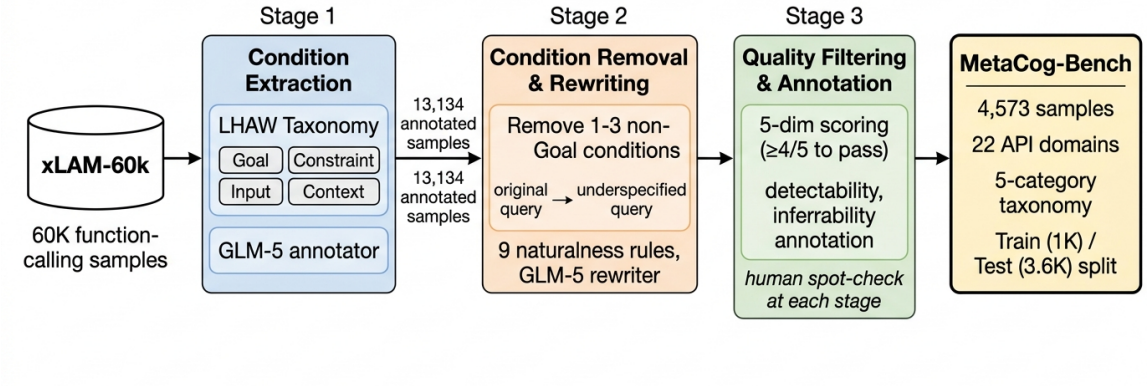

Figure 2: Three-stage construction pipeline of MetaCog-Bench: condition extraction via LHAW taxonomy, condition removal with naturalness-preserving rewriting, and quality filtering with detectability/inferrability annotation. Human spot-checks are conducted at each stage.

too incomplete to resemble natural user input.

Stage 2: Condition Removal and Rewriting. We randomly remove 1 to 3 non-Goal conditions per sample and use GLM-5 to rewrite the query so that the resulting text reads naturally. The rewriting must satisfy two constraints: remaining information is fully preserved, and the information gap is invisible to a reader who has not seen the original query.

Stage 3: Quality Filtering and Annotation. Each sample undergoes five-dimension quality scoring; samples scoring below 4/5 on any dimension are discarded. For each removed condition, we annotate **detectability** (whether a model can notice the gap from the rewritten query alone) and **inferrability** (whether a reasonable default exists). Conditions with high inferrability are filtered out, because fabricating a culturally standard default (e.g., "USD" for currency in an English query) does not constitute a meaningful failure.

Human spot-checks at each stage (100 samples per stage, two annotators) confirm over 92% agreement with automated scores.

3.3 Response Taxonomy and Annotation

We classify model responses into five categories that capture the full spectrum of behavior. REFUSE_FULL and REFUSE_PARTIAL indicate that the model correctly identifies missing information and declines to execute, detecting all or some missing conditions respectively. EXEC_CORRECT_PARTIAL (ECP) indicates correct execution without fabricating any removed parameters. HALLUCINATE_PARTIAL and HALLUCINATE_FULL indicate that some or all tool calls contain erroneous arguments for the removed conditions.

At the per-argument level, each removed condition's model output falls into one of three categories. Consider a hotel booking query where the original request specified 2 guests, but this condition was removed. If the model fills in `guests=1` (a plausible default), this is **fabrication**: format-legal, undetectable by schema validation, and reflecting a systematic default bias. If it fills in `guests=2` (matching the removed ground truth), this is **ground-truth leakage**: coincidentally correct but without basis in the query. If it omits the parameter entirely, this is **ignored**. The fabrication/leakage distinction matters because the two error types have different risk profiles and require different mitigation strategies.

Classification Pipeline. Classification is fully rule-based and requires no LLM judge, ensuring reproducibility. Model outputs are first parsed via regex to extract tool calls in JSON format. Each extracted argument is matched to removed conditions by parameter name. Values are then compared: an exact match with the suppressed ground truth is classified as *leakage*; any other non-empty value is classified as *fabrication*; absence of the parameter is classified as *ignored*. Responses containing no parseable tool calls are classified as refusals. Synonym and unit normalization are not applied, as the ground-truth values in xLAM-60k are already in canonical form matching the tool schema.

3.4 Dataset Overview

The final dataset contains 4,573 samples (4,273 under-specified, 300 complete) spanning 22 API domains, split into 1,000 training and 3,573 test samples (seed=42). The distribution of removed conditions is: n=1 (2,077 samples, 45.4%), n=2 (2,005, 43.8%), and n=3 (491, 10.7%). No domain has fewer than 80 samples, ensuring adequate coverage for per-domain analysis. The complete set of queries serves as a control group for measuring false positive rates.

4 Experiments

4.1 Experimental Setup

Models. We evaluate 14 models spanning four families: Qwen3 (8B/14B/32B in both think and nothink modes), DeepSeek-R1-Distill-Qwen (8B/14B), Gemma3 (12B/27B), and Mistral (Ministral-8B/14B, Mistral-Small-24B), with Kimi-K2.5 as an API-scale reference. This selection covers four architectural families and parameter counts from 8B to 32B.

Prompt conditions. We define two prompt conditions. The **standard prompt** provides the tool JSON schema and user query with no verification instructions, mirroring how models are deployed in practice (Appendix A). The **structured audit prompt** additionally requires the model to trace each required parameter back to an explicit mention in the user query before making any tool call; if no source is found, the model must flag the parameter as missing and request clarification. The audit prompt is a *diagnostic instrument*, not a deployment strategy: real users do not provide such scaffolding (Wang et al., 2025).

Metrics. **HR** (Hallucination Rate): the proportion of under-specified queries where the model produces at least one erroneous argument. **DR** (Detection Rate): the proportion where the model correctly identifies information insufficiency. **Cross-domain CV**: the coefficient of variation (standard deviation divided by mean) of per-domain HR, capturing how consistently a model behaves across API categories. All metrics are stratified by n_{removed} (1/2/3) to control for detection difficulty (Appendix B).

Model	HR%↓	DR%↑	Fab%
Without reasoning (nothink)			
Qwen3-8B	81.0	19.0	84.2
Qwen3-14B	72.0	28.0	70.3
Qwen3-32B	68.0	32.0	65.0
With reasoning (think / distill)			
Qwen3-8B	37.5	62.5	33.2
Qwen3-14B	32.1	67.9	26.0
Qwen3-32B	36.1	63.9	31.9
DS-R1-8B	38.5	61.5	37.2
DS-R1-14B	50.8	49.2	53.2
Other edge families			
Gemma3-12B	41.7	58.3	38.1
Gemma3-27B	28.8	71.2	24.0
Ministral-8B	28.4	71.6	21.5
Ministral-14B	**4.2**	**95.8**	2.9
Small-24B	**1.9**	**98.1**	1.5
Kimi-K2.5 (API)	8.5	91.5	6.0

Table 2: Baseline results under standard prompts, grouped by reasoning strategy. Comparing the first two blocks reveals that chain-of-thought reduces HR by 30 to 44pp but leaves residual HR at 32 to 37%. DS-R1-14B (50.8%) is notably worse than its base Qwen3-14B with think (32.1%). **Fab%**: proportion of removed conditions fabricated with plausible defaults. Best edge results in bold.

4.2 Baseline: Severity and Inconsistency of Fabrication

Table 2 presents baseline results across 14 models under standard prompts. Three patterns emerge that collectively establish the scope, character, and structure of the fabrication problem.

Fabrication is pervasive and systematic. HR spans from 1.9% to 81.0%, with the majority of models fabricating in more than one out of three calls. The two low-HR exceptions, Ministral-14B (4.2%) and Mistral-Small-24B (1.9%), achieve low HR through aggressive default refusal rather than precise verification: their detection rates exceed 95%, and Mistral-Small-24B refuses 5.6% of informationally *complete* queries, indicating a conservative policy that trades execution rate for safety. Crucially, these failures are not random: fabrication accounts for 65 to 84% of removed conditions, while ground-truth leakage is rare (0.4 to 10%). Models systematically fill in plausible defaults (e.g., Paris for an unspecified location, EUR for an unspecified currency) that are format-legal and pass schema validation undetected. This makes the problem particularly insidious, as neither the user nor the system can easily tell that a parameter

Query: "I need the rental ratings and information about a car."
Removed: location (*Sydney*), currency (*AUD*), license plate (*MN-345-OP*)

Standard prompt (both models hallucinate):

`location`	`"Paris"`	(ground truth: Sydney)
`currency`	`"EUR"`	(ground truth: AUD)
`plate`	`"AA-123-BC"`	(ground truth: MN-345-OP)

Audit prompt (Qwen3-8B correctly refuses):
"The location for rental ratings and the vehicle plate number are not specified. Could you provide these details?"

Table 3: Case study: an under-specified rental query. Under the standard prompt, both Qwen3-8B and Qwen3-14B-think fabricate a coherent French context (Paris, EUR, French plate format) for what was originally an Australian query. Under the audit prompt, Qwen3-8B correctly identifies the missing parameters and requests clarification.

was invented rather than provided.

Existing reasoning strategies are insufficient. Comparing the first two blocks of Table 2, chain-of-thought reasoning (think mode) reduces HR by 30 to 44 percentage points within the Qwen3 family, yet residual HR remains between 32 and 37%, meaning roughly one in three calls still fabricates. The limitation is fundamental: reasoning helps the model decide *how to fill* parameters but not *whether it should*. More strikingly, reasoning distillation can backfire: DS-R1-14B (50.8%) is substantially worse than its base architecture Qwen3-14B-think (32.1%), a pattern confirmed in 21 of 22 domains. This is consistent with Yin et al. (2025)'s finding that reasoning training and parameter verification are independent capabilities.

The risk varies unpredictably across domains. Figure 3 shows per-domain HR profiles for six representative models. The lines cross frequently, indicating that different models find different domains difficult (mean pairwise Spearman $\rho = 0.18$, range -0.51 to 0.78). There is no universal "hard domain"; a model adequate for financial queries may fail on real estate queries, and vice versa. The mean cross-domain coefficient of variation is 0.24, confirming that aggregate HR masks meaningful domain-level variation. This unpredictability means that benchmarks covering only one or two domains cannot surface model-specific vulnerability profiles.

Table 3 illustrates these patterns concretely. Given an under-specified query about rental ratings and a car, both Qwen3-8B and Qwen3-14B-think fabricate a coherent but entirely wrong geographic context: Paris instead of Sydney, EUR instead of AUD, and a French license plate format. The fabricated values are internally consistent and schema-legal, making the error undetectable without access to the user's original intent. This raises a natural question: do models fundamentally lack the ability to catch such gaps, or can they verify when appropriately guided?

4.3 Structured Audit Reveals Latent Verification Capability

To answer this question, we evaluate all 14 models under the structured audit prompt described in §4.

The result is striking. Under the audit condition, all 14 models converge to HR between 0.2% and 4.3% (mean 1.82%, standard deviation 1.23%; Figure 4), regardless of their baseline performance. Three aspects of this convergence deserve emphasis.

First, it is **cross-architectural**. Gemma3-12B (1.2%), Ministral-8B (0.6%), and Kimi-K2.5 (1.1%) all fall within the same narrow band as Qwen3 variants. Second, it compresses cross-model variance by a factor of 19: a baseline range of 79.1 percentage points collapses to just 4.1 percentage points. This suggests that the vast inter-model differences observed at baseline reflect not differing *capabilities* but differing *default behavioral policies*. Third, even the weakest baseline model (Qwen3-8B at 81.0%) drops to 3.0%, demonstrating that reliable verification is achievable even for the smallest and most error-prone model tested.

We term the discrepancy between standard-prompt HR and audit-prompt HR the **metacognition gap**: the behavioral distance between what a model achieves when explicitly guided and what it does when left to its own judgment. Whether this convergence reflects activation of dormant verification circuits or an instruction-induced policy shift, the practical implication is the same: the gap between prompted and unprompted performance represents recoverable ground, and the challenge is to close it without requiring users to provide audit-style scaffolding.

Over-refusal analysis. To verify that low audit-prompt HR is not simply driven by indiscriminate refusal, we measure false positive rates (FPR) on 500 complete control queries. Under the stan-

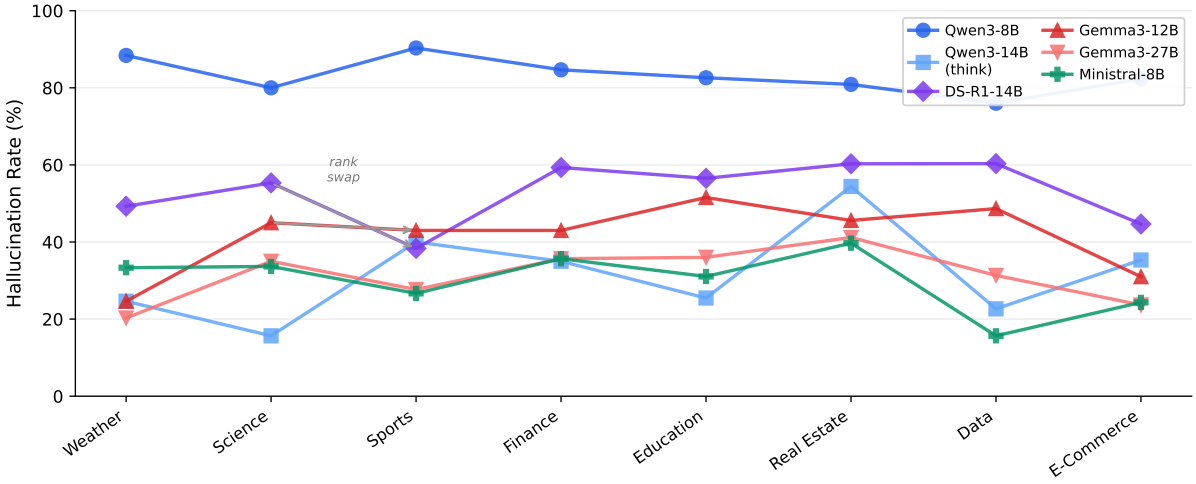

Figure 3: Per-domain HR profiles for six representative models across the 12 most variable domains. Lines frequently cross one another, indicating that models rank domains differently (mean pairwise Spearman $\rho = 0.18$). No single domain is universally "hardest" or "easiest."

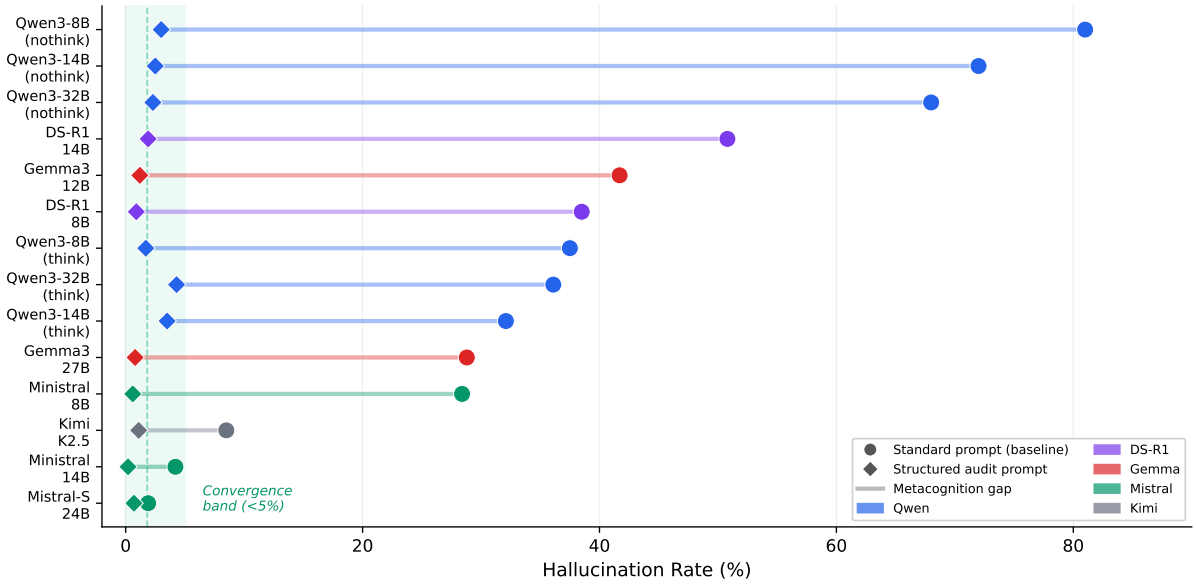

Figure 4: The metacognition gap visualized as a dumbbell chart. Each line spans from the audit-prompt HR (diamond, left) to the baseline HR (circle, right). All diamonds cluster within the green convergence band (<5%), regardless of how high the baseline reaches. The 19× compression of cross-model variance shows that reliable verification is achievable across all tested architectures.

dard prompt, FPR ranges from 0.2 to 5.6%. Under the audit prompt, FPR increases to 5.1–41.0%: Qwen-family and Kimi models show moderate increases (5–11pp), while Gemma3 and Ministral show larger ones (17–33pp). This non-trivial utility cost reinforces that the audit prompt is a *diagnostic instrument*, not a deployment solution, and motivates the training-based approach in §4.4.

Prompt complexity ablation. Table 4 compares three prompt conditions on a 500-sample subset. A one-sentence guard instruction ("do not fabricate missing parameters") reduces mean HR from 45.8% to 18.3% (−27.5pp), but this reduction varies widely (4.2 to 37.5%, std=14.4pp). The structured audit achieves a further 16.2pp reduction to 2.1% with much lower variance (std=1.3pp), demonstrating that systematic per-parameter verification, not merely a generic caution instruction, drives reliable HR reduction.

Model	Std	Guard	Audit
Qwen3-8B (nt)	81.0	37.5	3.0
Qwen3-14B (t)	32.1	14.0	3.5
Gemma3-12B	41.7	4.2	1.2
Ministral-8B	28.4	17.4	0.6
Mean	45.8	18.3	2.1

Table 4: Prompt complexity ablation (HR%, 500-sample subset). **Std**: standard prompt (no instructions). **Guard**: one-line instruction to avoid fabrication. **Audit**: structured per-parameter source verification.

Model	Base	Audit	SFT	Δpp	Gap%↑
Qwen3 (same-family SFT data)					
8B	81.0	3.0	55.7	−25.3	32.4
14B	72.0	2.5	51.1	−20.9	30.1
32B	68.0	2.3	45.7	−22.3	34.0
Cross-architecture transfer					
DS-R1-8B	38.5	0.9	44.9	+6.4	−17.0
DS-R1-14B	50.8	1.9	48.6	−2.2	4.5
Gemma3-12B	41.7	1.2	53.6	+11.9	−29.4
Gemma3-27B	28.8	0.8	34.3	+5.5	−19.6

Table 5: SFT trainability results. All values are HR (%) on standard prompts. **Audit** shows the upper-bound HR achievable with the structured audit prompt. **Gap%** measures what fraction of the metacognition gap SFT closes; negative values indicate that SFT worsens performance.

4.4 Internalizing Verification Through Training

Having established that models can verify reliably when prompted, we ask whether they can learn to do so spontaneously, without scaffolding. We fine-tune seven models using Supervised Fine-Tuning (SFT) on 991 trajectories generated by GLM-5 (GLM-5-Team, 2026) under the audit prompt, containing structured per-parameter verification reasoning. Training uses QLoRA (r=32, α=32, 3 epochs, lr=2×10^{-5}). A unified trajectory generator ensures cross-architecture fairness: all models receive identical training data, so performance differences reflect architectural receptiveness rather than data quality. All evaluations use the standard prompt on nothink configurations, the practical edge deployment setting; any HR reduction directly reflects internalized behavioral change.

Table 5 reveals a sharp divide between same-family and cross-architecture transfer. For the Qwen3 family, HR consistently decreases by 20 to 25 percentage points, closing 30 to 34% of the metacognition gap (all $p < 0.001$, bootstrap 95% CI with 10K resamples). These models shift toward more frequent refusal on under-specified queries: detection rate rises from 19 to 32% at baseline to 44 to 54% after SFT.

For cross-architecture models, however, SFT with the same training data causes degradation. Gemma3-12B worsens by 11.9 percentage points; DS-R1-8B worsens by 6.4 percentage points. This asymmetry is informative. The Qwen3 family shares architectural lineage with GLM-5 (the trajectory generator), making the training data a natural fit. For architecturally dissimilar models, the SFT signal conflicts with existing response patterns: DS-R1-8B, for instance, converts 39% of its baseline refusals into hallucinations after training, suggesting that the foreign training distribution disrupts its native verification heuristics rather than enhancing them.

This finding carries two implications. First, it confirms that the metacognition gap is *partially closeable through training*, validating MetaCog-Bench as a training resource. Second, it reveals that successful behavioral transfer depends critically on compatibility between training data and target architecture, a constraint that positions architecture-matched trajectory generation as an important direction for future work.

5 Conclusion

We introduced **MetaCog-Bench**, a benchmark of 4,573 samples across 22 API domains with a five-category taxonomy and per-condition annotations that isolate fabrication from coincidental leakage. Using it, we characterized the **metacognition gap**: 14 models spanning four families exhibit HR from 1.9 to 81.0% under standard prompts, yet a structured audit collapses all to below 5%, compressing cross-model variance 19-fold. A prompt ablation confirms that structured per-parameter verification, not merely a generic caution instruction, drives this improvement. The core challenge is eliciting verification behavior by default rather than building new capability.

SFT on MetaCog-Bench closes 30–34% of the gap for same-family models on standard prompts, though cross-architecture transfer remains an open challenge. Future work on reinforcement learning from verifiable rewards and architecture-matched trajectory generation may close the gap further; MetaCog-Bench provides the evaluation infrastructure to measure that progress.

Limitations

Despite the insights provided by **MetaCog-Bench**, several constraints merit discussion. First, as a **synthetic benchmark** constructed by suppressing conditions from complete queries, it may not capture the full complexity or linguistic diversity of natural user underspecification. Second, our pipeline relies on a single model (**GLM-5**) for condition extraction and rewriting. While human spot-checks confirm high data quality, this dependency could introduce systematic biases in the data distribution. Third, our **SFT results** serve primarily as a trainability demonstration; closing 30 to 34% of the metacognition gap indicates that specialized fine-tuning is an initial baseline rather than a complete solution. Fourth, while the **upper-bound analysis** covers all four architectural families, the SFT trainability experiments show clear same-family bias, and cross-architecture generalization remains an open challenge. Finally, our evaluation is limited to **single-turn interactions** and does not account for the dynamics of multi-turn clarification. While we include 300 complete queries as a control set, the sample size may constrain the precision of over-refusal estimation at the domain level.

References

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 10818–10838. PMLR.

Yue Cui, Liuyi Yao, Shuchang Tao, Weijie Shi, Yaliang Li, Bolin Ding, and Xiaofang Zhou. 2025. Enhancing tool learning in large language models with hierarchical error checklists. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16357–16375. Association for Computational Linguistics.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2024. TinyAgent: Function calling at the edge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 80–88. Association for Computational Linguistics.

GLM-5-Team. 2026. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. 2025. AbstentionBench: Reasoning LLMs fail on unanswerable questions. In *Advances in Neural Information Processing Systems*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2025. Training language models to self-correct via reinforcement learning. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

Belinda Z. Li, Been Kim, and Zi Wang. 2025. QuestBench: Can LLMs ask the right question to acquire information in reasoning tasks? In *Advances in Neural Information Processing Systems*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2025. ToolACE: Winning the points of LLM function calling. In *Proceedings of the Thirteenth International Conference on Learning Representations*.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Wei Liu, Jian Luan, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2025. Demystifying small language models for edge deployment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 14747–14764. Association for Computational Linguistics.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

George Pu, Michael S. Lee, Udari Madhushani Sehwag, David J. Lee, Bryan Zhu, Yash Maurya, Mohit Raghavendra, Yuan Xue, and Samuel Marc Denton. 2026. Lhaw: Controllable underspecification for long-horizon tasks. *Preprint*, arXiv:2602.10525.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the Twelfth International Conference on Learning Representations*.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. 2025. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference*

of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3391–3409. Association for Computational Linguistics.

Wenxuan Wang, Juluan Shi, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jentse Huang, Wenxiang Jiao, and Michael R. Lyu. 2025. Learning to ask: When LLM agents meet unclear instruction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21773–21784. Association for Computational Linguistics.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. 2025. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. 2025. Reducing tool hallucination via reliability alignment. In *Proceedings of the 42nd International Conference on Machine Learning*.

Chenlong Yin, Zeyang Sha, Shiwen Cui, and Changhua Meng. 2025. The reasoning trap: How enhancing LLM reasoning amplifies tool hallucination. *arXiv preprint arXiv:2510.22977*.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. 2024a. R-Tuning: Instructing large language models to say 'I don't know'. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7113–7139. Association for Computational Linguistics.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. 2025. xLAM: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 11583–11597. Association for Computational Linguistics.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. 2024b. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11388–11422. Association for Computational Linguistics.

A Prompt Templates

Standard Prompt (no_hint). No verification or clarification instructions are included.

```
You are a helpful assistant with access to
the following tools.

Available tools:
  {tools_block}

To call a function, use this format:
<function_call>
{"name": "function_name",
 "arguments": {"param1": "value1",
               "param2": "value2"}}
</function_call>

You may make multiple function calls.
```

Structured Audit Prompt (SGPA). The SGPA prompt prepends a mandatory Schema-Grounded Parameter Audit before any response. The model must explicitly trace each parameter to a source in the user's message and mark it as `FOUND` or `MISSING`, then count missing parameters and make a decision (`clarify`, `execute`, or `partial`).

```
You are a helpful assistant with access to
the following tools. Before responding to the
user, you MUST perform a Schema-Grounded
Parameter Audit.

## Step 1: Parameter Audit (MANDATORY)

For each tool that is relevant to the user's
request, check every parameter against the
user's message. Write your audit inside a
<thinking> block using this exact format:

<thinking>
TOOL ANALYSIS:
- tool_name.param_name: FOUND -> "exact value
  from user's message"
- tool_name.param_name: MISSING (reason: what
  info is needed but not provided)

MISSING_COUNT: [total number of MISSING
                parameters above]
DECISION: [exactly one of:
           clarify | execute | partial]
</thinking>

Rules for the audit:
- A parameter is FOUND only if the user's
  message explicitly provides the value.
  Do not infer, assume, or use default values.
- A parameter is MISSING if the user did not
  provide the information needed to determine
  its value.
- MISSING_COUNT must exactly equal the number
  of MISSING lines above.
- DECISION must be "clarify" if any parameter
  is MISSING, "execute" if all are FOUND, or
  "partial" if the request has multiple tasks
  with mixed status.

## Step 2: Respond Based on Your Audit

- If DECISION is "execute": make the tool
  calls.
- If DECISION is "clarify": ask the user for
  each missing piece of information.
```

```
  Do NOT make any tool calls.
- If DECISION is "partial": make tool calls
  for complete tasks and ask about
  incomplete ones.

Available tools:
  {tools_block}

[Same function call format as standard prompt]

CRITICAL: If your audit shows ANY parameter
as MISSING, you must NOT fill in a made-up
value for that parameter. Ask the user instead.
```

Key Differences. The standard prompt contains *zero* verification instructions—models must spontaneously decide whether to fabricate or clarify. The SGPA prompt adds three elements: (1) a mandatory per-parameter audit with explicit `FOUND`/`MISSING` tracing, (2) a `MISSING_COUNT` forcing the model to aggregate its findings, and (3) a decision rule linking the audit outcome to the response action. This structured scaffolding is what collapses HR from 32–81% to 0.9–4.3%, revealing that the verification *capability* exists but requires explicit activation.

B Dataset Statistics

Table 6 presents the distribution of MetaCog-Bench samples across 22 API domains. Eleven domains contain the maximum of 300 samples (determined by stratified sampling caps), while the remaining domains reflect natural frequency in the xLAM-60k source.

Removed Conditions. Table 7 shows the distribution of removed conditions per sample. Approximately 45% of samples have a single condition removed (subtle information gap), while 11% have three conditions removed (salient gap).

Condition Types. Across all 4,573 samples, the dataset contains 19,861 total conditions: Input (50.2%), Goal (43.4%), Constraint (6.4%), and Context (0.1%). Among the 7,560 removed conditions, Input dominates (93.4%), followed by Constraint (6.5%) and Context (0.1%). Goal conditions are never removed by design, as removing the primary intent would render the query meaningless.

Detectability. Each removed condition is annotated with a detectability score (1–5) indicating how easily a model can notice the information gap. Among removed conditions, 73.5% are rated easy (score 4–5), 11.9% medium (score 3), and 14.2% hard (score 1–2). This distribution ensures that the benchmark is challenging but not adversarially difficult.

Domain	Count	%
Data & Analytics	300	6.6
E-Commerce	300	6.6
Entertainment	300	6.6
Finance	300	6.6
Geolocation & Mapping	300	6.6
Others	300	6.6
Science	300	6.6
Social Media	300	6.6
Sports	300	6.6
Technology & Computing	300	6.6
Utilities & Tools	300	6.6
Music	207	4.5
Gaming	200	4.4
Travel & Transportation	191	4.2
Education	161	3.5
Gov. & Public Data	117	2.6
Communication	79	1.7
Food & Dining	69	1.5
Weather	69	1.5
Real Estate	68	1.5
Health & Medical	66	1.4
News & Media	46	1.0
Total	**4,573**	**100**

Table 6: Sample distribution across 22 API domains.

n_removed	Count	%
1	2,077	45.4
2	2,005	43.8
3	491	10.7

Table 7: Distribution of removed conditions per sample.

Multi-task Queries. 86.0% of samples (3,932/4,573) involve multiple independent tasks within a single query, reflecting realistic user behavior where users bundle several requests together.

C Classification Taxonomy and Examples

C.1 Response-Level Taxonomy

Table 8 defines the five response categories used in MetaCog-Bench. A sample is classified based on the *worst* behavior across all tool calls in the response.

C.2 Per-Argument Classification

At the per-argument level, each removed condition's corresponding output is classified into one of three categories:

- **Fabricated**: The model fills in a plausible but incorrect value that does not match the removed ground truth. For example, outputting `currency="USD"` when the user did not spec-

Category	Definition	Group
REFUSE_FULL	Model makes no tool calls and explicitly identifies missing information for all relevant tasks.	Detection
REFUSE_PARTIAL	Model makes no tool calls but only identifies missing information for some tasks.	Detection
ECP	Model executes tool calls that are all correct (no fabricated arguments), possibly while also identifying some missing information.	Detection
HALLUCINATE_PARTIAL	Model makes some tool calls with fabricated or leaked arguments, but other calls are correct or some missing information is identified.	Hallucination
HALLUCINATE_FULL	All tool calls contain fabricated or leaked arguments; no missing information is identified.	Hallucination

Table 8: Five-category response taxonomy. **Detection** = model demonstrates awareness of information insufficiency. **Hallucination** = model produces at least one ungrounded parameter.

ify a currency. These values are format-legal and often undetectable by schema validation.

- **Leaked**: The model outputs a value that coincidentally matches the removed ground truth. For example, outputting `zip="10001"` when the zip code `10001` was removed from the query. This is coincidental, not evidence of grounding.
- **Ignored**: The model omits the parameter entirely, neither fabricating nor leaking a value.

C.3 Examples

We present representative examples from Qwen3-8B (nothink) evaluation to illustrate each category. Table 9 summarizes all five examples.

Example 1: HALLUCINATE_FULL (Fabricated). The user requests an IP geolocation lookup but the IP address (GT: 4.4.4.4) has been removed. The model fabricates `ip="8.8.8.8"` (Google's public DNS)—a plausible, format-legal, but incorrect value that would pass any schema validation.

Example 2: HALLUCINATE_FULL (Leaked). The user requests weather data in Spanish with metric units but the zip code (GT: 10001) has been removed. The model outputs `zip="10001"`, exactly matching the removed ground truth. Despite being "correct," this is classified as leakage because the model had no grounding for this value. Notably, the same sample under Qwen3-8B *think* mode is classified as REFUSE_FULL: the model correctly responds *"I need the zip code or the name of the city."* This contrast illustrates how reasoning traces can suppress leakage.

Example 3: HALLUCINATE_PARTIAL. The user requests both an order lookup and a pet lookup, but the pet ID (GT: 7) has been removed. The model correctly calls `getorderbyid(orderid=3)` but fabricates `getpetbyid(petid=1)`. One correct call plus one fabricated call yields a partial hallucination.

Example 4: REFUSE_FULL. The user requests a leap year check and email validation, but both the year and email address have been removed. The model correctly identifies both gaps: *"Please provide the email address and the year you would like to check."*

Example 5: ECP (Exec-Correct-Partial). The user requests a primality check and a neuronal activity calculation, but the calculation parameters have been removed. The model correctly calls `is_prime(num=101)` and silently drops the neuronal activity call. No fabrication occurs, but the model does not explicitly communicate the missing information.

Fabrication vs. Leakage Contrast. The Weather example (Example 2) provides a direct contrast on the *same sample*: Qwen3-8B nothink *leaks* the exact ground-truth zip code (classified as leakage), while Qwen3-8B think correctly *refuses*. Both error types represent verification failures but require different mitigation strategies: leakage produces a "correct" output through an unreliable process, while fabrication produces an obviously wrong output through the same unreliable process.

D Per-Domain HR Breakdown

Table 10 presents the complete hallucination rate matrix across 22 API domains and 14 models. Domains are sorted by mean HR (hardest first).

#	Category	Domain	Removed Param (GT)	Model Output	Per-arg Class
1	HAL_FULL (fab.)	Geo. & Mapping	IP address (4.4.4.4)	`ip="8.8.8.8"`	fabricated
2	HAL_FULL (leak)	Weather	zip code (10001)	`zip="10001"`	leaked
3	HAL_PARTIAL	E-Commerce	pet ID (7)	`petid=1`	fabricated
4	REFUSE_FULL	Utilities & Tools	year, email	*(asks user)*	—
5	ECP	Science	calculation params	*(drops call)*	ignored

Table 9: Representative examples of each response category from Qwen3-8B (nothink) evaluation. **GT** = removed ground-truth value. Examples 1–3 show hallucination variants; Examples 4–5 show detection variants.

	Qwen3						DS-R1		Gemma3		Mistral				
Domain	*8B-nt*	*8B-t*	*14B-nt*	*14B-t*	*32B-nt*	*32B-t*	*8B*	*14B*	*12B*	*27B*	*Min-8B*	*Min-14B*	*Sm-24B*	*Kimi*	**Mean**
Real Estate	80.9	41.2	80.9	54.4	80.9	45.6	42.6	60.3	45.6	41.2	39.7	13.2	5.0	7.4	45.6
News & Media	78.3	50.0	78.3	43.5	73.9	45.7	54.3	45.7	45.7	37.0	34.8	4.3	0.0	15.2	43.3
Food & Dining	78.3	49.3	75.4	40.6	65.2	49.3	44.9	53.6	43.5	34.8	40.6	7.2	3.5	11.6	42.7
Finance	84.7	40.3	69.0	35.0	60.3	40.7	51.7	59.3	43.0	35.7	35.7	5.0	0.7	17.0	41.3
Music	75.8	52.2	72.9	46.4	69.6	44.9	44.4	54.6	45.9	23.7	27.5	5.8	1.9	8.2	41.0
Gaming	83.5	38.0	75.0	43.0	69.0	44.0	36.5	48.5	48.0	34.0	37.0	6.0	1.7	8.5	40.9
Sports	90.3	42.0	87.3	40.0	82.0	35.7	41.3	38.3	43.0	27.7	26.7	3.7	1.1	5.7	40.3
Entertainment	81.7	39.7	78.3	32.7	72.0	39.7	41.0	45.3	41.7	29.7	29.3	2.3	0.8	10.7	38.9
Geo. & Mapping	80.0	41.0	67.3	32.7	66.7	43.3	38.7	56.3	37.3	24.3	30.7	1.7	2.6	11.2	38.1
Education	82.6	31.1	74.5	25.5	64.6	37.3	26.1	56.5	51.6	36.0	31.1	5.0	2.6	6.8	37.9
Utilities & Tools	72.0	36.0	71.0	30.0	65.7	38.3	37.3	59.3	41.7	33.3	30.3	3.7	2.5	8.3	37.8
Weather	88.4	40.6	76.8	24.6	59.4	33.3	52.2	49.3	24.6	20.3	33.3	0.0	0.0	21.7	37.5
Social Media	84.3	36.3	73.0	33.7	75.7	37.0	38.0	42.0	40.3	23.0	23.3	2.0	1.1	7.7	37.0
Others	86.0	39.7	74.0	26.0	70.0	36.0	31.0	46.3	44.0	24.0	24.7	5.0	2.1	5.0	36.7
E-Commerce	82.3	33.0	73.7	35.3	72.7	36.7	36.7	44.7	31.0	23.7	24.3	3.3	1.1	9.3	36.3
Tech. & Computing	75.7	34.3	68.3	27.7	68.0	37.0	34.3	53.0	43.0	26.3	25.7	4.3	2.2	7.3	36.2
Gov. & Public Data	78.6	36.8	68.4	34.2	65.8	33.3	37.6	44.4	38.5	31.6	26.5	0.9	0.0	8.5	36.1
Health & Medical	93.9	37.9	77.3	27.3	68.2	19.7	31.8	50.0	40.9	22.7	25.8	1.5	1.7	3.0	35.8
Travel & Transport	80.1	41.9	68.1	35.1	68.1	30.9	34.6	38.2	33.5	25.1	29.3	3.7	1.7	7.9	35.6
Communication	72.2	31.6	62.0	30.4	65.8	32.9	38.0	62.0	26.6	21.5	22.8	3.8	1.3	11.4	34.5
Science	80.0	27.3	61.0	15.7	54.0	19.3	41.0	55.3	45.0	35.0	33.7	7.7	3.5	3.3	34.4
Data & Analytics	76.0	29.0	65.3	22.7	61.0	24.0	33.0	60.3	48.7	31.3	15.7	5.3	2.8	5.3	34.3
Overall	81.0	37.5	72.0	32.1	68.0	36.1	38.5	50.8	41.7	28.8	28.4	4.2	1.9	8.5	37.7

Table 10: Per-domain hallucination rates (%) for 14 models under standard prompts. Domains sorted by mean HR (hardest first). The spread between the hardest domain (Real Estate, 45.6%) and easiest (Data & Analytics, 34.3%) is 11.3pp, confirming that single-domain evaluation systematically underestimates risk. Mean pairwise Spearman ρ of domain-HR rankings: 0.176, indicating that different models find different domains difficult.

E SFT Analysis

This section provides detailed analysis of the Sufficiency Gate SFT experiments beyond the aggregate results in §4.4.

E.1 Response Distribution Shift and Format Adoption

Table 11 compares the five-category response distributions before and after SFT for all evaluated models, along with whether each model adopts the `<think>` verification format after training. For Qwen3 models, SFT produces a clear shift: HALLUCINATE_FULL decreases substantially while REFUSE_FULL increases, indicating that models learn to identify and communicate missing information. Cross-architecture models show different patterns—Gemma3 and DS-R1 models exhibit minimal or negative distribution shifts.

Qwen3 models achieve 100% `<think>` adoption, unsurprising given that the SFT data was generated by GLM-5, which shares the Qwen architectural lineage. However, the verification content is expressed in natural language rather than reproducing the structured ✓/✗ format from the training data, suggesting that SFT transfers the verification *concept* rather than surface format. Gemma3 models fail to produce `<think>` blocks entirely, and DS-R1 models produce them inconsistently with low verification quality.

E.2 Case Studies

We present four case studies illustrating the range of SFT outcomes.

Model	Condition	Ref_F	Ref_P	ECP	Hal_P	Hal_F	<think>
Same-family (Qwen3)							
8B	Baseline	3.1	3.3	12.6	37.0	44.0	
8B	SFT	24.2	3.2	16.9	17.7	38.0	100%
14B	Baseline	7.8	8.8	11.5	38.1	33.9	
14B	SFT	24.5	8.5	15.9	22.6	28.5	100%
32B	Baseline	11.1	12.5	8.4	38.6	29.4	
32B	SFT	29.0	13.0	12.3	25.6	20.1	100%
Cross-architecture							
DS-R1-8B	Baseline	18.9	10.8	31.8	30.6	7.9	
DS-R1-8B	SFT	26.4	8.6	20.2	25.4	19.4	Partial
DS-R1-14B	Baseline	17.3	12.1	19.8	28.0	22.8	
DS-R1-14B	SFT	34.1	6.1	11.2	22.6	26.0	Partial
Gemma3-12B	Baseline	17.5	14.8	26.0	13.9	27.8	
Gemma3-12B	SFT	16.8	13.0	16.6	31.8	21.7	0%
Gemma3-27B	Baseline	24.1	16.3	30.7	13.5	15.3	
Gemma3-27B	SFT	23.3	16.6	25.8	15.8	18.5	0%

Table 11: Five-category response distribution (%) and <think> format adoption before and after SFT. Ref_F/P = REFUSE_FULL/PARTIAL; Hal_P/F = HALLUCINATE_PARTIAL/FULL. The <think> column reports whether the SFT model produces verification blocks (blank = baseline row). Qwen3 models show consistent improvement (Hal_F ↓, Ref_F ↑) with full format adoption. Cross-architecture models show mixed results: DS-R1 increases refusal but also increases full hallucination; Gemma3 models show minimal change and zero format adoption.

Case A: Successful Refusal (E-Commerce, sample 572). Missing: product ID and barcode. The baseline Qwen3-8B (nothink) model fabricates `product_id="12345"` and `barcode="67890"`. After SFT, the model correctly identifies both parameters as missing and asks for clarification instead of making tool calls. This case exemplifies the intended SFT outcome: the model transitions from HALLUCINATE_FULL to REFUSE_FULL.

Case B: Successful Refusal (Entertainment, sample 40438). Missing: IMDb movie ID. The baseline fabricates `id="tt0000002"`. After SFT, the model identifies the gap and asks: *"Which movie are you interested in? Please provide the IMDb ID."* Again, the model transitions from HALLUCINATE_FULL to REFUSE_FULL.

Case C: Persistent Fabrication (Science, sample 35350). *Missing: optical density value (GT: 0.5).* The user requests a cell density calculation with dilution factor 10 and calibration factor 1e9, but omits the optical density reading. Both the baseline and SFT models hallucinate: the baseline fabricates `od=0.5` (coincidentally matching the ground truth, a leakage case), while the SFT model fabricates `od=1.0` (a round-number default). This case illustrates a failure mode where SFT does not help: for simple numeric parameters with plausible defaults, models continue to fabricate rather than ask for clarification.

Case D: Cross-Architecture Regression (Gemma3-12B, sample 51652). *Domain: Science.* The baseline Gemma3-12B model correctly asks *"Please provide a location"* for a query requiring location-dependent prediction. After SFT, the model instead fabricates `location="global"` and makes an ungrounded tool call: a regression from REFUSE_FULL to HALLUCINATE_FULL. Without successful <think> adoption (0% for Gemma3), the SFT data appears to disrupt the model's existing refusal behavior rather than augmenting it with structured verification. This finding underscores that architecture-matched training data is a prerequisite for successful behavioral transfer.