TRACE: Business Rule-Grounded Reasoning Curriculum for Knowledge-Preserving Parametric Tool Retrieval in Enterprise LLMs

Sai Shruthi Sistla* **Ashutosh Hathidara*** **Christopher Toukmaji**
Mayank Shrivastava **Karthikeyan Asokkumar**

SAP Labs

{sai.shruthi.sistla, ashutosh.hathidara, chris.toukmaji,
mayank.shrivastava02, karthikeyan.asokkumar}@sap.com

Abstract

Parametric retrieval enables LLMs to retrieve tools implicitly by assigning each API a unique virtual token and training the model to generate it via constrained beam search (Wang et al., 2025). Toolsense (Anonymous, 2026) shows that this regime has two critical drawbacks: it destroys parametric tool knowledge during training, and its beam-search decoding is too slow for real-time deployment. We introduce TRACE (**T**ool **R**etrieval via **A**ugmented **C**hain-of-thought and **E**nterprise rules), a two-stage curriculum that resolves this dissociation. Stage 1 reuses the multi-format memorization SFT from ToolSense to seed tool knowledge with LoRA. Stage 2 is our core contribution: the model is trained to emit a thinking trace before producing a JSON list of tool tokens, using two data sources — RRB pairs from ToolSense and queries synthesized to target business rules curated by domain experts — both augmented with reasoning traces. This training objective preserves Stage 1 MCQ and QA probing accuracy while enabling single-beam greedy decoding at production latency. Evaluated on a combined enterprise catalog of **8,300+** tools across two enterprise product lines, TRACE training for Stage 2 not only preserves but *improves* tool understanding: MCQ accuracy gains **+3.2 pp** and QA probing gains **+9 pp** over Stage 1. On retrieval, TRACE achieves ∼**86%** recall on Domain A and ∼**60%** on Domain B — compared to embedding baseline performance of ∼27% & ∼52% — both with single-beam greedy decoding, making it directly deployable at production latency.

1 Introduction

Routing user queries to the right API across thousands of proprietary enterprise tools is a core challenge for AI copilots. Embedding-based retrieval (Karpukhin et al., 2020) treats this as semantic matching, but plateaus at low recall on terse, overlapping tool descriptions (§4.3) and *critically* never internalizes tool knowledge, foreclosing richer downstream capabilities. Parametric tool retrieval (Wang et al., 2025) in LLMs offers solution to both problems, but its retrieval training objective catastrophically destroys parametric tool knowledge in the process (Anonymous, 2026).

In our production enterprise AI copilot serving ∼**39k** monthly active users, embedding-based retrieval accounts for ∼**60%** of incorrect tool retrievals, the dominant failure mode at scale and a hard bottleneck for the entire system, since no downstream component can recover from retrieving the wrong tool. Analysis of 609 stratified sampled sessions reveals two recurring patterns: ∼**81.7%** of failures involve many (> 10) semantically overlapping tools that require domain business rules to disambiguate — API deprecations, versioning changes, and domain-specific routing constraints — that embedding models cannot track, leaving both the retriever and any downstream model blind to the rules that govern correct tool selection. These failure modes cannot be fixed by improving the embedding model alone; they demand a retrieval system that can efficiently *reason* about tools.

ToolGen (Wang et al., 2025) introduced a paradigm to enable LLMs to retrieve tools implicitly by assigning each tool a unique virtual token string (e.g. «WeatherAPI/GetForecast») and training the model to generate that token given a user query. Constrained beam search over a prefix trie of valid token strings enables the tool retrieval, given a query. Later, ToolSense (Anonymous, 2026) performed empirical analysis also on tool representation made of hierarchical tokens (e.g. «WeatherAPI»«GetForecast»). ToolGen achieves a high recall of more than 90% on in-distribution benchmark for the ToolBench (Qin et al., 2024) catalog of ∼47,000+ tools.

ToolSense (Anonymous, 2026) introduced

*Equal contribution.

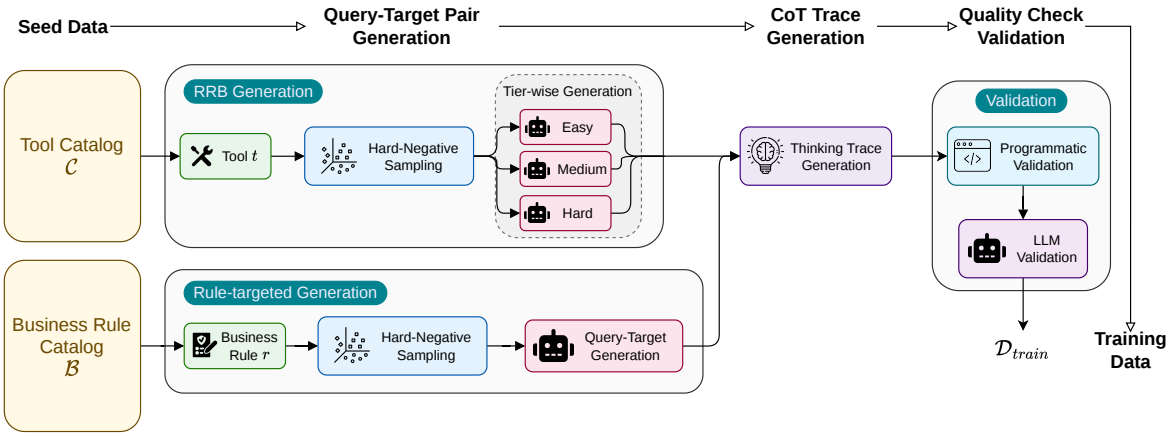


Figure 1: Methodology for Generating Data $\mathcal{D}_{train}$ for Stage 2 TRACE training

method to synthesize 3 benchmarks: Realistic Retrieval Benchmark (RRB) to evaluate retrieval performance on out-of-distribution queries, MCQ probing benchmark to assess multiple-choice, and QA probing benchmark to evaluate factual yes/no question answering. Empirical analysis on the same ToolBench catalog revealed that the model capable of high retrieval recall on the in-distribution evaluation benchmark doesn't generalize to the out-of-distribution RRB benchmark, and more importantly, the retrieval training to map queries to token strings catastrophically destroys the model's parametric tool knowledge as measured by MCQ and QA probing. This dissociation between retrieval performance and parametric tool knowledge raises a critical question: *can we design a training curriculum that preserves tool knowledge while enabling retrieval?*

We introduce TRACE, a two-staged curriculum that resolves the knowledge-retrieval dissociation to enable faithful retrieval with reasoning at production scale. LLM is trained to reason-and-retrieve using data synthesized with business grounded & ungrounded query-reasoning pairs, and the inference is enabled at single beam greedy decoding that helps reduce latency to meet production requirements. On an 8,283-tool enterprise catalog, TRACE achieves $\sim$ 73% retrieval recall on combined tools from two domains while preserving parametric tool knowledge (**+28 pp** $\text{MCQ}_{\text{expert}}$ over the non-reasoning baseline), with $\sim$200$\times$ higher throughput than constrained beam search.

To summarize, our contributions are threefold: (1) **TRACE curriculum**: a two-stage training recipe that preserves tool knowledge while delivering competitive retrieval performance with single-beam greedy decoding at production latency; (2) **Business rule grounding**: a data augmentation mechanism that injects enterprise business rules into reasoning traces, lifting retrieval on semantically overlapping tools by up to +23 pp; (3) **Production deployment study**: end-to-end validation on a combined catalog of 8,300+ proprietary enterprise tools across two business domains, establishing reasoning retrieval as the knowledge-safe path for enterprise AI copilots.

2 Related Work

Tool retrieval and function calling. A large body of work trains LLMs to use external tools, ranging from function-calling fine-tuning (Schick et al., 2023; Tang et al., 2023) to full API-call generation (Patil et al., 2024; Qin et al., 2024). These approaches assume the correct tool is either provided in context or selected from a small candidate set; they do not address *tool retrieval* — identifying the right tool from a large catalog where the correct answer is unknown. This retrieval step is a distinct and under-studied problem: catalog size makes in-context enumeration infeasible, and semantic overlap between tools makes embedding-based matching unreliable. Wang et al. (2025) close this gap with ToolGen, which maps each tool to a unique virtual token and trains the model to generate it via constrained beam search over a prefix trie. ToolSense (Anonymous, 2026) shows that such training systematically destroys learned knowledge about tool catalog, preventing any downstream pos-training that relies on this knowledge. TRACE addresses this blind spot: it treats tool knowledge preservation as a first-class training objective, while eliminating the need for

an external retriever at inference time.

Reasoning traces in supervised fine-tuning. Chain-of-thought prompting (Wei et al., 2022) demonstrated that eliciting intermediate reasoning steps substantially improves LLM performance. Subsequent work distils this capability through fine-tuning: STaR (Zelikman et al., 2022) bootstraps rationale generation iteratively, Orca (Mukherjee et al., 2023) distils GPT-4 explanation traces into smaller models, and DeepSeek-R1 (Guo et al., 2025) incentivises reasoning via reinforcement learning. These methods target generic reasoning capabilities. TRACE applies reasoning SFT to parametric retrieval to *preserve* parametric tool knowledge that the constrained retrieval objective destroys.

Catastrophic forgetting and knowledge retention. Fine-tuning LLMs on narrow task objectives is known to degrade previously acquired knowledge (Luo et al., 2025). Biderman et al. (2024) show that LoRA forgets less than full fine-tuning, providing theoretical grounding for our Stage 1 design choice. Anonymous (2026) and Mallen et al. (2023) establish probing evaluation as a reliable measure of parametric factual knowledge, which we adapt to the tool-understanding setting. TRACE leverages these insights directly: LoRA at Stage 1 buffers knowledge retention across sequential fine-tuning stages, and MCQ/QA probing serves as our primary diagnostic for the knowledge-retrieval dissociation.

3 TRACE: Two-Stage Curriculum with Reasoning Traces

TRACE trains an LLM to *reason-then-retrieve*: given a user query, the model first emits a thinking trace that deliberates over candidate tools in its parametric memory and the business rules that govern them, then commits to a JSON list of virtual tool tokens. Training follows a two-stage curriculum. **Stage 1** (§3.2) seeds parametric tool knowledge via multi-format memorization SFT with LoRA, reusing the recipe of ToolSense (Anonymous, 2026). **Stage 2** (§3.3) is our core contribution: a reasoning-augmented retrieval SFT trained on a corpus $\mathcal{D}_{train}$ synthesized by the three-phase pipeline in Figure 1 — query-target pair generation, CoT trace generation, and quality-check validation. We first formalize the setting and seed data (§3.1) before describing each stage.

3.1 Problem Setup and Seed Data

We are given an enterprise tool catalog of $|N| = 8{,}283$ tools spanning two domains: Domain A (HR; 918 tools) and Domain B (Finance; 7,365 tools),

$$\mathcal{C} = \{\, t_i = (\text{name}_i, \text{desc}_i) \,\}_{i=1}^{|N|}, \qquad (1)$$

where each tool t is assigned a unique virtual token v_t appended to the model vocabulary, following the parametric retrieval paradigm (Wang et al., 2025). In addition and unique to the enterprise setting, we are given a business rule catalog of size $|M| = 123$,

$$\mathcal{B} = \{\, r_j = \big(\mathcal{T}(r_j),\ \text{text}(r_j)\big) \,\}_{j=1}^{|M|}, \qquad (2)$$

curated by domain experts. Each rule r_j is a pair consisting of (i) a *confusible tool set* $\mathcal{T}(r_j) \subseteq \mathcal{C}$, a small group of tools whose surface descriptions overlap heavily and differ only in fine-grained, domain-specific details, and (ii) a natural-language rule body $\text{text}(r_j)$ that specifies how to disambiguate among the tools in $\mathcal{T}(r_j)$, capturing API deprecations, versioning changes, and domain-specific routing constraints (e.g., "for product line X, requests about year 2024 and later must route to API ABC and earlier should route to API PQR."). These rules are precisely the knowledge that is only limited to domain-experts and domain-developers, and is not inferable from tool descriptions alone, making them critical for correct retrieval.

Given a query q, the retrieval task produces the tool set $\hat{A}(q) \subseteq \mathcal{C}$. TRACE models this as conditional generation of a trace–answer pair $(\hat{z}, \hat{y})$, where $\hat{z}$ is a free-form thinking trace and $\hat{y}$ is an ordered list of virtual tool tokens $[\, v_x : x \in \hat{A}(q) \,]$. Later, this inference result is compared with the corresponding ground truth y and $A(q)$.

3.2 Stage 1: Multi-Format Memorization

Stage 1 seeds parametric tool knowledge by fine-tuning the base LLM on multiple complementary views of every tool $t \in \mathcal{C}$, following the multi-format memorization recipe of ToolSense (Anonymous, 2026) (details in Appendix A). The objective is not retrieval but *internalization*: by Stage 2, every v_t must be internalized in the model's representation space, tightly bound to the tool's semantics, name, and surface form.

3.3 Stage 2: Reasoning-Augmented Retrieval

Stage 2 trains the model to emit a thinking trace $\hat{z}$ before committing to an ordered list of virtual-tokens $\hat{y}$. Its training corpus $\mathcal{D}_{\text{train}}$ is synthesized by the three-phase pipeline of Figure 1; we summarize the design choices that drive performance and defer mechanics to Appendix B.

Two-branch query generation. We synthesize (q, A) pairs from two complementary branches. *(i) RRB generation* instantiates the ToolSense RRB pipeline (Anonymous, 2026) over $\mathcal{C}$: for each anchor tool we form a hard-negative pool from its top-K nearest neighbours under a sentence encoder, and prompt a generator LLM at three difficulty tiers ($|A|{=}1$, $|A| \in \{2, 3\}$, $|A| \geq 4$) to elicit the ambiguity spectrum of real user queries. *(ii) Rule-targeted generation* iterates over every $r \in \mathcal{B}$ and builds its pool from $\mathcal{T}(r)$ together with each confusible's nearest neighbours. For each rule we synthesize queries in three phrasing styles — *explicit* (keywords directly signal the rule), *implicit* (the rule applies but must be inferred from context without obvious signals), and *exception* (the query appears to match the rule's primary tool but an edge case redirects to a confusable). Such rule-targeted generation densifies coverage of domain understanding through user-like queries that require business rules to disambiguate between confusables.

Trace generation: name-token coupling. A teacher LLM produces a reference trace z following a fixed deliberative schema: scope the user's access permissions, identify and compare relevant APIs, narrow to endpoint candidates, apply the governing business rule (quoting it by name), and commit to the final tool selection. Both in reference z and the answer reference tools, the tool names at training time are replaced by its virtual token v_t, yielding the supervision target $o = (\tilde{z},\, y)$ where $\tilde{z}$ is the name-substituted trace and $y = \text{JSON}([\, v_a : a \in A\,])$. This substitution is the mechanism by which Stage 2 *reinforces* Stage 1 knowledge: every v_a appears inside the deliberative context that justifies it ("…the billing document domain is handled by v_{t_1}, but the rule states header-level queries belong to v_{t_2},…"), preventing v_a from decoupling from desc_a.

Validation. Each candidate (q, z, A) passes a two-filter cascade: a deterministic filter enforcing grounding, no-leakage, JSON consistency, and rule attribution; followed by an LLM judge scoring naturalness, label correctness, and *trace faithfulness*. Failures are regenerated with judge feedback up to a retry budget of 5; samples that exhaust retries are dropped. Surviving samples form $\mathcal{D}_{\text{train}} = \{(q_i, z_i, A_i)\}_{i=1}^{D}$. Filter definitions, judge rubric, and yield statistics are in Appendix B.

Both stages use standard autoregressive SFT with prompt-side loss masking, following the ToolSense settings (Appendix C). At inference, TRACE decodes greedily in a single beam — emitting the trace $\hat{z}$ followed by the JSON token list $\hat{y}$ — with no constrained beam search or prefix-trie expansion, making it directly deployable at production latency (Appendix D).

4 Experiments and Results

We evaluate TRACE on a combined enterprise catalog of 8,283 tools and our experiments answer five questions: (1) Does constrained retrieval training destroy tool knowledge as measured by ToolSense? (2) Does TRACE's curriculum resolve this dissociation? (3) How much does business-rule grounding help? (4) Does the recipe generalize across token formats? (5) Does this recipe deliver production-latency with competitive retrieval and preserved knowledge?

4.1 Experimental Setup

All experiments use **Gemma4-E4B-it** (Google DeepMind, 2026) as the base model (Appendix C) with combined tool catalog and metrics are reported per-domain to surface domain-specific behavior.

Evaluation suite. We evaluate retrieval on **PRB** (Production Retrieval Benchmark), a hand-curated set of golden production query-tool pairs from our enterprise copilot ($n_A{=}131$, $n_B{=}123$). We report recall (R) metric under two decoding views: constrained beam search returns the top-10 candidates per query and reports standard *R@10*; greedy single beam decoding with reasoning lets the model commit to its own answer set $\hat{A}(q)$ and reports *R@gen*. We probe parametric tool knowledge with three benchmarks: $\textbf{MCQ}_{\textbf{ts}}$ and $\textbf{QA}_{\textbf{ts}}$, multiple-choice and yes/no benchmarks synthesized from $\mathcal{C}$ via the ToolSense (Anonymous, 2026) (300+400 and 200+400 questions across the two domains; random baselines 25% and 50%); and $\textbf{MCQ}_{\textbf{expert}}$, a 454-question held-out multiple-choice benchmark authored by domain experts and never seen during training (random baseline 29.3% from non-uniform answer-set sizes; full details in F). $\text{MCQ}_{\text{expert}}$ is our primary knowledge diagnostic; the synthesized probes serve as in-distribution

Axis	Levels
F (format)	a flat, b bare-hier, c wrapped-hier
M (Stage 1)	∅ none, m memo, mr reasoning memo
R (Stage 2)	∅ none, n non-reasoning, r reasoning, r+R reasoning + rules

Table 1: Configuration grid.

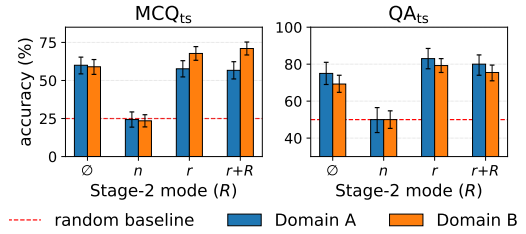


Figure 2: **In-distribution probing.** MCQ_{ts} (left) and QA_{ts} (right) accuracy across the four Stage-2 modes.

controls.

Configuration axes. Every run is described by a triple (F, M, R) enumerated in Table 1: token format F, Stage 1 memorization recipe M, and Stage 2 retrieval recipe R. The three formats are a flat (e.g. «GetForecast»), b bare-hierarchical (e.g. «WeatherAPI»«GetForecast»), and c wrapped-hierarchical (e.g. <tid>«API»«EP»</tid>).

4.2 Knowledge Preservation

Figure 2 reports MCQ_{ts} and QA_{ts} for the four Stage-2 modes on the wrapped-hierarchical format (F=c). Full per-format results are in Appendix E.1.

The pattern is consistent across all three probes. Non-reasoning retrieval training (R=n, the ToolGen recipe) collapses every benchmark to near its random baseline, replicating the dissociation finding of (Anonymous, 2026) on a fundamentally different enterprise catalog and on a held-out probe authored by domain experts. Reasoning retrieval (R=r, r+R) reverses the collapse and even improves on MCQ_{ts} and QA_{ts} over the Stage-1 ceiling. The same pattern carries over to the MCQ_{expert} probe (random baseline 29.3%): ∅: 69.2 n: 33.7 r: 61.5 r+R: 56.4. Unlike the in-distribution probes, MCQ_{expert} targets general domain knowledge rather than tool knowledge specifically, so any tool-focused fine-tuning is expected to incur a small drop; TRACE still substantially outperforms the non-reasoning baseline on this probe.

4.3 Retrieval Performance

We compare TRACE against the non-reasoning retrieval baseline (c, m, n) on the wrapped-hierarchical format under both decoding views, and then map the *rule-grounded data* axis to expose the Pareto trade-off between retrieval recall and expert tool knowledge.

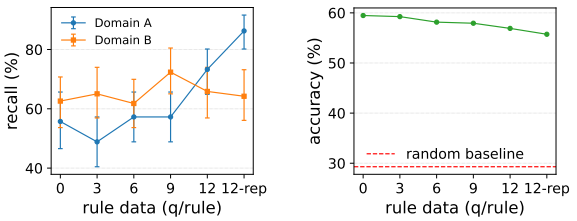
	Domain A		Domain B	
Method (F, M, R)	R@1	R@10	R@1	R@10
text-embedding-3-large	13.74	27.48	27.59	52.71
ToolGen (c, m, n)	45.0	**87.0**	35.8	78.0
TRACE$_{-R}$ (c, m, r)	40.5	74.0	**46.3**	**79.7**
TRACE $(c, m, r{+}R)$	**48.9**	86.3	36.6	77.2

Table 2: Fair retrieval comparison between non-reasoning & reasoning variants under 10-beam constrained decoded inference. F=c for all variants.

	Domain A		Domain B	
Method (F, M, R)	R@1	R@gen	R@1	R@gen
TRACE$_{-R}$ (c, m, r)	32.1	53.4	**56.1**	**67.5**
TRACE $(c, m, r{+}R)$	**51.1**	**85.5**	43.1	60.2

Table 3: Comparison of reasoning retrieval variants under no constrained decoding and single beam with JSON list of tool tokens as output. F=c for all variants.

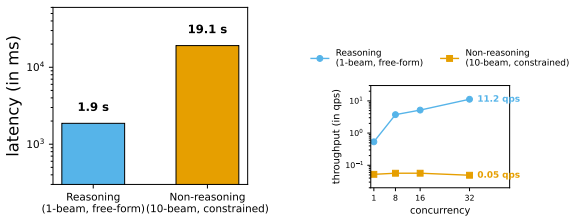
Comparison with non-reasoning retrieval. Tables 2 and 3 report PRB recall for the three relevant Stage-2 variants on F=c.The embedding basline (text-embedding-3-large) achieves only 27.5% R@10 on Domain A and 52.7% on Domain B(Table 2), confirming that semantic matching alone is insufficient. Under apple-to-apple constrained-decoded evaluation (Table 2), TRACE with rule grounding $(c, m, r{+}R)$ matches non-reasoning retrieval baseline on R@10 and slightly improves R@1 on Domain A. Without rule grounding, (c, m, r) trails non-reasoning baseline by $\sim$13 pp R@10 on Domain A even under the same constrained decoder, suggesting rule data is necessary for retrieval-grade recall, not a free add-on. Under single-beam reasoning decoding (Table 3), $(c, m, r{+}R)$ retains 85.5% R@gen on Domain A while Domain B sits at 60.2%. The Domain-B gap reflects *incomplete rule coverage*: $\mathcal{B}$ is curated incrementally and does not yet cover every confusable cluster in the 7,365-tool Domain B catalog, so rule-grounded queries densify disambiguation pressure on the subset of Domain B that has rules at the cost of the subset that does not. The flat-token result confirms this is a coverage artifact rather than a method limitation: rules do not regress Domain B retrieval on F=a (62.6 → 64.2).



(a) Retrieval recall ($R@gen$) on the two domains.

(b) $\text{MCQ}_{\text{expert}}$ accuracy (random baseline 29.3%).

Figure 3: **Stage 2 Rule-grounded Pareto sweep** on F=a, M=m and $R = r+R$ where we vary the number of queries per rule.



(a) Single-user latency at batch=1.

(b) Throughput vs concurrency.

Figure 4: **Latency comparison** of TRACE $(c, m, r+R(12-rep))$ on two inference models using vLLM on a single H200, 100 Domain-A PRB queries (log axes).

Full results appear in Appendix E.2.

4.4 Business Rule Grounding

Figure 3 sweeps rule-data density (0–12 query/rule, plus a replace strategy that swaps RRB pairs for rule-targeted ones) on F=a, measuring PRB recall and $\text{MCQ}_{\text{expert}}$. All-format results in Appendix E.2.

Retrieval scaling. On Domain A, R@gen climbs from 55.7% at 0 q/rule to 73.3% at 12 (+17.6 pp) and to **86.3**% under the replace strategy (+30.6 pp). Domain B, where the rule catalog is still sparse (§4.3), stays within its baseline CI across the sweep — the gain manifests only where rules cover the confusable clusters.

Knowledge cost. $\text{MCQ}_{\text{expert}}$ decreases monotonically from 59.5% at 0 q/rule to 55.7% at 12-*rep* (a 3.8 pp drop). The decline is modest given the probe targets general domain knowledge rather than tool knowledge specifically, and the in-distribution probes MCQ_{ts} and QA_{ts} stay within their CIs across the same axis (Table 6).

Rule citation analysis. To confirm the model genuinely reasons about rules rather than memorizing answer mappings, we analyze the generated traces of $(c, m, r+R(12-rep))$ on Domain A PRB. After rule-grounded training, the rule citation rate rises from 0% (no-rules baseline) to 71% of traces; on traces that cite a rule, recall reaches 94.6%, versus 63.2% on traces that do not.

4.5 Token Format Generalization

Repeating the pipeline across all three token formats from §4.1 confirms the dissociation and its resolution are not format-specific. Non-reasoning retrieval (R=n) collapses $\text{MCQ}_{\text{expert}}$ to near random in every format (34.8%, 30.0%, 33.7% for $F \in \{\text{a}, \text{b}, \text{c}\}$); reasoning retrieval ($R$=r) recovers it to 59.5, 50.9, 61.5. PRB R@gen tracks the same direction (Table 4): F=a and F=c both reach ∼86% on Domain A under the rule-grounded configuration, while F=b lags by ∼ 10 pp on every metric.

The n→r curriculum (bolting reasoning onto a non-reasoning checkpoint) shows a sharper format dependence: on F=a it partially recovers $\text{MCQ}_{\text{expert}}$ to 49.8%, but on F=b and F=c it remains at the random-baseline floor (28.2% and 31.5%). The n→r result is not part of the headline curriculum, but it strengthens the case for avoiding the non-reasoning retrieval objective altogether: damage caused by R=n is, at best, only partly reversible.

4.6 Latency Improvement

We compare end-to-end inference latency on the same TRACE headline checkpoint $(c, m, r+R(12-rep))$ under (i) single-beam free-form decoding with reasoning and (ii) 10-beam constrained decoding without reasoning, both served by vLLM (Kwon et al., 2023) on a single H200 with 100 Domain-A PRB queries. free-form decoding answers a single user in ∼ 1.9 s versus ∼ 19 s for constrained beam-10 (Figure 4a), and at concurrency 32 it sustains **11.2** qps versus **0.05** qps (Figure 4b), a ∼200× throughput gap.

5 Conclusion

We presented TRACE, a two-stage curriculum for parametric tool retrieval that decouples tool knowledge from the retrieval objective via a business-rule-grounded reasoning trace. Stage 1 memorizes the tool catalog; Stage 2 trains the model to reason over business rules before emitting tool tokens. On 8,283 enterprise tools across two domains, TRACE reaches ∼86% recall on Domain A and ∼60% on Domain B under single-beam greedy decoding, while improving Stage-1 MCQ and QA probing

by +7.6 pp and +4.5 pp respectively. TRACE thus delivers parametric tool retrieval that is knowledge-preserving and deployable at production latency.

Limitations

TRACE is evaluated within a deliberately scoped regime for enterprise use case, and three boundaries of that regime are (i) our experiments cover two enterprise domains (HR and Finance) at a single model size, and behavior at higher parameters size or in adjacent verticals (e.g., procurement, healthcare) is an open question. (ii) the benchmarks corresponds to proprietary tool catalogs and thus cannot be released, though we describe the full details in appendix so that the protocol can be reproduced on any analogous catalog. (iii) the business-rule corpus underpinning Stage 2 is authored by domain experts — which is exactly what gives the rules their grounding power, but also means the curation cost is non-trivial; we do not evaluate automatic rule extraction, and identifying when a rule corpus is sufficiently dense (§4.3 suggests $\geq$6 queries per rule) remains a recipe rather than a closed-form criterion. Finally, on the larger Domain B catalog (7,365 tools) greedy decoding still trails constrained-beam; closing this scale gap — through richer rule coverage — is the most concrete future direction.

Ethical Considerations

We conducted experiments within the provisions of the ACL Ethics Policy and relevant research-integrity guidelines. There are, to the best of our knowledge, no remaining ethical risks that have not been addressed.

References

Anonymous. 2026. ToolSense: A diagnostic framework for auditing parametric tool knowledge in LLMs. Under review.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. LoRA learns less and forgets less. *Transactions on Machine Learning Research*. Featured Certification.

Google DeepMind. 2026. Gemma 4 E4B Instruction-Tuned Model. https://huggingface.co/google/gemma-4-E4B-it. Model overview: https://ai.google.dev/gemma/docs/core/model_card_4. License: Apache 2.0.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves

to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *Preprint*, arXiv:2306.05301.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2025. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

A Multi-Format Tool Memorization

For the flat-token format we train on three formats jointly: (i) the forward mapping $\text{desc}\rightarrow v_t$, which grounds the description in the tool token; (ii) the reverse mapping $v_t\rightarrow\text{desc}$, which forces the model to recover the description from the token; and (iii) a discriminative Multi-Choice Tool Selection (MCTS) objective in which, given desc_t, the model must select the gold token v_t from a contrast set of $K+1$ candidates (v_t plus K hard negatives mined from semantically adjacent tools). The reverse and MCTS formats together break the shortcut of name-to-token surface memorization and force genuine description-token alignment. For the hierarchical-token formats, we add two formats that target the API/endpoint factorization: $\text{desc}\rightarrow v_t^{\text{api}}$ and $(\text{desc}, v_t^{\text{api}})\rightarrow v_t^{\text{endpoint}}$.

B Stage 2 Data Synthesis: Full Pipeline

This appendix provides the mechanics deferred from §3.3.

Hard-negative pools. Let $\phi : \mathcal{C} \rightarrow \mathbb{R}^d$ be a sentence encoder. For a tool t we retrieve the top-K neighbours under cosine similarity,

$$\mathcal{H}_K(t) = \underset{t' \in \mathcal{C}\setminus\{t\}}{\arg\,\text{top-K}} \left\langle \phi(t), \phi(t') \right\rangle. \tag{3}$$

The RRB anchor pool is $\mathcal{P}(t) = \{t\} \cup \mathcal{H}_K(t)$. The rule pool is

$$\mathcal{P}(r) \;=\; \mathcal{T}(r) \cup \bigcup_{t \in \mathcal{T}(r)} \mathcal{H}_K(t), \tag{4}$$

which, by construction, contains every tool whose surface form is close enough to a tool in $\mathcal{T}(r)$ to make the problem more challenging — thereby forcing the model to learn finer details of these tools.

Tier definitions (RRB). Three tier-wise sub-pipelines run in parallel for each anchor t, sampled stratified by domain. Each prompts the generator θ_{RRB} with $(t, \mathcal{P}(t), \text{tier}, \mathcal{E})$, where $\mathcal{E}$ is a small set of dynamic few-shot style real user queries to ground the generation towards realistic query distribution: *easy* ($|A|$=1, fully-specified intent), *medium* ($|A| \in \{2, 3\}$, mild ambiguity), and *hard* ($|A| \geq 4$, multi-intent or under-specified). This tiered structure exposes the model to the ambiguity spectrum of real user queries rather than the verbose, fully-specified queries that we rarely encounter in production.

Rule-targeted generator. A separate generator θ_{rule} receives $(r, \mathcal{P}(r), \mathcal{E})$ and is instructed to synthesize queries in three phrasing styles, each targeting a distinct mode in which a rule must be brought to bear at retrieval time:

- **Explicit:** q cites the rule verbatim or near-verbatim, supervising the model to recognize a rule when it is stated outright by the user.
- **Implicit:** q alludes to the rule indirectly — e.g., via a domain artifact (product line, year, region) that the rule conditions on — without naming the rule itself, supervising the model to surface the rule from contextual cues.
- **Violation:** q is deliberately under-specified so that, absent r, the answer set spans multiple tools in $\mathcal{T}(r)$; once r is applied the answer collapses to a single tool. This is the style embedding retrievers cannot resolve at any threshold and the principal target of TRACE's reasoning trace.

Each rule contributes a equal mix of the three styles, and every sample retains a pointer to its source rule r^*.

Trace generator. The teacher θ_{CoT} receives the tuple $(q, A, \mathcal{P}, r^*)$ — where $r^* = \varnothing$ for RRB samples — and follows the deliberative schema in §3.3: (i) restate the user intent; (ii) analyze the plausible candidates from $\mathcal{P}$ and articulate what distinguishes them; (iii) where applicable, state r^* and apply it to eliminate confusables; (iv) commit to the answer set.

Programmatic filter $\mathcal{V}$. A sample is rejected if any of the following fail:

- *grounding*: $A \subseteq \mathcal{P}$ and every tool referenced in z exists in $\mathcal{P}$ (no hallucinated tools);
- *leakage*: no answer tool name appears verbatim in q, i.e., $\forall\, t' \in A : \text{name}(t') \notin q$;
- *consistency*: the answer set committed at the end of z exactly matches y, and y is well-formed JSON over valid virtual tokens;
- *rule attribution*: for rule-targeted samples, z explicitly invokes r^*.

LLM judge $\mathcal{J}$. Samples passing $\mathcal{V}$ are scored at temperature 0 on four axes: query naturalness, tier compliance, label correctness, and *trace faithfulness* — whether the reasoning in z genuinely entails A rather than rationalizing it post hoc. Rejected samples are regenerated with the judge's critique injected as feedback (up to a fixed retry budget of 5) and dropped thereafter.

C Training Setup

Both Stage 1 and Stage 2 use standard autoregressive supervised fine-tuning. Each example is rendered as a chat-style prompt-completion pair, and the cross-entropy loss is masked over the user/system prompt tokens so that the model is supervised only on the assistant completion — in Stage 2 this is the trace–answer pair $o = (\tilde{z},\, y)$ given the query q, and in Stage 1 the format-specific completion (e.g. the gold token v_t in the $\text{desc}\rightarrow v_t$ format). Optimizer, learning-rate schedule, batch size, sequence length, and LoRA rank/α/dropout follow the ToolSense recipe (Anonymous, 2026).

Models and hyperparameters. All primary experiments use Gemma4-E4B-it with LoRA with embedding layers fully finetuned ($r = 64$, $\alpha = 128$) both for Stage 1 and Stage 2. Both stages use AdamW with cosine schedule (Stage 1: lr =5e-5, Stage 2: lr = 1e-4) with minimum LR ratio of 0.1, batch size of 8 for Stage 1 and 16 for Stage 2, bf16 precision, on a single H200 GPU.

Statistical reporting. All retrieval and probing metrics are reported with 95% confidence intervals via paired bootstrap (B=1,000) over per-example predictions. $\text{MCQ}_{\text{expert}}$ is reported as a point estimate (n=454).

D Inference

At inference time TRACE decodes greedily in a single forward pass: given a user query q, the model emits the trace $\hat{z}$ followed by the JSON token list $\hat{y} = [\, v_{a_1}, v_{a_2}, \ldots]$, terminating on the closing JSON bracket. We do not employ constrained beam search or prefix-trie expansion over the virtual-token vocabulary as in ToolGen (Wang et al., 2025); the parametric grounding from Stage 1 plus the deliberative scaffold from Stage 2 are sufficient to keep the answer list inside the valid token alphabet $\{v_t : t \in \mathcal{C}\}$ in practice (off-vocabulary rate of mere 3% for TRACE $(c, m, r{+}R)$). The retrieved tool set is recovered as $\hat{A}(q) = \{t : v_t \in y\}$. This single-beam, no-constraint decoding is the source of TRACE's production-latency benefit: end-to-end retrieval cost is one prompt-prefill plus a short autoregressive generation, with no per-step trie lookup. Latency measurements are reported in §4.6.

E Detailed Evaluation Results

This section provides the full evaluation results for the experiments summarized in the main body, including confidence intervals and all metrics.

E.1 Probing Evaluations

This appendix tabulates per-format probing results for every TRACE variant we trained, completing the results in §4.2. Throughout, runs are referenced by their configuration triple (F, M, R) where F is the token format (a flat, b bare-hier, c wrapped-hier), M is the Stage-1 mode (m memo, mr reasoning memo, $\varnothing$ none), and R is the Stage-2 mode ($\varnothing$ none, n non-reasoning retrieval, n→r non-reasoning then reasoning, r reasoning, r+R reasoning with rule grounding). All numbers are accuracy in %; brackets are 95% paired-bootstrap CIs (B=1,000). $\text{MCQ}_{\text{expert}}$ is reported as a point estimate.

Format sweep (Table 4). The dissociation under R=n is consistent across all three token formats: every $(\cdot, \text{m}, \text{n})$ row sits around the random baseline

F	M	R	MCQ_{ts}-A	MCQ_{ts}-B	QA_{ts}-A	QA_{ts}-B	$\text{MCQ}_{\text{expert}}$
a	m	∅	52.3 [47.0, 57.7]	50.2 [45.5, 55.5]	80.0 [74.5, 85.5]	67.8 [62.7, 72.2]	63.4
	m	n	27.3 [22.7, 32.7]	29.8 [25.2, 34.2]	51.5 [44.5, 58.5]	52.5 [47.5, 57.0]	34.8
	m	n→r	65.0 [60.0, 70.7]	71.5 [67.2, 76.0]	76.5 [71.0, 82.0]	80.0 [76.0, 83.8]	49.8
	m	r	68.7 [63.7, 73.7]	70.0 [65.5, 74.5]	87.5 [83.0, 91.5]	85.5 [82.0, 88.8]	59.5
b	m	∅	43.3 [37.3, 49.0]	37.2 [32.8, 42.0]	61.5 [54.5, 67.5]	55.8 [50.7, 60.8]	58.6
	m	n	28.3 [23.3, 33.3]	27.0 [22.5, 31.8]	45.5 [38.5, 52.5]	43.0 [38.2, 48.0]	30.0
	m	n→r	28.7 [23.7, 34.0]	32.0 [27.5, 36.5]	80.5 [75.0, 85.5]	69.0 [64.5, 73.2]	28.2
	m	r	46.3 [41.0, 52.0]	44.0 [39.2, 48.8]	84.0 [79.0, 88.5]	79.8 [75.8, 83.5]	50.9
c	m	∅	60.0 [54.3, 65.3]	59.0 [54.0, 63.7]	75.0 [69.0, 81.0]	69.2 [64.8, 74.0]	69.2
	m	n	24.3 [19.3, 29.3]	23.5 [19.5, 27.5]	50.0 [43.0, 56.5]	50.0 [45.2, 54.8]	33.7
	m	n→r	24.7 [19.7, 29.7]	24.8 [20.2, 28.7]	50.0 [43.0, 57.0]	50.0 [45.5, 54.8]	31.5
	m	r	57.7 [52.3, 63.0]	67.8 [63.2, 72.2]	83.0 [77.5, 88.5]	79.2 [75.5, 83.0]	61.5

Table 4: **Token-format sweep.** Probing accuracy across the three token formats and four Stage-2 modes, all with non-reasoning Stage-1 (M=m). Non-reasoning retrieval (R=n) collapses every probe across all formats; reasoning retrieval (R=r) recovers it. Bolting reasoning onto a non-reasoning checkpoint (R=n→r) only partially recovers MCQ accuracy and fails entirely for $F\in\{\text{b}, \text{c}\}$.

F	M	R	MCQ_{ts}-A	MCQ_{ts}-B	$\text{MCQ}_{\text{expert}}$
a	∅	r	74.0 [69.0, 78.7]	81.8 [78.0, 85.2]	67.8
a	m	r	68.7 [63.7, 73.7]	70.0 [65.5, 74.5]	59.5
a	mr	r	62.3 [57.0, 68.0]	73.5 [69.0, 77.5]	65.2
b	m	r	46.3 [41.0, 52.0]	44.0 [39.2, 48.8]	50.9
b	mr	r	62.0 [56.3, 67.3]	71.5 [67.0, 76.0]	63.2
c	m	r	57.7 [52.3, 63.0]	67.8 [63.2, 72.2]	61.5
c	mr	r	58.3 [53.0, 63.7]	67.8 [63.2, 72.2]	65.2

Table 5: **Stage-1 ablation under reasoning retrieval.** Holding R=r fixed, we vary $M \in \{\varnothing, \text{m}, \text{mr}\}$ to isolate Stage-1's contribution. The $(\text{a}, \varnothing, \text{r})$ row is the no-Stage-1 ablation: it recovers $\text{MCQ}_{\text{expert}}$ to within 1 pp of the base model, but has the lowest retrieval. Reasoning Stage-1 (M=mr) outperforms non-reasoning Stage-1 (M=m) on $\text{MCQ}_{\text{expert}}$ for every F.

F	M	R	MCQ_{ts}-A	MCQ_{ts}-B	$\text{MCQ}_{\text{expert}}$
a	m	r	68.7 [63.7, 73.7]	70.0 [65.5, 74.5]	59.5
a	m	r+R(3q)	66.7 [61.3, 72.0]	62.5 [57.7, 67.2]	59.2
a	m	r+R(6q)	68.0 [63.0, 73.7]	66.2 [61.7, 70.8]	58.1
a	m	r+R(9q)	66.3 [61.0, 71.7]	62.0 [57.5, 67.0]	57.9
a	m	r+R(*app*)	62.3 [56.3, 67.7]	66.8 [61.8, 71.5]	—
a	m	r+R(*rep*)	65.7 [60.3, 71.0]	63.0 [58.2, 68.0]	56.2
a	m	r+R	58.7 [53.0, 64.3]	68.0 [63.2, 72.3]	55.7
c	m	r	57.7 [52.3, 63.0]	67.8 [63.2, 72.2]	61.5
c	m	r+R	56.7 [51.0, 62.3]	71.0 [66.8, 75.2]	56.4

Table 6: **Rule-grounded scaling.** Probing accuracy along the rule-data axis. Rows above the rule entries reproduce the no-rules baseline (F, m, r) for direct comparison. Held-out $\text{MCQ}_{\text{expert}}$ falls monotonically with rule-data volume; in-distribution probes are stable.

on all probes. Replacing R=n with R=r recovers the in-distribution probes to within a few points of the $(\cdot, \text{m}, \varnothing)$ ceiling for every F. The n→r curriculum is informative: on flat tokens (F=a) the late reasoning bolt-on partially recovers MCQ; on hierarchical formats ($F\in\{\text{b}, \text{c}\}$) it does not, suggesting that the damage caused by non-reasoning training to the description–token binding is irrecoverable when the token vocabulary itself carries structural content. Wrapped-hierarchical (F=c) achieves the highest $(\cdot, \text{m}, \varnothing)$ ceiling on $\text{MCQ}_{\text{expert}}$ (69.2), which is why we showcased the results corresponding to that format in the main body.

Stage-1 ablation (Table 5). The $(\text{a}, \varnothing, \text{r})$ row is striking: training reasoning retrieval directly from the base model (no Stage-1) recovers $\text{MCQ}_{\text{expert}}$ to 67.8 — within 1.4 pp of the strongest Stage-1 ceiling $(\text{c}, \text{m}, \varnothing) = 69.2$ — but its retrieval recall is the lowest of any TRACE variant. This suggests that Stage-1 alone is not responsible for building tool understanding. Switching M=m → mr (reasoning at Stage-1) yields a consistent 2–13 pp gain on $\text{MCQ}_{\text{expert}}$ across formats, suggesting that reasoning at the memorization stage helps in improving tool understanding.

Rule-grounded scaling (Table 6). The rule-data axis is reported in two layers: the upper block sweeps $(F$=a, m, r+R$(\cdot))$ across rule-data volumes (k q/rule, append vs. replace strategies); the lower block reproduces the headline $(F$=c, m, $\cdot)$ pair for direct comparison with the body. Two trends are visible. First, in-distribution probes are stable across the rule-data axis: MCQ_{ts} and QA_{ts} accuracy moves within their own CIs as we scale rule-data volume. Second, $\text{MCQ}_{\text{expert}}$ exhibits the Pareto cost: each rule-data increment costs roughly 1–2 pp of held-out general-domain knowledge, with the lowest landing at 55.7 (F=a)

and 56.4 (F=c), still well above the R=n collapse but below the no-rules ceiling. The (*app*) row's missing $\text{MCQ}_{\text{expert}}$ entry is because the corresponding checkpoint has not yet been evaluated.

E.2 Retrieval Evaluations

Tables 7 and 8 tabulate the full set of retrieval results for every (F, M, R) checkpoint we trained, completing the comparisons in §4.3. Each checkpoint is reported under whichever decoding mode it was evaluated with (non-reasoning checkpoints under constrained decoding, reasoning checkpoints under single-beam reasoning decoding); for the two TRACE-class checkpoints that we additionally re-evaluated under constrained decoding (suffix *cstr*, marked with *), both decoding modes appear. Numbers are recall in %; brackets are 95% paired-bootstrap CIs (B=1,000) over per-example predictions.

E.3 Qualitative Trace Example

Figure 5 shows a representative side-by-side comparison of model reasoning traces before and after rule grounding, illustrating how the <think> trace mechanism enables explicit rule citation for ambiguous queries. The left panel shows the ungrounded model reasoning plausibly but arriving at the wrong tool; the right panel shows how citing the governing business rule within the trace corrects the retrieval.

F $\text{MCQ}_{\text{expert}}$ Evaluation Details

$\text{MCQ}_{\text{expert}}$ is a multiple-choice question dataset handcrafted by domain experts, used to evaluate model performance on Domain A and Domain B knowledge.

F.1 Dataset Construction

We acquire a large dataset of multiple-choice questions and answers written by domain experts, used to assess domain understanding when preparing for certification exams. Each question belongs to a lesson group which is a curated set of instructional materials such as videos and documentation. We retain only questions relevant to Domain A and Domain B by filtering on keywords 'A' and 'B' in the lesson group title, followed by de-duplication to remove questions appearing in multiple lessons. This leaves 454 multiple-choice questions, each with 2 to 5 answer choices and exactly one correct answer, with the distribution of answer choices shown in Table 9. The random chance baseline is computed as $\frac{1}{N}\sum_{i=1}^{N}\frac{1}{k_i}$, where $N = 454$ and k_i is the number of choices for question i. With n_k denoting the number of questions with k choices (Table 9), this gives:

$$\begin{aligned}\text{MCQ}_{\text{expert}}^{\text{random}} &= \frac{1}{N}\sum_{k=2}^{5}\frac{n_k}{k} \\ &= \frac{1}{454}\left(\frac{58}{2}+\frac{74}{3}+\frac{301}{4}+\frac{21}{5}\right) \\ &\approx 0.2932\end{aligned} \tag{5}$$

F.2 Model Inference

During inference, we prompt the model with the question content and the answer choices, where each answer choice is prefixed by the corresponding indexed letter of the English alphabet. We perform greedy constrained decoding, restricting the output vocabulary to the set of valid answer letters. Formally, given a question with k choices, the model's prediction is:

$$\hat{a} = \arg\max_{l \in \mathcal{A}_k} P(l \mid \text{prompt}) \tag{6}$$

where $\mathcal{A}_k = \{\texttt{A}, \texttt{B}, \dots\}$ is the set of k valid answer letters and $P(l \mid \text{prompt})$ is the model's predicted probability for letter l. The model is evaluated by comparing $\hat{a}$ to the ground truth label a^*, and accuracy is reported as the fraction of correct predictions over all N questions.

F.3 Example Question

The following example is selected at random from the dataset of 454 questions.

$\text{MCQ}_{\text{expert}}$ Example Question

```
Answer the following question.
Respond with the answer letter only.

Which of the following steps is NOT
typically part of the basic financing
process for indirect materials?
A: Requirements Determination
B: Source of Supply Determination
C: Production Planning and Control
D: Purchase Order Processing
Answer (single letter only):
```

The correct answer is C.

			Domain A			Domain B		
F	M	R	R@1	R@5	R@10	R@1	R@5	R@10
Non-reasoning retrieval baselines								
a	m	n	39.7 [31.3, 48.1]	80.9 [74.0, 87.8]	91.6 [86.3, 96.2]	31.7 [23.6, 40.7]	69.9 [61.8, 77.2]	80.5 [73.2, 87.0]
b	m	n	44.3 [35.9, 52.7]	80.2 [73.3, 86.3]	86.3 [80.2, 91.6]	40.7 [31.7, 49.6]	70.7 [63.4, 77.2]	79.7 [73.2, 86.2]
c	m	n	45.0 [36.6, 54.2]	80.9 [74.0, 87.0]	87.0 [80.9, 92.4]	35.8 [27.6, 43.9]	73.2 [65.9, 80.5]	78.0 [70.7, 85.4]
TRACE*-class re-evaluations under constrained decoding (suffix *)*								
c	m	r*	40.5 [32.1, 48.9]	67.2 [58.8, 75.6]	74.0 [66.4, 81.7]	46.3 [37.4, 54.5]	75.6 [67.5, 82.9]	79.7 [72.4, 87.0]
a	m	r+R*	24.4 [16.8, 32.1]	44.3 [35.9, 52.7]	54.2 [45.8, 62.6]	25.2 [17.9, 33.3]	52.8 [43.9, 61.8]	65.0 [56.9, 73.2]
c	m	r+R*	48.9 [40.5, 57.3]	78.6 [71.0, 85.5]	86.3 [79.4, 91.6]	36.6 [27.6, 45.5]	69.1 [60.2, 77.2]	77.2 [69.9, 84.6]

Table 7: **Constrained-decoded retrieval (R@1, R@5, R@10).** Includes the non-reasoning retrieval baselines $(\cdot, \mathsf{m}, \mathsf{n})$ across all three token formats and the apple-to-apple re-evaluations (suffix *) of two TRACE-class reasoning checkpoints.

Before Rule Grounding

Query: *"Show my cost center details"*

```
<think>
The user wants cost center details. . . .
Cost centers relate to org units.
WorkAssignment has org unit fields.
. . . </think>
```

Output: `["«User&&WorkAssignment»"]`

× **Incorrect** – wrong entity selected

After Rule Grounding

Query: *"Show my cost center details"*

```
<think>
The user wants cost center details. . . .
Per MASTER_DATA rule: employee-attribute
queries incl. cost center → EmployeeInfo.
WorkAssignment does not hold cost center.
. . . </think>
```

Output: `["«Employee&&EmployeeInfo»"]`

✓ **Correct** – rule citation guides routing

Figure 5: **Rule grounding corrects ambiguous tool routing.** Without rule grounding (left), the model reasons plausibly but selects the wrong tool because it lacks the domain routing constraint. After rule grounding (right), the reasoning trace explicitly cites the business rule, producing the correct retrieval.

G Business Rule Catalog

The business rule catalog $\mathcal{B}$ comprises **123 rules** across the two domains: 20 rules for Domain A and 103 rules for Domain B . Each rule is authored by domain experts and specifies how to disambiguate between a set of semantically overlapping tools (the *confusible set* $\mathcal{T}(r)$). Rules capture API deprecations, versioning constraints, product-line routing logic, and domain-specific data-model boundaries that are not inferable from tool descriptions alone.

G.1 Rule Structure

Each rule $r \in \mathcal{B}$ is stored as a JSON object with three fields: **`tool_name`** (the target tool the rule governs), **`confusables`** (the set of tools $\mathcal{T}(r)$ that overlap semantically with the target), and **`rule_text`** (a natural-language directive specifying when to route to this tool vs. its confusables).

G.2 Example Rule

The following example from Domain A illustrates a business rule.

Example Rule

```
{
  "tool_name": "API1/EndpointA",
  "confusables": [
    "API1/EndpointB",
    "API1/EndpointC",
    "API1/EndpointD"
  ],
  "rule_text": "EndpointA is the default
      for current-state queries that do
      not require historical data. Route
      to EndpointB when the query
      involves historical records or
      date-range tracking. Route to
      EndpointC when the query targets
      transactional or line-item level
      details, and to EndpointD
      otherwise."
}
```

F	M	R	Domain A R@1	Domain A R@gen	Domain B R@1	Domain B R@gen
Non-reasoning then reasoning (M=m, R=n→r)						
a	m	n→r	44.3 [35.9, 53.4]	58.0 [49.6, 66.4]	44.7 [35.8, 53.7]	52.0 [43.1, 61.0]
b	m	n→r	48.9 [40.5, 57.3]	61.8 [53.4, 69.5]	56.9 [48.0, 65.9]	63.4 [54.5, 71.5]
c	m	n→r	34.4 [26.7, 42.7]	45.0 [36.6, 53.4]	48.0 [39.0, 56.9]	54.5 [45.5, 63.4]
Reasoning retrieval, no rules (M=m, R=r)						
a	m	r	38.2 [29.8, 46.6]	55.7 [46.6, 65.6]	55.3 [46.3, 64.2]	62.6 [53.7, 70.7]
b	m	r	30.5 [22.9, 38.2]	46.6 [38.2, 55.0]	54.5 [45.5, 63.4]	60.2 [51.2, 68.3]
c	m	r	32.1 [24.4, 40.5]	53.4 [45.0, 61.1]	56.1 [47.2, 65.0]	67.5 [59.3, 74.8]
Reasoning memo + reasoning retrieval (M=mr, R=r)						
a	mr	r	31.3 [23.7, 39.0]	41.2 [33.6, 49.6]	44.7 [35.8, 53.7]	61.0 [51.2, 69.9]
b	mr	r	42.0 [33.6, 50.4]	52.7 [44.3, 61.1]	52.0 [43.1, 61.0]	65.0 [56.1, 73.2]
c	mr	r	41.2 [33.6, 49.6]	49.6 [41.2, 58.0]	53.7 [44.7, 62.6]	62.6 [53.7, 71.5]
Direct reasoning, no Stage-1 (M=∅, R=r)						
a	∅	r	29.8 [22.1, 37.4]	45.8 [37.4, 54.2]	46.3 [37.4, 55.3]	61.8 [52.8, 70.7]
Rule-grounded reasoning retrieval on F=a scaling sweep						
a	m	r+R(3q)	25.2 [17.6, 32.8]	48.9 [40.5, 57.3]	59.3 [50.4, 68.3]	65.0 [56.1, 73.2]
a	m	r+R(6q)	42.0 [33.6, 50.4]	57.3 [48.9, 65.6]	52.0 [43.1, 61.0]	61.8 [52.8, 70.7]
a	m	r+R(9q)	38.9 [30.5, 47.3]	57.3 [48.9, 65.6]	68.3 [59.3, 77.2]	72.4 [64.2, 80.5]
a	m	r+R(12)	45.8 [37.4, 54.2]	73.3 [65.6, 80.9]	62.6 [53.7, 71.5]	65.9 [56.9, 74.0]
a	m	r+R(12-rep)	44.3 [35.9, 52.7]	78.6 [71.8, 84.7]	52.8 [43.9, 61.8]	61.8 [52.8, 70.7]
a	m	r+R	48.1 [40.5, 57.3]	86.3 [80.9, 91.6]	61.0 [52.0, 69.9]	64.2 [56.1, 72.4]
TRACE *on F=c (headline)*						
c	m	r+R	51.1 [42.7, 60.3]	85.5 [79.4, 90.8]	43.1 [35.0, 52.1]	60.2 [51.2, 69.1]

Table 8: **Single-beam reasoning retrieval (R@1, R@gen).** Each row is a checkpoint evaluated by letting the model commit to its own answer set; R@gen is recall over the generated set. Sections group runs by (M, R).

No. of Choices	No. of Questions
2	58
3	74
4	301
5	21
Total	454

Table 9: Distribution of answer choices in the $\text{MCQ}_{\text{expert}}$ dataset.

G.3 Rule Statistics

	Domain A	Domain B
Rules	20	103
Avg. confusables	4.2	2.8
Avg. length (words)	∼180	∼120

Table 10: Business rule catalog statistics.

H System Prompts

This section reproduces the system prompts used in the Stage 2 data synthesis pipeline (§3.3). All prompts use Jinja2 templating; variables in double braces (e.g., `{{tool_name}}`) are filled at runtime.

H.1 Rule-Targeted Query Generation

The following prompt generates queries in three categories (explicit, implicit, exception) for a given business rule. It is invoked once per (rule, category) pair..

Prompt: Rule-Targeted Query Generator

```
You are tasked at generating diverse user
queries to evaluate a business routing rule
for a specific tool.

<BUSINESS RULE>
Tool: {{tool_name}}
Rule: {{rule_name}}
{{rule_text}}
</BUSINESS RULE>

<TOOL DESCRIPTION>
{{tool_description}}
```

</TOOL DESCRIPTION>

<CATEGORY>
Generate queries in the "{{category}}" category:
- **explicit**: The query wording makes it obvious that the business rule applies. Keywords, time references, or field names directly signal the rule.
- **implicit**: The query requires applying the same rule, but the wording does NOT directly signal it. No obvious keywords — the connection is indirect.
- **exception**: The query hits an edge case or exception path within the rule. It might seem like the rule's primary tool is correct at first glance, but the exception clause redirects to a different tool mentioned in the rule.

You are generating "{{category}}" queries.
</CATEGORY>

Generate exactly {{n_queries}} diverse user queries where this business rule determines the correct tool selection. For each query, provide:
- The generated query
- The correct target tool in API/ENDPOINT format
- A one-sentence explanation of why the chosen tool is the target based on the rule

Important context: This rule is FOR the tool "{{tool_name}}". For explicit and implicit categories, the target tool should primarily be "{{tool_name}}". For exception category, the target tool may be a DIFFERENT tool mentioned in the rule.

Requirements:
1. Each query should be a realistic question a user would ask a system.
2. Use different parameters that could go as input to the query where applicable.
3. Vary phrasing: questions, imperatives, natural language, formal, informal
4. Do NOT reference the rule name, tool name, or any API/technical terminology in the query
5. Each query must have a different semantic focus — avoid duplicates
6. The target tool MUST be one of the tools mentioned in the business rule

H.2 Reasoning Trace Generation

The following prompt generates a structured reasoning trace for a given user query, as described in the trace generation step of §3.3

Prompt: Reasoning Trace Generator

You are a tool selection reasoning engine. Given a user query, a set of candidate tools, and the correct tool(s) for the query, generate a structured reasoning trace that explains why the selected tool(s) are the best match.

[START OF INPUT]
<USER QUERY> {{user_query}} </USER QUERY>
<USER PERMISSION TARGETS>
{{user_query_permission_targets}} </USER PERMISSION TARGETS>
<CANDIDATE TOOLS> {{candidate_pool}} </CANDIDATE TOOLS>
<SELECTED TOOLS> {{selected_tools}} </SELECTED TOOLS>
[END OF INPUT]

[START OF TOOL STRUCTURE]
Each tool has a tool_name in the format API/ENDPOINT.
- API is the first part before the "/" — it represents the API service.
- ENDPOINT is the second part after the "/" — it represents a specific endpoint within that API service.
- Multiple tools can share the same API but differ in Endpoints.
[END OF TOOL STRUCTURE]

[START OF TASK DESCRIPTION]
Generate a structured reasoning trace that explains why the selected tool(s) are the correct match for the user query. Follow the reasoning steps, and always follow business rule (if given below) that captures the discriminating principle for selection, and populate the output fields accordingly.
[END OF TASK DESCRIPTION]

[START OF BUSINESS RULE]
{{business_rule}}
[END OF BUSINESS RULE]

[START OF REASONING STEPS]
STEP 1: PERMISSION ACCESS FILTERING
- Each tool has permission targets representing required permissions.
- Exclude any tool whose permission targets share no overlap with the user's permission targets.
- Briefly state this filtering in 1-2 sentences. Do NOT name specific permission tags or tools. Simply describe the user's permission scope in general terms.
STEP 2: SEMANTIC REASONING
Reason through why the selected tool(s) are the best match. Follow this hierarchy strictly:
1. API first: Identify and compare the most relevant services.
2. Then entities: Within the relevant API(s), explain why the selected endpoint(s) are correct over alternatives.
3. Conclude: Summarize why the selected tool(s) win.

When candidate tools share the same API and differ only in endpoint, apply domain-specific routing rules to ground endpoint selection. When applying a routing principle, state the full rule text verbatim.

Keep reasoning tight — only discuss genuine contenders. Do not enumerate clearly irrelevant tools.
[END OF REASONING STEPS]

```
[START OF OUTPUT FIELDS]
- permission_filtering.reasoning: Step 1 access scope text.
- semantic_reasoning.reasoning: Step 2 semantic reasoning text.
[END OF OUTPUT FIELDS]

[START OF INSTRUCTIONS]
1. Only reason over tools explicitly provided — do not hallucinate.
2. Use exact tool names as given — no abbreviations.
3. Do not infer functionality beyond what is stated in descriptions.
4. Reason progressively: APIs -> Endpoints -> selection.
5. Permission step: do not mention specific Permission tags.
6. Trace must read as clean single-pass reasoning — no mention of feedback.
7. No mention of candidate pool/set — reason as if from own knowledge.
8. business_rule must be followed if applicable and provided.
9. No meta-phrases referencing this prompt or instructions.
[END OF INSTRUCTIONS]
```